

DEVELOPMENT OF A SMALL SCALE COREXY TYPE 3D PRINTER

YEOH XING YUAN


**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Engineering
(Honours) Mechatronics Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

May 2021

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :  _____
Name : YEOH XING YUAN _____
ID No. : 16UEB03286 _____
Date : 5 MAY 2021 _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**DEVELOPMENT OF A SMALL SCALE COREXY TYPE 3D PRINTER**” was prepared by **YEOH XING YUAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature :  _____

Supervisor : DR. LEE JER VUI

Date : 5 MAY 2021

Signature :  _____

Co-Supervisor : IR. DR. CHUAH YEA DAT

Date : 8 April 2021

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2021, Yeoh Xing Yuan. All right reserved.

ABSTRACT

The 3D printing technology is a high demand field of study as every components of the 3D printing technology required sufficient and detailed research and development before it can reach the functional stage today; expand, improve and grow in the future. It can be claimed that 3D printing is among the important field to study for the development and enhancement of skills to prepare ourselves for the future world of industry 4.0.

This project aims to step up the normal printing speed of low-cost FDM 3D printer in the market priced below MYR 1, 000 which usually in the coreXZ structure by the structural change and other modifications made. This research provides a detailed development process of a small scale coreXY type 3D printer from scratch. The developed prototype targeted to achieved more than 80mm/s in normal printing speed with the manufacturing cost limited to MYR 1, 000. This project can expose us to various fields included but not limited to mechanical design, electronics circuit connection, components selection, firmware configuration, quality control, 3D printing knowledge, idea realization and material study.

For the prototype developed, the base model of the design derived from Hypercube model by Tech2C. The firmware used is the open source Marlin Firmware 2.0.7. With the improvement on printing speed, a shorter printing time permissible for the same model and the efficiency of the printer increased. Besides, the operation cost mainly induced by the heating elements, hot end and heated bed can be reduced. In short, more parts can be printed in a shorter period of time, it takes shorter time for product realization and able to promote the creativity and innovation of people in a budgeted way. This project has achieved a 100mm/s normal printing speed for the FDM 3D printer developed using core XY structure that can be claimed to have escaped from the “slow” category among the 3D printers in the similar price range.

TABLE OF CONTENTS

DECLARATION	i
APPROVAL FOR SUBMISSION	ii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF TABLES	viii
LIST OF FIGURES	x
LIST OF SYMBOLS / ABBREVIATIONS	xvi
LIST OF APPENDICES	xviii

CHAPTER

1	INTRODUCTION	1
	1.1 General Introduction	1
	1.2 Importance of the Study	3
	1.3 Problem Statement	3
	1.4 Aim and Objectives	4
	1.5 Scope and Limitation of the Study	4
2	LITERATURE REVIEW	6
	2.1 Introduction	6
	2.2 FDM Technology	7
	2.2.1 FDM 3D Printing Optimization	9
	2.3 Core XY Structure	10
	2.4 Electronics Components	13
	2.4.1 Stepper Motor	14
	2.4.2 Heated Bed	15
	2.4.3 Hot End	16
	2.4.4 Extruder	17
	2.4.5 Motherboard	19
	2.5 Human Machine Interface	24
	2.6 Firmware	27

	2.6.1	G-code	28
	2.6.2	Temperature Control	31
	2.7	Summary	32
3		METHODOLOGY AND WORK PLAN	33
	3.1	Introduction	33
	3.2	Requirements	34
	3.3	Components Selection	34
	3.3.1	Main Frame	35
	3.3.2	Motion Related Components	36
	3.3.3	Fastener	39
	3.3.4	Electronics Parts	40
	3.4	Mechanical Design	47
	3.4.1	Dimensional Computation	47
	3.4.2	3D Printable Part Design	49
	3.4.3	Assembly	58
	3.5	Electronics System	65
	3.6	Firmware Configuration	68
	3.6.1	Parameters Calculation	69
	3.7	Summary of Experiments	70
4		RESULT AND DISCUSSION	71
	4.1	Project Prototype	71
	4.1.1	Fiscal	72
	4.2	Calibration	74
	4.2.1	PID Auto Tune	74
	4.2.2	E-steps Calibration	75
	4.2.3	Slicer Flow Tuning	76
	4.2.4	Retraction Tuning	77
	4.2.5	Temperature Tuning	79
	4.2.6	Acceleration Tuning	79
	4.2.7	Linear Advance	81
	4.3	Feature and Performance of Prototype	82
	4.3.1	Accuracy of Printed Products	83
	4.3.2	Speed Limit of Printing Process	85
	4.4	Problem Encountered and Improvement	88

5	CONCLUSION	91
5.1	Conclusion	91
5.2	Limitation of The Prototype	92
5.3	Recommendation for Future Work	93
	REFERENCES	95
	APPENDICES	106

LIST OF TABLES

Table 1.1: Benefits of 3D Printing Technology in Different Industrial Fields.	2
Table 1.2: Comparison Between Additive Manufacturing Technology and Traditional Manufacturing Technology. (Sunil and Abdullah, 2015)	3
Table 1.3: Classification of FDM 3D Printer in Term of Speed. (Joel, 2020)	3
Table 2.1: Motion Study in CoreXY Structure.	11
Table 2.2: Summary of the Benefits of CoreXY Structure in 3D Printing. (Grames, 2019)	12
Table 2.3: Comparison Between Direct Drive and Bowden Extruder Types. (Landry, 2016)	18
Table 2.4: 3D Printer's Firmware List. (RepRap, 2020d)	27
Table 2.5: Thermal Protections by Marlin Firmware. (Marlin, 2020)	32
Table 3.1: FDM 3D Printer Prototype Specification Rough Plan.	34
Table 3.2: Tabulation of Constants and Presumptions Made in Calculation.	48
Table 3.3: Approximate Dimensional Calculation of Carriages and Restricted Regions.	49
Table 3.4: Draft Calculation of Material Length Required.	49
Table 3.5: Final Decision of Materials' Length Required.	62
Table 3.6: Bill of Materials of Proposed Assembly.	63
Table 3.7: Calculation of Motion Parameters Required for All the Stepper Motors.	69
Table 4.1: Tabulation of Cost for Developed Prototype.	73
Table 4.2: Procedure in E-steps Calibration.	75
Table 4.3: Slicer's Extrusion Flow Computation.	77

Table 4.4: Acceleration and Junction Deviation Parameters Determination with Accerelation Towers.	80
Table 4.5: Prototype's Accuracy Evaluation Based on 20mm Cube Printed.	83
Table 4.6: Update on Step/mm for Each Axis Based on Printed Cube Result.	84
Table 4.7: Prototype's Accuracy Evaluation Based on 20mm Cube Printed with Updated Step/mm for Each Axis.	84
Table 4.8: Computation of Prototype's Speed Limit and Safe Extrusion Limit.	86
Table 4.9: Printing Speed Test for Prototype.	87

LIST OF FIGURES

Figure 1.1: FDM 3D Printers with Different Movement Control Mechanism. (Alex, 2017)	2
Figure 1.2: Industry 4.0 Key Components Mapping. (PlumLogix, 2019)	3
Figure 2.1: Rough Map of Main Components for 3D Printer Development.	6
Figure 2.2: FDM Illustration. (Sidambe, 2014)	7
Figure 2.3: FDM Printing Without Appropriate Support Structure. (Higgs, 2018)	8
Figure 2.4: Relationship Between Tensile Modulus and Infill Density for PLA. (Abeykoon, Sri-Amphorn and Fernando, 2020)	9
Figure 2.5: Mechanism of CoreXY Structure. (Moyer, 2012)	10
Figure 2.6: Two Versions of CoreXY Belt System. (Hoge, n.d.)	11
Figure 2.7: Waveform of X-axis Displacement of the Machine. (Zhu and Lee, 2019)	12
Figure 2.8: Waveform of Y-axis Displacement of the Machine. (Zhu and Lee, 2019)	13
Figure 2.9: Functionality Diagram of FDM 3D Printer. (<i>3D Printer electronics design</i> , 2014)	14
Figure 2.10: Open-loop Position Control [above] and Closed-loop Position Control [below]. (Mike, 2014)	15
Figure 2.11: FDM 3D Printing Warping Deformation. (Alsoufi and El-Sayed, 2017)	16
Figure 2.12: Illustration of Typical Hot End Assembly. (Filament2Print, 2020)	16
Figure 2.13: Measurement of Extrudate Temperature at Different Feeding Rate for Different Materials. (Serdeczny, et al., 2020)	17
Figure 2.14: Different Types of Extruder for FDM 3D Printing. (Boichut, 2019)	18

Figure 2.15: MKS Gen V1.4 3D Printer Motherboard Layout. (Keyestudio, 2019)	19
Figure 2.16: Average Output Voltage from PWM with Different Duty Cycle. (Heath, 2017)	20
Figure 2.17: Micro-stepping. (TRINAMIC Motion Control GmbH & Co., 2020c)	22
Figure 2.18: StealthChop™. (TRINAMIC Motion Control GmbH & Co., 2020b)	22
Figure 2.19: SpreadCycle™. (TRINAMIC Motion Control GmbH & Co., 2020b)	23
Figure 2.20: StallGuard™. (TRINAMIC Motion Control GmbH & Co., 2020d)	24
Figure 2.21: CoolStep™. (TRINAMIC Motion Control GmbH & Co., 2020d)	24
Figure 2.22: Pronterface with Model Loaded and Ready to Print. (Horne and Hausman, n.d.a)	25
Figure 2.23: Full Graphic Control Panel with SD Card Reader. (RepRap, 2019b)	26
Figure 2.24: OctoPrint Web Interface. (Häußge, 2020)	27
Figure 2.25: Software Management of 3D Printer. (Lyubomirov, Nedeva and Bundeava, 2015)	29
Figure 2.26: Slicing Software with Model Loaded to Produce G-code for 3D Printing. (Carlota, 2019)	30
Figure 2.27: Sample of G-code Fragment. (RepRap, 2020c)	30
Figure 3.1: European Standard 2020 Aluminium Extrusion Profile 3D Model.	35
Figure 3.2: Corner Bracket for European Standard 2020 Extrusion 3D Model.	35
Figure 3.3: M5 T-nut for European Standard 2020 Extrusion 3D Model.	35
Figure 3.4: 20 Teeth GT2 Timing Pulley 3D Model.	36
Figure 3.5: GT2 Idler Pulley with Teeth 3D Model.	37

Figure 3.6: GT2 Idler Pulley without Teeth 3D Model.	37
Figure 3.7: Open Loop 6mm Width GT2 Timing Belt.	37
Figure 3.8: T8 Acme Thread Lead Screw with 8mm Pitch 3D Model.	37
Figure 3.9: POM Anti-backlash Nut Block 3D Model.	37
Figure 3.10: 5mm to 8mm Flexible Shaft Coupler 3D Model.	38
Figure 3.11: Linear Shaft Rods with 8mm, 10mm and 12mm Diameter 3D Model.	38
Figure 3.12: LM10LUU and LM8LUU Graphite Insert Brass Bushing 3D Model.	39
Figure 3.13: LM12UU Linear Ball Bearing 3D Model.	39
Figure 3.14: 608zz Ball Bearing 3D Model.	39
Figure 3.15: M3 Brass Knurled Threaded Insert 3D Model.	40
Figure 3.16: Brief Data of NEMA 17 4401S. (Handson Technology, 2014)	40
Figure 3.17: NEMA 17 4401S Stepper Motor 3D Model.	40
Figure 3.18: MK3 Heated Bed 3D Model.	41
Figure 3.19: MK3 Heated Bed Levelling Tool Set 3D Model.	41
Figure 3.20: E3D V6 Hot End 3D Model.	42
Figure 3.21: 5015 Radial Cooling Fan 3D Model.	42
Figure 3.22: BMG Extruder 3D Model.	43
Figure 3.23: Extruder Mounting Piece 3D Model.	43
Figure 3.24: Motherboard 3D Model.	44
Figure 3.25: Features of MKS Robin E3 Motherboard. (Makerbase, 2020)	44
Figure 3.26: 4010 Axial Cooling Fan 3D Model.	44
Figure 3.27: RepRap 2004 LCD 3D Model.	45
Figure 3.28: S-360-24 Switch Mode Power Supply 3D Model.	45

Figure 3.29: C14 Socket with Switch 3D Model.	46
Figure 3.30: Optical End Stop 3D Model.	46
Figure 3.31: Filament Sensor 3D Model.	46
Figure 3.32: Rough Model of Machine Structure.	47
Figure 3.33: X-carriage.	51
Figure 3.34: X-carriage Clamp.	51
Figure 3.35: Hot End Mount.	51
Figure 3.36: Hot End Clamp.	51
Figure 3.37: Y-carriage.	52
Figure 3.38: Y-carriage Clamp.	52
Figure 3.39: XY Stepper Mount (Left and Right).	52
Figure 3.40: Idler Mount (Left and Right).	52
Figure 3.41: Z Stepper Mount.	53
Figure 3.42: Z-linear Rod Bracket (Left and Right).	53
Figure 3.43: Belt Clamp.	53
Figure 3.44: Heated Bed Bracket.	53
Figure 3.45: Z-carriage.	54
Figure 3.46: Belt Tensioner.	54
Figure 3.47: Fan Duct.	54
Figure 3.48: 2004 LCD Panel Cover.	54
Figure 3.49: 2004 LCD Panel Support (Left and Right).	55
Figure 3.50: PSU Support.	55
Figure 3.51: PSU Holder.	55
Figure 3.52: PSU Holder Cap.	55
Figure 3.53: Spool Holder (Left Part and Right Part).	56

Figure 3.54: Spool Holder Bearing Retainer.	56
Figure 3.55: Optical End Stop Holder.	56
Figure 3.56: Optical End Stop Flag.	56
Figure 3.57: Filament Guide Arm.	57
Figure 3.58: Filament Guide Sheave (Half).	57
Figure 3.59: Filament Guide Bearing Retainer.	57
Figure 3.60: Filament Sensor Holder.	57
Figure 3.61: Motherboard Case.	57
Figure 3.62: Motherboard Case Cover.	58
Figure 3.63: Extruder Mount Spacer.	58
Figure 3.64: POM Nut Block Connector.	58
Figure 3.65: Front and Back Views of Proposed Assembly.	59
Figure 3.66: Left and Right Views of Proposed Assembly.	59
Figure 3.67: Top and Bottom Views of Proposed Assembly.	60
Figure 3.68: Isometric View of Proposed Assembly.	60
Figure 3.69: X Movement Evaluation.	61
Figure 3.70: Y Movement Evaluation.	61
Figure 3.71: Z Movement Evaluation.	62
Figure 3.72: Belt Length Evaluation.	62
Figure 3.73: MKS Robin E3 Pin Map. (Makerbase, 2020)	66
Figure 3.74: Circuit Connections of the Prototype.	67
Figure 3.75: Circuit Connection for Prototype.	68
Figure 4.1: CoreXY 3D Printer Prototype Developed.	71
Figure 4.2: The First Print by the Prototype Developed Prior to Calibration.	72

Figure 4.3: PID Auto Tune of Hot End at 200 Degree Celsius for Normal PLA Printing with Result Stored to RAM.	74
Figure 4.4: PID Auto Tune of Heated Bed at 60 Degree Celsius for Normal PLA Printing with Result Stored to RAM.	75
Figure 4.5: Save Data to EEPROM.	75
Figure 4.6: Single Walled Hollow Cube Printed with 100% Flow and High Precision Callipers.	77
Figure 4.7: Prints for String Test with Different Parameters Set on Every 5mm Layer Height.	78
Figure 4.8: Result for Optimum Retraction Setting Among the Combinations Made.	96 78
Figure 4.9: Optimal Parameters for Retraction Tuning of the Prototype.	78
Figure 4.10: Temperature Towers for PLA.	79
Figure 4.11: Outputs with K-factor from 0 to 2, Increment of 0.2.	82
Figure 4.12: Outputs with K-factor from 0 to 0.4, Increment of 0.05.	82
Figure 4.13: Results of Extrusion Limit Test with Extruder's Feed Rate from 120mm/min to 600mm/min and Hot End at 200 Degree Celsius.	85
Figure 4.14: Speed Versus Distance with Superimposed Acceleration Curve (1500mm/s^2). (Prusa Research a.s., 2021)	88
Figure 4.15: Improvement Made in Inserting Threaded Insert into Printed Part.	88
Figure 4.16: Modification Made on Serial Port Declaration from Datasheet for Successful Connection with Pronterface.	89
Figure 4.17: Prototype Developed with Enclosure and Glass Bed.	90
Figure 4.18: Cable Management of Prototype.	90
Figure 5.1: Detail Views of Non-contact Optical End Stop and Universal Spool Holder Designed.	93

LIST OF SYMBOLS / ABBREVIATIONS

V_{avg}	average voltage, V
V_{peak}	peak voltage, V
ΔX	motion of print head in x direction, m
ΔY	motion of print head in y direction, m
A	acceleration, mm/s^2
FR	base feed rate, mm/s
J	junction deviation
3D	three dimensional
ABS	acrylonitrile butadiene styrene
AC	alternating current
AM	additive manufacturing
AWG	American wire gauge
CAD	computer aided design
CCW	counter clockwise
CNC	computer numerical control
CPU	central processing unit
CW	clockwise
DC	direct current
EEPROM	electrically erasable programmable read-only memory
FDM	fused deposition modelling
FFF	fused filament fabrication
HMI	human machine interface
IC	integrated circuit
LCD	liquid crystal display
MYR	Malaysian ringgit
N/A	not applicable
NTC	negative temperature coefficient
PC	polycarbonate
PETG	polyethylene terephthalate glycol-modified

PID	proportional, integral, derivative
PJP	plastic jet printing
PLA	polylactic acid
POM	polyoxymethylene
PSU	power supply unit
PTC	positive temperature coefficient
PTFE	polytetrafluoroethylene
PWM	pulse-width modulation
RAM	random access memory
ROM	read only memory
SCARA	selective compliance assembly robotic arm
SD	secure digital
SPI	serial peripheral interface
TF	trans-flash
TPU	thermoplastic polyurethane
UART	universal asynchronous receiver-transmitter
USB	universal serial bus

LIST OF APPENDICES

APPENDIX A: Graphs of Survey Result	106
APPENDIX B: Flowchart	109
APPENDIX C: Older Assembly 3D Model (Version 1)	111
APPENDIX D: Marlin Firmware Configuration.h	112
APPENDIX E: Marlin Firmware Configuration_adv.h	152
APPENDIX F: CAD Drawings for Enclosure	206

CHAPTER 1

INTRODUCTION

1.1 General Introduction

The latest manufacturing technology can be mainly classified as additive, continuous addition of material like 3D printing; subtractive, removal of material like CNC milling; forming, moulding of material like injection moulding and casting. (Nutma, 2019) Additive manufacturing or often referred as the three dimensional printing technology or digital fabrication technology is a fast-emerging technology in current twenty-first century. By 3D printing, the physical 3D object can be produced by the layered development framework, where successive addition of materials on each layer utilising the numerical control and fused together to form a solid model based on the CAD drawing. (Zhou, 2019)

From the definition of the 3D printing, the main components of the 3D printing technology can be outlined as below:

1. The digital controlled system to deposit the material layer by layer.
2. The tool and mechanism for laying the material.
3. The feedstock that used to form the desired object.
4. The digital model (CAD drawing) of the object to be fabricated.

Each and every component for the 3D printing technology is a great field to study and explore. With the effort from researchers, there are a lot of variation in each component which result in the varieties of 3D printing technologies nowadays. The 3D printing technology is classified into seven groups by ISO/ASTM which are:

- | | |
|-------------------------------|-----------------------------|
| 1. binder jetting | 5. powder bed fusion |
| 2. directed energy deposition | 6. sheet lamination |
| 3. material extrusion | 7. vat photo polymerization |
| 4. material jetting | |

(International Organization for Standardization/ American Society for Testing and Materials, 2015)

The most common budget 3D printer in the market recently is the fused deposition modelling (FDM) type or so called fused filament fabrication (FFF) type due the expiry of patents and it is categorised under material extrusion where the filament is selectively dispensed via a nozzle to form the layers for 3D model. (Chen, et al., 2020) This type of printer able to work with many different kinds of thermoplastic feedstock included: PLA, ABS, PETG, TPU, PC and others by tuning the appropriate printing parameters. (Shen Zhen Esun Industrial Co.,Ltd, 2007)

There are vast variety of movement control mechanism for FDM 3D printer such as Cartesian, delta, polar and SCARA with its pros and cons respectively. (Alex, 2017) The coreXY type 3D printer using the Cartesian coordinate system (x, y, z) with the nozzle movement in x and y axes and printing platform moving in z axis. (Grames, 2019)



Figure 1.1: FDM 3D Printers with Different Movement Control Mechanism.
(Alex, 2017)

Nowadays, 3D printing is not only use for rapid prototyping but the technology can be applied in many fields and sectors. For instances, 3D printing is applied in healthcare for the learning and teaching purposes, for treatment organizing, as diagnostic help and for the therapeutic process. (Mardis, 2018)

1.2 Importance of the Study

The fifth edition of “The State of 3D Printing” with surveys answered by more than 1300 respondents around the world, has demonstrated the latest information regarding 3D printing. According Graph A-1 in Appendix A, most people believe that 3D printing technology will have a significant role in the manufacturing, business and individual life in the future. (Moreau, 2019) This indicate that the 3D printing technologies potential to be popularised not only for industrial usage but in-house application such as to fabricate the customised part instantly and economically, for learning purpose and personal hobby.

The study and research over the improvements of the current additive manufacturing technology is on high demand and ongoing to support the wide application in many industries as the step towards the fourth industrial revolution. (Özüdoğru, Ergün and Ammari, 2018) This is because the global industrial revolution has the dependency on these minor technological revolutions of the additive manufacturing which is one of the key component in the industry 4.0. (Stăncioiu, 2017)

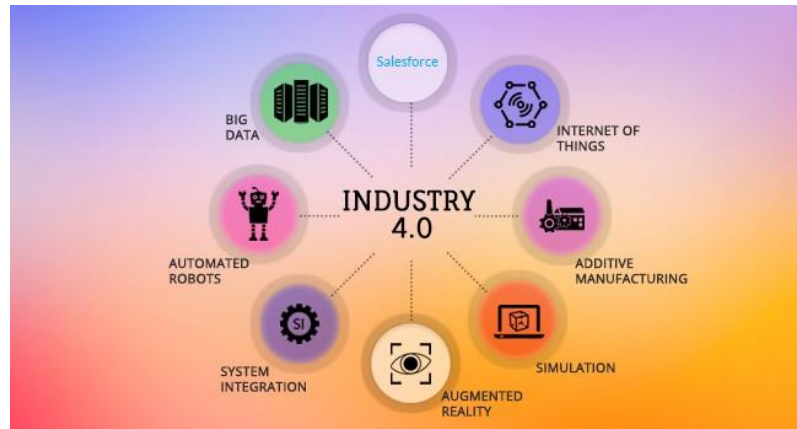


Figure 1.2: Industry 4.0 Key Components Mapping. (PlumLogix, 2019)

The wide range of industries that can applied 3D printing technology included but not limited to:

1. aerospace,
2. automotive,
3. food,
4. healthcare and medical,
5. architecture and construction,
6. fabric and fashion,
7. electrical and electronics.

(Shahrubudin, Lee and Ramlan, 2019)

Table 1.1: Benefits of 3D Printing Technology in Different Industrial Fields.

Fields	Advantages of 3D printing
Aerospace	Able to regulate infill, density, mechanical properties, produce complex geometry and lighter parts. (Kalender, et al., 2020)
Automotive	Able to combine many small parts in a unified composite structure, quick standardization and catalyse design work. (Tofail, et al., 2018)
Food	Allows for food customization to meet individual preferences and needs. (Shahrubudin, Lee and Ramlan, 2019)
Healthcare and medical	For educational purposes, for treatment organizing, as diagnostic help and for the therapeutic means. (Mardis, 2018)
Architecture and construction	For construction of cheaper, low energy buildings, facilitate special design concrete like the self-cleaning concrete. (Tofail, et al., 2018)
Fabric and fashion	Maximize the creative and innovative possibilities for novel design, personalised styling and small scale manufacturing is feasible. (Shahrubudin, Lee and Ramlan, 2019)
Electrical and electronics	Provide cheaper and time saving way to mass fabricate electrode materials and speed up the development of green electronics device. (Shahrubudin, Lee and Ramlan, 2019)

On the other hand, the 3D printing technology has the potential to replace or simplify the traditional manufacturing method with the advantages in various aspects as summarised in Table 1.2.

Table 1.2: Comparison Between Additive Manufacturing Technology and Traditional Manufacturing Technology. (Sunil and Abdullah, 2015)

Additive Manufacturing	Traditional Manufacturing
Able to manufacture object with complex geometry and design is mainly limited by the designers' imagination.	More difficult the fabricate complex product, separation into various parts usually take place and the design is limited by the manufacturing ability.
Optimum material consumption.	Huge material consumption.
Cheaper and time efficient in prototyping.	Expensive and time consuming in prototyping.
Post fabrication processing can be eliminated or very little.	Post fabrication processing usually required.

1.3 Problem Statement

The FDM 3D printers can be classified into four categories based on its normal printing speed.

Table 1.3: Classification of FDM 3D Printer in Term of Speed. (Joel, 2020)

Categories	Printing speed
Slow	$\leq 80mm/s$
Mid-speed	$\leq 100mm/s$
Rapid	$\leq 150mm/s$
Very fast	$> 150mm/s$

In the current market, the budget small-scale FDM 3D printer offered for sales is under slow categories with the suggested slow printing speed to maintain the good print quality. As a representative of the 3D printer in this category, the Creality Ender 3 Pro which is a global bestseller priced at MYR

988.00 (quoted from Cytron Technologies Sdn. Bhd. in 24 July 2020) suggest a normal printing speed of 30-60mm/s. (Shen Zhen Crealiti 3D Technology Co., Ltd., 2019)

This lead to a longer printing times that not only reduced the efficiency of the printer but incurred a higher cost of operation primarily due to the long period heating of the heated bed and hot end until the completion of printing. This issue will be much more obvious during the printing of bulky model which will easily consume more than few days and there might be a need to rest the budget 3D printer after specific long period of continuous printing.

1.4 Aim and Objectives

The core aim of this project is to resolve the issue stated in the problem statement by the development of a coreXY FDM 3D printer within the budget of MYR 1 000.00 that capable to print at higher speed ($>80\text{mm/s}$) meanwhile maintaining a good print quality. Four objectives have been clearly identified as a guide towards the defined goal.

1. To select for the easily available, appropriate and affordable material, components and equipment to manufacture a small-scale coreXY FDM 3D printer.
2. To design and develop a feasible, compact and sturdy mechanical structure for the coreXY FDM 3D printer.
3. To develop the electronics system to fit the coreXY FDM 3D printer.
4. To test and develop the coreXY FDM 3D printer prototype for higher reliability and accuracy.

1.5 Scope and Limitation of the Study

In consideration of the popularity according to Graph A-2 and Graph A-3 in Appendix A, the coreXY 3D printer prototype to study and develop is selected to be the FDM type with the thermoplastic as feedstock for the easier sourcing of components and materials supply online. Thus, the scope to study is narrowed by the FDM technology chosen for the printer prototype.

For the ease of manufacturing, the customised parts for the prototype will be fabricated with 3D printing technology. The application of FDM 3D

printed products in development of a 3D printer is feasible because of the good precision, acceptable strength, affordable cost and highly flexible. (Muzammal, et al., 2019). Thus, the brief understanding over the mechanical properties of various types of feedstock and printing parameters is required. With the budget limit, the budget 3D printer used to fabricate the customised parts has a maximum hot end temperature around 250°C which eliminate the usage of thermoplastics like PC and nylon with high melting point and better mechanical properties. For the design of the parts to be 3D printed, it must adhere to several rules such as reducing overhang in order to yield the most appealing result. Therefore, the inadequate experience on the operation of 3D printer will be the limitation of the study and fabrication.

On the other hand, the scope for this project is considerably broad which covered the mechanical, electronics, programming, control system and others. With the fixed amount of time allocated, the penetration into each field will be limited.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The revolutionary process of AM has taken place since the inception of 3D printing that described as the technologies capable of creating 3D components through the deposition of material at successive layers. (Kalender, et al., 2020) The 3D printing technology is believe to be able to incite the renaissance of manufacturing field as everyone now can easily run a manufacturing facility at home thanks to the AM technology advancement and the related open source projects available. (Horvath, 2014)

There are mainly two divisions of 3D printing industry based on the area of focus in the mid of 1990s. (Wohlers and Gornet, 2014)

1. High end, which aimed for the application such as medical and aerospace fields that required high precision and very complex designs.
2. User-friendly, which is more cost effective targeted to develop the concept and create functional prototypes.

There are now a wide variety of user-friendly desktop 3D printer in the market especially the FDM type.

The main components in developing a user-friendly 3D printer are roughly identified to foster a more focus study and better development progress as shown in Figure 2.1.

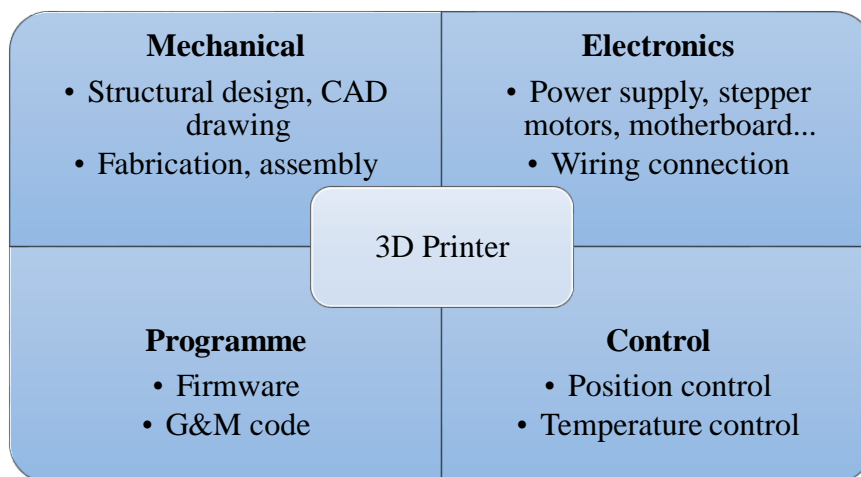


Figure 2.1: Rough Map of Main Components for 3D Printer Development.

2.2 FDM Technology

FDM 3D printer also known as FFF or PJP is the pioneer in 3D printing technology since 1990s by Stratasys and the typical FDM printer consisted of the printing platform, print head and the feedstock in the filament form. (DesignTech Systems, n.d.)

The popularity of FDM 3D printer was enhanced due to the emerging of open source projects such as Fab@Home and RepRap Movement which make the designs of 3D printer open for everyone for free after the expiry of patent in FDM technology by Stratasys around 2005. (Su and Al'Aref, 2018) One of the most famous example of the open source FDM 3D printer model is the Prusa Mendel, simplified version of 3D printer designed by Joseph Prusa who are the founder of the Prusa Research in the latter. (Prusa Research a.s., 2020b)

Under the ideology of getting open source, it embarks the journey of continuously growing of the FDM 3D printer in every aspects including the source codes, 3D model, blueprints, motherboard design and others thanks to the effort of experts and enthusiasts from varies sectors around the world. From here, the innovation and creativity of people globally in the 3D printing field support the endless improvements and development to yield the better and better versions of 3D printer.

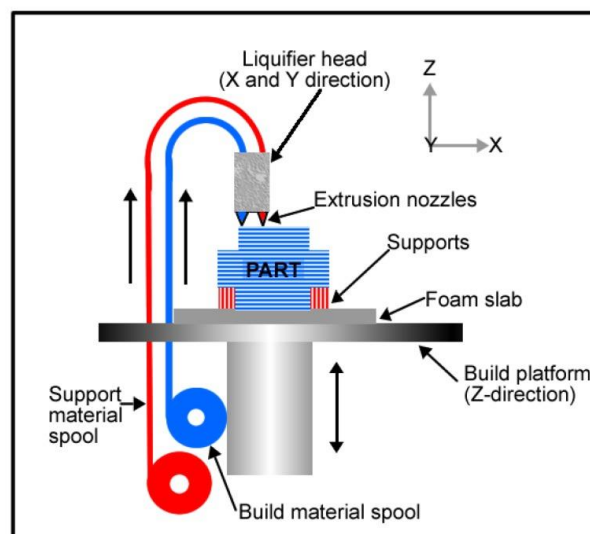


Figure 2.2: FDM Illustration. (Sidambe, 2014)

The typical FDM 3D printer will adhere to a coordinate system, for instances, the Cartesian coordinate system, it will have the precise motion carry

out by the stepper motors in the X, Y and Z directions. (Sidambe, 2014) The thermoplastic filament which is production-graded will be heated to its glass transition temperature by the hot end and force extrude through the nozzle by the extruder motion from the stepper motor. (Vincent, 2018) There are usually sensors in the printer to help detect the temperature for the hot end and heated bed ; home position of each axis for the moving parts as the feedback for the control system. The relative motion between the print head (nozzle) and printing platform (print bed) allow the deposition of the raw material layer-by-layer from the bottom up along the computed path as directed by the microcontroller according to the coding. (Stratasys Ltd., 2020) These layers fused together and harden upon cooling to form a 3D model.

FDM is suggested because of the benefits outlined by Stratasys Ltd. that are office-friendly, simple operation, clean, the supported feedstock with the production-grade are environmentally and mechanically stable and the ability to produce complex geometries and cavities. There are a lot of thermoplastics feedstock supported including flexible (TPU) and non-flexible (PETG, PC, PLA, ABS) with respective unique mechanical and chemical properties make it suitable for respective application. (Shen Zhen Esun Industrial Co.,Ltd, 2007)

The support structure for the overhang printing is critically important in FDM to ensure good print quality else the lengthy overhang part will fall down due to the gravitational pull as displayed in Figure 2.3. (Higgs, 2018) Some FDM printers will use another material to print the support structure which will then be able to wash away by specific solvent as in Figure 2.2. Besides, the print speed is relatively slow and the layering in FDM can possibly cause problems with shrinking and warping. (3DSourced, 2020)

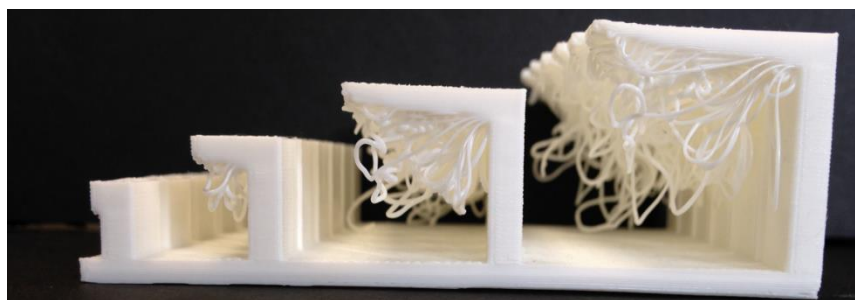


Figure 2.3: FDM Printing Without Appropriate Support Structure. (Higgs, 2018)

As a conclusion, FDM is very appeal to the public as observed from Graph A-6 where the machine capability, material supply and cost are the most important factors for the users.

2.2.1 FDM 3D Printing Optimization

The improvements on this technology has never stop with the ambition to get the ideal print in the more efficient manner. Every parameter in the setting will affect the print process, quality and behaviour. According to the research done by Chamil Abeykoon, Pimpisut Sri-Amphorn and Anura Fernando in 2020, the tensile properties of the FDM 3D printed part was analysed with different infill density, infill speed, infill pattern, printing temperature and printing material.

From Figure 2.4, it can be concluded that the Young's modulus (strength) of the 3D printed part will increase with the infill density. However, the infill density increase in the expense of longer printing time. The linear infill pattern result in the highest tensile strength among the tested patterns such as hexagonal and diamond because of the smaller gaps between individual layers. (Abeykoon, Sri-Amphorn and Fernando, 2020) Therefore, for normal usage and efficient fabrication time, 30-40% infill can be selected with linear infill pattern.

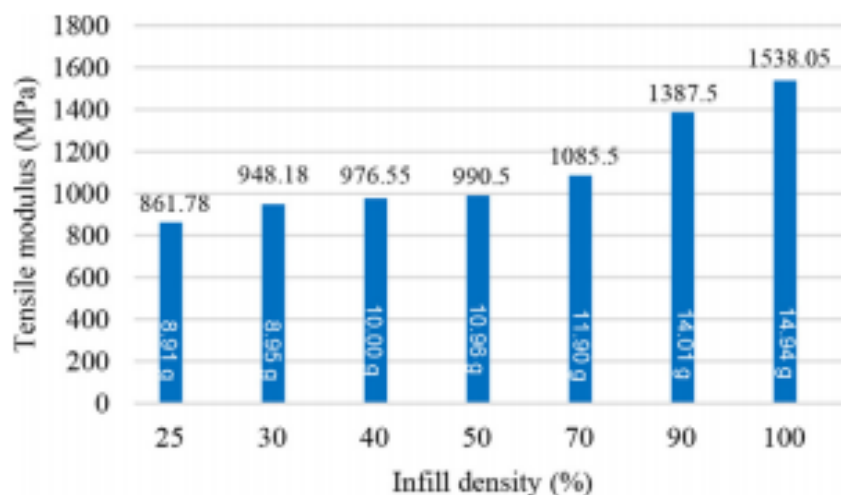


Figure 2.4: Relationship Between Tensile Modulus and Infill Density for PLA.
(Abeykoon, Sri-Amphorn and Fernando, 2020)

On the other hand, layering in FDM will possibly causing porosity of the printed product and affecting the print quality. This microscopic structural

defects can be solved by applying a thicker print wall, and applying higher value of extrusion multiplier (k-value that controlling extrusion flow rate of the material). (Gordeev, Galushko and Ananikov, 2018) As a conclusion, there are no fixed parameter setting for the ideal print, the optimization process is a continuous process and affected by many factors.

2.3 Core XY Structure

CoreXY is the mechanical arrangement that is suitable and feasible to apply in the development of a 3D printer. (Soon, et al., 2020) CoreXY structure is under Cartesian arrangement which allow the print head to move easily in a straight line along the axes which perpendicular to each other, for instances the X and Y axes in Cartesian coordinate system. (RepRap, 2020b) For the application in 3D printing, CoreXY as it name implied is only the motion control in two axes (X,Y) and is usually applied for the print head while the Z axis is accomplished by the vertical print bed motion. (Avdeev, Shvets and Torubarov, 2020)

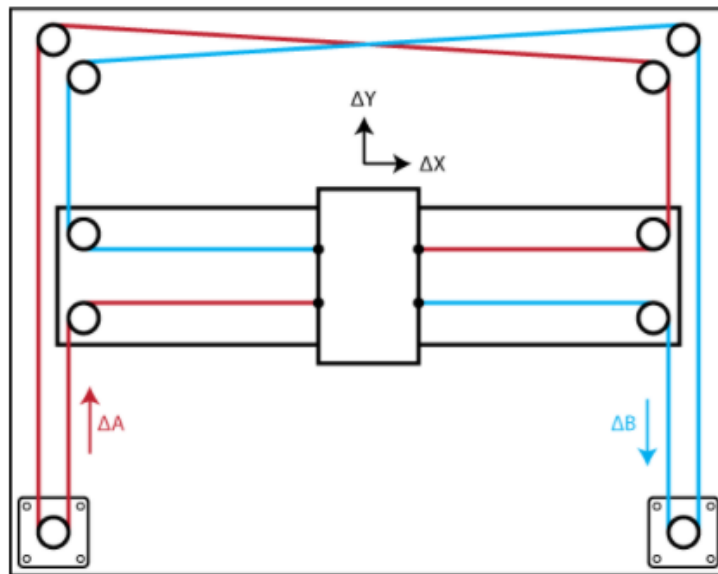


Figure 2.5: Mechanism of CoreXY Structure. (Moyer, 2012)

$$\Delta X = \frac{\Delta A + \Delta B}{2} \quad (2.1)$$

$$\Delta Y = \frac{\Delta A - \Delta B}{2} \quad (2.2)$$

The mechanism of coreXY is different with those direct motion control by a stepper motor for each axis that applied in most of the budget 3D printers, because the print head will move diagonally with the single motor rotation in coreXY structure. (Moyer, 2012) Both the stepper motors will operate for the motion in X and Y directions as shown in Table 2.1.

Table 2.1: Motion Study in CoreXY Structure.

Motion	Requirements	Result
+ x direction	$\Delta Y = 0$ From (2.2), $\Delta A = \Delta B$ Motors rotate in same direction, same magnitude.	From (2.1), Magnitude, $\Delta X = \Delta B$ Left motor rotate CCW, Right motor rotate CCW.
+ y direction	$\Delta X = 0$ From (2.1), $\Delta B = -\Delta A$ Motors rotate in different direction, same magnitude.	From (2.2), Magnitude, $\Delta Y = \Delta A$ Left motor rotate CCW, Right motor rotate CW.

The implementation of coreXY belts system have two variations as shown in Figure 2.6. The first version has both belts located at the same plane with a crossing point whereas the second version have each belt located at different height with the pulleys stacked together.

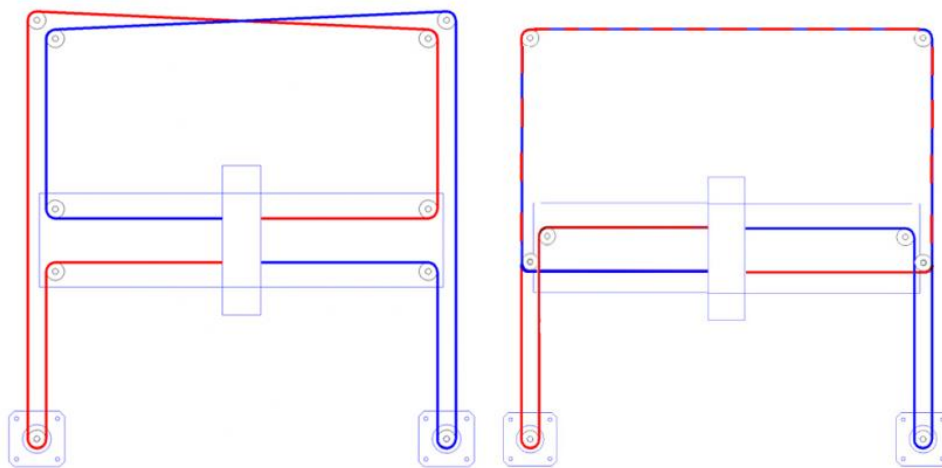


Figure 2.6: Two Versions of CoreXY Belt System. (Hoge, n.d.)

CoreXY structure has many advantages for the application in FDM 3D printing due to its unique construction and motion mechanism as summarised in Table 2.2.

Table 2.2: Summary of the Benefits of CoreXY Structure in 3D Printing.
(Grames, 2019)

Advantages	Description
Enhance printing speed	No significant loads (stepper motors, heated bed) in continuous motion during printing process and causing less vibration thus able to boost the printing speed and yet maintaining acceptable quality.
Reduce printer size	Smaller base can be achieved for the print head to access the whole build area as compared to common i3-style. Lower height than the delta type 3D printer.

The accuracy of the coreXY structure is very high. For instances, the position error for the x and y axes in imitated-handwriting machine utilised the coreXY structure developed by Guan-Han Zhu and Jin-Shyan Lee in 2019 are very low as shown in Figure 2.7 and Figure 2.8 respectively where position error plot line is nearly a straight line at 0 level. (Zhu and Lee, 2019)

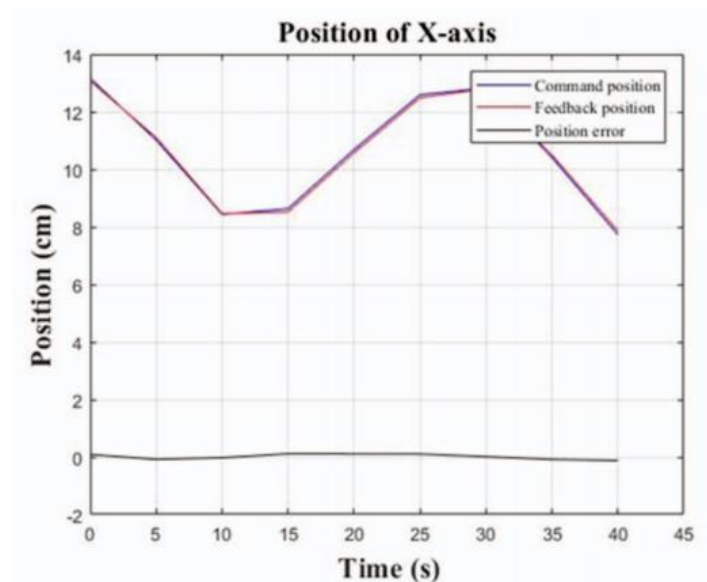


Figure 2.7: Waveform of X-axis Displacement of the Machine. (Zhu and Lee, 2019)

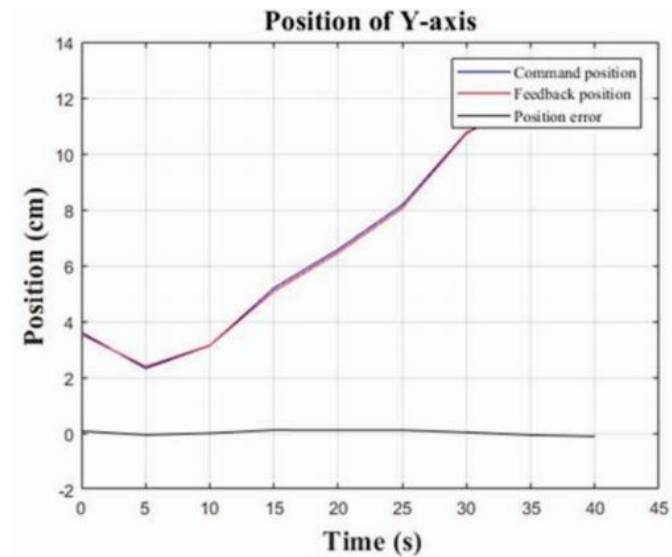


Figure 2.8: Waveform of Y-axis Displacement of the Machine. (Zhu and Lee, 2019)

In contrast, the coreXY structure utilise the lengthy belts to actuate the print head which make the effect of belt tension and alignment being magnified. (Grames, 2019) Thus, the optimised assembly of the coreXY structure is vital to give the desired quality of print.

2.4 Electronics Components

Electronic components are very important part of a 3D printer that cannot be neglected. The understanding on the electronics components and its function is critical to select the appropriate part in development of a 3D printer.

The main electronics components for a 3D printer can be identified from the functionality diagram required by a 3D printer in Figure 2.9. The required components should be able to provide the specified functions stated in order to create a successful 3D model. As observed from Figure 2.9, there are basically five main functions to be served by the electronics components required to create a 3D solid with FDM technology: movement, material heating, material solidification, material obtaining and control.

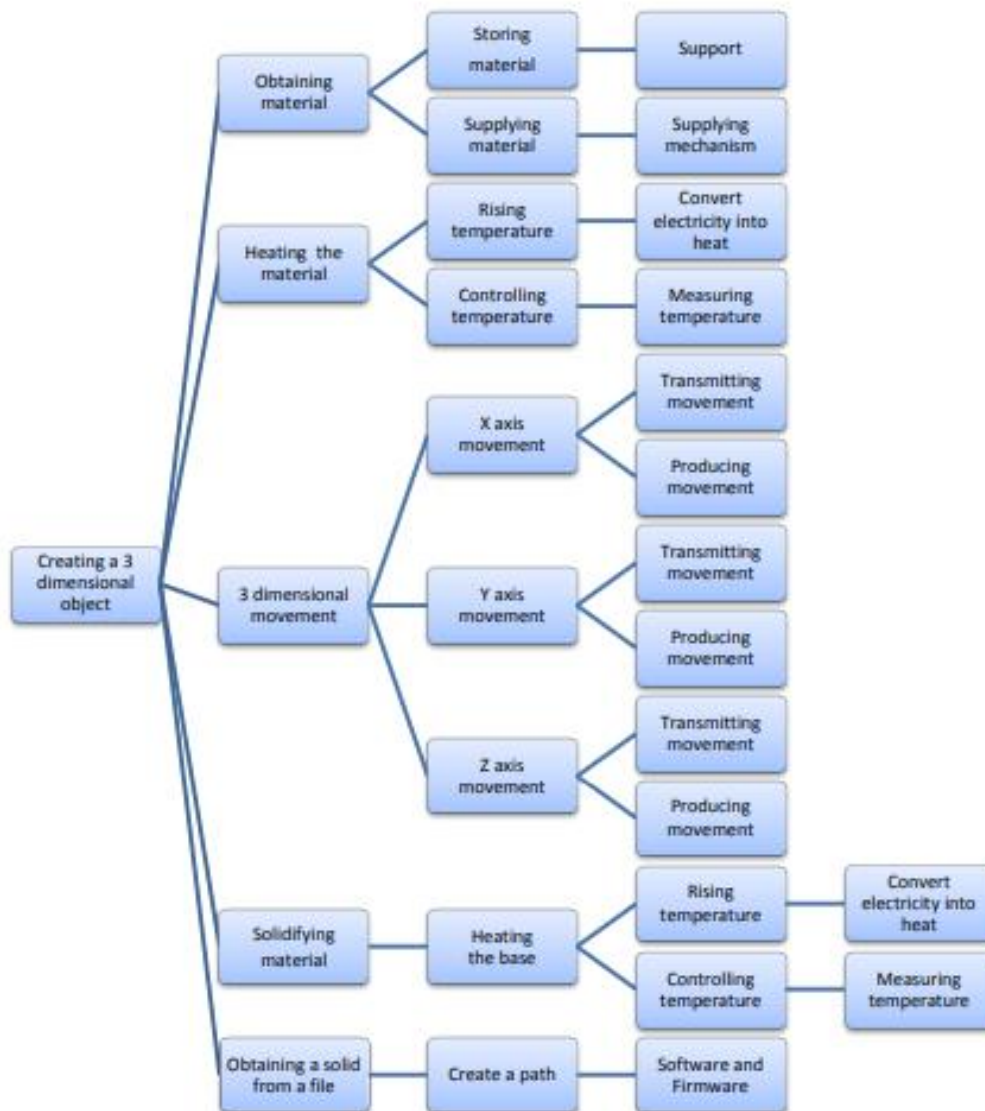


Figure 2.9: Functionality Diagram of FDM 3D Printer. (*3D Printer electronics design*, 2014)

2.4.1 Stepper Motor

Stepper motor, digital electric DC motor widely used in 3D printer to provide the precise motion for relative movement between head and print platform in 3D and material extrusion. Stepper motor able to divide the rotation into small angle instead of direct spinning like normal DC motor when activated.

This inherently digital characteristic makes stepper motor suitable for open-loop control in the low cost system such as budget 3D printer with the assumption that no external interruption during the operation. (Considine and Considine, 1986) This is also the main demarcation with the servomotor which

is operated in closed loop system with the feedback by the encoder and thus is more costly as shown in Figure 2.10.

The stepper motor will be driven by a driver as shown in Figure 2.10, which come in many types and versions in a form of IC chip such as A4988, DRV8825, TMC2103, TB6600, L298 and many more which allow for different driving options with respective specification and advantages. (RepRap, 2018c)

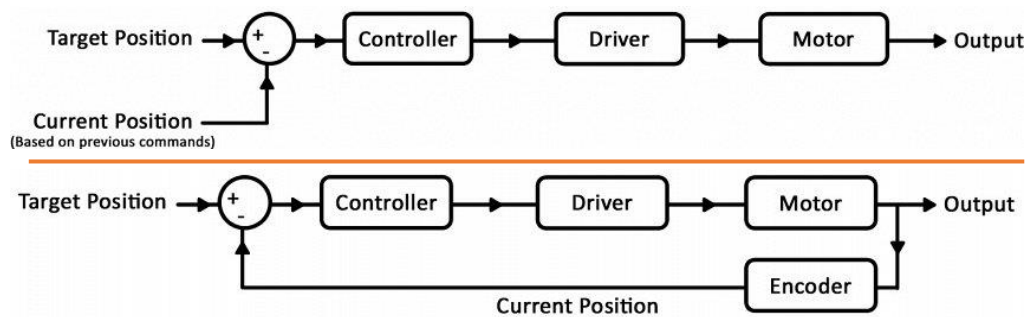


Figure 2.10: Open-loop Position Control [above] and Closed-loop Position Control [below]. (Mike, 2014)

The stepper motor can be classified by the frame size as according to the National Electrical Manufacturers Association (NEMA). The common stepper motor applied in small scale 3D printer, for instance the RepRap Mendel, is the bipolar NEMA 17 stepper motor with the torque around 13.7Ncm. (RepRap, 2018b)

2.4.2 Heated Bed

Heated bed is a flat build platform for the FDM 3D printer that used to solve the warping problem and improve print quality. For most of the printing materials, the edges of the printed goods tend to be lifted due to the high temperature gradient between the printing platform and extruding filament and resulting a warping phenomenon which affecting the print quality as shown in Figure 2.11. (Alsoufi and El-Sayed, 2017) Thus, the heated bed is used to reduce the excessive material shrinkage due to low temperature of printing platform during printing process.

The heated bed is to supply the heat and it will usually couple with the surface material or bed material such as Kapton tape, blue tape, PEI (Polyetherimide) sheet or others to cover on it for better adhesion and removal

of prints. (RepRap, 2016) For the precise temperature control of the heated bed, a closed-loop control with the thermistor temperature feedback is commonly used. For instances, the heated bed temperature around 100°C is suggested for printing ABS successfully and thus the temperature control is significant. (3D Printer electronics design, 2014)

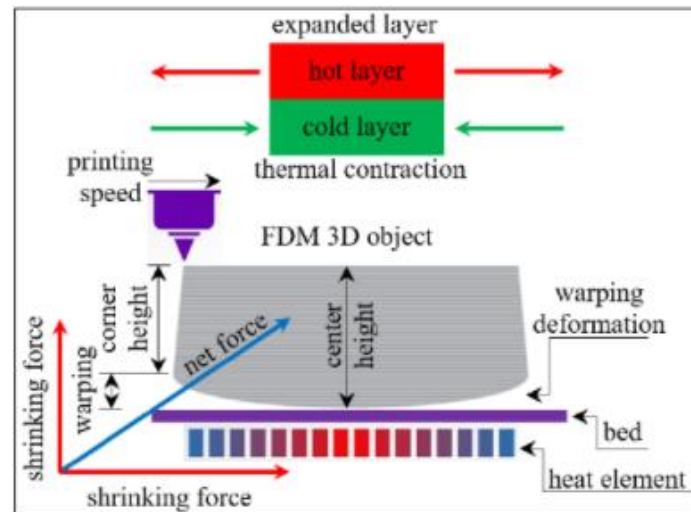


Figure 2.11: FDM 3D Printing Warping Deformation. (Alsoufi and El-Sayed, 2017)

2.4.3 Hot End

Hot end is the vital part in FDM 3D printing, it is to realise the FDM principle by heat up the filament to melt it for the extrusion process. (Beaudoin, Boulanger and DiPersio, 2017) A typical hot end consisted of heatsink, heat break, heater block, nozzle, heater cartridge and thermistor as demonstrated in Figure 2.12.

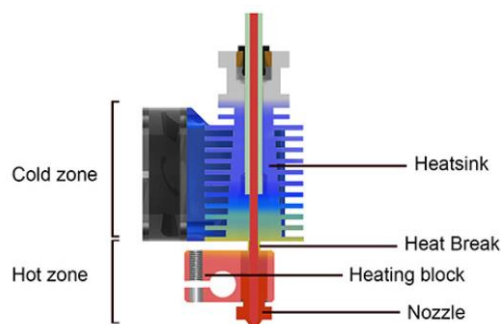


Figure 2.12: Illustration of Typical Hot End Assembly. (Filament2Print, 2020)

The heatsink and heat break is to radiate the heat and reduce heat transfer from the hot zone to cold zone respectively. The cooling fan will usually be used for forced air flow to remove heat more efficiently. This is important to prevent filament from soften too early in the cold zone and cause clog in the hot end, that is the condition known as heat creep and will result in fail print. (Prusa Research a.s., 2020a) The heater block is heat up by the heater cartridge to melt the filament so it is able to be extruded via the nozzle. The thermistor will detect the temperature of the heater block as the feedback for closed-loop control.

The hot end temperature control is the most important aspect in FDM 3D printing, because different material requires different temperature setting. Besides, the hot end temperature need to be increase with the increase in feed rate due to the shorter time for heat transfer from the heater block to the extruding filament. For the high speed printing, the feed rate will be higher to allow more material to be deposited at the same time. The variation in extrudate temperature at different feed rate can surge up to 15 °C, for instance, extrudate temperature drop from the liquefier temperature with increasing feed rate as shown in Figure 2.13. (Serdeczny, et al., 2020)

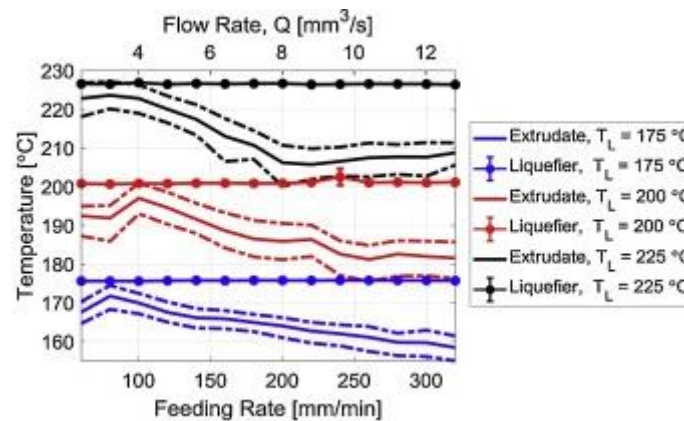


Figure 2.13: Measurement of Extrudate Temperature at Different Feeding Rate for Different Materials. (Serdeczny, et al., 2020)

2.4.4 Extruder

Extruder is used to drive and regulate the filament into the hot end for extrusion. The extruder stepper motor will provide the energy to the filament drive gear which force the filament through the hot end. The filament will be supported against the drive gear by an idler, or the another drive gear in the case

of dual gear drive extruder to provide sufficient pressure for perfect driving. (Beaudoin, Boulanger and DiPersio, 2017)

The extruder can be grouped into three main categories which are direct drive, Bowden and remote motor as illustrated in Figure 2.14.

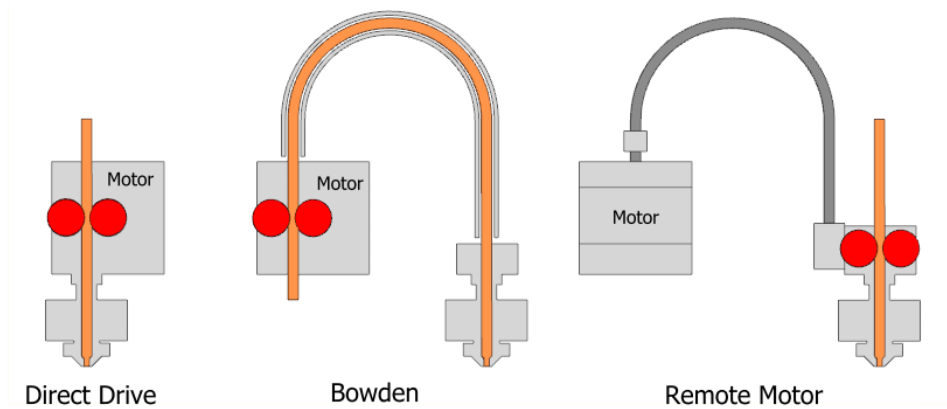


Figure 2.14: Different Types of Extruder for FDM 3D Printing. (Boichut, 2019)

Each types of extruder have their unique advantages and disadvantages as summarised in Table 2.3 due to their structural different. The remote motor extruder is the hybrid that have the advantages from both the direct drive and Bowden. The power transmission from the separately attached motor to the filament drive gear just above the hot end is achieved by a flexible shaft in remote motor extruder model. (Boichut, 2019)

Table 2.3: Comparison Between Direct Drive and Bowden Extruder Types. (Landry, 2016)

	Direct Drive	Bowden
Responsiveness	Higher, shorter distance between driving mechanism and hot end.	Lower, long PTFE tubing path to reach hot end from driving mechanism.
High speed printing	Poorer, heavier moving load due to attached motor and inertia is higher.	Better, lighter moving load due to separated motor and inertia is lower.

2.4.5 Motherboard

A microcontroller has four main elements: CPU to perform all the computation, memory for data storage, I/O interface and the interconnecting system buses. Motherboard or so called a main control board is a microcontroller that acts as the brain of a 3D printer that control and process the information by allowing the communication between the electronics components such as the CPU and memory. (Harris, 2020) The motherboard will basically handle all the logic control in 3D printer, such as movement control, temperature regulation and G-code parsing. (Mika, 2019)

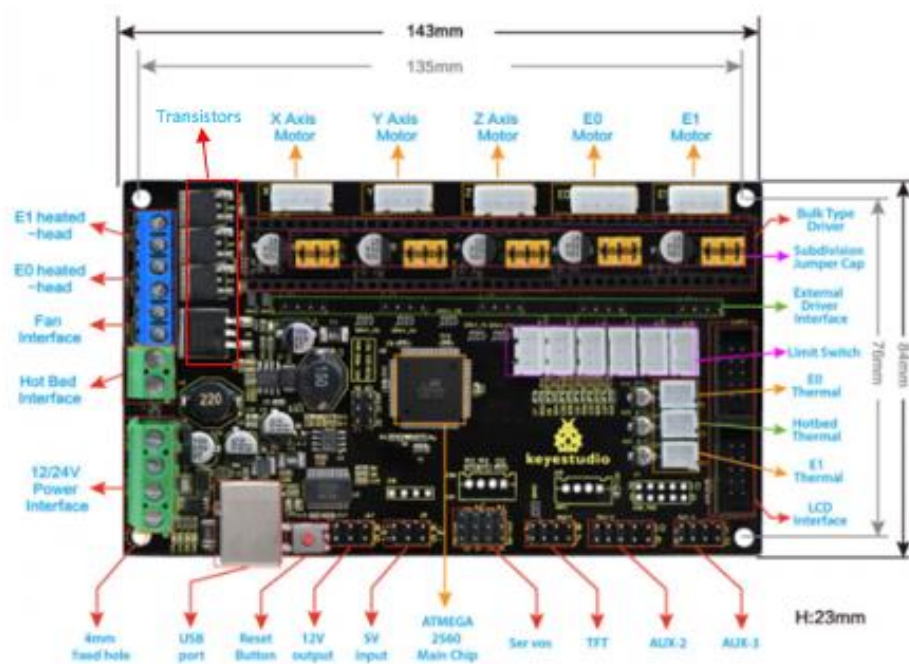


Figure 2.15: MKS Gen V1.4 3D Printer Motherboard Layout. (Keyestudio, 2019)

A typical 3D printer motherboard has all the screw terminals and ports to connect the inputs such as power supply, sensors; outputs such as motors, heater cartridges and user interface such as USB port and control panel as shown in Figure 2.15. There are many aspects to consider in the motherboard selection, which included but not limited to the number of I/O devices supported, the type of firmware supported, the type of stepper driver supported, the type of user interface supported, CPU type and speed. (RepRap, 2020a)

The main chip of the 3D printer motherboard in the market can be divided into two main categories based on its architecture, the 8-bit and 32-bit with the 16-bit leaned toward to 32-bit categories as from the performance benefits. (Thornton, 2016) The bit number means the data bus width. The higher bit number will allow more data transfer per clock cycle thus will speed up the computation time for complicated tasks. Besides, the processing capability of 32-bit is higher where it has more instructions that can be finished in single cycle as compare to an 8-bit board.

Other than that, the power transistors are also the crucial components on the motherboard to control the magnitude of voltage output to the devices like cooling fan and heaters. The magnitude of voltage is corresponding to the output magnitude like the fan speed, and heating power. Power transistor can operate in switching mode where it only has either on and off states other than the traditional linear way. The advantages of switch mode are lower power loss, more compatible with the digital controller and easy to implement. (Patel, et al., 2009) The variation of magnitude can be achieved by the PWM as demonstrated in Figure 2.16.

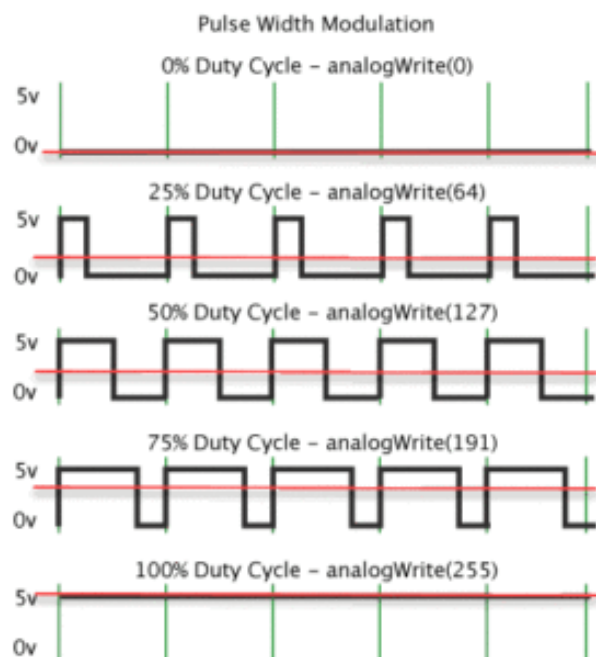


Figure 2.16: Average Output Voltage from PWM with Different Duty Cycle.

(Heath, 2017)

$$V_{avg} = V_{peak} \times Duty\ Cycle \quad (2.3)$$

As shown in equation 2.3, the output voltage can be computed and it is depending on the duty cycle if the input voltage is fixed.

Last but not least, the stepper motor drivers either build in or in the form of carrier board are the important components in a 3D printer motherboard as well. The driver can differ in the minimum and maximum operating voltage, maximum continuous current per phase, micro-stepping and other special features. (Pololu Corporation, 2020) The common control of the stepper motor with the popular stepper drivers such as A4988 and DRV8825 are the step and direction mode where only two pins required from the microcontroller to control the stepper motor. (Nedelkovski, 2019)

There are more advance and variations of motion control and driver chip communication available for the driver chips offered by TRINAMIC Motion Control such as TMC 2208, TMC 2209 and TMC 2130. Other than micro-stepping, TMC provide the unique motor control technologies such as StealthChop™, SpreadCycle™, StallGuard™ and CoolStep™.

As the operating principle of a stepper motor, the rotation of the magnetic rotor is in the step mode with the attraction from the electromagnetic field from the stator coils. Micro-stepping can smoothen the motor operation, increase torque and accuracy by introducing additional current states other than full steps specified by the motor structure as shown in Figure 2.17. With micro-stepping, the noise, step loss and vibration during the stepper operation can be reduced as well. The TMC drivers can provide up to 256 times micro-stepping to a hybrid stepper motor. In other words, the stepper motor with 200 steps per revolution can now achieve 51 200 steps per revolution by 256 times micro-stepping.

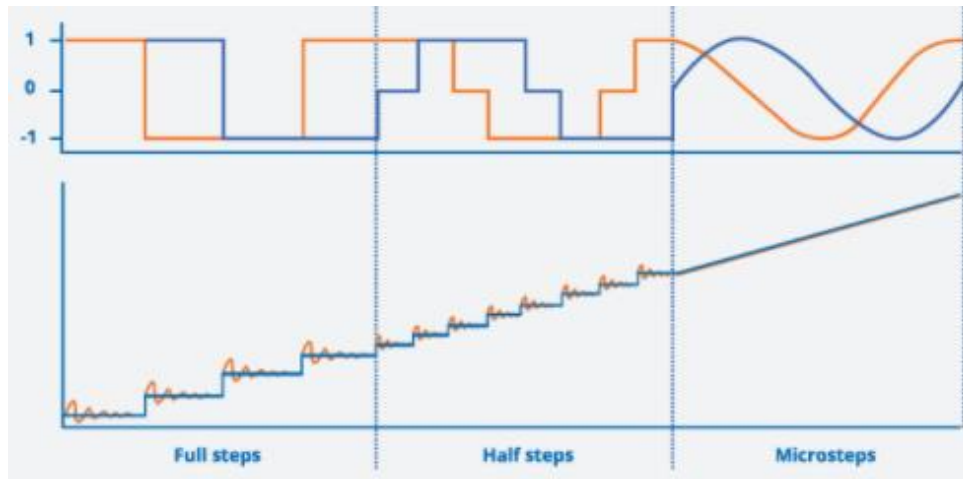


Figure 2.17: Micro-stepping. (TRINAMIC Motion Control GmbH & Co., 2020c)

StealthChop™ is the voltage-regulated chopper by TMC that can silent the stepper motor at standstill till the moderate speed operation. It can minimize the current ripple from the modulation of current according to the PWM duty cycle as the PWM frequency is same as shown in Figure 2.18. Besides, the reduction in current ripple can yield lesser power loss due to the reduction of Eddy current in the stator. It has successfully achieved a low noise level of 10 dB under classical control and thus is suitable for applications that required quiet motion.

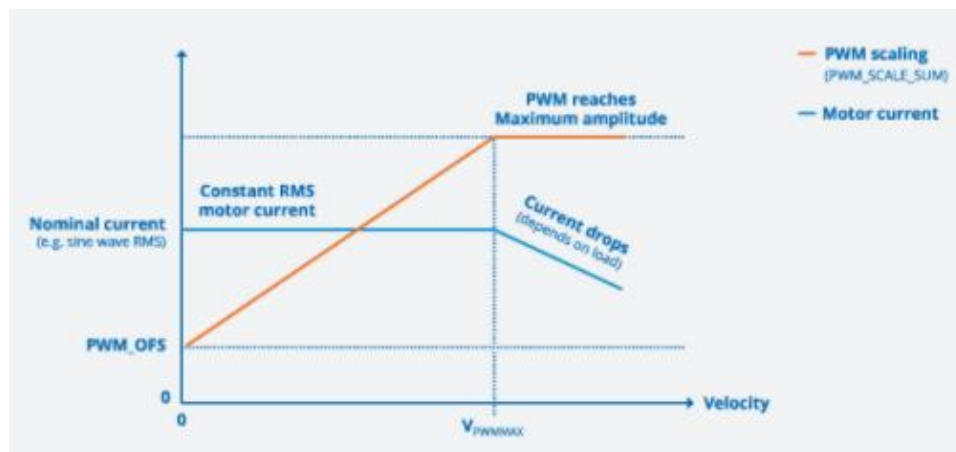


Figure 2.18: StealthChop™. (TRINAMIC Motion Control GmbH & Co., 2020b)

SpreadCycle™ able to resolve the vibration problem of stepper motor by the hysteresis function added. The hysteresis function will slow down the current dropping rate so the matching to the desired current value is achieved by the average current as shown in Figure 2.19. This technology also minimizes the current and torque ripples. The automatically tuned hysteresis function will optimize the “fd” phase (fast decay) and keep effective for high speed operation.



Figure 2.19: SpreadCycle™. (TRINAMIC Motion Control GmbH & Co., 2020b)

StallGuard™ technology is the load measurement for stepper motor using the feedback of load angle from the back EMF. The sensitivity can be adjusted and there are up to 1 024 different load levels can be detected. It is usually applied for the assurance of part operating within safety margin, sensorless homing and distance measurement. It can maintain step count reliability by halting the motor when reaching the maximum load value to avoid overload as shown in Figure 2.20.

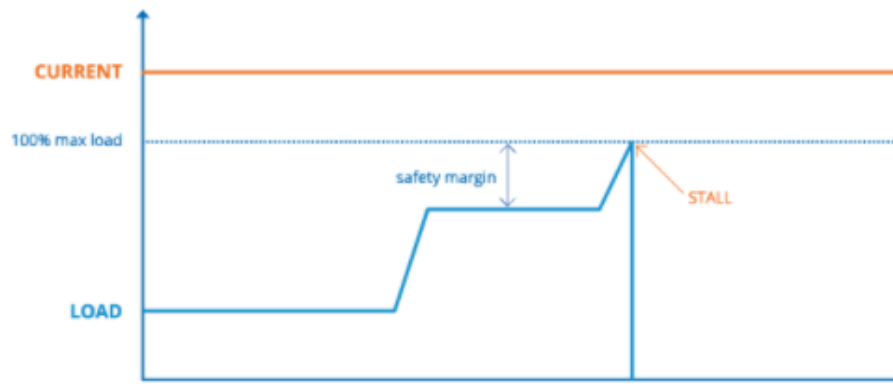


Figure 2.20: StallGuard™. (TRINAMIC Motion Control GmbH & Co., 2020d)

CoolStep™ can reduce the energy dissipation up to 90% and thus the heat generation from the motor operation. It can drive the motor at the minimum current required by using the dynamic current control based on the load condition as shown in Figure 2.21. Besides, it allows for temporary current boosts and thus a smaller motor can be used for certain applications due to smaller torque reserve.

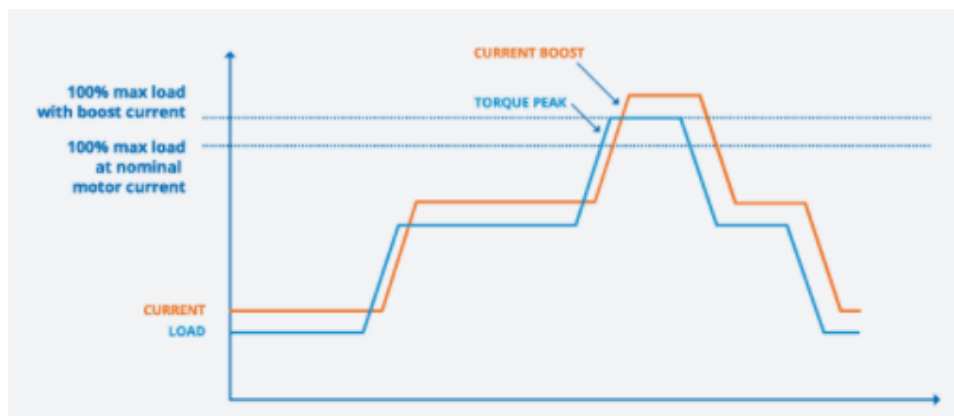


Figure 2.21: CoolStep™. (TRINAMIC Motion Control GmbH & Co., 2020d)

2.5 Human Machine Interface

HMI or known as a user interface is a crucial component that forms the connection between a person and a machine system. There are wide variety of HMI types such as buttons and switches based interface, graphical user interface and advanced interface which can exert control to a system via gesture, voice or even the electrical signal from human body.

The main features of a good HMI are able to supply sufficient information, easy to use and strict to minimize the possibility of human errors. (Roibu, et al., 2018) HMI served to monitor the I/O of the machine, display data of the machine to users, process tracking and other interactions between users and machine. (Inductive Automation, 2018)

Firstly, a 3D printer can be controlled through a USB connection between motherboard and computer with Pronterface. Pronterface is an open-source graphical user interface by Kliment Yanev specialised for 3D printer or CNC machines licensed under the GNU General Public License. (Yanev and Seguin, 2020) Users can send commands to the 3D printer via Pronterface to perform printing, tuning, and calibration. (Horne and Hausman, n.d.a) It gives access to most of the important settings, allows machine code (M and G codes) directly send by users, and run time tuning is achievable. Besides, Pronterface is supported by major operating systems like Windows, Linux and MacOS. (Morse, 2019)

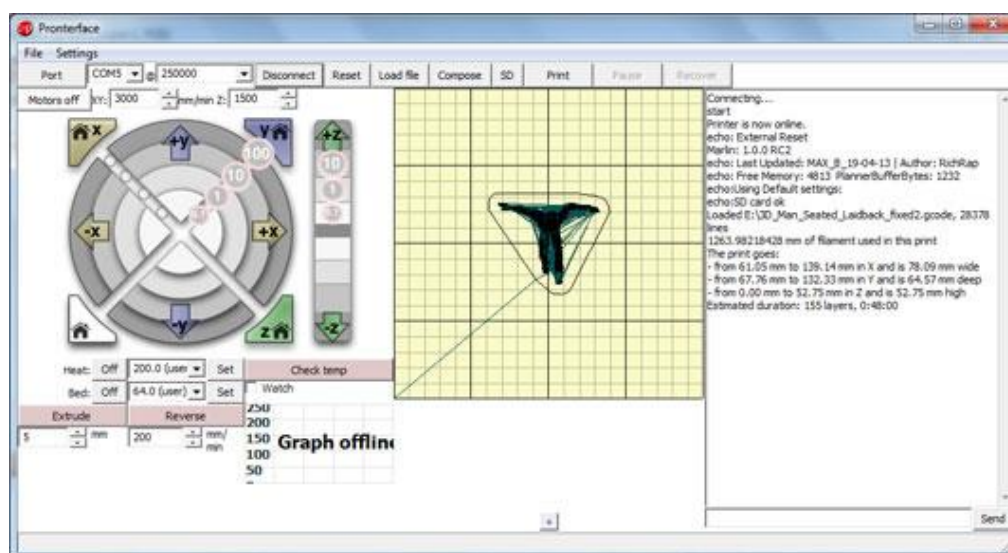


Figure 2.22: Pronterface with Model Loaded and Ready to Print. (Horne and Hausman, n.d.a)

Secondly, a 3D printer can perform off-line printing by reading the machine code stored in a memory storage device such as SD card depends on the type of slot available on the motherboard or control panel. The printing status will be displayed and settings can be fine-tuned through the button, rotary

encoder or other means on the controller without computer link to the machine as shown in Figure 2.23. (RepRap, 2019b)



Figure 2.23: Full Graphic Control Panel with SD Card Reader. (RepRap, 2019b)

The machine code is generated by the slicer software from the CAD file (usually in STL format). The user will load the 3D model and specify the printing parameters in the slicer, then the slicer will compute the path and output as machine code readable by the 3D printer and store in the memory storage devices for off-line printing. (*3D Printer electronics design*, 2014)

The settings appeared on the control panel depends on the firmware and hardware used. Some common settings available on a 3D printer control panel including homing, temperature setting, preheat and kinematic setting. For instances, the TMC driver allow bus interfaces such as SPI or UART besides the low-level communication. (TRINAMIC Motion Control GmbH & Co., 2020a) Then, more parameters of the stepper motor can be tuned from the control panel provided it is allowed in the firmware and using the specified driver.

Last but not least, there are web-based interface available for 3D printer such as OctoPrint and Duet. These web-based HMI provide the wireless connect, set up and control over a 3D printer provided the electronics used can support the application. (Horne and Hausman, n.d.b) For instances, Raspberry Pi computer is needed for OctoPrint. Most of these interface is free and open source like OctoPrint is open source under GNU Affero General Public License (AGPL) and every individual can tune it to meet their needs. (Häußge, 2020)

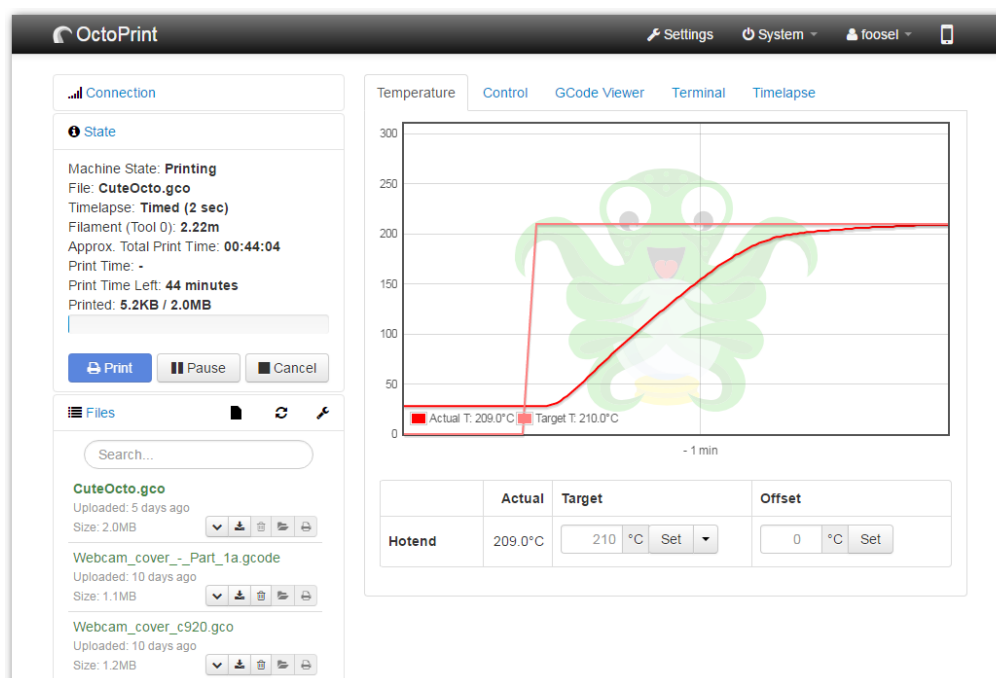


Figure 2.24: OctoPrint Web Interface. (Häußge, 2020)

2.6 Firmware

Firmware is the permanent programme that programmed and stored as ROM in the microprocessor of a machine such as 3D printer. (Lyubomirov, Nedeva and Bundevea, 2015) Firmware will determine the hardware respond from the software input commands. For instances, a 3D printer firmware will interpret the commands into specific electrical signals that is readable by the 3D printer components such as stepper motors, heaters, cooling fans, and others to perform the desired outputs. (Jones, 2019) The optimum firmware settings will be the root in development of a functional and reliable 3D printer.

There are a lot of open-source firmwares for 3D printer nowadays which have heavily simplified the development process for a 3D printer. Table 2.4 tabulated few popular 3D printer's firmware and some features out of it.

Table 2.4: 3D Printer's Firmware List. (RepRap, 2020d)

Firmware	Features
Marlin	<ul style="list-style-type: none"> Compatible with coreXY and delta structure other than normal Cartesian. Compatible with different platforms such as LPC, AVR, STM32 and ESP32.

Smoothie	<ul style="list-style-type: none"> • Ability to support laser and CNC milling. • Complete documentation and easy configuration.
RepRap	<ul style="list-style-type: none"> • Availability of web server usage. • All configuration is G-code based.
Repetier	<ul style="list-style-type: none"> • Availability of pressure control for nozzle. • Path planning system for speed improvement.
Klipper	<ul style="list-style-type: none"> • Compatible with coreXY and delta structure other than normal Cartesian. • Support multi-controllers on a machine.

The selection of firmware for a 3D printer is depends on few factors such as the compatibility of motherboard used, the features required and the ease of uses. (Jones, 2019) For instances, certain firmware is not supporting 32-bit motherboard and delta or coreXY structure.

2.6.1 G-code

G-code is the machine instruction developed for the CNC machines like 3D printer. G-code is known as preparatory which specified the tasks setting for a machine whereas M-code is the auxiliary. (Lyubomirov, Nedeva and Bundevea, 2015) In the field of 3D printing, firmware will translate the G-code generated that is originated from the CAD file as demonstrated in Figure 2.25 and then electronics components will work accordingly obeying the commands.

The interpretation of G-code for movement in 3D printer is mainly following the National Institute of Standards and Technology (NIST) RS274/NGC G-code standard, but there are still minor different between the interpretation of different firmware for some new or unique features offered by respective firmware. (RepRap, 2020c)

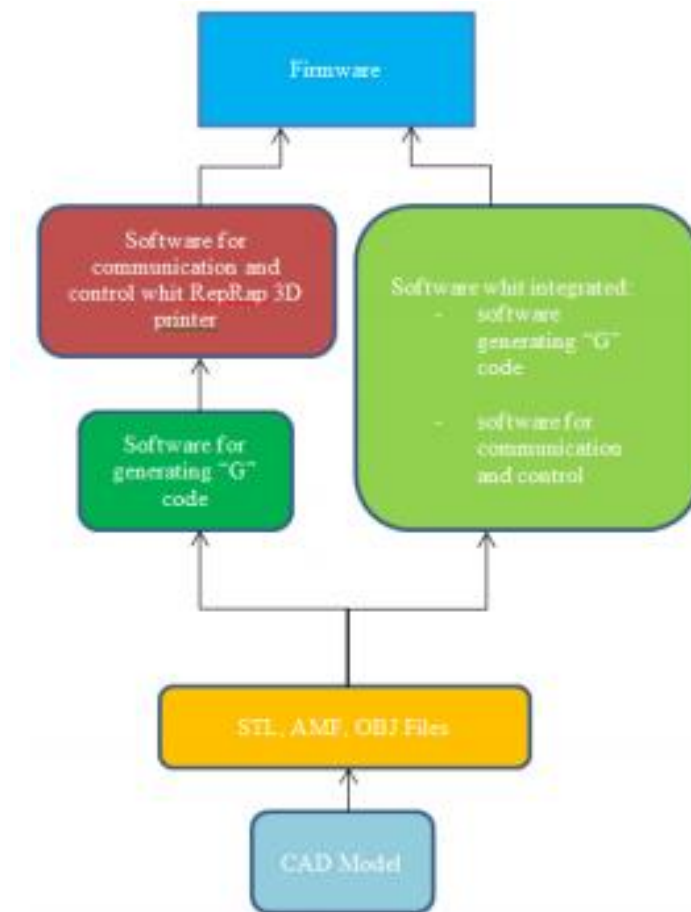


Figure 2.25: Software Management of 3D Printer. (Lyubomirov, Nedeva and Bundevea, 2015)

The easiest way to generate G-code for 3D printing is by using slicing software such as Cura, Prusa Slicer, Simplify3D and Slic3r which will slice the CAD model into layers and produce G-code needed to print each layer. (RepRap, 2020c) The path of the nozzle is planned with the slicing software and can be fine-tuned with some parameters such as layer height, shell thickness and infill pattern. The 3D object is formed by the combination of every layers as the principle of AM. Other than that, the G-code can be written manually as well to give highest flexibility at point level, but it is not feasible for bulky and complex geometry. (RepRap, 2020c)

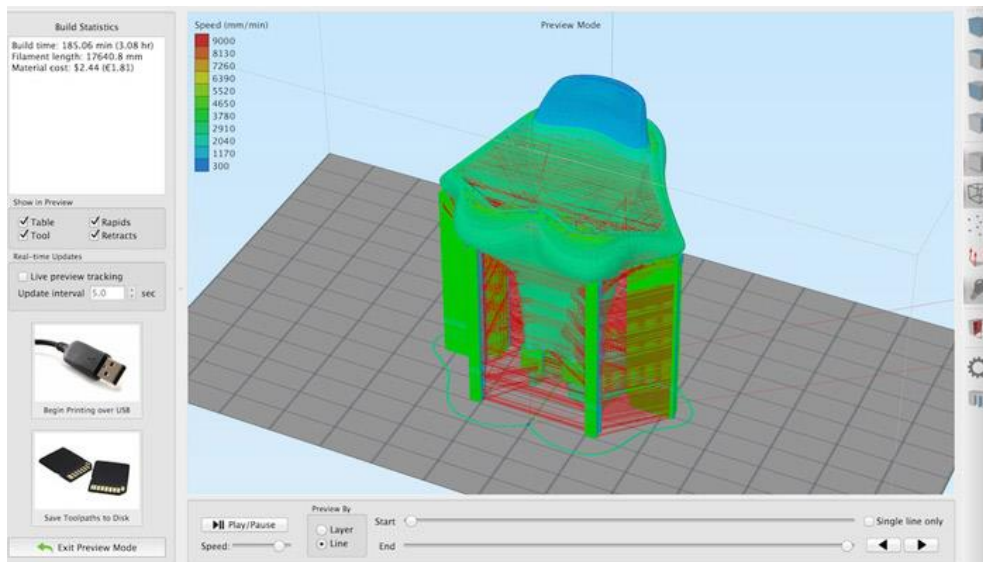


Figure 2.26: Slicing Software with Model Loaded to Produce G-code for 3D Printing. (Carlota, 2019)

G-code constituted by an uppercase letter followed by the number which will indicate different action and parameters. Some common G-code which is usually identical between different firmware are those movements related and follow the RS274/NGC G-code standard. For example, G00 indicate the rapid positioning, G01 is the linear interpolation, G02 is the CW circular interpolation and G03 mean the CCW circular interpolation. (National Institute of Standards and Technology, 2000) The information like the coordinate and feed rate can be added behind the G command to provide complete information for the specified action as shown in Figure 2.27.

```
G28
G1 F1500
G1 X2.0 Y2.0 F3000
G1 X3.0 Y3.0
```

Figure 2.27: Sample of G-code Fragment. (RepRap, 2020c)

Each line of the G-code directs the machine to carry out a discrete action and it will be performed sequentially. The motion in specific route is achieved by stringing of the sets of instruction at point level. (Vincent, 2018)

2.6.2 Temperature Control

Temperature regulation in a 3D printer is also monitored by the firmware setting. For a typical FDM 3D printer, it will have the temperature sensor such as thermistor or thermocouple at the hot end and heated bed to measure the temperature as the feedback for control system.

Thermocouple is a temperature measuring device formed by two different metals which will produce the proportional voltage difference from the temperature variation detected known as “Seebeck effect”. (John and Hoshino, 2019) Thermistor also known as thermal resistor which resistance value will change respond to the temperature different. Thermistor can mainly group into NTC and PTC types. As its name implied, resistance will decrease with increasing temperature for NTC, while resistance will increase with increasing temperature for PTC. (Awalt, 2020) These electrical signals from the temperature sensor are readable by the microcontroller to exert proper control. For a proper temperature regulation, the correct model of temperature sensor must be defined in the firmware.

The control of temperature is critical as the stable temperature for hot end and heated bed is needed for uniform and success printing. PID control algorithm applied in 3D printing field able to provide accurate temperature control, reduce overshoot and fluctuation of temperature. (Wu, Wang and He, 2020) The P, I and D value of the control algorithm will define the behaviour of the controlled waveform.

The incorrect PID tuning will badly affect the temperature control such as the deviation from target temperature, fluctuation of temperature, and large overshoot might occur. Most of the firmware provide PID auto tune function. For instance, running the G-code, “M303 E0 S200 C8” will return the P, I and D value for the firmware configuration. (RepRap, 2019a)

The temperature control in 3D printer can protect the machine from damages. It can be set in the firmware to prevent the under temperature extrusion where the extruder motor will only move if the hot end temperature reached the targeted value. The under temperature extrusion exerts excessive amount of pressure to the nozzle, overload the extruder motor and hence can damage the proper functionality of parts.

The thermal protection is also the important feature in firmware setting for 3D printer. The thermal protection serves to turn off the heater to prevent fire hazard in case that the thermal sensor become loose or broken where the temperature reading keep low and cause the heater to heat up indefinitely. (Marlin, 2020) For Marlin Firmware, there are two levels of thermal protection as summarised in Table 2.5.

Table 2.5: Thermal Protections by Marlin Firmware. (Marlin, 2020)

Output Error	Reason and Causes
Heating failed	<ul style="list-style-type: none"> • The temperature of heater does not rise up within the specified period. • Possibly cause by improper connection of heater or thermal sensor, wrong configuration of thermal sensor.
Thermal runaway	<ul style="list-style-type: none"> • The temperature deviates from the target temperature for a long period. • Possibly cause by poor contact between sensor and heater, bad PID configuration and cold surrounding.

2.7 Summary

In short, prototype developed will be using FDM technology with coreXY structure. As a compromise between cost and functions, closed-looped control for axes motion and web-based interface were decided not be implemented. The prototype developed will included the 5 main components which are opened-loop controlled stepper motor, heated bed, hot end, geared dual drive Bowden extruder and a 32-bits motherboards. For HMI, a control panel and terminal software should be applicable to the prototype for machine control and calibration. In consideration of the compatibility of technologies including StealthChop™, SpreadCycle™, StallGuard™ and CoolStep™, TMC drivers decided to be used in the prototype. Marlin firmware will be used due to the compatibility and features provided including the safety features.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

AM is a broad field and this project will only focus on the study related to the development of a small scale coreXY 3D printer using the FDM print technology that aimed to achieve higher printing speed and maintain acceptable budget and print quality. FDM 3D printer is very convenient for users who willing to create printed parts of different material, texture, properties because it can handle wide variety of feedstock. It is also famous by the scalability of prints, affordable cost and simplicity in use.

The development process of a 3D printer can be initiated by the mechanical structure with proper components selected then follow by firmware installation and calibration. (Sood and Pradhan, 2020) The work plan of this development project is divided into four main phases. The first two phases aimed to be carried out within three months and another three months for the next two phases.

First and foremost, main parts of the 3D printer will be identified, compared and determined. The optimisation between performance and cost is desired. Procurement of the parts is aimed to be done merely at the online shopping platforms for the ease of access by everyone.

Secondly, the mechanical structure will be planned and designed to fit the selected components. The assembly of the design will be carried out using the CAD tools to ensure the proper fitting and logical design before the procurement.

After that, the electronics components with proper connections will be carried out. The suitable firmware will be modified and programmed into the motherboard. The calibration, debugging and tuning will be performed.

Last but not least, the performance evaluation of the prototype will be carried out. The improvement works will be continued whenever necessary to yield the functional prototype. The cost of development will be tabulated and evaluated as well.

3.2 Requirements

The step taken before making decision on the components to be used is to decide the criteria and specification of the 3D printer prototype desired. Then, the components will be selected based on the fulfilment of the requirement, cost and other factors.

Table 3.1: FDM 3D Printer Prototype Specification Rough Plan.

Structure	CoreXY
Operating voltage	24V
Minimum build volume	$200 \times 200 \times 200mm$
Nozzle number	1
Offline printing	Available
Extruder model	Bowden
Achievable printing speed	$> 80mm/s$
Added features: <ul style="list-style-type: none"> • Silent print at low speed • LCD display screen • Sensorless homing (X and Y axes) • Filament shortage detection • Power loss recovery 	

3.3 Components Selection

Components selection should be done earlier to provide more data, information and restrictions in the mechanical design stage. For instances, the 3D model or the dimensional drawing of the selected components is required to design a part that properly fit to it. The 3D models of the selected parts will be found out, modified or produced according to the engineering drawing or dimensional information to simplify the mechanical design and assembly in the next phase.

The decision made in components selection is based on the comparison of price, features or the material properties. The price is compared merely between different sellers and products at different online shopping platforms as it is easy to access and to purchase when required. Besides, the parts that is

commonly used in 3D printers on the market will be selected where possible for easier sourcing and finding of replacement parts.

3.3.1 Main Frame

The construction of main frame is decided to use the European standard 2020 T-slot aluminium alloy extrusion profile connected using the corner brackets, 8mm M5 bolts and M5 T-nuts.

The advantages of using the aluminium extrusion profile is contributed by its material properties and the profile design. Besides, there are many standard parts and accessories available for the standard aluminium extrusion profile such the corner bracket. These fixed dimension reduce the decision made in design and reduce the work load.



Figure 3.1: European Standard 2020 Aluminium Extrusion Profile 3D Model.

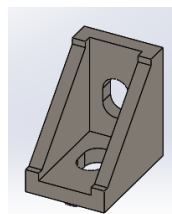


Figure 3.2: Corner Bracket for European Standard 2020 Extrusion 3D Model.

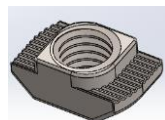


Figure 3.3: M5 T-nut for European Standard 2020 Extrusion 3D Model.

Aluminium alloy has the low density, around 33% of other usual alloys such as steel and brass but it possessed high strength to weight ratio. (Misiolek and Kelly, 2005) Thus, it is suitable to produce light weight frame with high strength. In addition, the shipping fees of the parts purchased will be lesser by the lower weight as compared with other alloy materials.

The profile design allows for high flexibility where the mounting of other components on it can be easily done by the T-nut and bolt along the uniform length. This made the construction work be much simple by avoiding additional processes in mounting of parts and the part can be removed easily as well. It is suitable for the prototype construction which is tend to change from time to time.

3.3.2 Motion Related Components

In the development of a 3D printer, the synchronization in movement is critically important, where the print head need to move accordingly from the action of stepper motor to ensure precise motion achieved. For this purpose, the belt system will be implemented using the synchronous belt or known as timing belt together with the timing pulleys set to realize the coreXY structure.

The selected belt is open loop GT2 timing belt with 6mm belt width. The correspond timing pulleys chosen have 20 teeth and 5mm bore diameter to mount on the shaft of stepper motor used, while selected idler pulleys have the bore diameter of 3mm. The idler pulley with teeth is to contact with the side of the belt with teeth to protect the teeth from deformation, whereas the idler pulley without teeth will contact with back side of the belt.

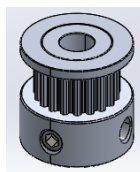


Figure 3.4: 20 Teeth GT2 Timing Pulley 3D Model.

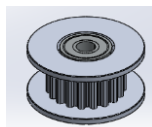


Figure 3.5: GT2 Idler Pulley with Teeth 3D Model.

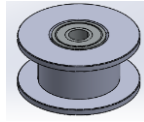


Figure 3.6: GT2 Idler Pulley without Teeth 3D Model.



Figure 3.7: Open Loop 6mm Width GT2 Timing Belt.

The z axis linear motion is achieved by using T8 Acme thread lead screw with 8mm pitch paired with POM anti-backlash nut block for larger load carrying ability and avoid back-driven cause by gravity. POM is the engineering thermoplastic with high strength, low friction and high dimensional stability, which is commonly use in sliding applications. (Ensinger, 2020) For the connection between the lead screw and stepper motor shaft, the 5mm to 8mm flexible shaft coupler is used to tolerate minor misalignment.



Figure 3.8: T8 Acme Thread Lead Screw with 8mm Pitch 3D Model.

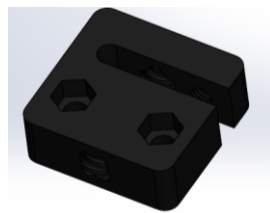


Figure 3.9: POM Anti-backlash Nut Block 3D Model.

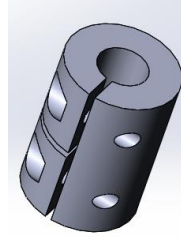


Figure 3.10: 5mm to 8mm Flexible Shaft Coupler 3D Model.

The linear motion guide is achieved by the linear shaft rod paired with linear bearing or bushing. The selected linear shaft rod is chrome-plated hardened steel rod for high strength properties. Chromium has high hardness (Brinell hardness of 800-1000) which can sustain long wear other than providing protection from corrosion and better appearance. (Bloch and Geitner, 2019)

With the aspiration to produce coreXY 3D printer with higher printing speed and silent printing feature. The graphite insert brass bushing is selected to apply in x and y axes, whereas z axis is using the usual linear ball bearing. Polarized graphite can form the lubricating film by adherence to metal surface and exhibit high load carrying capability. (Caenn, Darley and Gray, 2017) The self-lubricating properties of the graphite insert brass bushing is used to support the high speed movement desired. The linear ball bearing is cheaper and thus is used at z-axis which will experience a slower movement during printing because fast movement of linear ball bearing will produce a noticeable noise due to the mechanical vibration of the metal balls.

The shaft diameter selected for x, y and z axes are 8mm, 10mm and 12mm respectively. The larger diameter is to provide higher bending resistance to support larger load. The linear bearing/ bushing selected for x, y and z axes are LM8LUU, LM10LUU and LM12UU.

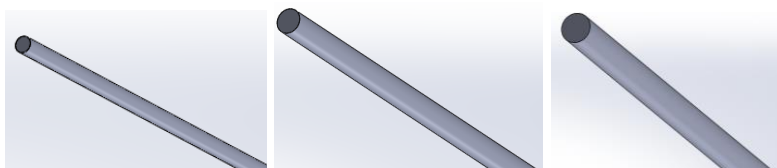


Figure 3.11: Linear Shaft Rods with 8mm, 10mm and 12mm Diameter 3D Model.

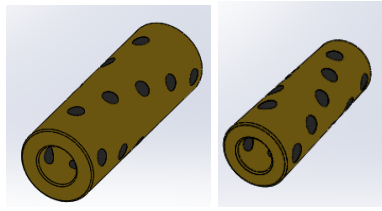


Figure 3.12: LM10LUU and LM8LUU Graphite Insert Brass Bushing 3D Model.

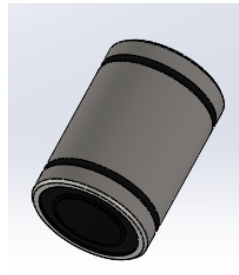


Figure 3.13: LM12UU Linear Ball Bearing 3D Model.

The 608zz ball bearing is selected for applications that required the usage of ball bearing such as spool holder and filament guide. The 608 ball bearing is easily available as the skateboard bearing in the market.

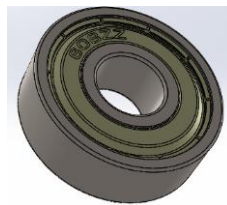


Figure 3.14: 608zz Ball Bearing 3D Model.

3.3.3 Fastener

The fasteners such as bolt and nut in this project used will referred to the ISO metric screw threads. For a better standardization and simpler management, all the connections related to the main frame are selected to use the M5 bolt and M5 T-nut. The length of bolt to mount the printed part on the frame is selected as 10mm which mean the thickness of the printed part need to be 5mm for the another 5mm in the T-nut.

For other designed connection in the customised part, the M3 bolts is selected to pair with the M3 nut or M3 insert. The knurled threaded insert is selected to be embedded into the printed part by the heat supply from the soldering iron or other means for the strong threaded connection. The selected M3 brass knurled threaded insert has 5mm height and 5mm maximum outer diameter. The indented outer diameter at the middle portion of the knurled insert is 4.7mm.

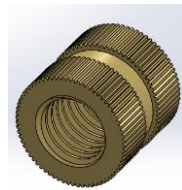


Figure 3.15: M3 Brass Knurled Threaded Insert 3D Model.

3.3.4 Electronics Parts

The stepper motor selected for all the axis movement and extruder is the same that is NEMA 17 4401S with holding torque 43Ncm.

<ul style="list-style-type: none"> • Nema17 Bipolar. • Number of Phase: 2. • Step Angle: 1.8°. • Phase Voltage: 2.6Vdc. • Phase Current: 1.7A. • Resistance/Phase: $1.5\Omega \pm 10\%$. • Inductance: 2.8mH $\pm 20\%$ (1KHz). • Number of Wire: 4 (100cm Length). 	<ul style="list-style-type: none"> • Holding Torque: 43Ncm. • Shaft Diameter: Ø5mm. • Motor Length: 40mm. • Rotor Inertia: 54gcm². • Temperature rise: 80°C Max. • Insulation Class: B. • Dielectric Strength: 500VAC/1-minute. • Mass: 280g.
---	--

Figure 3.16: Brief Data of NEMA 17 4401S. (Handson Technology, 2014)

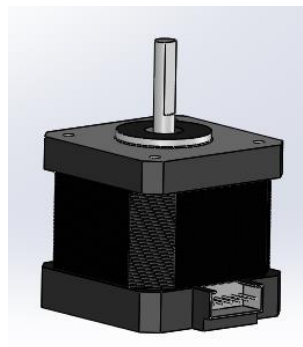


Figure 3.17: NEMA 17 4401S Stepper Motor 3D Model.

The heated bed selected is the MK3 heated bed with 3mm aluminium core operate at 24V. The size is 220×220mm with the four countersink holes for M3 bolt distance at 210mm from each other at four corners. The thermistor is attached to the bottom side of heated bed for temperature measurement. The aluminium pieces provide the flatness to the build platform and the usage of surface material can be eliminated. (RepRap, 2020e)

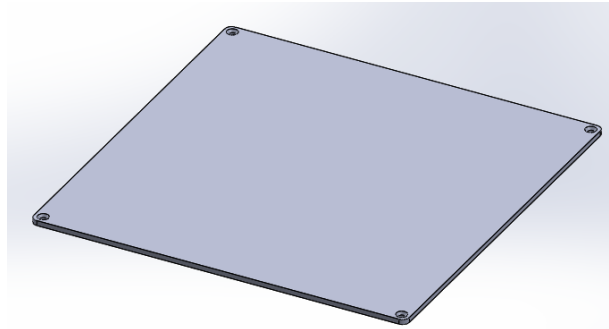


Figure 3.18: MK3 Heated Bed 3D Model.

The four holes are used to attach the bed levelling tool sets which included the spring, M3 bolts and the levelling nut.

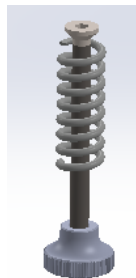


Figure 3.19: MK3 Heated Bed Levelling Tool Set 3D Model.

Hot end selected is the single nozzle E3D V6 hot end which is one of the most common and famous hot end in the 3D printing field. The replacement parts and accessories like silicon shield to cover the heater block is widely available due to its popularity. The set of hot end including the fan mount and cooling fan for the heatsink. The heater cartridge and fan are selected to be 24V.

The nozzle diameter is as default, 0.4mm. The aluminium heater block accommodated thermistor cartridge for temperature measurement. (Gabe, 2020)

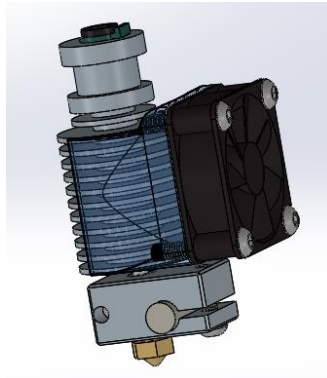


Figure 3.20: E3D V6 Hot End 3D Model.

The cooling fan selected to cool the printed part is the DC 5015 radial fan. The operating voltage is decided to be 24V.

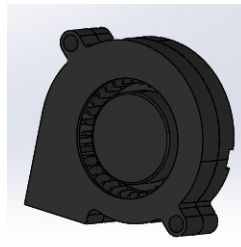


Figure 3.21: 5015 Radial Cooling Fan 3D Model.

Extruder selected is the BMG extruder. This extruder has the dual drive system to eliminate filament slipping problem. The internal gearing ratio of 3:1 increase the driving torque to ensure smooth extrusion; possessed high force to weight ratio which can minimize wobbling during printing. (Clarke, 2017) Besides, it gives the flexibility as it can work with both direct and Bowden drive model.

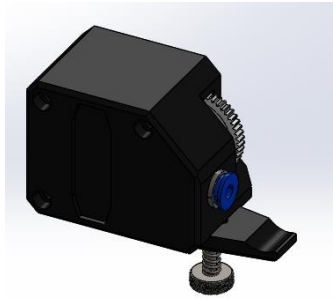


Figure 3.22: BMG Extruder 3D Model.

The extruder is to connect to the NEMA 17 stepper motor with 5mm shaft diameter and sandwiched a 3mm thick mounting piece between for mounting to the frame with M5 bolts and T-nuts.

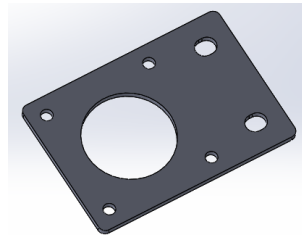


Figure 3.23: Extruder Mounting Piece 3D Model.

Motherboard selected is the MKS Robin E3 because of the attractive features offered at the affordable price. It has higher computational power by 32-bit CPU at 72MHz while Arduino Mega with ATmega2560 is only 8-bit at 16MHz. (Arduino mega 2560 datasheet, n.d.)

Besides, it has integrated TMC2209 stepper drivers which support StealthChop™ for silent movement, SpreadCycle™ for high precision current control, StallGuard™ for sensorless homing, CoolStep™ to reduce heat generated, micro-stepping up to 256 steps and UART bus interface option. (TRINAMIC Motion Control GmbH & Co., 2019) It has the TF (also known as micro SD) card reader to support offline printing and firmware update.

In addition, this motherboard is the direct upgrade replacement for the Creality Ender 3/3Pro, which mean the size and locations of the mounting holes, TF card slot and USB port which required for the casing design are similar. (Makerbase, 2020) Thus, the 3D model can be obtained from the open-source

3D printer by Creality instead of drawing from the dimensional data to ease the casing design.

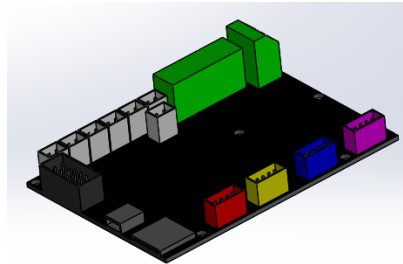


Figure 3.24: Motherboard 3D Model.

- ARM® 32-bit Cortex®-M3 CPU STM32F103RCT6, 72 MHz maximum frequency
- Dcin-Dc5v use MP1584EN, 3A, 1.5MHz, 28V Step-Down Converter
- HOT-BED use HY3403D, parameters is 30V/100A
- HOT-END use HY1403D, parameters is 30V/42A
- 2 NTC100K Temperature measurement
- Integrated 4 TMC2209 UART Mode drive and Support SENSORLESS_HOMING
- 3D TOUCH leveling
- Filament detection
- Support Neopixel light strip, W2812B and so on
- Support Ender3 LCD12864 display and CR-10 LCD12864 display
- Marlin2.0 firmware
- TF card update firmware

Figure 3.25: Features of MKS Robin E3 Motherboard. (Makerbase, 2020)

The cooling fan selected to cool the motherboard in the enclosed case is DC 4010 axial fan. The operating voltage is decided to be 24V.



Figure 3.26: 4010 Axial Cooling Fan 3D Model.

The interface selected is the RepRap 2004 LCD which is commonly use in 3D printer in the market like the Prusa I3 MK3S and there are many designs

for its cover available freely due to its popularity. The price is also cheaper as compared with other display options like the LCD 12864.

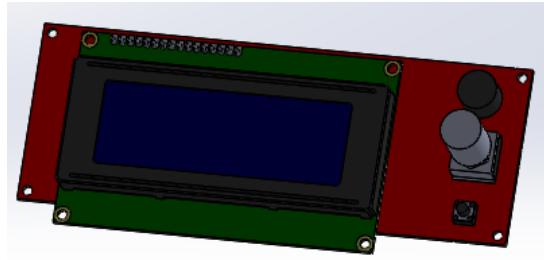


Figure 3.27: RepRap 2004 LCD 3D Model.

The selected power supply is the switching power supply which can convert AC input from the household plug to the DC 24V for the usage in the machine. Switching mode power supply is usually applied in 3D printer because it is less bulky, more efficient and lighter weight than the linear power supply. (Coates, 2020)

The power consumption of the 3D printer mainly caused by the heated bed and hot end. For single hot end and heated bed up to $300 \times 300mm$ size with the motherboard, 5 stepper motors, few cooling fan and some sensors, 360W is the standard selection. (Grames, 2018) Therefore, 360W power rating is far enough for decided system with only 4 stepper motors, $220 \times 220mm$ heated bed and other similar set up. The selected power supply is the S-360-24 switching power supply.

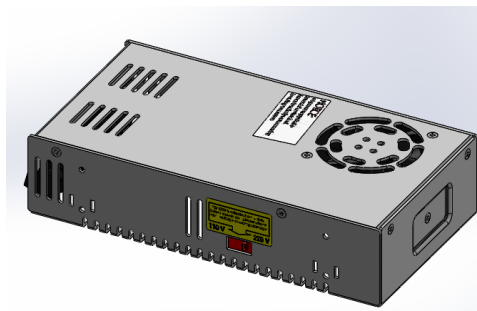


Figure 3.28: S-360-24 Switch Mode Power Supply 3D Model.

The connection from the AC socket to the power supply is to be done by the C13 connector and C14 socket under IEC (International Electrotechnical

Commission) 60320 standard rated at 10A/250VAC. (Interpower, 2020) The 3 pins C13/C14 connection is widely use in the household electric appliances such as rice cooker and computer. The selected C14 socket has a fuse and switch to allow immediate cut off of power supply when required.

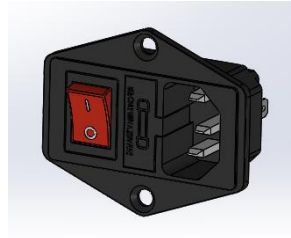


Figure 3.29: C14 Socket with Switch 3D Model.

The end-stop selected for the z axis homing is the contactless optical end stop because the sensorless homing feature's sensitivity for z axis will varies as the printed parts' weight change.

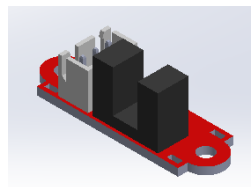


Figure 3.30: Optical End Stop 3D Model.

In order to detect the filament shortage during print, a filament sensor is used. It will serve to sense the present of filament by the switching action triggered by the filament passing through it.

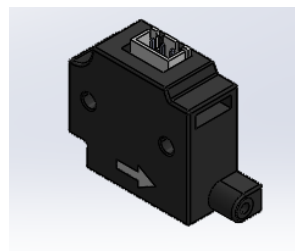


Figure 3.31: Filament Sensor 3D Model.

3.4 Mechanical Design

The 3D printer mechanical design is initiated with reference to the open source coreXY 3D printer, Hypercube that is constructed by a YouTube vlogger from Australia known as Tech2C. (RepRap, 2018a) Hypercube is selected because of the tidy belt system, sturdy structure and proven feasibility.

With all the 3D models available for each and every parts selected, the design process can proceed smoothly where the appropriate dimension can be determined easily by the evaluation of the 3D models with CAD tools.

3.4.1 Dimensional Computation

The frame to be constructed using the European standard 20 series T-slot aluminium extrusion profile, thus the only variable that need to decide is the length of the material. The same applied for the standard linear shaft rods and T8 lead screw. The rough structure is first drawn out with reference to the Hypercube model as a visual aid for computation as shown in Figure 3.32.

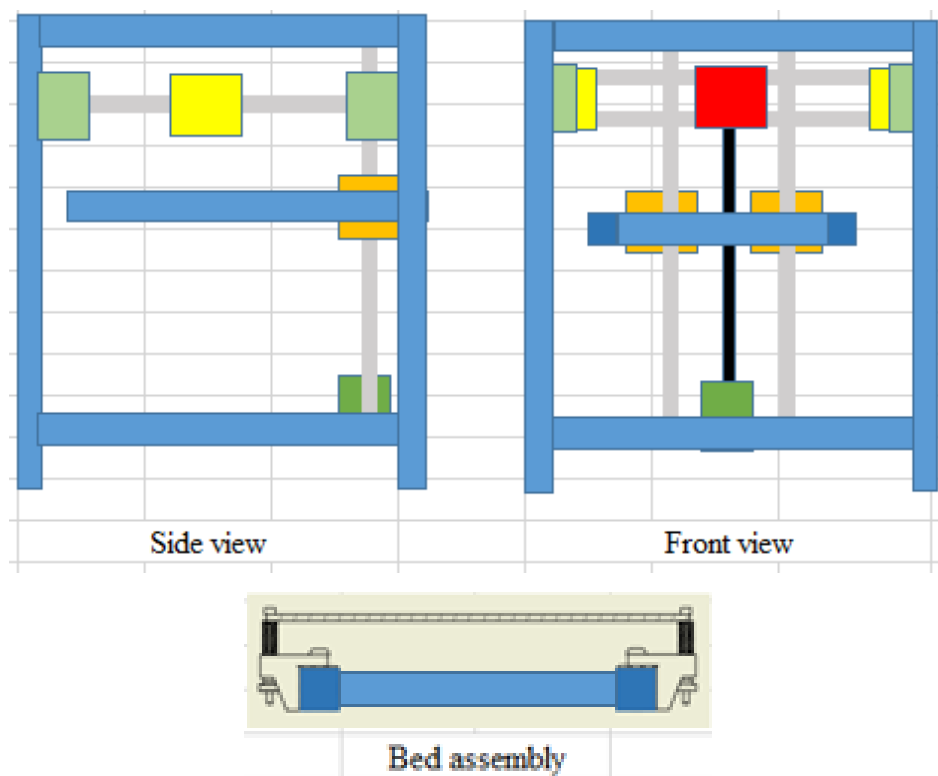


Figure 3.32: Rough Model of Machine Structure.

With the components selected in previous phase, the constant values can be obtained to ease calculation as shown in Table 3.2. The presumption, 50mm offset between horizontal frame piece and ground is made to provide space for the mounting of z-axis stepper motor at the bottom frame piece besides the actual offset. The bed bracket offset is taken for the mounting of heated bed on the aluminium extrusion with the specified bed bracket. The 5mm thickness of printed part is decided so that the printed stepper mount can has at least 5mm thread contact with 10mm M3 bolts used to fix the stepper motor.

Table 3.2: Tabulation of Constants and Presumptions Made in Calculation.

Symbol	Constant		
A	HeatedBed_length	220	mm
B	HeatedBed_width	220	mm
C	HeatedBed_holeSpacing	210	mm
D	HeatedBed_thickness	3	mm
E	LevelingSpring_height	20	mm
F	LM8LUU_length	45	mm
G	LM10LUU_length	55	mm
H	LM12UU_length	30	mm
I	NEMA17_height	64	mm
J	NEMA17_width	42	mm
K	E3DV6_height	63.5	mm
L	AntiBacklashNut_height	33	mm
M	Coupler_height	30	mm
N	IdlerPulley_maxDiameter	18	mm
P	LM8LUU_maxDiameter	15	mm
Q	TimingBelt_width	6	mm
R	MinBuild_height	220	mm
S	AlExrusion_width	20	mm
	Presumption		
X	FrameGround_offset	50	mm
Y	BedBracket_offset	15	mm
Z	PrintedPart_thickness	5	mm

The dimensions are determined with the restriction from the size of selected components such as heated bed and some presumptions made. The parts with approximate dimension computed as in Table 3.3 and Table 3.4 can

be referred to Figure 3.30 according to the colour specified and Table 3.2 for the specified symbol representation.

Table 3.3: Approximate Dimensional Calculation of Carriages and Restricted Regions.

		Approx.Height	Approx. Length
	Y-carriage	$2P+2Q+3Z$	G
	Z-carriage	$2H$	
	X-carriage	$2P+2Q+3Z$	F
	Restricted Region 1	$I+M/2+Z$	
	Restricted Region 2	$I+Z$	J

Table 3.4: Draft Calculation of Material Length Required.

	Approximate length	Chosen length
	2020 Aluminium Extrusion	
Frame_Horizontal	$2J+A+G$	359
Frame_Vertical	$R+(I+M/2+Z)+(2P+2Q+3Z)/2+K+3S+X$	506
Bed_x	$C-2Y-2S$	140
Bed_y	$2J+A+G-J+S$	337
	T8 Lead Screw	
T8 lead screw	$R+L+M/2$	268
	Linear Shaft Rod	
8mm Diameter	Length of Frame_Horizontal+2S	399
10mm Diameter	Length of Frame_Horizontal	359
12mm Diameter	Length of Frame_Vertical-X	456

The computed dimension is to initiated the CAD drawing, it will be modified accordingly in the CAD assembly when needed. The frame has only the horizontal and vertical part as all the horizontal parts is decided to have same length (square shape for top view).

3.4.2 3D Printable Part Design

The customised parts are aimed to be 3D printable to simplify the manufacturing process and lower the cost of fabrication. The design work need to obey few principles to minimize the post fabrication process after the FDM 3D printing. These principles are obtained from the common knowledge, past experience and the test prints produced with FDM 3D printing using Creality Ender 3 Pro with

nozzle of 0.4mm diameter which will also be used to fabricate the prototype in the later.

The principles followed in the mechanical design of the customised parts to be fabricated with AM in this project are:

1. The hole/ cavity is offset 0.4mm from the desired dimension to ensure proper fitting (loose fit).
2. The overhang region should be minimized and the overhang angle should not exceed 45°.
3. There must have at least one flat surface in the design for proper adherence on to the build platform.
4. The maximum build size for a single part is $220 \times 220 \times 250\text{mm}$ according to the 3D printer used.
5. The small feature with dimension below nozzle diameter, 0.4mm is avoided.

The printable parts for main structural components as shown in Figure 3.33 to Figure 3.46 are decided and drawn out with reference from the Hypercube model. The fan duct as shown in Figure 3.47 is contributed by Sean Moe in Thingiverse platform which is the largest community for 3D printing with a lot of downloadable models. The LCD display cover with its support as shown in Figure 3.48 and Figure 3.49 are extracted and modified from open-source Prusa I3 MK3S. The modifications made for the parts to fit the different components selected. The parts as shown in Figure 3.50 to Figure 3.64 are custom designed to accommodate components which included power supply unit, circuit box, extruder, filament spool, optical sensor, filament sensor and some other miscellaneous within the frame volume for compact design to reduce space occupation.

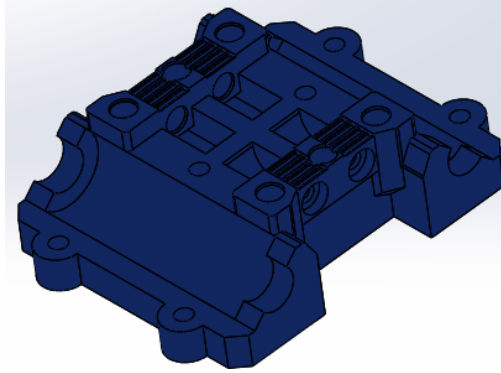


Figure 3.33: X-carriage.

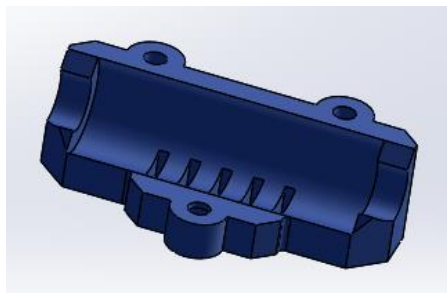


Figure 3.34: X-carriage Clamp.

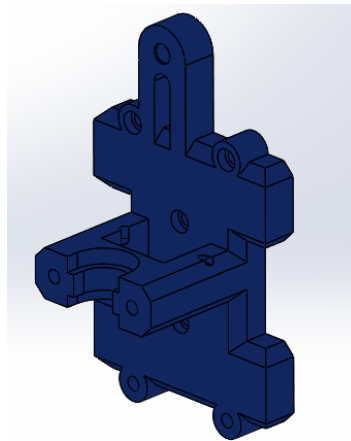


Figure 3.35: Hot End Mount.

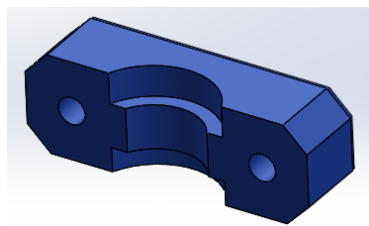


Figure 3.36: Hot End Clamp.

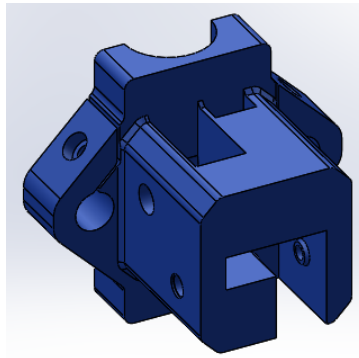


Figure 3.37: Y-carriage.

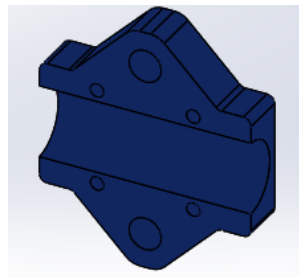


Figure 3.38: Y-carriage Clamp.

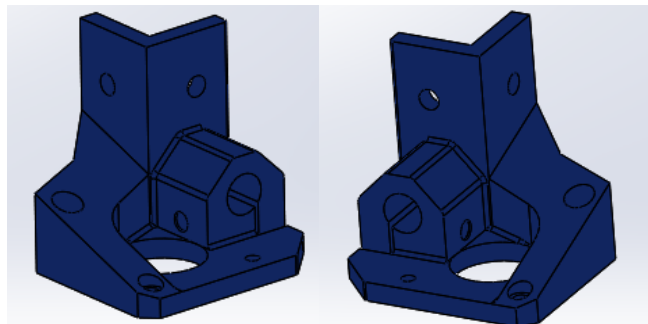


Figure 3.39: XY Stepper Mount (Left and Right).

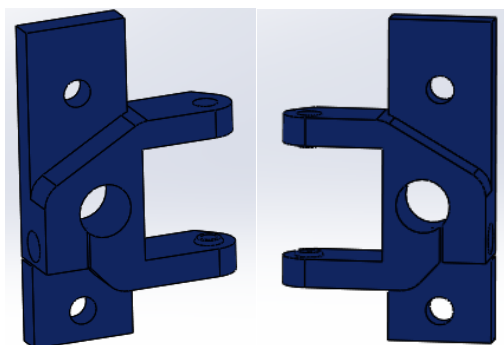


Figure 3.40: Idler Mount (Left and Right).

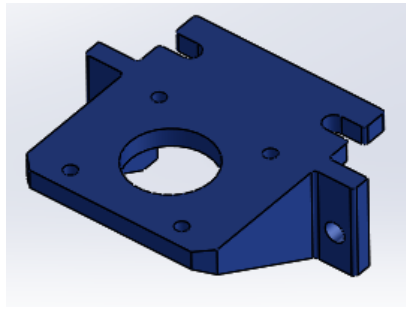


Figure 3.41: Z Stepper Mount.

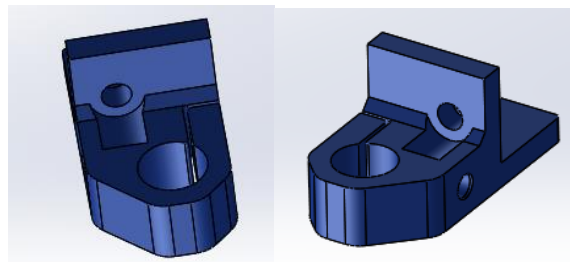


Figure 3.42: Z-linear Rod Bracket (Left and Right).

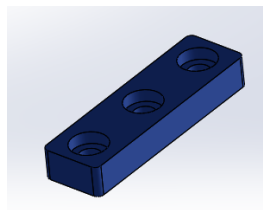


Figure 3.43: Belt Clamp.

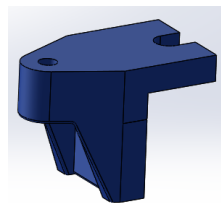


Figure 3.44: Heated Bed Bracket.

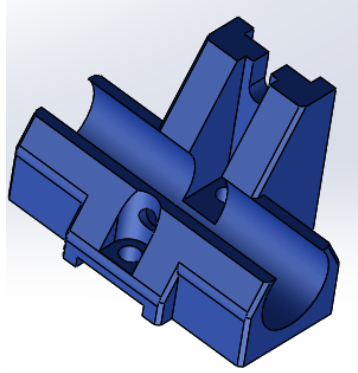


Figure 3.45: Z-carriage.

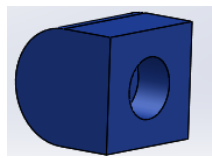


Figure 3.46: Belt Tensioner.

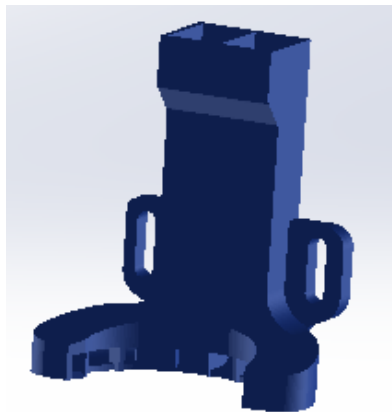


Figure 3.47: Fan Duct.

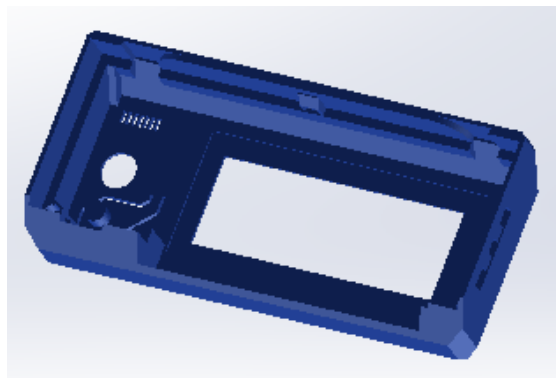


Figure 3.48: 2004 LCD Panel Cover.

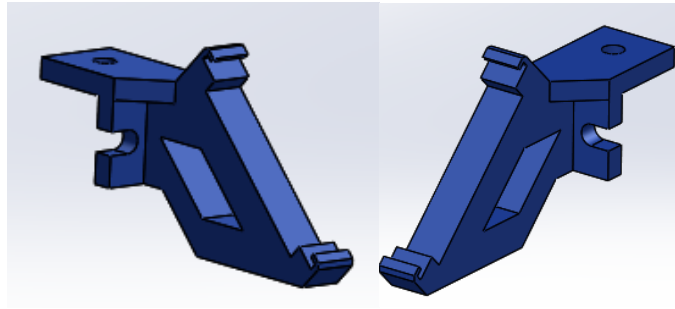


Figure 3.49: 2004 LCD Panel Support (Left and Right).

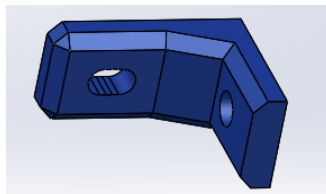


Figure 3.50: PSU Support.

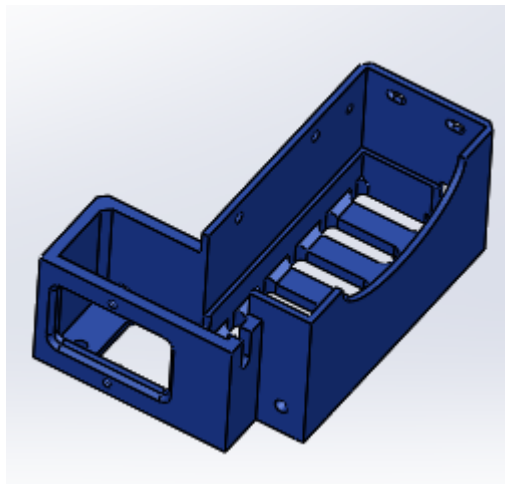


Figure 3.51: PSU Holder.

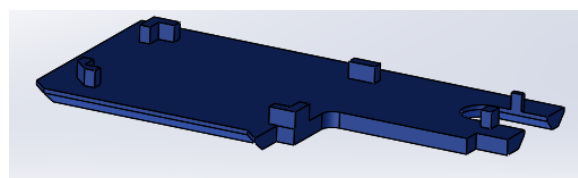


Figure 3.52: PSU Holder Cap.

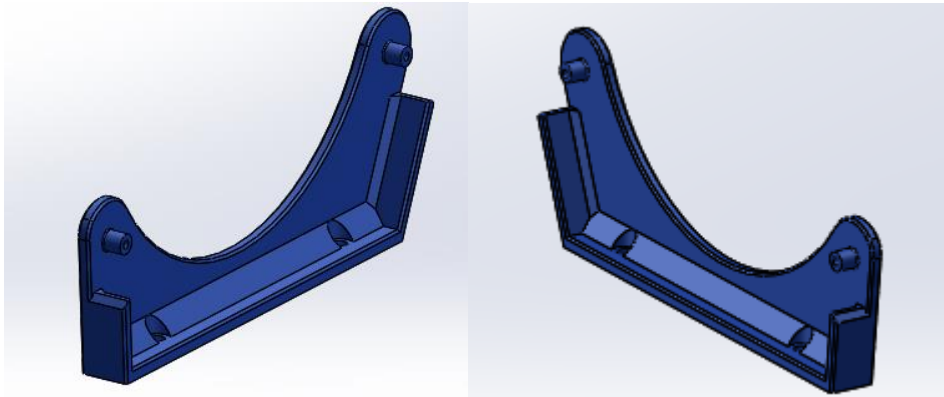


Figure 3.53: Spool Holder (Left Part and Right Part).

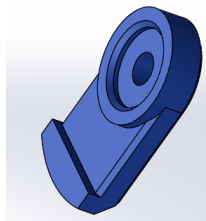


Figure 3.54: Spool Holder Bearing Retainer.

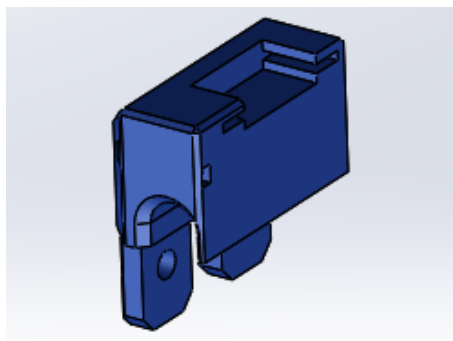


Figure 3.55: Optical End Stop Holder.

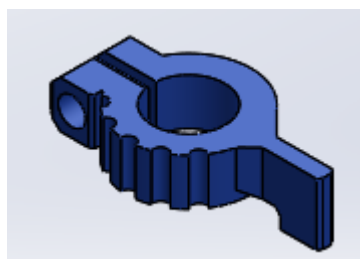


Figure 3.56: Optical End Stop Flag.

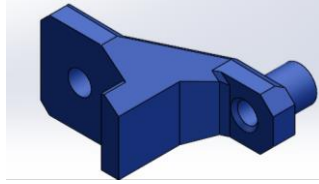


Figure 3.57: Filament Guide Arm.

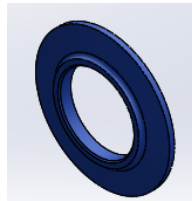


Figure 3.58: Filament Guide Sheave (Half).

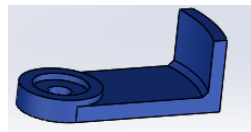


Figure 3.59: Filament Guide Bearing Retainer.

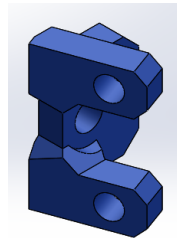


Figure 3.60: Filament Sensor Holder.

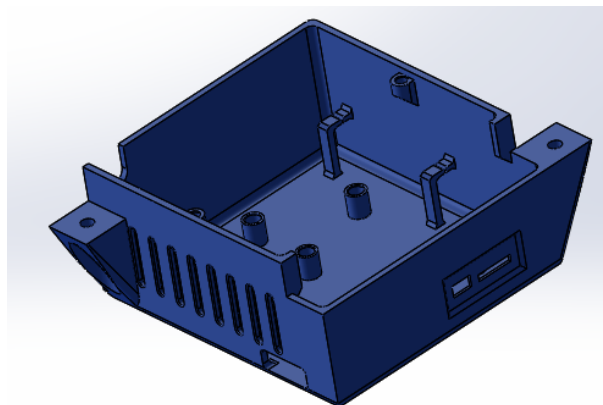


Figure 3.61: Motherboard Case.

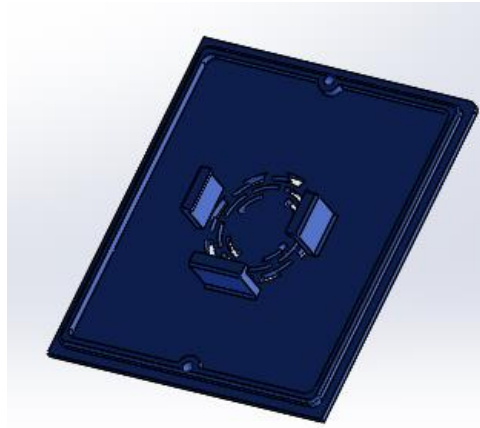


Figure 3.62: Motherboard Case Cover.

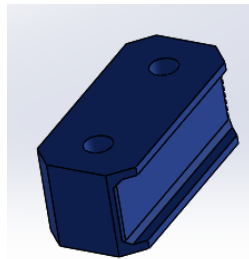


Figure 3.63: Extruder Mount Spacer.

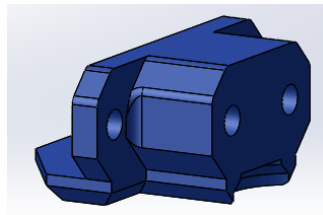


Figure 3.64: POM Nut Block Connector.

3.4.3 Assembly

The assembly made with all the 3D models of designed and selected parts to give the proposed outlook of 3D printer structure as shown in Figure 3.65 to Figure 3.68. The parts are modified and improved in the assembly to eliminate interference between parts, reduce footprint and offer a better outlook. [Older version of assembly can be referred to Appendix C]

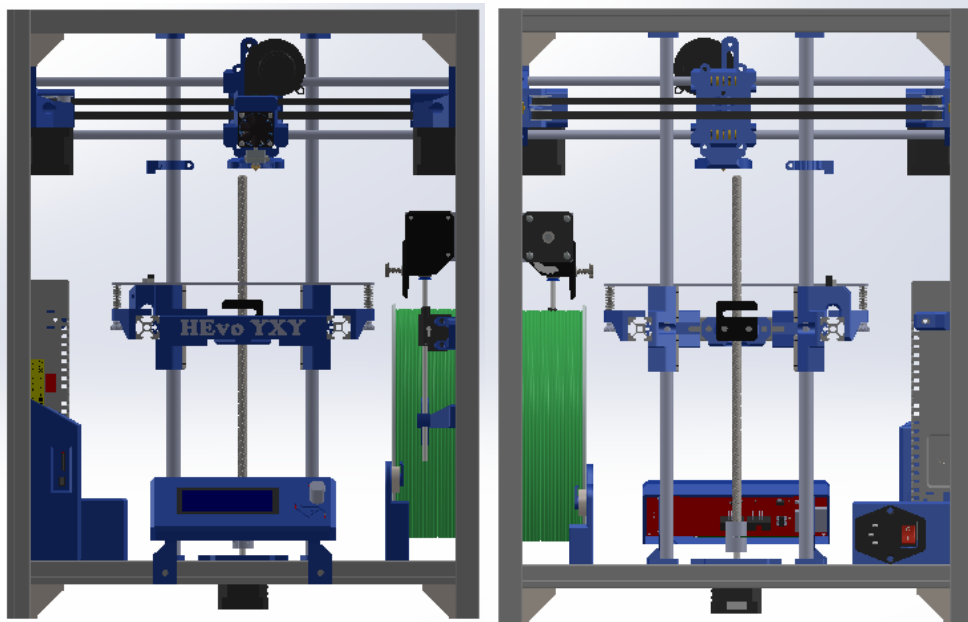


Figure 3.65: Front and Back Views of Proposed Assembly.

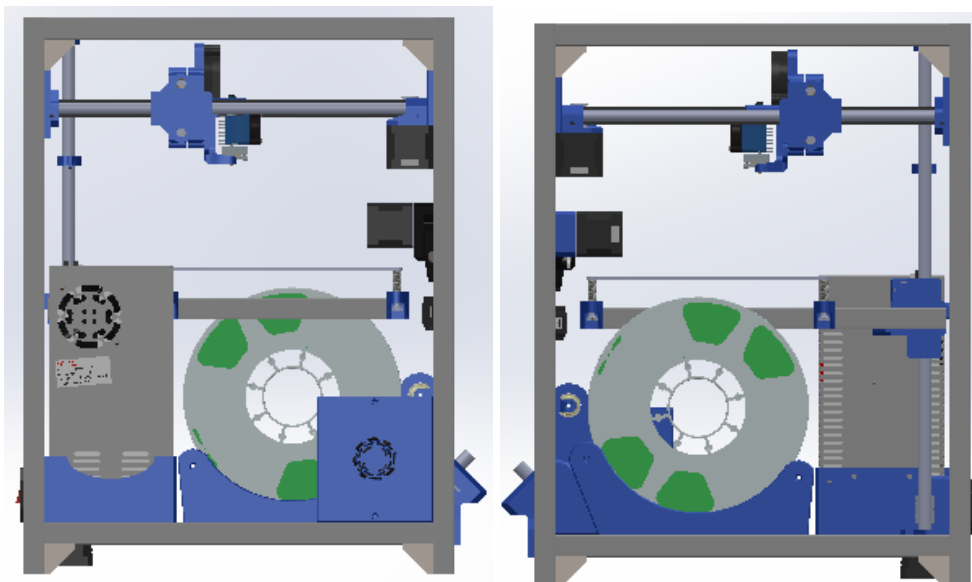


Figure 3.66: Left and Right Views of Proposed Assembly.

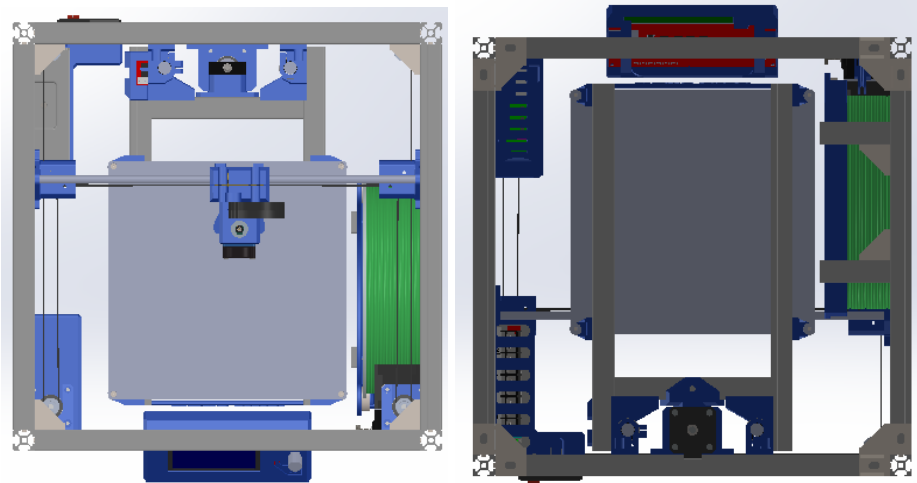


Figure 3.67: Top and Bottom Views of Proposed Assembly.

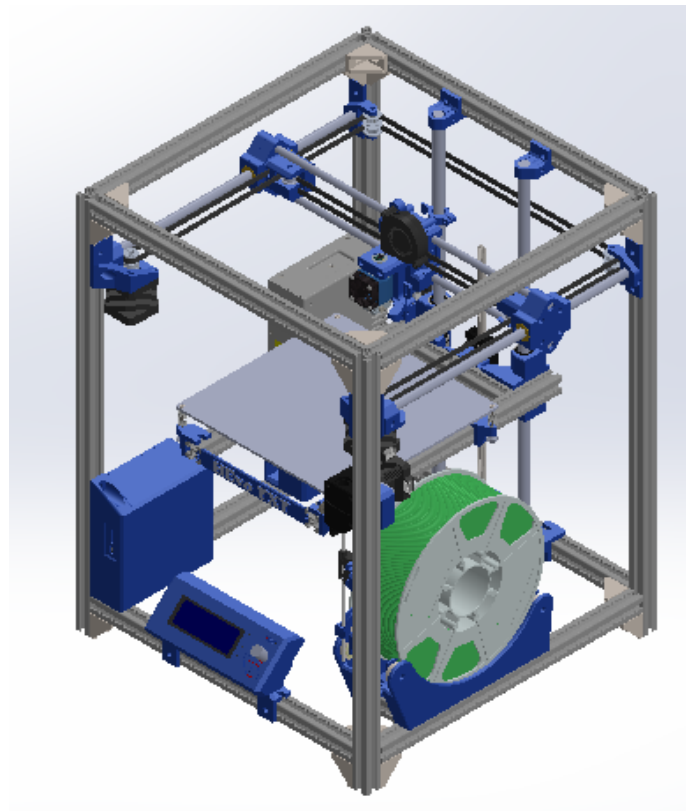


Figure 3.68: Isometric View of Proposed Assembly.

Evaluation of the assembly used to determine the final length for the aluminium extrusion profiles , lead screw, timing belts and shaft rods required while providing minimum relative movement between print head and printing platform of $(x, y, z) = (220mm, 220mm, 220mm)$ and fulfil the design compactness to accommodate the components within the frame.

The cutting tolerance of the material acquired from the sellers will be taken into account to reduce post processing work during assembly of prototype. For instance, the horizontal frame piece need to be slightly longer than the y-axis linear rod to accommodate it without any processing step needed if the actual rod is slightly longer than specified.

The relative movement between print head and printing platform available for the assembly is measured as $(225.6mm, 225.83mm, 233.1mm)$ and fulfil the requirement as proven in Figure 3.69 to Figure 3.71.

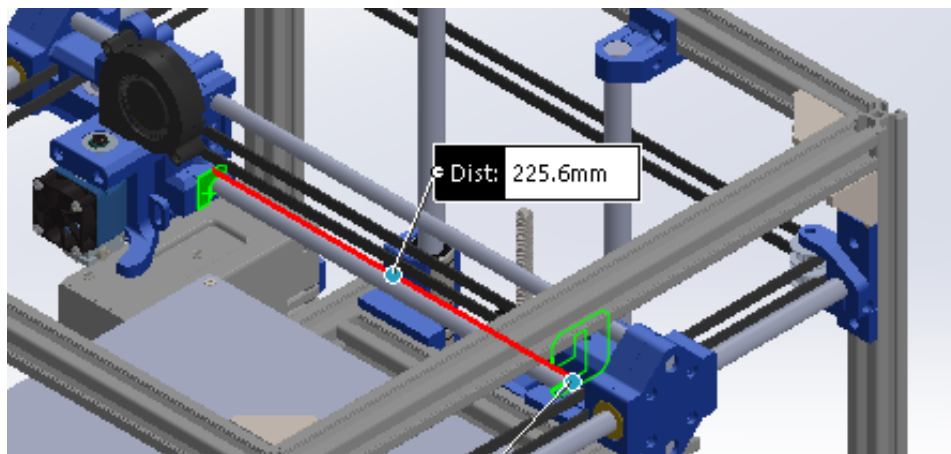


Figure 3.69: X Movement Evaluation.

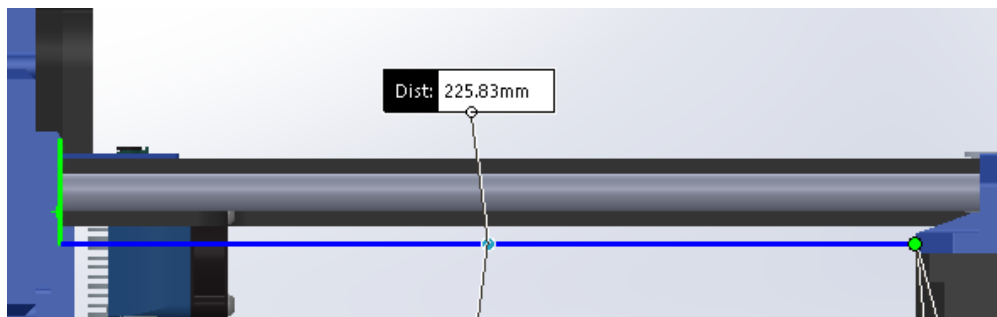


Figure 3.70: Y Movement Evaluation.

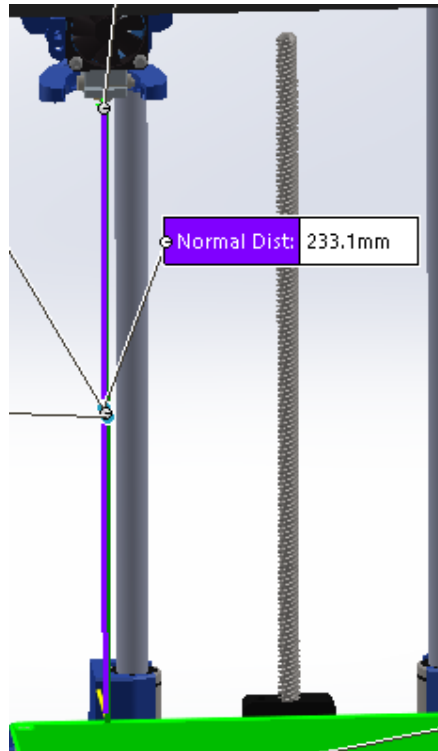


Figure 3.71: Z Movement Evaluation.

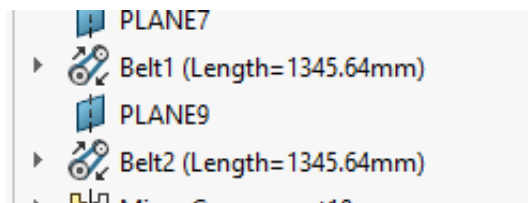


Figure 3.72: Belt Length Evaluation.

Table 3.5: Final Decision of Materials' Length Required.

Material	Part	Length (mm)		
		Evaluated	Tolerance	Decided
2020 Extrusion Profile	Frame-Horizontal	355	± 0.5	355
	Frame-Vertical	510		510
	Bed-x	140		140
	Bed-y	330		330
Linear Shaft Rod	Rod-x	395	± 2	390
	Rod-y	355		350
	Rod-z	450		450
-	T8 Lead Screw	300	N/A	300
Timing Belt	Belt 1	1 346	N/A	1 500
	Belt 2	1 346		1 500

The bill of material to construct a 3D Printer can be obtained from the assembly as shown in Table 3.6. To reduce the working time, the quantity required of the fasteners such as bolts and nuts are just the approximate number. The wires and connectors required are not included in Table 3.6.

Table 3.6: Bill of Materials of Proposed Assembly.

No.	Part Name	Quantity
1	2020Extrusion_BedX	1
2	2020Extrusion_BedY	2
3	2020Extrusion_CornerBracket	26
4	2020Extrusion_FrameHorizontal	8
5	2020Extrusion_FrameVertical	4
6	2020Extrusion_SpoolSupport	2
7	4010_DC-Fan	1
8	5015_RadialCoolingFan	1
9	Ball Bearing 608ZZ	5
10	BedLeveling_M3BoltFlatHead	4
11	BedLeveling_M3NutBlock	4
12	BedLeveling_Spring	4
13	Belt_Clamp	2
14	Belt_Tensioner	4
15	BMG_Extruder	1
16	CopperGraphiteBushing_LM10LUU	2
17	CopperGraphiteBushing_LM8LUU	2
18	Coupler_5x8mm	1
19	E3D_V6_Hotend	1
20	ExtruderMountSpacer	1
21	FanDuct_MK3_HypercubeModded	1
22	FilamentGuide_Arm	1
23	FilamentGuide_BearingRetainer	1
24	FilamentGuide_SheaveHalf	2
25	FilamentSensor	1

26	FilamentSensor_Holder	1
27	FusedAC_MalePowerSocket	1
28	GT2_TimingIdlerPulley_With T	6
29	GT2_TimingIdlerPulley_Without T	2
30	GT2_TimingPulley	2
31	Heatbed_MK3_220x220	1
32	HeatedBed_Bracket	4
33	HotEnd_Clamp	1
34	HotEnd_Mount	1
35	LCD_PanelCover	1
36	LCD_Support_Left	1
37	LCD_Support_Right	1
38	LinearBallBearing_LM12UU	4
39	M3 Bolt 10mm	~20
40	M3 Bolt 16mm	~20
41	M3 Bolt 20mm	~10
42	M3 Bolt 25mm	~10
43	M3 Bolt 30mm	~10
44	M3 Bolt 45mm	~10
45	M3 Nut	~20
46	M3_KnurledInsert	~60
47	M4 Bolt 10mm	~10
48	M5 Bolt 10mm	~100
49	M5 Bolt 8mm	~50
50	M5_T-nut	~150
51	Motherboard	1
52	MotherboardCase	1
53	MotherboardCaseCover	1
54	NamingPlate	1
55	OpticalSensor	1
56	OpticalSensor_Flag	1
57	OpticalSensor_Holder	1

58	POM_Anti-BacklashNutBlock	1
59	POM_NutBlockConnector	1
60	PowerSupply	1
61	PSU_Holder	1
62	PSU_Support	1
63	RepRap 2004 LCD	1
64	SpoolHolder_BearingRetainer	4
65	SpoolHolder_LeftPart	1
66	SpoolHolder_RightPart	1
67	StepperMotor_NEMA17HS4401S	4
68	StepperMountingPlate	1
69	T8_LeadScrew	1
70	TimingBelt	2
71	X_Carriage	1
72	X_Carriage_Clamp	2
73	X_LinearRod	2
74	XY_IdlerMount_Left	1
75	XY_IdlerMount_Right	1
76	XY_StepperMount_Left	1
77	XY_StepperMount_Right	1
78	Y_Carriage	2
79	Y_Carriage_Clamp	2
80	Z_Carriage	2
81	Z_LinearRod	2
82	Z_LinearRodBracket_Left	2
83	Z_LinearRodBracket_Right	2
84	Z_StepperMount	1

3.5 Electronics System

The circuit connection is carried out with reference from the datasheet with the modification made for 3 pins optical end stop (5V, Ground, Signal) used, where the end stop port on the motherboard have only 2 pins (Ground, Signal) for usual

mechanical end stop. From the pin map in Figure 3.73, the 5V pin is obtained from the bottom left corner of AUX-1 where is for the connection of touch screen display that will not be used in this case.

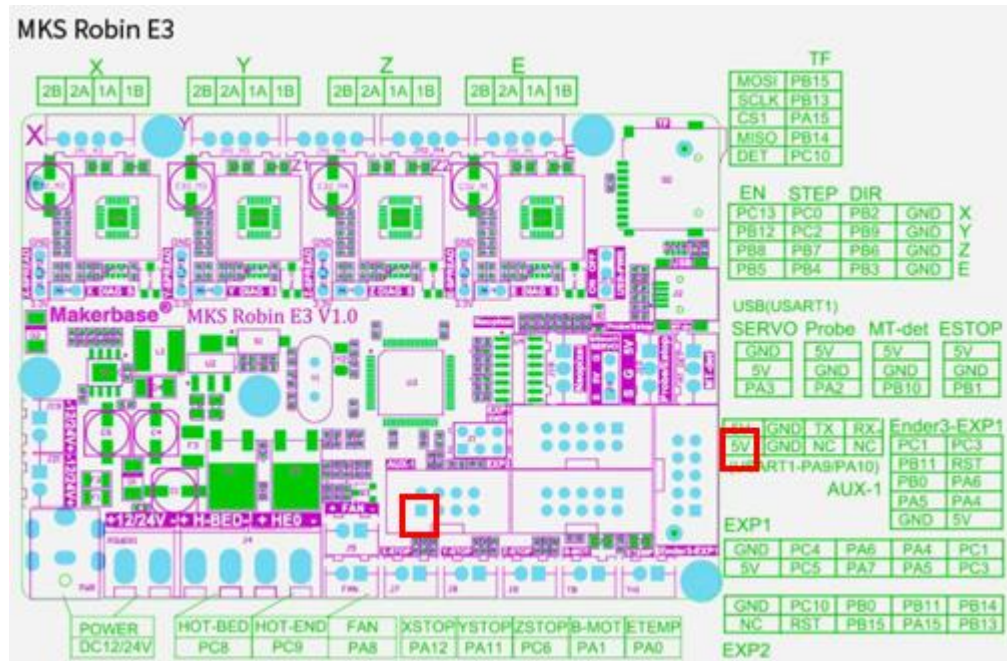


Figure 3.73: MKS Robin E3 Pin Map. (Makerbase, 2020)

The electronics connection for the prototype can be referred to Figure 3.74 with yellow marks indicating the jumper cap inserted on the motherboard. The polarity for the components should be followed carefully except the heater cartridges and thermistors are reversible for their polarity. The 24V outputs are connected to the 3010 heatsink cooling fan for hot end and 4010 axial cooling fan for motherboard. The PWM controlled fan port will connect the 5010 radial part cooling fan. All the TMC2209 stepper drivers are set to StealthChop mode so that it is convertible to SpreadCycle using Hybrid threshold in the firmware configuration. The USB power turned off to prevent power drawing via USB port and only allow the power draw from the 24V 360W main power supply.

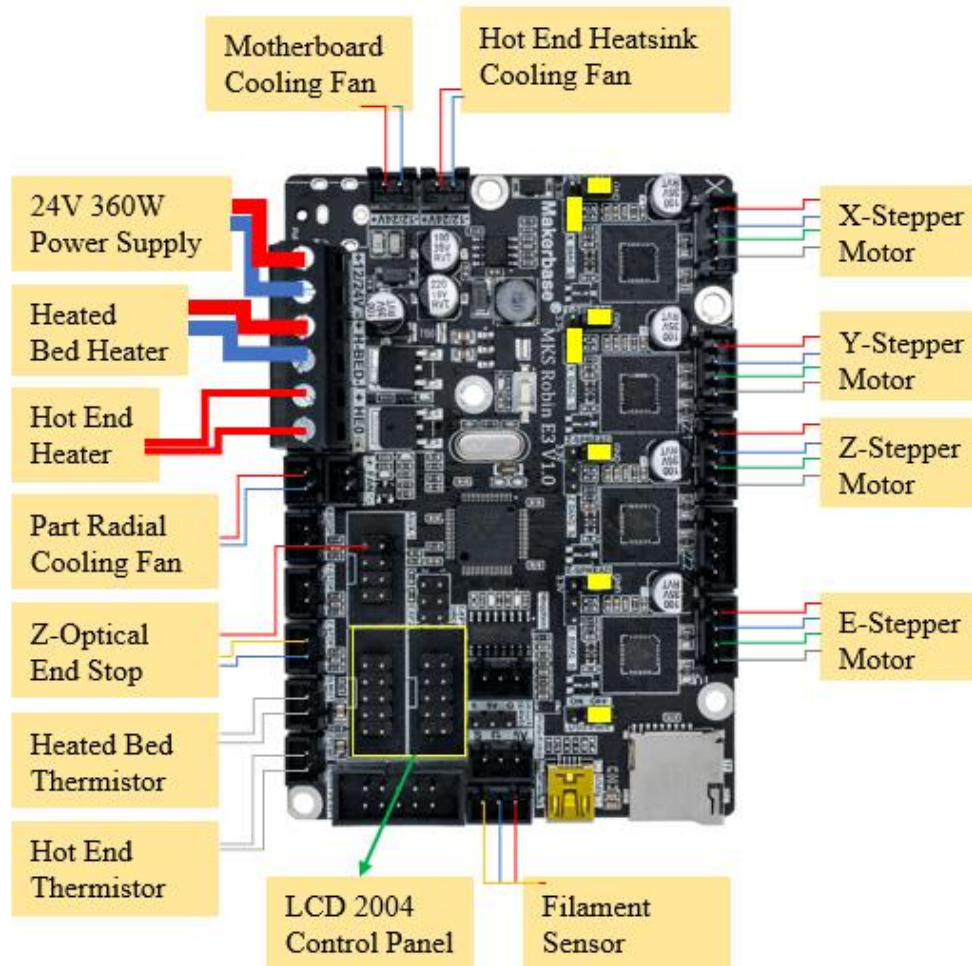


Figure 3.74: Circuit Connections of the Prototype.

For most of the connections, the connecting wires are provided together with the purchasing of parts and there is not much concern allocated for wire selection of the low current drawing components. However, the main power supply and heated bed which will have higher current pass through are decided to use the 14AWG wires which can hold more than 15A of current in performing the connection to avoid the burning of wires as shown in Figure 3.75.

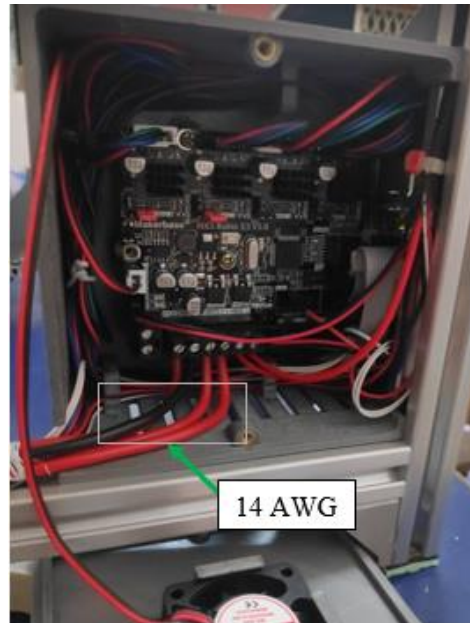


Figure 3.75: Circuit Connection for Prototype.

3.6 Firmware Configuration

The configuration of latest Marlin firmware version 2.0.7 on to the 32-bits MKS Robin E3 motherboard are carried out using Platform IO in the Visual Studio Code. The Marlin firmware used for the prototype is configured based on the components used and computation of the respective parameters. The first step is to declare the components used and related parameters need to be calculated and set. Some of the parameters are just temporary values such as the PID values and stall sensitivity values for sensorless homing, a further tuning via the control panel during the testing period required as it is dependent on the mechanical structure.

Then, the advanced features can be slowly adding in after confirming the basic functions fulfilled. The most important tasks in firmware configuration is to ensure the safety features such as thermal run away protection is enabled and the boundary conditions like travel limits are set in the firmware for safety use of machine. The configuration steps are proceeded by studying the detail comments in the firmware programme. The configuration of firmware mainly done in `configuration.h` [referred to Appendix D] and `configuration_adv.h` [referred to Appendix E]. Some important modifications made over the original firmware are highlighted in yellow as shown in Appendices D and E. The further update on the firmware can be done by saving the latest parameters in EEPROM

via USB connection (Pronterface) or LCD control panel and thus the firmware code shown might not reflect the latest parameters.

3.6.1 Parameters Calculation

Table 3.7: Calculation of Motion Parameters Required for All the Stepper Motors.

Components Selected	Constant Parameters Computation
GT2 Timing Pulley	$\text{Pitch} = 2\text{mm}$ $\text{Number of teeth} = 20$ $\text{Pitch diameter} = \frac{\text{Pitch} \times \text{Number of teeth}}{\pi}$ $= 12.7324\text{mm}$
NEMA 17 Stepper Motor	$\text{Step angle} = 1.8^\circ \text{ per full step}$ $\text{For } \frac{1}{16} \text{ microstepping,}$ $\text{Step angle} = 1.8^\circ \times \frac{1}{16} = 0.1125^\circ/\text{step}$
T8 Lead Screw	$\text{Number of start} = 4$ $\text{Pitch} = 2\text{mm}$ $\text{Lead} = \text{Number of start} \times \text{Pitch}$ $= 8\text{mm/rev}$
Calculation for step/mm in X and Y axes motors	
<p><i>Motor Rotation: Belt Movement</i></p> $360^\circ: \text{Circumference of Pulley} = \pi(\text{Pitch Diameter})$ $360^\circ: 40\text{mm}$ $\text{Revolution angle per mm} = 9^\circ/\text{mm}$ <p>For operation in 1/16 micro step,</p> $\text{Step per mm} = \frac{9^\circ/\text{mm}}{0.1125^\circ/\text{step}} = 80 \text{ steps/mm}$	
Calculation for step/mm in Z axis motor	

<p><i>Motor Rotation: Linear Travel</i></p> <p>$360^\circ: \text{Lead} \times 1\text{rev}$</p> <p>$360^\circ: 8\text{mm}$</p> <p><i>Revolution angle per mm = $45^\circ/\text{mm}$</i></p> <p>For operation in 1/16 micro step,</p> $\text{Step per mm} = \frac{45^\circ/\text{mm}}{0.1125^\circ/\text{step}} = 400 \text{ steps/mm}$
Step/mm for dual-drive geared extruder motor (E-step)
<p>Value recommended by seller,</p> $\text{Step per mm} = 382.2 \text{ steps/mm}$

3.7 Summary of Experiments

In order to test the performance of the prototype developed, some experiments were carried out with results as reported in Chapter 4. The experiments carried out will mainly use for prototype calibration, printing speed limit test, printing accuracy and printing quality evaluation. The calibration steps taken with referenced from fully opened sourced Teaching Tech 3D Printer Calibration by Michael Laws. The printing speed limit test was carried out with the method proposed by Martin Pirringer in 2019. The printing accuracy and quality were evaluated based on the printed 20mm calibration cubes.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Project Prototype

The 3D printer prototype developed according to the design is shown in Figure 4.1. The specifications of the 3D printer developed can be referred to Table 3.1.

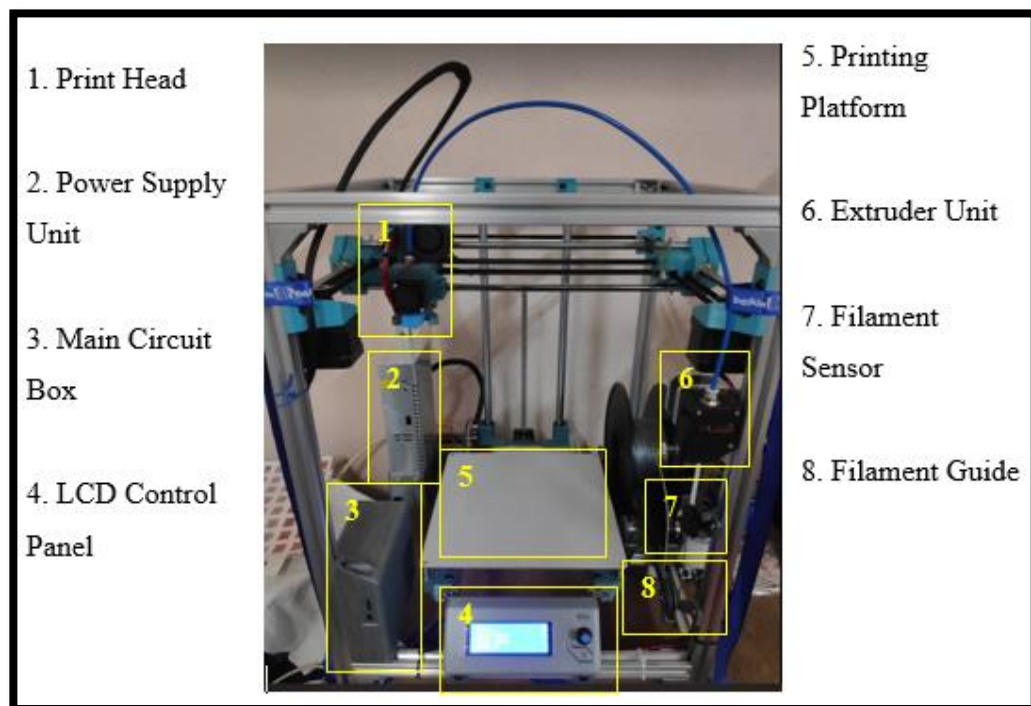


Figure 4.1: CoreXY 3D Printer Prototype Developed.

The first print from this prototype was carried out at low speed, high quality printing profile to test for the performance of prototype in printing generic PLA at the optimal condition prior to the calibration and further tests. The result shown in Figure 4.2 indicated the main function of the prototype is achieved but accompanied with some imperfections which included stringing and inconsistent extrusion width that required the prototype to undergo calibration and fine tune before further tests and evaluation.

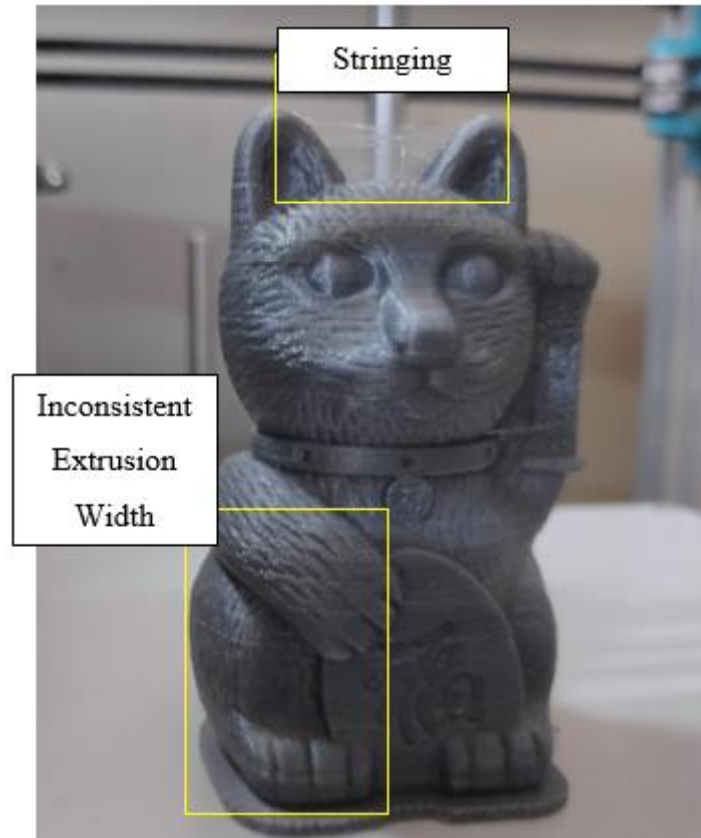


Figure 4.2: The First Print by the Prototype Developed Prior to Calibration.

4.1.1 Fiscal

The 3D printer prototype developed spent around MYR 700.00 for the components and materials procured as detailed in Table 4.1. The electronics components mainly contributed by the motherboard, stepper motors, heated bed and switching power supply spent the largest amount of the budget. Next, the delivery charges and filaments cost also spent a noticeable amount following the electronics components.

The components cost was suppressed by using the clone parts for some costly components like E3D hot end and BMG extruder. The large amount of delivery charges incurred because of the long distance shipping from China to Malaysia for most of the components in the exchange for more complete product list available and cheaper components' cost.

Table 4.1: Tabulation of Cost for Developed Prototype.

Item's Category	Item's Detail	Quantity	Total Price (MYR)	(MYR)
3D Printing Filament	1 kg eSun PETG_Blue	1	60.72	
	1 kg SunDcreate PETG_Not Specified Colour	1	28.50	
	1kg SunDcreate PLA_White	1	24.73	113.95
2020L Aluminium Extrusion Profile	510mm	4		
	355mm	8		
	330mm	2	37.24	
	140mm	1		
	65mm	2		37.24
Hardened Chrome Plated Linear Shaft	D8x 390mm	2		
	D10x 350mm	2	20.88	
	D12x 450mm	2		20.88
Bearing/ Bushing/ Pulley	Graphite Filled Brass Bushing LM8LUU	2	32.88	
	Graphite Filled Brass Bushing LM10LUU	2		
	GT2 Timing Idler Pulley_With T	6	9.96	
	GT2 Timing Idler Pulley_Without T	2	2.41	
	GT2 Timing Pulley	2	1.44	
	Linear Ball Bearing LM12UU	4	8.36	
	608zz Ball Bearing	5	3.00	58.05
Fastener/ Connector	M5x 8mm Hex Socket Bolt	50		
	M5x 10mm Hex Socket Bolt	100	30.00	
	2020 extrusion M5 T Sliding Nut	100		
	M3x 5x 5mm Brass Cylindrical Knurled Threaded Insert	100		
	2020 Corner Bracket	26	14.12	
	M4x 10mm Hex Socket Bolt	20	1.20	
	NEMA 17 Stepper Mounting Plate	1	2.14	
	MK3 Bed Leveling Set (Spring+ M3x35mm Bolt +Nut)	4	3.08	
	Openbuilds Antibacklash POM Nut Block	1	1.40	
	T8 Lead Screw_300mm	1	5.05	
	M3 Bolt Set (10mm, 16mm, 20mm, 25mm, 30mm, 45mm)	1	15.00	
	5-8mm Flexible Coupling_H-25mm	1	2.74	74.73
Electronic Components	MK3 220x 220 Aluminium Heated Bead (+14AWG wire)	1	32.96	
	XT60 Male+Female Connector Set_30cm wire_14AWG	1	4.95	
	Optical Endstop (+wire)	1	1.66	
	RepRap LCD 2004 Control Panel (+wire)	1	14.03	
	NEMA 17 4401S Stepper Motor (+wire)	4	76.00	
	2 Pins Male to Female Connecting Wire_70cm	2	1.20	
	E3D-V6 Hot End (+Thermistor, 3010 Fan, Heater)_Clone	1	16.31	
	BMG Dual Drive Extruder_Clone	1	17.22	
	5015 Radial Fan_24V (+30cm wire)	1	1.85	
	4010 Axial Fan_24V (+30cm wire)	1	1.34	
	Filament Sensor (+wire)	1	1.88	
	Fused AC Male Power Socket (+wire)	1	4.08	
	MKS Robin E3 Motherboard (+Heatsink, Jumper Cap)	1	75.28	
	S-360-24 Switching Power Supply	1	35.09	283.85
Miscellaneous	Silicon Cover for E3D-V6 Heat Block	1	3.69	
	GT2 6mm Timing Belt_5m	1	9.17	
	Teflon Tube Cutter	1	1.91	
	PET 8mm Expandable Braided Sleeving_1m	1	0.51	
	2*4mm Teflon Tube_1m	1	0.77	
	STARS-922 HeatSink Plaster_50g	1	1.60	17.65
Delivery Charges	4.79kg_TaoBao Shipping Fee	1	30.76	
	TaoBao Courier Fee	1	9.05	
	4.405kg_TaoBao Shipping Fee	1	29.80	
	1.814kg_TaoBao Shipping Fee	1	16.80	
	3.03kg_TaoBao Shipping Fee	1	24.74	111.15
SUM			717.50	

4.2 Calibration

The result from calibration and testing of the prototype's performance will deviate according to the type of filament feedstock and other conditions such as air flow and ambient temperature. For this case, the tests and calibrations processes are carried out at the same place with stable air flow and using the same feedstock, generic PLA for the whole process.

The prototype undergoes a series of calibration process with reference from Teaching Tech 3D Printer Calibration including: PID auto tune, E-steps calibration, slicer flow tuning, retraction tuning, temperature tuning, acceleration tuning and linear advance.

4.2.1 PID Auto Tune

The PID auto tune carried out for the stable and consistent heating of the hot end and heated bed to pre-set value. With PID heating feature enabled in Marlin firmware, PID auto tune was done by the G-code communication with the 3D printer via a terminal software, Pronterface.

The G-code sent in the format M303 E[1] S[2] U[3], where [1] is the part selected, [2] temperature required, [3] storing place as shown in Figure 4.3 and Figure 4.4. (Laws, n.d.)

```
>>> M303 E0 S200 U1
SENDING:M303 E0 S200 U1
PID Autotune start
bias: 79 d: 79 min: 196.98 max: 207.75
bias: 75 d: 75 min: 196.60 max: 204.03
bias: 72 d: 72 min: 196.98 max: 203.56 Ku: 27.83 Tu: 25.17
Classic PID
Kp: 16.70 Ki: 1.33 Kd: 52.55
bias: 72 d: 72 min: 196.98 max: 203.03 Ku: 30.27 Tu: 24.53
Classic PID
Kp: 18.16 Ki: 1.48 Kd: 55.70
bias: 71 d: 71 min: 197.10 max: 203.50 Ku: 28.25 Tu: 24.85
Classic PID
Kp: 16.95 Ki: 1.36 Kd: 52.66
PID Autotune finished! Put the last Kp, Ki and Kd constants from below into Configuration.h
#define DEFAULT_Kp 16.95
#define DEFAULT_Ki 1.36
#define DEFAULT_Kd 52.66
```

Figure 4.3: PID Auto Tune of Hot End at 200 Degree Celsius for Normal PLA Printing with Result Stored to RAM.

```

>>> M303 E-1 S60 U1
SENDING:M303 E-1 S60 U1
PID Autotune start
bias: 33 d: 33 min: 59.84 max: 62.10
bias: 29 d: 29 min: 59.85 max: 60.17
bias: 27 d: 27 min: 59.83 max: 60.17 Ku: 196.44 Tu: 15.55
No overshoot
Kp: 39.29 Ki: 5.05 Kd: 203.69
bias: 27 d: 27 min: 59.84 max: 60.14 Ku: 229.18 Tu: 16.03
No overshoot
Kp: 45.84 Ki: 5.72 Kd: 244.98
bias: 26 d: 26 min: 59.83 max: 60.15 Ku: 203.72 Tu: 16.03
No overshoot
Kp: 40.74 Ki: 5.08 Kd: 217.76
PID Autotune finished! Put the last Kp, Ki and Kd constants from below into Configuration.h
#define DEFAULT_bedKp 40.74
#define DEFAULT_bedKi 5.08
#define DEFAULT_bedKd 217.76

```

Figure 4.4: PID Auto Tune of Heated Bed at 60 Degree Celsius for Normal PLA Printing with Result Stored to RAM.

After the each PID auto tune, the PID values directly saved to EEPROM with M500 instead of modifying the firmware and upload again as in Figure 4.5.

```

>>> M500
SENDING:M500
echo:Settings Stored (670 bytes; crc 17841)


```


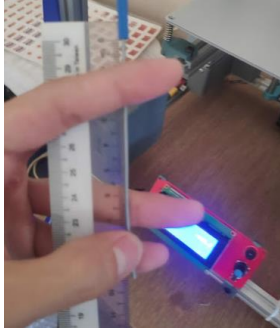

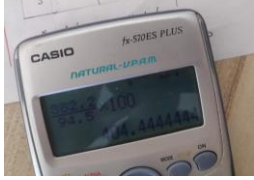


Figure 4.5: Save Data to EEPROM.

4.2.2 E-steps Calibration

Extruder step calibration required to tolerate the slight deviation of physical extruder to the theoretical case in order to avoid print defects such as under-extrusion or over-extrusion. The steps performed in doing the E-step calibration are tabulated in Table 4.2.

Table 4.2: Procedure in E-steps Calibration.

No.	Step Description	Demonstration
1	Load in the filament and cut it flat with a reference point, end of Teflon tube.	

2	Set a value of extrusion for the filament, L_{ideal} .	
3	Measure the actual extruded length, L_{actual} with respect to the reference point.	
4	Take note of current E-step value, E_{old}	
5	Calculate the new E-step, $E_{new} = \frac{E_{old}}{L_{actual}} \times L_{ideal}$	
6	Update the new E-step value calculated.	
7	Stored the setting to EEPROM.	

4.2.3 Slicer Flow Tuning

The slicer flow tuning aimed to figure out the ideal amount of extruded filament by the slicer used to increase the accuracy of print. This process started by printing out a single walled hollow cube with the prototype and measure the walls thickness with high precision tools such as Vernier callipers with 0.01mm accuracy as shown in Figure 4.6. Then, compute and update the value of flow or extrusion multiplier in the slicer as shown in Table 4.3.



Figure 4.6: Single Walled Hollow Cube Printed with 100% Flow and High Precision Callipers.

Table 4.3: Slicer's Extrusion Flow Computation.

Printed Cube's Wall	Wall Thickness Measured (mm)					
	1	2	3	4	5	Average
1	0.43	0.42	0.43	0.42	0.43	0.426
2	0.41	0.42	0.43	0.41	0.41	0.416
3	0.40	0.41	0.40	0.40	0.40	0.402
4	0.41	0.41	0.42	0.42	0.42	0.416
Average wall thickness, T_{actual} :						0.415
<p><i>Ideal wall thickness, $T_{ideal} = \text{nozzle diameter} = 0.4 \text{ mm}$</i></p> <p><i>New extrusion flow, $Flow_{new} = \frac{Flow_{old}}{T_{actual}} \times T_{ideal}$</i></p> <p>$= \frac{100}{0.415} \times 0.4 = 96.39\%$</p>						

4.2.4 Retraction Tuning

Retraction tuning targeted to reduce the stringing effect from the printed products by adjustment on few parameters including retraction distance, retraction speed, extra restart distance, prime speed and z hop. (Laws, n.d.) The prints for stringing test with different combination of the values from these parameters have been fabricated as shown in Figure 4.7.

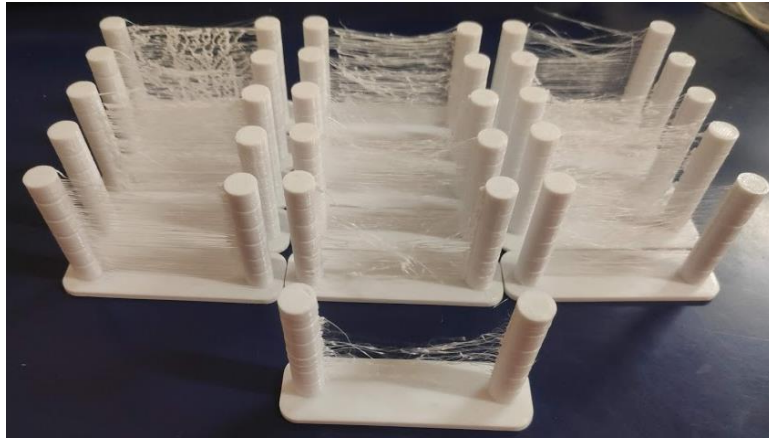


Figure 4.7: Prints for String Test with Different Parameters Set on Every 5mm Layer Height.

The optimum condition for string reduction of PLA printing among the 96 combinations made with the prototype developed are shown in Figure 4.8 and Figure 4.9. The optimal parameters set are 6mm retraction, 30mm/s retraction speed, 40mm/s prime speed, no extra restart distance and zero z hop.

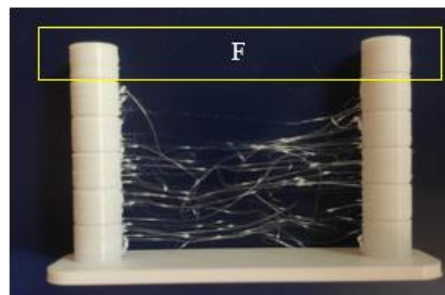


Figure 4.8: Result for Optimum Retraction Setting Among the 96 Combinations Made.


Reference Diagram		Segment	Retraction distance (mm)	Retraction speed (mm/sec)	Extra restart distance (mm)	Prime (unretract) speed (mm/sec)	Z hop (mm)
	F	<input type="text" value="6"/>	<input type="text" value="30"/>	<input type="text" value="0"/>	<input type="text" value="40"/>	<input type="text" value="0"/>	
	E	<input type="text" value="5"/>	<input type="text" value="30"/>	<input type="text" value="0"/>	<input type="text" value="40"/>	<input type="text" value="0"/>	
	D	<input type="text" value="4"/>	<input type="text" value="30"/>	<input type="text" value="0"/>	<input type="text" value="40"/>	<input type="text" value="0"/>	
	C	<input type="text" value="3"/>	<input type="text" value="30"/>	<input type="text" value="0"/>	<input type="text" value="40"/>	<input type="text" value="0"/>	
	B	<input type="text" value="2"/>	<input type="text" value="30"/>	<input type="text" value="0"/>	<input type="text" value="40"/>	<input type="text" value="0"/>	
	A	<input type="text" value="1"/>	<input type="text" value="30"/>	<input type="text" value="0"/>	<input type="text" value="40"/>	<input type="text" value="0"/>	

Figure 4.9: Optimal Parameters for Retraction Tuning of the Prototype.

4.2.5 Temperature Tuning

The temperature tuning used to find out the optimum printing temperature for the filament used. Temperature towers shown in Figure 4.10 showed that the PLA used have a better overhang and smooth appearance for the printing temperature in the range of $180^{\circ}\text{C} - 210^{\circ}\text{C}$. Rough surface finish resulted for printing temperature above 215°C .

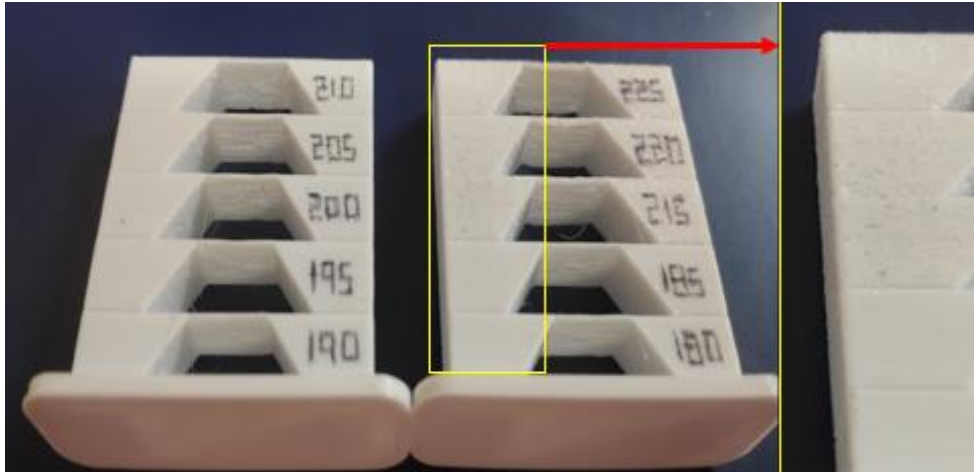


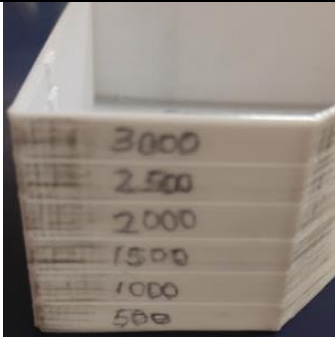
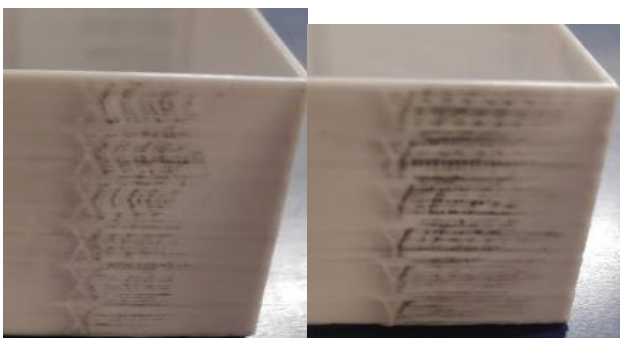
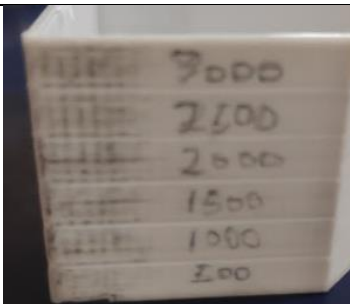
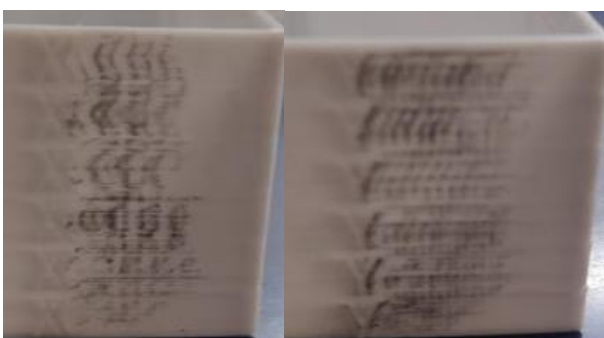
Figure 4.10: Temperature Towers for PLA.


4.2.6 Acceleration Tuning

Acceleration tuning needed for the compromise between faster print speed and print quality assurance. A higher acceleration and junction deviation will result in a faster print time desired because the printer achieve top speed limit more quickly and maintains a higher speed when cornering respectively. (Laws, n.d.) However, this may result in shorter lifespan of components and surface defects such as ghosting.

The results of acceleration towers printed with different values of Junction Deviation, J, Acceleration, A and Base feed rate, FR to figure out the optimal point for the compromise between speed and quality are tabulated and analysed in Table 4.4. The acceptance on the printed output results is subjective and may varies among person.

Table 4.4: Acceleration and Junction Deviation Parameters Determination with Acceleration Towers.

No	Parameters	Output Results
1	$J = 0.01$ $FR = 100 \text{ mm/s}$ $A =$ $500 - 3000 \frac{\text{mm}}{\text{s}^2}$ (Increment of 500)	 
2	$J = 0.01$ $FR = 150 \text{ mm/s}$ $A =$ $500 - 3000 \frac{\text{mm}}{\text{s}^2}$ (Increment of 500)	 

3	$FR = 150 \text{ mm/s}$ $A = 1500 \frac{\text{mm}}{\text{s}^2}$ $J = 0.01 - 0.10$ (0.01, 0.02, 0.04, 0.06, 0.08, 0.10)	
<p><u>Analysis of Result</u></p> <ul style="list-style-type: none"> • Result in set 1 showed ghosting effect diminished below 1500 mm/s^2 for base feed rate at 100 mm/s with $J=0.01$. • Result in set 2 showed 1500 mm/s^2 of acceleration is the upper limit for acceptable amount of surface defect with base feed rate of 150 mm/s, $J=0.01$. • Result in set 3 showed not much variation of printed output between different junction deviation values below 0.10 at base feed rate of 150 mm/s with acceleration at 1500 mm/s^2. <p><u>Conclusion</u></p> <p>For faster print speed with consideration on the print quality, base feed rate = 150 mm/s, acceleration = 1500 mm/s^2 and junction deviation = 0.10.</p>		

4.2.7 Linear Advance

Linear advance used to tune the timing of extrusion to cope with the delay between extruder motion and physical extrusion of molten filament from nozzle for more consistent extrusion width. (Laws, n.d.) This process carried out by using “Marlin Linear Advance Pattern Generator” and observe the result from the printed outputs as shown in Figure 4.11 and Figure 4.12. The optimal K factor required will generate a most consistent line width. From Figure 4.11, the

desired K factor fall in the range between 0 to 0.4. A higher precision reading in Figure 4.12 indicate that the optimum value for K factor is 0.35.



Figure 4.11: Outputs with K-factor from 0 to 2, Increment of 0.2.



Figure 4.12: Outputs with K-factor from 0 to 0.4, Increment of 0.05.




4.3 Feature and Performance of Prototype

The feature and performance of prototype for this project will be tested and evaluated mainly on the accuracy of print and speed limit of the printing process by using PLA as the feedstock.

4.3.1 Accuracy of Printed Products

The accuracy of 3D printer developed tested by printing out a 20mm cube with the 3D printer prototype, measure and average the actual dimension on each axis and compare with the ideal case as shown in Table 4.5.

Table 4.5: Prototype's Accuracy Evaluation Based on 20mm Cube Printed.

Axis	Demonstration Image	Actual Readings (mm)	Percentage Error
X		20.20	$\frac{20.228 - 20}{20} \times 100\% = 1.14\%$
		20.16	
		20.28	
		20.24	
		20.26	
		Average= 20.228	
Y		20.07	$\frac{20.066 - 20}{20} \times 100\% = 0.33\%$
		20.05	
		20.07	
		20.06	
		20.08	
		Average= 20.066	
Z		19.94	$\frac{20 - 19.946}{20} \times 100\% = 0.27\%$
		19.99	
		19.96	
		19.94	
		19.90	
		Average= 19.946	

The percentage error for x axis is higher than 1% where there will be more than 1mm tolerance for 100mm printed. The accuracy of the prototype was improved with the fine tuning on step/mm declared during firmware configuration for each axis as shown in Table 4.6 and then accuracy test repeated for latest update on accuracy of prototype as shown in Table 4.7.

Table 4.6: Update on Step/mm for Each Axis Based on Printed Cube Result.

$[Step/mm_{new}] = \frac{Ideal\ Value \times [Step/mm_{old}]}{Actual\ Value}$
$New\ X_step/mm = \frac{20 \times 80}{20.228} = 79.098$
$New\ Y_step/mm = \frac{20 \times 80}{20.066} = 79.737$
$New\ Z_step/mm = \frac{20 \times 400}{19.946} = 401.083$

Table 4.7: Prototype's Accuracy Evaluation Based on 20mm Cube Printed with Updated Step/mm for Each Axis.

Axis	Actual Readings (mm)	Percentage Error	Evaluation
X	20.03	$\frac{20.050 - 20}{20} \times 100\%$ $= 0.25\%$	Accepted, percentage error less than 0.3%.
	20.05		
	20.07		
	20.06		
	20.04		
	Average= 20.050		
Y	19.98	$\frac{20.016 - 20}{20} \times 100\%$ $= 0.08\%$	Accepted, percentage error less than 0.3%.
	20.03		
	20.04		
	20.02		
	20.01		
	Average= 20.016		
Z	19.98	$\frac{20 - 19.978}{20} \times 100\%$ $= 0.11\%$	Accepted, percentage error less than 0.3%.
	19.97		
	19.97		
	19.95		
	20.01		
	Average= 19.978		

4.3.2 Speed Limit of Printing Process

The extrusion limit test carried out prior to the printing speed test to prevent damages to the components of the prototype. The extrusion limit occurred in this test when hot end not able to heat up the filament consistently at a given extruder feed rate and hence resulted in sharp change in extruded diameter as shown in Figure 4.13.

The procedure of the extrusion limit test is first to direct the printer to extrude certain length of filament at different feed rate with constant hot end temperature, observe the outputs and compute the safe parameters based on extrusion limit obtained as shown in Table 4.8. (Pirring, 2019)

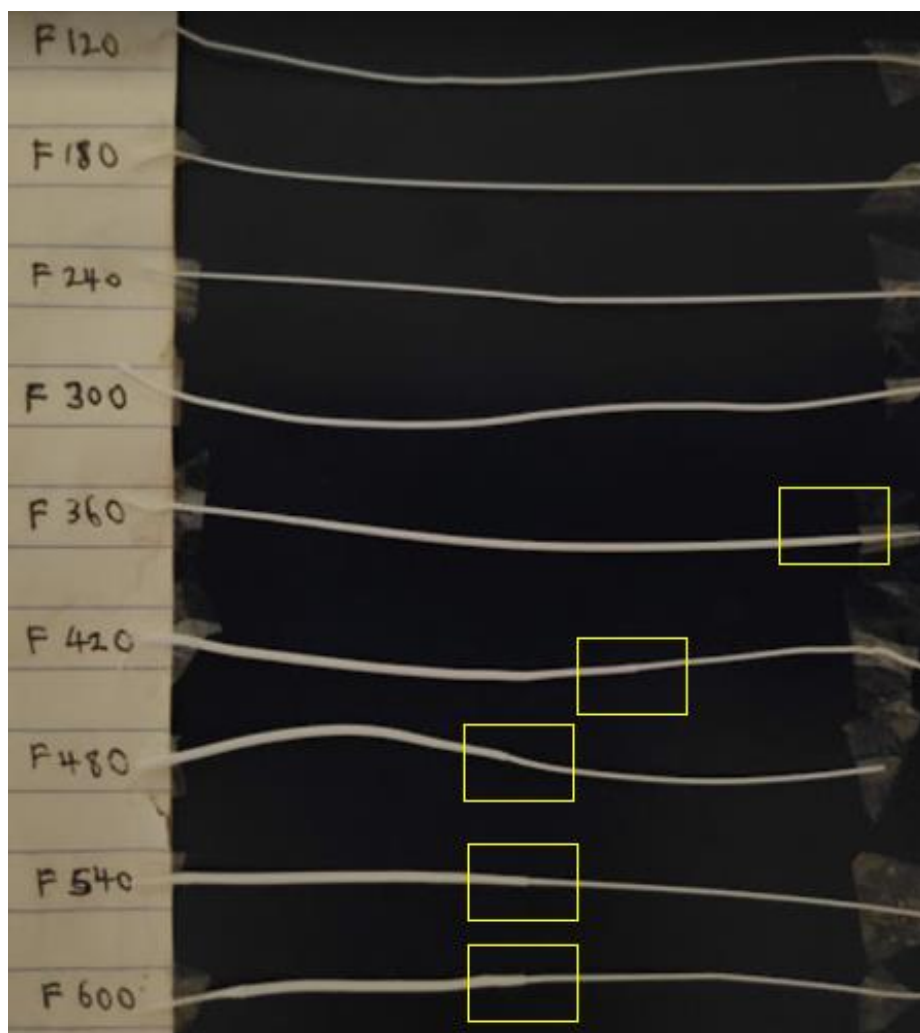






Figure 4.13: Results of Extrusion Limit Test with Extruder's Feed Rate from 120mm/min to 600mm/min and Hot End at 200 Degree Celsius.

Table 4.8: Computation of Prototype's Speed Limit and Safe Extrusion Limit.

No.	Computation Steps
1	Constant Parameters, $\text{Nozzle Diameter} = 0.4\text{mm}$ $\text{Filament Diameter} = 1.75\text{mm}$
2	Observation from Experiment, $\text{Extrusion Limit} = \frac{300\text{mm}}{\text{min}} = \frac{5\text{mm}}{\text{s}}$
3	Interpretation of Experimental Result, $\text{Volume per unit extrusion} = \pi \left(\frac{1.75\text{mm}}{2} \right)^2 \times 1 = \frac{2.4053\text{mm}^3}{\text{mm}}$ $\text{Safe Extrusion Rate} = \frac{5\text{mm}}{\text{s}} \times \frac{2.4053\text{mm}^3}{\text{mm}} = \frac{12.0265\text{mm}^3}{\text{s}}$ (Pirringer, 2019)
4	Take 0.2mm layer height printing with 0.4mm extrusion width, $\text{Maximum XY Axes Movement Speed}$ $= \frac{12.0265\text{mm}^3}{\text{s}} \times (0.2\text{mm} \times 0.4\text{mm}) = \frac{150.33\text{mm}}{\text{s}}$ (Laws, n.d.)

In order to avoid any destruction on machine parts as there are no extra parts available for replacement, the speed for axes movement is suppressed below 120mm/s to leave a safety buffer during the printing speed test. The 20mm cube printed with different base feed rate used to inspect on the ability of the prototype developed in higher speed printing above 80mm/s as shown in Table 4.9. From the results, the prints shown minor surface defect due to vibration as printing speed increases, but there are no major defects occurred since all of the prints look similar.

Table 4.9: Printing Speed Test for Prototype.

Printing Speed	Printed Output	Description
80 mm/s		Good appearance.
90 mm/s		Good appearance.
100 mm/s		Acceptable quality with some minor surface defects.
110 mm/s		Acceptable quality with slightly more minor surface defects.

In addition, for a 20mm distance travelled, with acceleration of 1500mm/s^2 , around 62.5% of the total length are travelled with top speed of 110mm/s as shown in Figure 4.14. Therefore, the further testing on the higher speed with the 20mm cube is insufficient and not performed in this test.

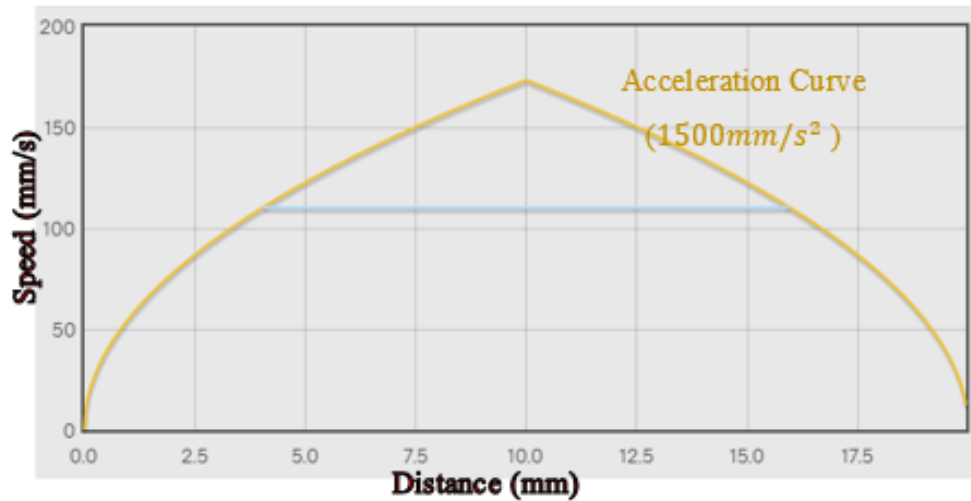


Figure 4.14: Speed Versus Distance with Superimposed Acceleration Curve (1500mm/s^2). (Prusa Research a.s., 2021)

4.4 Problem Encountered and Improvement

First problem encountered was the pressing of M3 knurled threaded insert into the printed part with heat supply from the soldering iron will cause the molten thermoplastic to clog the hole where the bolt need to pass through. The clog is very difficult to remove without M3 thread tap and drill bit. The improvement made on assembly process by using a long bolt (M3x 45mm) to screw on the knurled insert from the opposite side, then hot press the insert together with bolt into place as shown in Figure 4.15. The hole will not be clogged as the molten thermoplastic cannot fill the hole where preoccupied by the bolt.



Figure 4.15: Improvement Made in Inserting Threaded Insert into Printed Part.

Secondly, communication of 3D printer to the terminal software was unsuccessful. I was not able to connect printer with Pronterface via USB connection to perform G-code input to printer for calibration and tuning. After a series of trial and error and confirmed on baud rate to be same as firmware setting (115200). The problem solved by modify serial port declaration from -1 to 1 in the firmware and reboot the printer (-1 is the setting given initially by the Makerbase motherboard datasheet) as summarised in Figure 4.16.

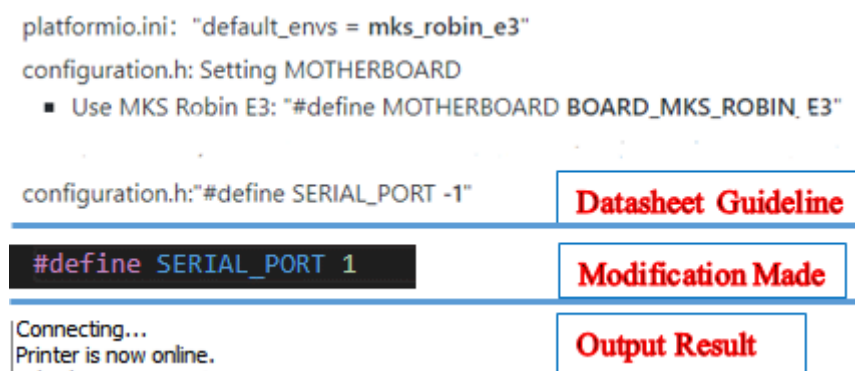


Figure 4.16: Modification Made on Serial Port Declaration from Datasheet for Successful Connection with Pronterface.

Thirdly, the enclosure has been designed and produced for a more stable ambient temperature during printing to minimize warping issue and other related quality issues. Besides, the enclosure reduces the cleaning time by protecting the internal parts from dust deposition. The enclosure with frameless glass door was developed from glass pieces, plywood and some printed parts from the prototype as shown in Figure 4.17. The enclosure developed have the openings for the control panel for on time tuning during printing without the need to open the door. Other than that, enclosure designed have the openings at the sides for air flow of cooling fans and for ease of filament spool change. [Refer to Appendix F for CAD drawings of enclosure parts designed.]

On the other hand, proper bed levelling is required for a perfect first layer adhesion in FDM 3D printing which will affect the subsequent layer addition. Even the slight imperfection as small as 0.1mm of the heated bed will affect the printed result. The improvement made was adding a 3mm thick glass piece on the heated bed for perfect flatness as shown in Figure 4.17. Other than

the corner levelling, auto linear bed levelling feature allowed with the nozzle as probe without the need of other levelling sensor by the firmware set.

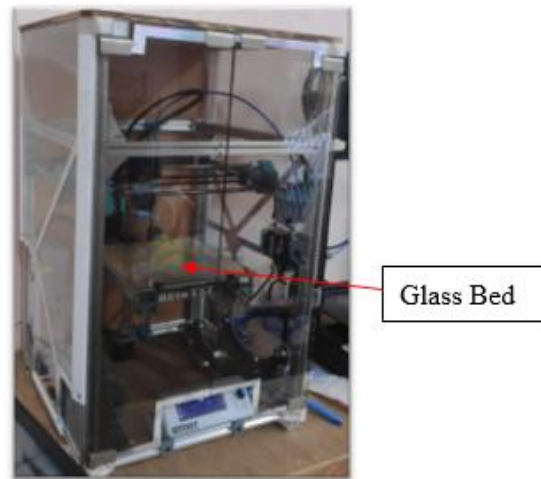


Figure 4.17: Prototype Developed with Enclosure and Glass Bed.

Last but not least, the cable management has been carried out for a tidier outlook, easier identification and prevent entanglements. Different size of clips has been printed with the prototype to fix the cables on specific location of the main frame as shown in Figure 4.18. Besides, the Teflon tube and cables for print head has been guided at around 40cm from the end to the side of the main frame to avoid hitting the printed part and moving axes.

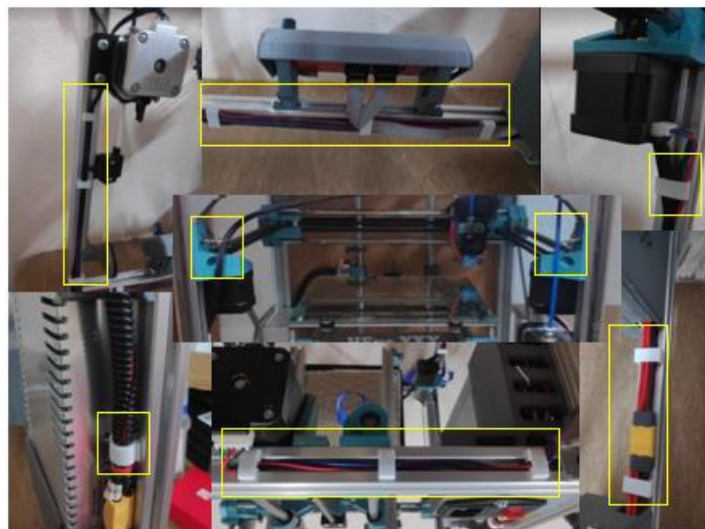


Figure 4.18: Cable Management of Prototype.

CHAPTER 5

CONCLUSION

5.1 Conclusion

The aim and 4 objectives of this project have been achieved. The small-scale coreXY FDM 3D printer developed spent approximately MYR 700.00 for all the materials and components used. The printing speed achievable by the prototype developed without much effect on printing quality is about 100mm/s.

The maximum printing speed of the prototype according to extrusion limit test for PLA under 0.2mm layer height at 200°C hot end temperature is 150mm/s. In short, the prototype can be claimed as a 3D printer in mid-speed ($\leq 100\text{mm/s}$) category and highly possible to achieve the rapid ($\leq 150\text{mm/s}$) category according to Joel's categorization in term of printing speed with some tunings and improvements. Besides, the prototype is reliable with high printing accuracy where the percentage error for each axis is less than 0.3%. The high accuracy in printed products was expected as the imitated-handwriting machine utilised the coreXY structure developed by Guan-Han Zhu and Jin-Shyan Lee in 2019 with only negligible position error.

This study showed the importance in popularization of 3D printing technology with the improvements made for more efficient printing by the low-cost 3D printer. As observed from the Graph A-4 in Appendix A, cost of entry is the main consideration for the public to adopt 3D printing. From Graph A-5, most people felt that the top challenges for using 3D printers is the quality control. Thus, it is important to enhance the printing speed and yet maintaining the print quality and keep the price of the 3D printer within a typical budget which have been accomplished in this study.

Last but not least, this study can contribute to the fourth industrial revolution in term of preparing more talents with creative and innovative ideas. For instances, with the wide adoption of 3D printing technology, product realization or prototyping process being economic and more accessible by the public which will assist them to preserve their creativity and innovation for the improvements in ample of fields.

5.2 Limitation of The Prototype

The prototype developed has a few limitations including the maximum hot end temperature limited at $275^{\circ}C$ in the firmware setting and thus the filament type can be used with the prototype is limited. Furthermore, the heat break with PTFE insert will limit the temperature allowed as the PTFE tube will degrade at high temperature ($> 250^{\circ}C$).

Besides, single Z structure for the heated bed of the prototype is less rigid because the loads only supported on a single side as illustrated in Figure 5.1. The heated bed tends to be tilted slightly with imperfect tightening of fasteners especially with the glass bed added. The corner levelling might need to be repeated when adding new surface material although with the same thickness.

Other than that, the Z homing location is determined by the activation of optical end stop and this is depending on the position of the end stop flag on the linear rod as demonstrated in Figure 5.1. The crashing between heated bed and print head during homing will occur if the flag not set properly. For the universal spool holder designed as shown in Figure 5.1, the adjustable piece need to adjust in position to fit different spool width. This adjustment needed is considerably inconvenient where it requires the manual fixing of the adjustable piece.

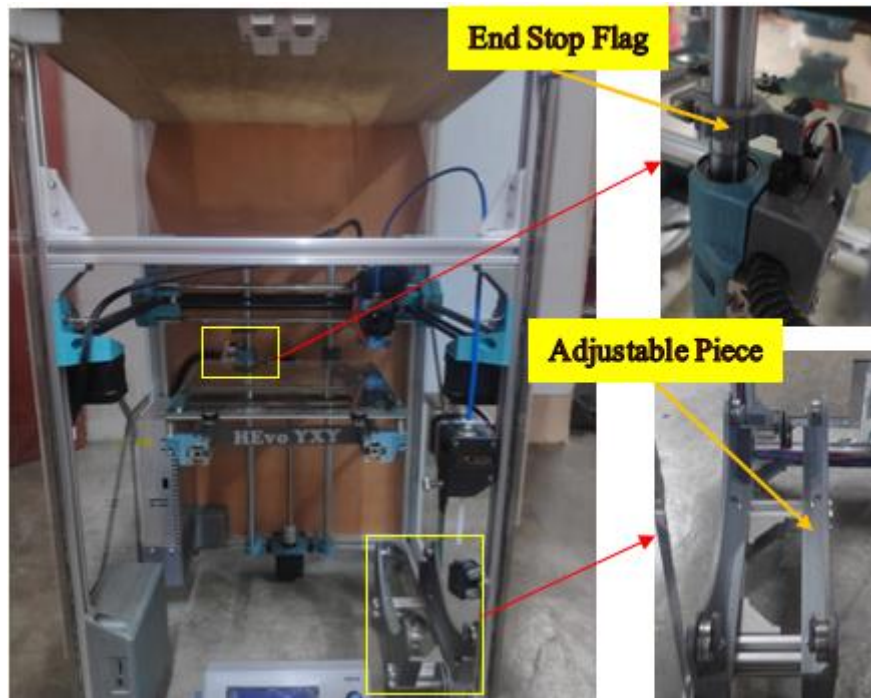


Figure 5.1: Detail Views of Non-contact Optical End Stop and Universal Spool Holder Designed.

5.3 Recommendation for Future Work

First and foremost, there are still gap for further test in term of printing speed since the test carried out stop at 110mm/s but the maximum speed allowed for this design is 150mm/s. Other than PLA, there are many different feedstock pending for test for the optimum print setting with the prototype. With the dual drive BMG extruder used, flexible filament, TPU can be another field to explore with the prototype as it is more difficult in printing as compare to non-flexible filaments.

In addition, the E3D hot end components used in this prototype is upgradable, where the 40W heating cartridge, 3950 NTC 100k thermistor and heat break with PTFE insert can directly replace with 50W heating cartridge, high temperature (up to 350⁰C) NTC 100k thermistor and all metal heat break respectively to prepare for the high temperature (~300⁰C) printing and exploration of stronger printed part like PC. However, the maximum hot end temperature and components used need to declare and change accordingly in the firmware.

Besides, the MKS Robin E3 motherboard used have another port for second Z axis stepper motor readily, thus the design on the dual Z motors structure of prototype can be proceeded to give more stable printing platform provided cost is not the limiting factor.

On the other hand, the x-gantry of the prototype developed can be lighten by using the hollow shaft or other lighter material to replace the 8mm steel rods as an effort in vibration reduction and enhance the life cycle of the timing belt. Furthermore, size of the brass bushing other that inner diameter is recommended to be reduced for the weight reduction for the gantry.

Last but not least, the recommendation on future work is to use the branded filament in testing and parts printing for consistent filament properties, reliable test results and minimize printing errors like nozzle clog due to impurities.

REFERENCES

2014. *3D Printer electronics design*. Bachelor Thesis. Available at:
 <https://upcommons.upc.edu/bitstream/handle/2099.1/24670/Bachelor_Thesis_3D_Printer_Electronics_Design.pdf?sequence=1&isAllowed=y> [Accessed 19 August 2020].

3DSourced, 2020. *Fused deposition modeling: Everything you need to know about FDM 3D printing*. [online] Available at:
 <<https://3dsourced.com/guides/fused-deposition-modeling-fdm/>> [Accessed 14 August 2020].

Abeykoon, C., Sri-Amphorn, P. and Fernando, A., 2020. Optimization of fused deposition modeling parameters for improved PLA and ABS 3D printed structures. *International Journal of Lightweight Materials and Manufacture*, [e-journal] 3(3), pp. 284–297. <http://dx.doi.org/10.1016/j.ijlmm.2020.03.003>.

Alex, M., 2017. *The 4 types of FFF / FDM 3D printer explained (Cartesian, Delta, Polar)*. [online] Available at: <<https://www.3dnatives.com/en/four-types-fdm-3d-printers140620174/#!>> [Accessed 11 July 2020].

Alsoufi, M. S. and El-Sayed, A., 2017. Warping deformation of desktop 3D printed parts manufactured by open source fused deposition modeling (FDM) system. *International Journal of Mechanical & Mechatronics Engineering*, 17(4), pp. 7–16.

n.d. *Arduino mega 2560 datasheet*. [online] Available at:
 <<http://eprints.polsri.ac.id/4598/8/File%20VIII%20%28Lampiran%29.pdf>> [Accessed 31 August 2020].

Avdeev, A. R., Shvets, A. A. and Torubarov, I. S., 2020. *Investigation of kinematics of 3D printer print head moving systems. Proceedings of the 5th International Conference on Industrial Engineering ICIE 2019*: Springer.

Awalt, A., 2020. *Basics of NTC and PTC thermistors*. [online] Available at:
 <<https://www.digikey.com/en/blog/basics-of-ntc-and-ptc-thermistors>> [Accessed 2 September 2020].

- Beaudoin, A., Boulanger, R. and DiPersio, J., 2017. *Multihead 3D printer*. Bachelor dissertation. Available at: <<https://digitalcommons.wpi.edu/mqp-all/10>>.
- Bloch, H. P. and Geitner, F. K., 2019. *Protecting machinery parts against the loss of surface*. 4th ed.
- Boichut, P., 2019. *Direct drive, bowden, remote motor, the differences*. [online] Available at: <<http://www.spiderbot.eu/direct-drive-bowden-remote-motor-the-differences/?lang=en>> [Accessed 22 August 2020].
- Caenn, R., Darley, H.C.H. and Gray, G. R., 2017. *Drilling fluid components*. 7th ed.
- Carlota, V., 2019. *Top 10 best slicer software for all levels*. [online] Available at: <<https://www.3dnatives.com/en/top-10-slicer-software-200520194/>> [Accessed 29 August 2020].
- Chen, M. Y., Skewes, J., Daley, R., Woodruff, M. A. and Rukin, N. J., 2020. Three-dimensional printing versus conventional machining in the creation of a meatal urethral dilator: development and mechanical testing. *Biomedical engineering online*, [e-journal] 19(1), p. 55–55.
<http://dx.doi.org/10.1186/s12938-020-00799-8>.
- Clarke, C., 2017. *Bondtech releases lightweight BMG extruder with “best force to weight” ratio on the market*. [online] Available at: <<https://3dprintingindustry.com/news/bondtech-releases-lightweight-bmg-extruder-best-force-weight-ratio-market-108969/>> [Accessed 31 August 2020].
- Coates, E., 2020. *Switched mode power supplies*. [online] Available at: <<https://learnabout-electronics.org/PSU/psu30.php>> [Accessed 31 August 2020].
- Considine, D. M. and Considine, G. D., 1986. *Standard Handbook of Industrial Automation: Stepper Motors and Controls*. Boston MA: Springer.
- DesignTech Systems, n.d. *How fused deposition modeling (FDM) printers work*. [online] Available at:

<<https://www.designtechproducts.com/index.php?/articles/working-fdm-3d-printers>> [Accessed 14 August 2020].

Ensinger, 2020. *POM - Acetal polyoxymethylene*. [online] Available at: <<https://www.ensingerplastics.com/en/shapes/engineering-plastics/pom-acetal>> [Accessed 30 August 2020].

Filament2Print, 2020. *Direct extrusion and bowden systems*. [online] Available at: <https://filament2print.com/gb/blog/94_bowden-direct-extrusion.html> [Accessed 22 August 2020].

Gabe, S., 2020. *V6 Assembly*. [online] Available at: <<https://e3d-online.dozuki.com/Guide/V6+Assembly/6?lang=en>> [Accessed 31 August 2020].

Gordeev, E. G., Galushko, A. S. and Ananikov, V. P., 2018. Improvement of quality of 3D printed objects by elimination of microscopic structural defects in fused deposition modeling. *PloS one*, [e-journal] 13(6), 1-19. <http://dx.doi.org/10.1371/journal.pone.0198370>.

Grames, E., 2018. *3D Printer power supply: How to choose the right one?* [online] Available at: <<https://all3dp.com/2/3d-printer-power-supply-how-to-choose-the-right-one/>> [Accessed 31 August 2020].

Grames, E., 2019. *CoreXY 3D printer: why it makes a difference*. [online] Available at: <<https://all3dp.com/2/corexy-3d-printer-is-it-worth-buying/>> [Accessed 12 July 2020].

Handson Technology, 2014. *17HS4401S Motor datasheet*. [online] Available at: <<https://datasheetpdf.com/pdf/1310364/Handson/17HS4401S/1>> [Accessed 30 August 2020].

Harris, K., 2020. *How to select the right 3D printing motherboard?* [online] Available at: <<https://prototypeinfo.com/right-3d-printing-motherboard-selecting/>> [Accessed 24 August 2020].

Häußge, G., 2020. *OctoPrint*. [online] Available at: <<https://octoprint.org/>> [Accessed 27 August 2020].

Heath, J., 2017. *Pulse width modulation: What is it and how does it work?* [online] Available at: <<https://www.analogictips.com/pulse-width-modulation-pwm/>> [Accessed 24 August 2020].

Higgs, B., 2018. *3D Print overhangs and how to deal with them.* [online] Available at: <<https://medium.com/bravovictornovember/3d-print-overhangs-and-how-to-deal-with-them-9eed6a7bcb5d>> [Accessed 14 August 2020].

Hoge, G., n.d. *CoreXY gantry system.* [online] Available at: <<https://greghoge.com/portfolio/large-format-3d-printer/corexy-gantry-system/>> [Accessed 14 August 2020].

Horne, R. and Hausman, K. K., n.d.a. *How to calibrate your 3D printer?* [online] Available at: <<https://www.dummies.com/computers/pcs/printers/calibrate-3d-printer/>> [Accessed 27 August 2020].

Horne, R. and Hausman, K. K., n.d.b. *Using a web-based 3D printing interface.* [online] Available at: <<https://www.dummies.com/computers/pcs/printers/using-web-based-3d-printing-interface/>> [Accessed 27 August 2020].

Horvath, J., 2014. *Mastering 3D printing: A brief history of 3D printing:* Apress, Berkeley, CA.

Inductive Automation, 2018. *What is HMI?* [online] Available at: <[https://www.inductiveautomation.com/resources/article/what-is-hmi#:~:text=A%20Human%2DMachine%20Interface%20\(HMI,context%20of%20an%20industrial%20process.](https://www.inductiveautomation.com/resources/article/what-is-hmi#:~:text=A%20Human%2DMachine%20Interface%20(HMI,context%20of%20an%20industrial%20process.)> [Accessed 27 August 2020].

International Organization for Standardization/ American Society for Testing and Materials, 2015. *Standard . International standard ISO/ASTM 52900.* Switzerland.

Interpower, 2020. *Designing products with IEC 60320 C13 connectors and C14 inlets.* [online] Available at: <<https://www.interpower.com/ic/InfoPower/iec-60320-c13-c14.html>> [Accessed 31 August 2020].

Joel, C., 2020. *3D printing: How to obtain the best possible speeds*. [online] Available at: <<https://www.3dprintersonlinestore.com/how-to-obtain-best-3d-printing-speed>> [Accessed 24 July 2020].

John, X.J.Z. and Hoshino, K., 2019. *Molecular sensors and nanodevices: Mechanical transducers: Cantilevers, acoustic wave sensors, and thermal sensors*. 2nd ed.

Jones, M., 2019. *3D Printer firmware: Which to choose & how to change it?* [online] Available at: <<https://all3dp.com/2/3d-printer-firmware-which-to-choose-and-how-to-change-it/>> [Accessed 29 August 2020].

Kalender, M., Bozkurt, Y., Ersoy, S. and Salman, S., 2020. Product development by additive manufacturing and 3D printer technology in aerospace industry. *Journal of aeronautics and space technologies*, 13(1), pp. 129–138.

Keyestudio, 2019. *Ks0282 Keyestudio 3D MKS Gen V1.4 Printer Motherboard Control Board (Black & Environmental-protection)*. [online] Available at: <[https://wiki.keyestudio.com/Ks0282_Keyestudio_3D_MKS_Gen_V1.4_Printer_Motherboard_Control_Board\(Black_%26Environmental-protection\)](https://wiki.keyestudio.com/Ks0282_Keyestudio_3D_MKS_Gen_V1.4_Printer_Motherboard_Control_Board(Black_%26Environmental-protection))>.

Landry, T., 2016. *Extruders 101: A crash course on an essential component of your 3D printer*. [online] Available at: <<https://www.matterhackers.com/articles/extruders-101:-a-crash-course-on-an-essential-component-of-your-3d-printer>> [Accessed 23 August 2020].

Laws, M., n.d. *Teaching Tech 3D Printer Calibration*. [online] Available at: <<https://teachingtechyt.github.io/calibration.html>>.

Lyubomirov, S. Y., Nedeva, M. I. and Bundeve, M. K., 2015. Software management of 3D printer. *Annual Journal of Electronics*, pp. 96–99.

Makerbase, 2020. *Robin_E3*. [online] Available at: <https://github.com/makerbase-mks/MKS-Robin-E3-E3D/wiki/Robin_E3> [Accessed 31 August 2020].

Mardis, N. J., 2018. Emerging technology and applications of 3D printing in the medical field. *Science of medicine*, pp. 368–373.

- Marlin, 2020. *Configuring Marlin*. [online] Available at:
<<https://marlinfw.org/docs/configuration/configuration.html#thermal-settings>>
[Accessed 3 September 2020].
- Mika, Y., 2019. *5 Best 3D printer controller boards*. [online] Available at:
<<https://all3dp.com/2/5-fantastic-3d-printer-controller-boards/>> [Accessed 24 August 2020].
- Mike, 2014. *Steppers with encoders: When open-loop control is not enough*. [online] Available at:
<<https://www.phidgets.com/?view=articles&article=SteppersWithEncoders>>
[Accessed 19 August 2020].
- Misiolek, W. Z. and Kelly, R. M., 2005. Extrusion of aluminum alloys. *ASM Handbook*, [e-journal] 14, pp. 522–527.
<http://dx.doi.org/10.1361/asmhba0004015>.
- Moreau, C., 2019. *The state of 3D printing*. The data you need to understand the 3D printing world. 2019th ed.
<https://cdn2.hubspot.net/hubfs/5154612/downloads/Sculpteo_The%20State%20of%203D%20Printing_2019.pdf> [Accessed 9 July 2020].
- Morse, R., 2019. *Printrun – A beginner’s tutorial*. [online] Available at:
<<https://all3dp.com/2/printrun-a-beginner-s-tutorial/>> [Accessed 27 August 2020].
- Moyer, I. E., 2012. *CoreXY*. [online] Available at:
<<https://web.archive.org/web/20140718064219/http://www.corexy.com/>>
[Accessed 14 August 2020].
- Muzammal, H., Mehedi, H. J., Mehedi, H. and Hasnat, K., 2019. *Design and implementation of an FDM based 3D printer*. *International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)*, July 11, 2019: IEEE.
- National Institute of Standards and Technology, 2000. *Standard . The NIST RS274NGC Interpreter - Version 3*. Maryland.
- Nedelkovski, D., 2019. *How to control a stepper motor with A4988 driver and arduino*. [online] Available at:

<<https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/>> [Accessed 25 August 2020].

Nutma, M., 2019. *3 types of manufacturing - additive, subtractive, and forming*. [online] Available at:

<<https://www.techzone360.com/topics/techzone/articles/2019/09/09/443185-3-types-manufacturing-additive-subtractive-forming.htm>> [Accessed 11 June 2020].

Özüdoğru, A. G., Ergün, E. and Ammari, D., 2018. How industry 4.0 changes bussiness: A commercial perspective. *International Journal of Commerce and Finance*, 4(1), pp. 84–95.

Patel, M. A., Patel, A. R., Vyas, D. R. and Patel, K. M., 2009. Use of PWM techniques for power quality improvement. *International Journal of Recent Trends in Engineering*, 1(4), pp. 99–102.

Pirringer, M., 2019. *Dialing in a Filament and Specifying the Max Volumetric E(xtrusion) Value*. [online] Available at:

<<https://grabcad.com/tutorials/dialing-in-a-filament-and-specifying-the-max-volumetric-e-xtrusion-value>>.

PlumLogix, 2019. *Industry 4.0 — what it is and how salesforce is an integral part of it all*. [online] Available at: <<https://medium.com/plumlogix-blog/industry-4-0-what-it-is-and-how-salesforce-is-an-integral-part-of-it-all-64c474761b4f>> [Accessed 12 July 2020].

Pololu Corporation, 2020. *Stepper motor drivers*. [online] Available at:

<<https://www.pololu.com/category/120/stepper-motor-drivers>> [Accessed 25 August 2020].

Prusa Research a.s., 2020a. *Extrusion stopped mid-print (heat creep)*. [online] Available at: <https://help.prusa3d.com/en/article/extrusion-stopped-mid-print_1948> [Accessed 22 August 2020].

Prusa Research a.s., 2020b. *Time-line*. [online] Available at:

<<https://www.prusa3d.com/about-us/#timeline>>.

Prusa Research a.s., 2021. *Prusa Printers Blog*. [online] Available at:

<https://blog.prusaprinters.org/calculator_3416/>.

RepRap, 2016. *Bed material*. [online] Available at:

<https://reprap.org/wiki/Bed_material> [Accessed 19 August 2020].

RepRap, 2018a. *HyperCube*. [online] Available at:

<<https://reprap.org/wiki/HyperCube#:~:text=The%20HyperCube%203D%20Printer%2FCNC,your%203D%20printer%20much%20better.>> [Accessed 29 August 2020].

RepRap, 2018b. *Stepper motor*. [online] Available at:

<https://reprap.org/wiki/Stepper_motor> [Accessed 19 August 2020].

RepRap, 2018c. *Stepper motor driver*. [online] Available at:

<https://reprap.org/wiki/Stepper_motor_driver> [Accessed 22 August 2020].

RepRap, 2019a. *PID tuning*. [online] Available at:

<https://reprap.org/wiki/PID_Tuning> [Accessed 3 September 2020].

RepRap, 2019b. *RepRap discount full graphic smart controller*. [online]

Available at:

<https://reprap.org/wiki/RepRapDiscount_Full_Graphic_Smart_Controller> [Accessed 27 August 2020].

RepRap, 2020a. *Comparison of electronics*. [online] Available at:

<https://reprap.org/wiki/Comparison_of_Electronics> [Accessed 24 August 2020].

RepRap, 2020b. *CoreXY*. [online] Available at:

<<https://reprap.org/wiki/CoreXY>> [Accessed 14 August 2020].

RepRap, 2020c. *G-code*. [online] Available at: <<https://reprap.org/wiki/G-code>>

[Accessed 29 August 2020].

RepRap, 2020d. *List of firmware*. [online] Available at:

<https://reprap.org/wiki/List_of_Firmware> [Accessed 29 August 2020].

RepRap, 2020e. *PCB Heatbed*. [online] Available at:

<https://reprap.org/wiki/PCB_Heatbed> [Accessed 31 August 2020].

Roibu, H., Popescu, D., Abagiu, M.-M. and Bizdoaca, N.-G., 2018. *Human-machine interfaces for robotic system control*. Piscataway, NJ: IEEE.

- Serdeczny, M. P., Comminal, R., Pedersen, D. B. and Spangenberg, J., 2020. Experimental and analytical study of the polymer melt flow through the hot-end in material extrusion additive manufacturing. *Additive Manufacturing*, [e-journal] 32. <http://dx.doi.org/10.1016/j.addma.2019.100997>.
- Shahrubudin, N., Lee, T. C. and Ramlan, R., 2019. An overview on 3D printing technology: technological, materials, and applications. *Procedia Manufacturing*, [e-journal] 35, pp. 1286–1296. <http://dx.doi.org/10.1016/j.promfg.2019.06.089>.
- Shen Zhen Creality 3D Technology Co., Ltd., 2019. *Ender-3*. [online] Available at: <<https://www.creality.com/goods-detail/ender-3-3d-printer>>.
- Shen Zhen Esun Industrial Co.,Ltd, 2007. *Products and services: Filament*. [online] Available at: <<http://www.esun3d.net/Products/Filament>> [Accessed 11 July 2020].
- Sidambe, A. T., 2014. Biocompatibility of advanced manufactured titanium implants-A review. *Materials (Basel)*, [e-journal] 7(12), pp. 8168–8188. <http://dx.doi.org/10.3390/ma7128168>.
- Sood, R. and Pradhan, S. K., 2020. Design and development of a low-cost open-source 3D printer and its single response optimization using polylactic acid (PLA) material. *Materials Today: Proceedings*, [e-journal] 27, pp. 2981–2991. <http://dx.doi.org/10.1016/j.matpr.2020.04.905>.
- Soon, C. F., Ramilan, M. F., Hanafi, D., Zakaria, W.N.W., Tee, K. S., Khialdin, S.B.M. and Isa, H., 2020. Development of a 3D bio-printer using CoreXY mechanism and syringe-based extrusion. *Indonesian Journal of Electrical Engineering and Computer Science*, [e-journal] 18(3), pp. 1180–1187. <http://dx.doi.org/10.11591/ijeecs.v18.i3.pp1180-1187>.
- Stăncioiu, A., 2017. The fourth industrial revolution_ "Industry 4.0". *Fiabilitate Și Durabilitate*, (1), pp. 74–78.
- Stratasys Ltd., 2020. *What is FDM Technology?* [online] Available at: <<https://www.stratasys.com/fdm-technology>> [Accessed 14 August 2020].
- Su, A. and Al'Aref, S. J., 2018. *3D Printing Applications in Cardiovascular Medicine: History of 3D Printing*.

Sunil, C. J. and Abdullah, A. S., 2015. 3D printing in aerospace and its long-term sustainability. *Virtual and Physical Prototyping*, [e-journal] 10.

<http://dx.doi.org/10.1080/17452759.2015.1111519>.

Thornton, S., 2016. *Trade-offs in choosing 8-bit vs. 16- and 32-bit architectures*. [online] Available at:

<<https://www.microcontrollertips.com/trade-offs-choosing-8-bit-vs-16-32-bit-architectures/>> [Accessed 24 August 2020].

Tofail, S. A.M., Koumoulos, E. P., Bandyopadhyay, A., Bose, S., O'Donoghue, L. and Charitidis, C., 2018. Additive manufacturing: Scientific and technological challenges, market uptake and opportunities. *Materials Today*, [e-journal] 21(1), pp. 22–37.

<http://dx.doi.org/10.1016/j.mattod.2017.07.001>.

TRINAMIC Motion Control GmbH & Co., 2019. *TMC2209 Datasheet*.

[online] Available at:

<https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC2209_Datasheet_V103.pdf> [Accessed 31 August 2020].

TRINAMIC Motion Control GmbH & Co., 2020a. *Chip-level communication*.

[online] Available at: <<https://www.trinamic.com/technology/interfaces-protocols/chip-level-communication/>> [Accessed 27 August 2020].

TRINAMIC Motion Control GmbH & Co., 2020b. *Chopper Modes*. [online]

Available at: <<https://www.trinamic.com/technology/motor-control-technology/chopper-modes/#c3044>> [Accessed 25 August 2020].

TRINAMIC Motion Control GmbH & Co., 2020c. *Microstepping*. [online]

Available at: <<https://www.trinamic.com/technology/motor-control-technology/microstepping/https://www.trinamic.com/technology/motor-control-technology/microstepping/>> [Accessed 25 August 2020].

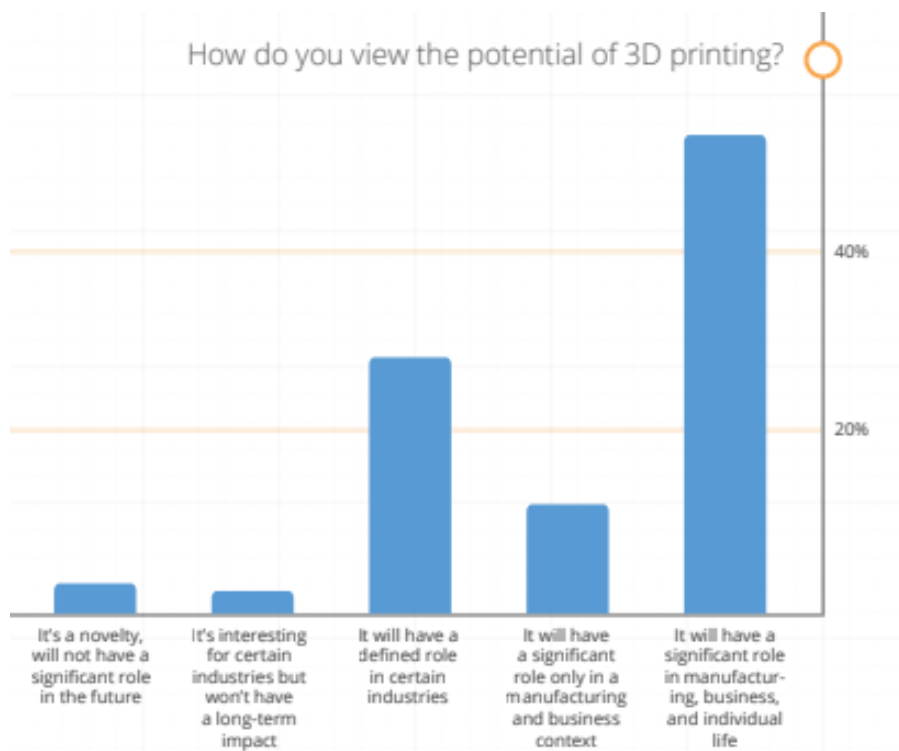
TRINAMIC Motion Control GmbH & Co., 2020d. *Sensorless StallGuard™ and CoolStep™ technologies*. [online] Available at:

<<https://www.trinamic.com/technology/motor-control-technology/stallguard-and-coolstep/>> [Accessed 25 August 2020].

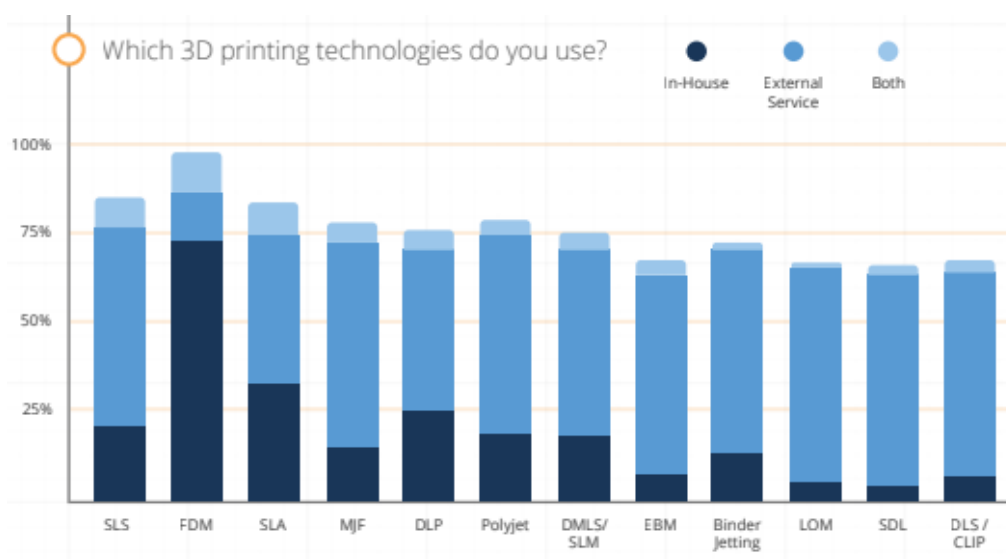
- Vincent, H. Y. S., 2018. *Development of 3D printing framework using PC-based PLC*. Bachelor's dissertation. University Tunku Abdul Rahman.
- Wohlers, T. and Gornet, T., 2014. *History of additive manufacturing*.
- Wu, X., Wang, X. and He, G., 2020. *A fuzzy self-tuning temperature PID control algorithms for 3D bio-printing temperature control system*. 2020 Chinese Control And Decision Conference (CCDC 2020).
- Yanev, K. and Seguin, G., 2020. *Printrun 2.X*. [online] Available at: <<https://github.com/kliment/Printrun/blob/master/README.md>> [Accessed 27 August 2020].
- Zhou, Y., 2019. *Research on development and problems of 3D printing technology under intelligent background*. 2019 12th International Conference on Intelligent Computation Technology and Automation (ICICTA). Guang Xi, China: IEEE.
- Zhu, G.-H. and Lee, J.-S., 2019. *Development of Imitated-Handwriting Systems Using PLC-Controlled CoreXY Mechanisms*. 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA). Xi'an, China, 19-21 June 2019: IEEE.

APPENDICES

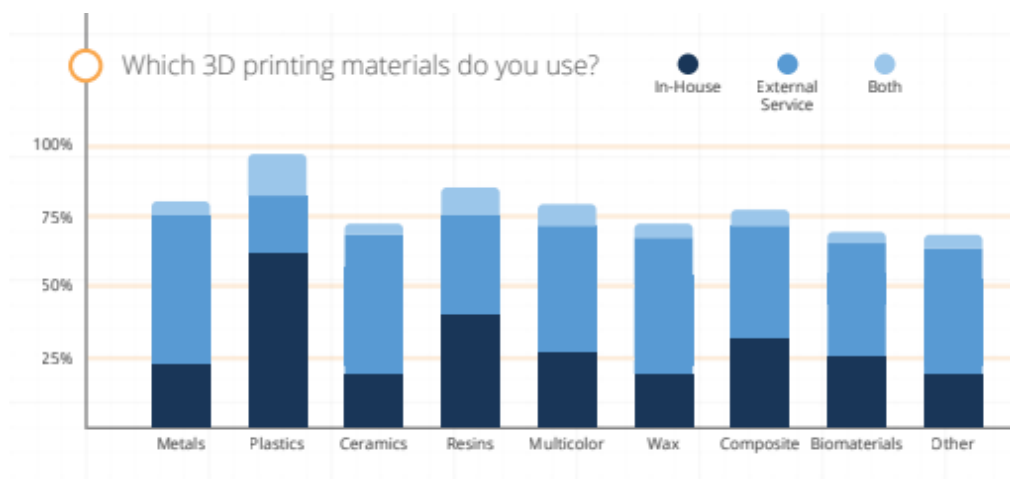
APPENDIX A: Graphs of Survey Result



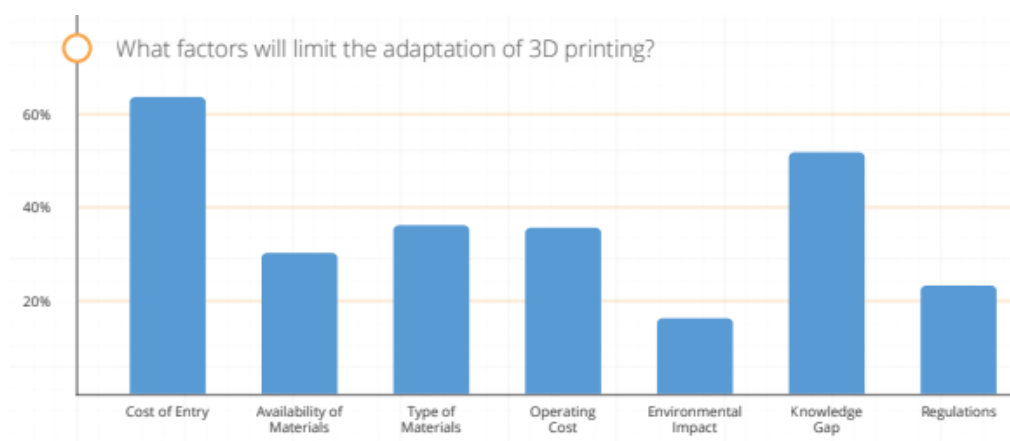
Graph A-1: “How Do You View the Potential of 3D Printing?” (Moreau, 2019)



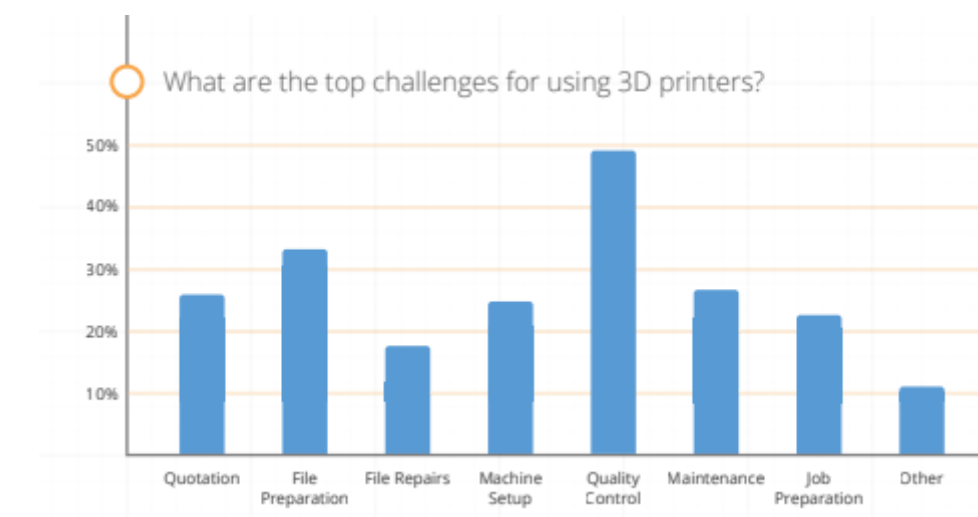
Graph A-2: “Which 3D Printing Technologies Do You Use?” (Moreau, 2019)



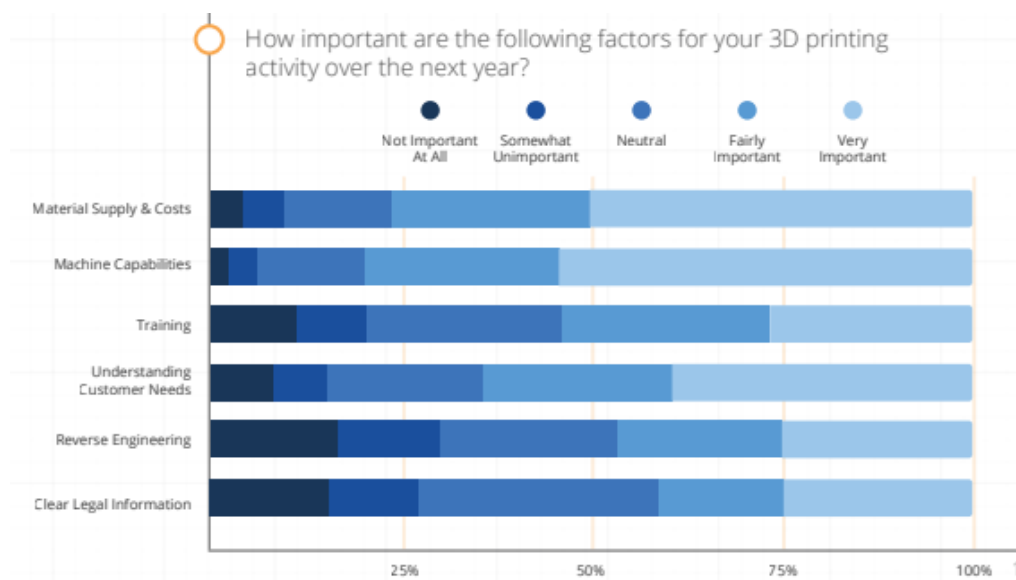
Graph A-3: “Which 3D Printing Materials Do You Use?” (Moreau, 2019)



Graph A-4: “What Factors Will Limit the Adaptation of 3D Printing?” (Moreau, 2019)



Graph A-5: “What are the Top Challenges for Using 3D Printers?” (Moreau, 2019)



Graph A-6: “How Important are the Following Factors for Your 3D Printing Activity Over the Next Year?” (Moreau, 2019)

APPENDIX B: Flowchart

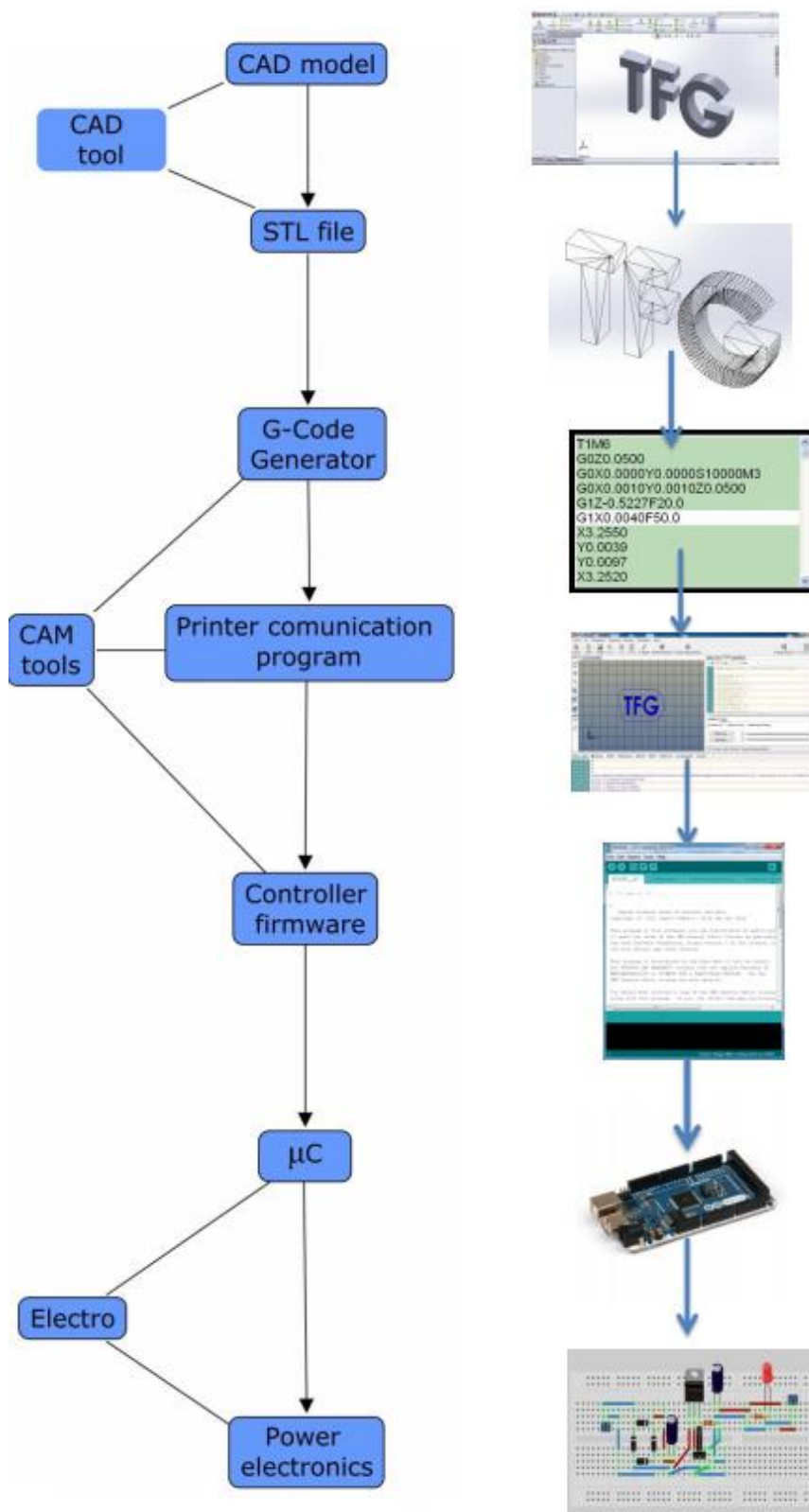


Figure B-1: Flowchart of 3D Printing Process. (*3D Printer electronics design*, 2014)

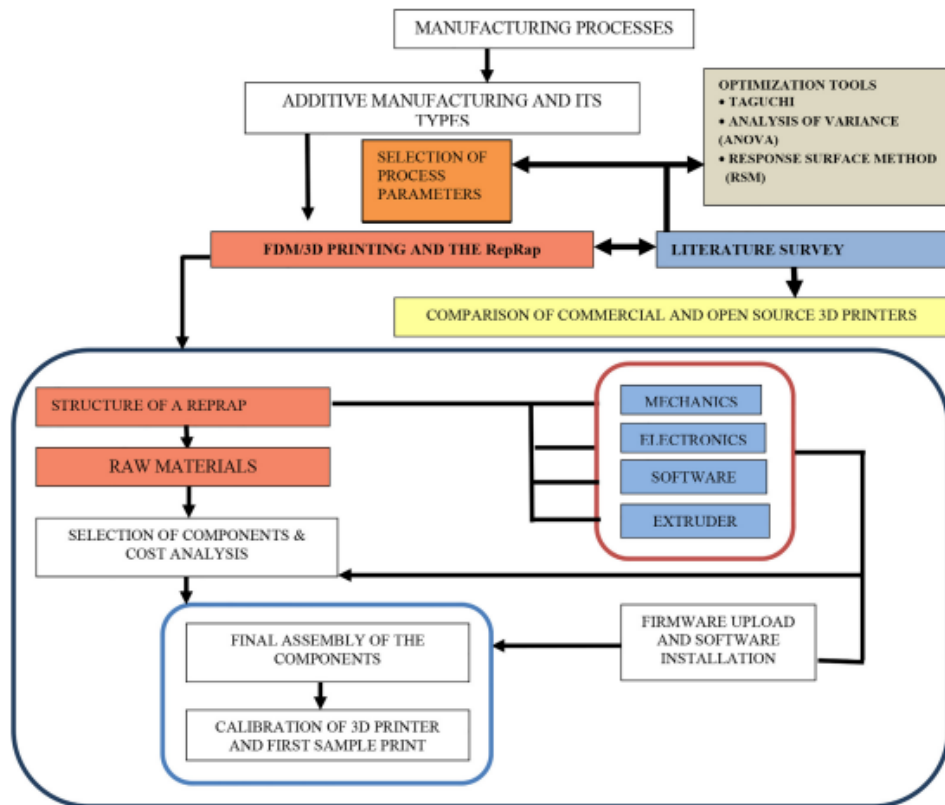


Figure B-2: Flowchart of 3D Printer Development. (Sood and Pradhan, 2020)

APPENDIX C: Older Assembly 3D Model (Version 1)

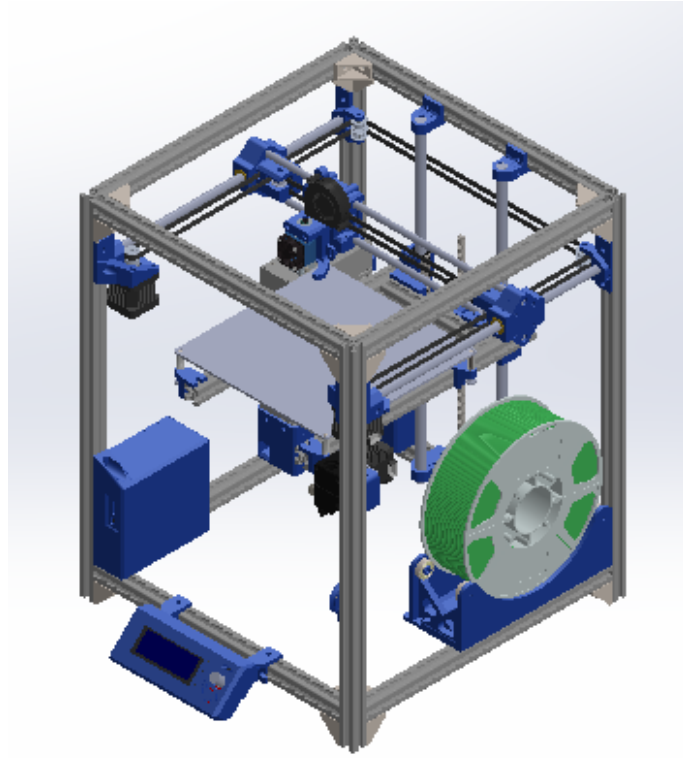


Figure C-1: Isometric View of 3D Printer Prototype Version 1.

APPENDIX D: Marlin Firmware Configuration.h

```

/**
 * Marlin 3D Printer Firmware
 * Copyright (c) 2020 MarlinFirmware [https://github.com/MarlinFirmware/Marlin]
 *
 * Based on Sprinter and grbl.
 * Copyright (c) 2011 Camiel Gubbels / Erik van der Zalm
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 */
#pragma once

/**
 * Configuration.h
 *
 * Basic settings such as:
 *
 * - Type of electronics
 * - Type of temperature sensor
 * - Printer geometry
 * - Endstop configuration
 * - LCD controller
 * - Extra features
 *
 * Advanced settings can be found in Configuration_adv.h
 */
#define CONFIGURATION_H_VERSION 020007

//=====
//===== Getting Started =====
//=====

/**
 * Here are some standard links for getting your machine calibrated:
 *
 * https://reprap.org/wiki/Calibration
 * https://youtu.be/wAL9d7FgInk
 * http://calculator.josefprusa.cz
 * https://reprap.org/wiki/Triffid_Hunter%27s_Calibration_Guide
 * https://www.thingiverse.com/thing:5573
 * https://sites.google.com/site/repraplogphase/calibration-of-your-reprap
 * https://www.thingiverse.com/thing:298812
 */

//=====
//===== DELTA Printer
//=====

```

```

//=====
=====
// For a Delta printer start with one of the configuration files in the
// config/examples/delta directory and customize for your machine.
//

//=====
=====
//===== SCARA Printer
=====
//=====
=====
// For a SCARA printer start with the configuration files in
// config/examples/SCARA and customize for your machine.
//

// @section info

// Author info of this build printed to the host during boot and M115
#define STRING_CONFIG_H_AUTHOR "(Yeoh Xing Yuan, default config)" // Who made the
changes.
// #define CUSTOM_VERSION_FILE Version.h // Path from the root directory (no quotes)

/**
 * *** VENDORS PLEASE READ ***
 *
 * Marlin allows you to add a custom boot image for Graphical LCDs.
 * With this option Marlin will first show your custom screen followed
 * by the standard Marlin logo with version number and web URL.
 *
 * We encourage you to take advantage of this new feature and we also
 * respectfully request that you retain the unmodified Marlin boot screen.
 */

// Show the Marlin bootscreen on startup. ** ENABLE FOR PRODUCTION **
#define SHOW_BOOTSCREEN

// Show the bitmap in Marlin/_Bootscreen.h on startup.
// #define SHOW_CUSTOM_BOOTSCREEN

// Show the bitmap in Marlin/_Statusscreen.h on the status screen.
// #define CUSTOM_STATUS_SCREEN_IMAGE

// @section machine

/**
 * Select the serial port on the board to use for communication with the host.
 * This allows the connection of wireless adapters (for instance) to non-default port pins.
 * Serial port -1 is the USB emulated serial port, if available.
 * Note: The first serial port (-1 or 0) will always be used by the Arduino bootloader.
 *
 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
 */
#define SERIAL_PORT 1

/**
 * Select a secondary serial port on the board to use for communication with the host.
 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
 */
// #define SERIAL_PORT_2 -1

/**
 * This setting determines the communication speed of the printer.
 *
 * 250000 works in most cases, but you might try a lower speed if
 * you commonly experience drop-outs during host printing.
 * You may try up to 1000000 to speed up SD file transfer.

```

```

*
* :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
*/
#define BAUDRATE 115200

// Enable the Bluetooth serial interface on AT90USB devices
// #define BLUETOOTH

// Choose the name from boards.h that matches your setup
#ifndef MOTHERBOARD
#define MOTHERBOARD BOARD_MKS_ROBIN_E3
#endif

// Name displayed in the LCD "Ready" message and Info menu
#define CUSTOM_MACHINE_NAME "HEvo YXY"

// Printer's unique ID, used by some programs to differentiate between machines.
// Choose your own or use a service like https://www.uuidgenerator.net/version4
// #define MACHINE_UUID "00000000-0000-0000-0000-000000000000"

// @section extruder

// This defines the number of extruders
// :[0, 1, 2, 3, 4, 5, 6, 7, 8]
#define EXTRUDERS 1

// Generally expected filament diameter (1.75, 2.85, 3.0, ...). Used for Volumetric, Filament
// Width Sensor, etc.
#define DEFAULT_NOMINAL_FILAMENT_DIA 1.75

// For Cyclops or any "multi-extruder" that shares a single nozzle.
// #define SINGLENOZZLE

// Save and restore temperature and fan speed on tool-change.
// Set standby for the unselected tool with M104/106/109 T...
// if ENABLED(SINGLENOZZLE)
//   #define SINGLENOZZLE_STANDBY_TEMP
//   #define SINGLENOZZLE_STANDBY_FAN
// endif

/**
 * Průša MK2 Single Nozzle Multi-Material Multiplexer, and variants.
 *
 * This device allows one stepper driver on a control board to drive
 * two to eight stepper motors, one at a time, in a manner suitable
 * for extruders.
 *
 * This option only allows the multiplexer to switch on tool-change.
 * Additional options to configure custom E moves are pending.
 */
// #define MK2_MULTIPLEXER
// if ENABLED(MK2_MULTIPLEXER)
//   // Override the default DIO selector pins here, if needed.
//   // Some pins files may provide defaults for these pins.
//   #define E_MUX0_PIN 40 // Always Required
//   #define E_MUX1_PIN 42 // Needed for 3 to 8 inputs
//   #define E_MUX2_PIN 44 // Needed for 5 to 8 inputs
// endif

/**
 * Průša Multi-Material Unit v2
 *
 * Requires NOZZLE_PARK_FEATURE to park print head in case MMU unit fails.
 * Requires EXTRUDERS = 5
 *
 * For additional configuration see Configuration_adv.h
 */

```

```

##define PRUSA_MMU2

// A dual extruder that uses a single stepper motor
##define SWITCHING_EXTRUDER
#if ENABLED(SWITCHING_EXTRUDER)
  #define SWITCHING_EXTRUDER_SERVO_NR 0
  #define SWITCHING_EXTRUDER_SERVO_ANGLES { 0, 90 } // Angles for E0, E1[, E2, E3]
  #if EXTRUDERS > 3
    #define SWITCHING_EXTRUDER_E23_SERVO_NR 1
  #endif
#endif

// A dual-nozzle that uses a servomotor to raise/lower one (or both) of the nozzles
##define SWITCHING_NOZZLE
#if ENABLED(SWITCHING_NOZZLE)
  #define SWITCHING_NOZZLE_SERVO_NR 0
  ##define SWITCHING_NOZZLE_E1_SERVO_NR 1 // If two servos are used, the index
of the second
  #define SWITCHING_NOZZLE_SERVO_ANGLES { 0, 90 } // Angles for E0, E1 (single
servo) or lowered/raised (dual servo)
#endif

/**
 * Two separate X-carriages with extruders that connect to a moving part
 * via a solenoid docking mechanism. Requires SOL1_PIN and SOL2_PIN.
 */
##define PARKING_EXTRUDER

/**
 * Two separate X-carriages with extruders that connect to a moving part
 * via a magnetic docking mechanism using movements and no solenoid
 *
 * project : https://www.thingiverse.com/thing:3080893
 * movements : https://youtu.be/0xCEiG9VS3k
 *            https://youtu.be/BqbcS0CU2FE
 */
##define MAGNETIC_PARKING_EXTRUDER

#if EITHER(PARKING_EXTRUDER, MAGNETIC_PARKING_EXTRUDER)

  #define PARKING_EXTRUDER_PARKING_X { -78, 184 } // X positions for parking the
extruders
  #define PARKING_EXTRUDER_GRAB_DISTANCE 1 // (mm) Distance to move
beyond the parking point to grab the extruder
  ##define MANUAL_SOLENOID_CONTROL // Manual control of docking solenoids
with M380 S / M381

  #if ENABLED(PARKING_EXTRUDER)

    #define PARKING_EXTRUDER_SOLENOIDS_INVERT // If enabled, the solenoid is
NOT magnetized with applied voltage
    #define PARKING_EXTRUDER_SOLENOIDS_PINS_ACTIVE LOW // LOW or HIGH pin
signal energizes the coil
    #define PARKING_EXTRUDER_SOLENOIDS_DELAY 250 // (ms) Delay for magnetic
field. No delay if 0 or not defined.
    ##define MANUAL_SOLENOID_CONTROL // Manual control of docking
solenoids with M380 S / M381

  #elif ENABLED(MAGNETIC_PARKING_EXTRUDER)

    #define MPE_FAST_SPEED 9000 // (mm/min) Speed for travel before last distance
point
    #define MPE_SLOW_SPEED 4500 // (mm/min) Speed for last distance travel to park
and couple
    #define MPE_TRAVEL_DISTANCE 10 // (mm) Last distance point
    #define MPE_COMPENSATION 0 // Offset Compensation -1 , 0 , 1 (multiplier) only
for coupling

```



```

#endif

#endif

/**
 * Switching Toolhead
 *
 * Support for swappable and dockable toolheads, such as
 * the E3D Tool Changer. Toolheads are locked with a servo.
 */
#define SWITCHING_TOOLHEAD

/**
 * Magnetic Switching Toolhead
 *
 * Support swappable and dockable toolheads with a magnetic
 * docking mechanism using movement and no servo.
 */
#define MAGNETIC_SWITCHING_TOOLHEAD

/**
 * Electromagnetic Switching Toolhead
 *
 * Parking for CoreXY / HBot kinematics.
 * Toolheads are parked at one edge and held with an electromagnet.
 * Supports more than 2 Toolheads. See https://youtu.be/JolbsAKTKf4
 */
#define ELECTROMAGNETIC_SWITCHING_TOOLHEAD

#if ANY(SWITCHING_TOOLHEAD, MAGNETIC_SWITCHING_TOOLHEAD,
ELECTROMAGNETIC_SWITCHING_TOOLHEAD)
  #define SWITCHING_TOOLHEAD_Y_POS 235 // (mm) Y position of the toolhead
  dock
  #define SWITCHING_TOOLHEAD_Y_SECURITY 10 // (mm) Security distance Y axis
  #define SWITCHING_TOOLHEAD_Y_CLEAR 60 // (mm) Minimum distance from
  dock for unobstructed X axis
  #define SWITCHING_TOOLHEAD_X_POS { 215, 0 } // (mm) X positions for parking the
  extruders
  #if ENABLED(SWITCHING_TOOLHEAD)
    #define SWITCHING_TOOLHEAD_SERVO_NR 2 // Index of the servo connector
    #define SWITCHING_TOOLHEAD_SERVO_ANGLES { 0, 180 } // (degrees) Angles for
    Lock, Unlock
  #elif ENABLED(MAGNETIC_SWITCHING_TOOLHEAD)
    #define SWITCHING_TOOLHEAD_Y_RELEASE 5 // (mm) Security distance Y axis
    #define SWITCHING_TOOLHEAD_X_SECURITY { 90, 150 } // (mm) Security distance X
    axis (T0,T1)
    #define PRIME_BEFORE_REMOVE // Prime the nozzle before release from
    the dock
    #if ENABLED(PRIME_BEFORE_REMOVE)
      #define SWITCHING_TOOLHEAD_PRIME_MM 20 // (mm) Extruder prime length
      #define SWITCHING_TOOLHEAD_RETRACT_MM 10 // (mm) Retract after priming
      length
      #define SWITCHING_TOOLHEAD_PRIME_FEEDRATE 300 // (mm/min) Extruder prime
      feedrate
      #define SWITCHING_TOOLHEAD_RETRACT_FEEDRATE 2400 // (mm/min) Extruder
      retract feedrate
    #endif
    #elif ENABLED(ELECTROMAGNETIC_SWITCHING_TOOLHEAD)
      #define SWITCHING_TOOLHEAD_Z_HOP 2 // (mm) Z raise for switching
    #endif
  #endif

/**
 * "Mixing Extruder"
 *
 * - Adds G-codes M163 and M164 to set and "commit" the current mix factors.
 *
 * - Extends the stepping routines to move multiple steppers in proportion to the mix.

```

```

* - Optional support for Repetier Firmware's 'M164 S<index>' supporting virtual tools.
* - This implementation supports up to two mixing extruders.
* - Enable DIRECT_MIXING_IN_G1 for M165 and mixing in G1 (from Pia Taubert's reference
implementation).
*/
#define MIXING_EXTRUDER
#if ENABLED(MIXING_EXTRUDER)
  #define MIXING_STEPPERS 2 // Number of steppers in your mixing extruder
  #define MIXING_VIRTUAL_TOOLS 16 // Use the Virtual Tool method with M163 and M164
  #define DIRECT_MIXING_IN_G1 // Allow ABCDHI mix factors in G1 movement
  commands
  #define GRADIENT_MIX // Support for gradient mixing with M166 and LCD
  #if ENABLED(GRADIENT_MIX)
    #define GRADIENT_VTOOL // Add M166 T to use a V-tool index as a Gradient alias
  #endif
#endif

// Offset of the extruders (uncomment if using more than one and relying on firmware to
position when changing).
// The offset has to be X=0, Y=0 for the extruder 0 hotend (default extruder).
// For the other hotends it is their distance from the extruder 0 hotend.
#define HOTEND_OFFSET_X { 0.0, 20.00 } // (mm) relative X-offset for each nozzle
#define HOTEND_OFFSET_Y { 0.0, 5.00 } // (mm) relative Y-offset for each nozzle
#define HOTEND_OFFSET_Z { 0.0, 0.00 } // (mm) relative Z-offset for each nozzle

// @section machine

/**
 * Power Supply Control
 *
 * Enable and connect the power supply to the PS_ON_PIN.
 * Specify whether the power supply is active HIGH or active LOW.
 */
#define PSU_CONTROL
#define PSU_NAME "Power Supply"

#if ENABLED(PSU_CONTROL)
  #define PSU_ACTIVE_STATE LOW // Set 'LOW' for ATX, 'HIGH' for X-Box

  #define PSU_DEFAULT_OFF // Keep power off until enabled directly with M80
  #define PSU_POWERUP_DELAY 250 // (ms) Delay for the PSU to warm up to full power

  #define AUTO_POWER_CONTROL // Enable automatic control of the PS_ON pin
  #if ENABLED(AUTO_POWER_CONTROL)
    #define AUTO_POWER_FANS // Turn on PSU if fans need power
    #define AUTO_POWER_E_FANS
    #define AUTO_POWER_CONTROLLERFAN
    #define AUTO_POWER_CHAMBER_FAN
    #define AUTO_POWER_E_TEMP 50 // (°C) Turn on PSU over this temperature
    #define AUTO_POWER_CHAMBER_TEMP 30 // (°C) Turn on PSU over this temperature
    #define POWER_TIMEOUT 30
  #endif
#endif

//=====
//===== Thermal Settings =====
//=====

// @section temperature

/**
 * --NORMAL IS 4.7kohm PULLUP!-- 1kohm pullup can be used on hotend sensor, using
correct resistor and table
 *
 * Temperature sensors available:
 *

```

```

* -5 : PT100 / PT1000 with MAX31865 (only for sensors 0-1)
* -3 : thermocouple with MAX31855 (only for sensors 0-1)
* -2 : thermocouple with MAX6675 (only for sensors 0-1)
* -4 : thermocouple with AD8495
* -1 : thermocouple with AD595
* 0 : not used
* 1 : 100k thermistor - best choice for EPCOS 100k (4.7k pullup)
* 331 : (3.3V scaled thermistor 1 table for MEGA)
* 332 : (3.3V scaled thermistor 1 table for DUE)
* 2 : 200k thermistor - ATC Semitec 204GT-2 (4.7k pullup)
* 202 : 200k thermistor - Copymaster 3D
* 3 : Mendel-parts thermistor (4.7k pullup)
* 4 : 10k thermistor !! do not use it for a hotend. It gives bad resolution at high temp. !!
* 5 : 100K thermistor - ATC Semitec 104GT-2/104NT-4-R025H42G (Used in ParCan, J-
Head, and E3D) (4.7k pullup)
* 501 : 100K Zonestar (Tronxy X3A) Thermistor
* 502 : 100K Zonestar Thermistor used by hot bed in Zonestar Průša P802M
* 512 : 100k RPW-Ultra hotend thermistor (4.7k pullup)
* 6 : 100k EPCOS - Not as accurate as table 1 (created using a fluke thermocouple) (4.7k
pullup)
* 7 : 100k Honeywell thermistor 135-104LAG-J01 (4.7k pullup)
* 71 : 100k Honeywell thermistor 135-104LAF-J01 (4.7k pullup)
* 8 : 100k 0603 SMD Vishay NTCS0603E3104FXT (4.7k pullup)
* 9 : 100k GE Sensing AL03006-58.2K-97-G1 (4.7k pullup)
* 10 : 100k RS thermistor 198-961 (4.7k pullup)
* 11 : 100k beta 3950 1% thermistor (Used in Keenovo AC silicone mats and most Wanhao
i3 machines) (4.7k pullup)
* 12 : 100k 0603 SMD Vishay NTCS0603E3104FXT (4.7k pullup) (calibrated for Makibox hot
bed)
* 13 : 100k Hisens 3950 1% up to 300°C for hotend "Simple ONE " & "Hotend "All In ONE"
* 15 : 100k thermistor calibration for JGAurora A5 hotend
* 18 : ATC Semitec 204GT-2 (4.7k pullup) Dagoma.Fr - MKS_Base_DKU001327
* 20 : Pt100 with circuit in the Ultimainboard V2.x with 5v excitation (AVR)
* 21 : Pt100 with circuit in the Ultimainboard V2.x with 3.3v excitation (STM32 \ LPC176x....)
* 22 : 100k (hotend) with 4.7k pullup to 3.3V and 220R to analog input (as in GTM32 Pro vB)
* 23 : 100k (bed) with 4.7k pullup to 3.3v and 220R to analog input (as in GTM32 Pro vB)
* 30 : Kis3d Silicone heating mat 200W/300W with 6mm precision cast plate (EN AW 5083)
NTC100K / B3950 (4.7k pullup)
* 201 : Pt100 with circuit in Overlord, similar to Ultimainboard V2.x
* 60 : 100k Maker's Tool Works Kapton Bed Thermistor beta=3950
* 61 : 100k Formbot / Vivedino 3950 350C thermistor 4.7k pullup
* 66 : 4.7M High Temperature thermistor from Dyze Design
* 67 : 450C thermistor from SliceEngineering
* 70 : the 100K thermistor found in the bq Hephestos 2
* 75 : 100k Generic Silicon Heat Pad with NTC 100K MGB18-104F39050L32 thermistor
* 99 : 100k thermistor with a 10K pull-up resistor (found on some Wanhao i3 machines)
*
* 1k ohm pullup tables - This is atypical, and requires changing out the 4.7k pullup for 1k.
* (but gives greater accuracy and more stable PID)
* 51 : 100k thermistor - EPCOS (1k pullup)
* 52 : 200k thermistor - ATC Semitec 204GT-2 (1k pullup)
* 55 : 100k thermistor - ATC Semitec 104GT-2 (Used in ParCan & J-Head) (1k pullup)
*
* 1047 : Pt1000 with 4k7 pullup (E3D)
* 1010 : Pt1000 with 1k pullup (non standard)
* 147 : Pt100 with 4k7 pullup
* 110 : Pt100 with 1k pullup (non standard)
*
* 1000 : Custom - Specify parameters in Configuration_adv.h
*
* Use these for Testing or Development purposes. NEVER for production machine.
* 998 : Dummy Table that ALWAYS reads 25°C or the temperature defined below.
* 999 : Dummy Table that ALWAYS reads 100°C or the temperature defined below.
*/
#define TEMP_SENSOR_0 11
#define TEMP_SENSOR_1 0
#define TEMP_SENSOR_2 0

```

```

#define TEMP_SENSOR_3 0
#define TEMP_SENSOR_4 0
#define TEMP_SENSOR_5 0
#define TEMP_SENSOR_6 0
#define TEMP_SENSOR_7 0
#define TEMP_SENSOR_BED 11
#define TEMP_SENSOR_PROBE 0
#define TEMP_SENSOR_CHAMBER 0

// Dummy thermistor constant temperature readings, for use with 998 and 999
#define DUMMY_THERMISTOR_998_VALUE 25
#define DUMMY_THERMISTOR_999_VALUE 100

// Resistor values when using a MAX31865 (sensor -5)
// Sensor value is typically 100 (PT100) or 1000 (PT1000)
// Calibration value is typically 430 ohm for AdaFruit PT100 modules and 4300 ohm for
AdaFruit PT1000 modules.
//#define MAX31865_SENSOR_OHMS 100
//#define MAX31865_CALIBRATION_OHMS 430

// Use temp sensor 1 as a redundant sensor with sensor 0. If the readings
// from the two sensors differ too much the print will be aborted.
//#define TEMP_SENSOR_1_AS_REDUNDANT
#define MAX_REDUNDANT_TEMP_SENSOR_DIFF 10

#define TEMP_RESIDENCY_TIME 10 // (seconds) Time to wait for hotend to "settle" in
M109
#define TEMP_WINDOW 1 // (°C) Temperature proximity for the "temperature
reached" timer
#define TEMP_HYSTERESIS 3 // (°C) Temperature proximity considered "close enough"
to the target

#define TEMP_BED_RESIDENCY_TIME 10 // (seconds) Time to wait for bed to "settle" in
M190
#define TEMP_BED_WINDOW 1 // (°C) Temperature proximity for the "temperature
reached" timer
#define TEMP_BED_HYSTERESIS 3 // (°C) Temperature proximity considered "close
enough" to the target

// Below this temperature the heater will be switched off
// because it probably indicates a broken thermistor wire.
#define HEATER_0_MINTEMP 5
#define HEATER_1_MINTEMP 5
#define HEATER_2_MINTEMP 5
#define HEATER_3_MINTEMP 5
#define HEATER_4_MINTEMP 5
#define HEATER_5_MINTEMP 5
#define HEATER_6_MINTEMP 5
#define HEATER_7_MINTEMP 5
#define BED_MINTEMP 5

// Above this temperature the heater will be switched off.
// This can protect components from overheating, but NOT from shorts and failures.
// (Use MINTEMP for thermistor short/failure protection.)
#define HEATER_0_MAXTEMP 275
#define HEATER_1_MAXTEMP 275
#define HEATER_2_MAXTEMP 275
#define HEATER_3_MAXTEMP 275
#define HEATER_4_MAXTEMP 275
#define HEATER_5_MAXTEMP 275
#define HEATER_6_MAXTEMP 275
#define HEATER_7_MAXTEMP 275
#define BED_MAXTEMP 150

//=====
=====

```

```

//===== PID Settings
=====
//=====
=====
// PID Tuning Guide here: https://reprap.org/wiki/PID\_Tuning

// Comment the following line to disable PID and enable bang-bang.
#define PIDTEMP
#define BANG_MAX 255 // Limits current to nozzle while in bang-bang mode; 255=full
current
#define PID_MAX BANG_MAX // Limits current to nozzle while PID is active (see
PID_FUNCTIONAL_RANGE below); 255=full current
#define PID_K1 0.95 // Smoothing factor within any PID loop

#if ENABLED(PIDTEMP)
  //#define PID_EDIT_MENU // Add PID editing to the "Advanced Settings" menu. (~700
bytes of PROGMEM)
  //#define PID_AUTOTUNE_MENU // Add PID auto-tuning to the "Advanced Settings"
menu. (~250 bytes of PROGMEM)
  //#define PID_PARAMS_PER_HOTEND // Uses separate PID parameters for each extruder
(useful for mismatched extruders)
    // Set/get with gcode: M301 E[extruder number, 0-2]

    #if ENABLED(PID_PARAMS_PER_HOTEND)
      // Specify between 1 and HOTENDS values per array.
      // If fewer than EXTRUDER values are provided, the last element will be repeated.
      #define DEFAULT_Kp_LIST { 22.20, 22.20 }
      #define DEFAULT_Ki_LIST { 1.08, 1.08 }
      #define DEFAULT_Kd_LIST { 114.00, 114.00 }
    #else
      //#define DEFAULT_Kp 22.20
      //#define DEFAULT_Ki 1.08
      //#define DEFAULT_Kd 114.00
      // HEvo YXY @200
      #define DEFAULT_Kp 16.95
      #define DEFAULT_Ki 1.36
      #define DEFAULT_Kd 52.66
      //#define DEFAULT_Kp 13.13
      //#define DEFAULT_Ki 0.69
      //#define DEFAULT_Kd 62.58
    #endif
#endif // PIDTEMP

//=====
=====
//===== PID > Bed Temperature Control =====
//=====
=====

/**
 * PID Bed Heating
 *
 * If this option is enabled set PID constants below.
 * If this option is disabled, bang-bang will be used and BED_LIMIT_SWITCHING will enable
hysteresis.
 *
 * The PID frequency will be the same as the extruder PWM.
 * If PID_dT is the default, and correct for the hardware/configuration, that means 7.689Hz,
 * which is fine for driving a square wave into a resistive load and does not significantly
 * impact FET heating. This also works fine on a Fotek SSR-10DA Solid State Relay into a
250W
 * heater. If your configuration is significantly different than this and you don't understand
 * the issues involved, don't use bed PID until someone else verifies that your hardware works.
 */
#define PIDTEMPBED

//define BED_LIMIT_SWITCHING

```

```

/**
 * Max Bed Power
 * Applies to all forms of bed control (PID, bang-bang, and bang-bang with hysteresis).
 * When set to any value below 255, enables a form of PWM to the bed that acts like a divider
 * so don't use it unless you are OK with PWM on your bed. (See the comment on enabling
 * PIDTEMPBED)
 */
#define MAX_BED_POWER 255 // limits duty cycle to bed; 255=full current

#if ENABLED(PIDTEMPBED)
  // #define MIN_BED_POWER 0
  // #define PID_BED_DEBUG // Sends debug data to the serial port.

  // 120V 250W silicone heater into 4mm borosilicate (MendelMax 1.5+)
  // from FOPDT model - kp=.39 Tp=405 Tdead=66, Tc set to 79.2, aggressive factor of .15
  // (vs .1, 1, 10)
  // #define DEFAULT_bedKp 10.00
  // #define DEFAULT_bedKi .023
  // #define DEFAULT_bedKd 305.4

  #define DEFAULT_bedKp 50.56
  #define DEFAULT_bedKi 9.02
  #define DEFAULT_bedKd 188.94
  // FIND YOUR OWN: "M303 E-1 C8 S90" to run autotune on the bed at 90 degreesC for 8
  // cycles.
#endif // PIDTEMPBED

#if EITHER(PIDTEMP, PIDTEMPBED)
  #define PID_DEBUG // Sends debug data to the serial port. Use 'M303 D' to toggle
  // activation.
  // #define PID_OPENLOOP // Puts PID in open loop. M104/M140 sets the output power
  // from 0 to PID_MAX
  // #define SLOW_PWM_HEATERS // PWM with very low frequency (roughly 0.125Hz=8s)
  // and minimum state time of approximately 1s useful for heaters driven by a relay
  #define PID_FUNCTIONAL_RANGE 10 // If the temperature difference between the target
  // temperature and the actual temperature
  // is more than PID_FUNCTIONAL_RANGE then the PID will be shut off
  // and the heater will be set to min/max.
#endif

// @section extruder

/**
 * Prevent extrusion if the temperature is below EXTRUDE_MINTEMP.
 * Add M302 to set the minimum extrusion temperature and/or turn
 * cold extrusion prevention on and off.
 *
 * *** IT IS HIGHLY RECOMMENDED TO LEAVE THIS OPTION ENABLED! ***
 */
#define PREVENT_COLD_EXTRUSION
#define EXTRUDE_MINTEMP 170

/**
 * Prevent a single extrusion longer than EXTRUDE_MAXLENGTH.
 * Note: For Bowden Extruders make this large enough to allow load/unload.
 */
#define PREVENT_LENGTHY_EXTRUDE
#define EXTRUDE_MAXLENGTH 1000

//=====
//===== Thermal Runaway Protection =====
//=====
//=====

/**

```

```

* Thermal Protection provides additional protection to your printer from damage
* and fire. Marlin always includes safe min and max temperature ranges which
* protect against a broken or disconnected thermistor wire.
*
* The issue: If a thermistor falls out, it will report the much lower
* temperature of the air in the room, and the the firmware will keep
* the heater on.
*
* If you get "Thermal Runaway" or "Heating failed" errors the
* details can be tuned in Configuration_adv.h
*/

```

```

#define THERMAL_PROTECTION_HOTENDS // Enable thermal protection for all extruders
#define THERMAL_PROTECTION_BED // Enable thermal protection for the heated bed
#define THERMAL_PROTECTION_CHAMBER // Enable thermal protection for the heated
chamber

```

```

//=====
=====
//===== Mechanical Settings =====
//=====
=====

```

```

// @section machine

```

```

// Enable one of the options below for CoreXY, CoreXZ, or CoreYZ kinematics,
// either in the usual order or reversed

```

```

#define COREXY
//#define COREXZ
//#define COREYZ
//#define COREYX
//#define COREZX
//#define COREZY
//#define MARKFORGED_XY // MarkForged. See
https://reprap.org/forum/read.php?152,504042

```

```

//=====
=====
//===== Endstop Settings =====
//=====
=====

```

```

// @section homing

```

```

// Specify here all the endstop connectors that are connected to any endstop or probe.
// Almost all printers will be using one per axis. Probes will use one or more of the
// extra connectors. Leave undefined any used for non-endstop and non-probe purposes.

```

```

#define USE_XMIN_PLUG
#define USE_YMIN_PLUG
#define USE_ZMIN_PLUG
#define USE_XMAX_PLUG
//#define USE_YMAX_PLUG
//#define USE_ZMAX_PLUG

```

```

// Enable pullup for all endstops to prevent a floating state
#define ENDSTOPPULLUPS
#if DISABLED(ENDSTOPPULLUPS)
// Disable ENDSTOPPULLUPS to set pullups individually
//#define ENDSTOPPULLUP_XMAX
//#define ENDSTOPPULLUP_YMAX
//#define ENDSTOPPULLUP_ZMAX
//#define ENDSTOPPULLUP_XMIN
//#define ENDSTOPPULLUP_YMIN
//#define ENDSTOPPULLUP_ZMIN
//#define ENDSTOPPULLUP_ZMIN_PROBE
#endif

```

```

// Enable pulldown for all endstops to prevent a floating state
#define ENDSTOPPULLDOWNS
#if DISABLED(ENDSTOPPULLDOWNS)
  // Disable ENDSTOPPULLDOWNS to set pulldowns individually
  //#define ENDSTOPPULLDOWN_XMAX
  //#define ENDSTOPPULLDOWN_YMAX
  //#define ENDSTOPPULLDOWN_ZMAX
  //#define ENDSTOPPULLDOWN_XMIN
  //#define ENDSTOPPULLDOWN_YMIN
  //#define ENDSTOPPULLDOWN_ZMIN
  //#define ENDSTOPPULLDOWN_ZMIN_PROBE
#endif

// Mechanical endstop with COM to ground and NC to Signal uses "false" here (most common
// setup).
#define X_MIN_ENDSTOP_INVERTING false // TMC2209 false, TMC2130 true, optical false.
#define Y_MIN_ENDSTOP_INVERTING false // Set to true to invert the logic of the endstop.
#define Z_MIN_ENDSTOP_INVERTING false // Set to true to invert the logic of the endstop.
#define X_MAX_ENDSTOP_INVERTING false // Set to true to invert the logic of the endstop.
#define Y_MAX_ENDSTOP_INVERTING false // Set to true to invert the logic of the endstop.
//#define Z_MAX_ENDSTOP_INVERTING false // Set to true to invert the logic of the endstop.
#define Z_MIN_PROBE_ENDSTOP_INVERTING false // Set to true to invert the logic of the
// probe.

/**
 * Stepper Drivers
 *
 * These settings allow Marlin to tune stepper driver timing and enable advanced options for
 * stepper drivers that support them. You may also override timing options in
 * Configuration_adv.h.
 *
 * A4988 is assumed for unspecified drivers.
 *
 * Options: A4988, A5984, DRV8825, LV8729, L6470, L6474, POWERSTEP01,
 *          TB6560, TB6600, TMC2100,
 *          TMC2130, TMC2130_STANDALONE, TMC2160, TMC2160_STANDALONE,
 *          TMC2208, TMC2208_STANDALONE, TMC2209, TMC2209_STANDALONE,
 *          TMC26X, TMC26X_STANDALONE, TMC2660, TMC2660_STANDALONE,
 *          TMC5130, TMC5130_STANDALONE, TMC5160, TMC5160_STANDALONE
 * :['A4988', 'A5984', 'DRV8825', 'LV8729', 'L6470', 'L6474', 'POWERSTEP01', 'TB6560',
 * 'TB6600', 'TMC2100', 'TMC2130', 'TMC2130_STANDALONE', 'TMC2160',
 * 'TMC2160_STANDALONE', 'TMC2208', 'TMC2208_STANDALONE', 'TMC2209',
 * 'TMC2209_STANDALONE', 'TMC26X', 'TMC26X_STANDALONE', 'TMC2660',
 * 'TMC2660_STANDALONE', 'TMC5130', 'TMC5130_STANDALONE', 'TMC5160',
 * 'TMC5160_STANDALONE']
 */
#define X_DRIVER_TYPE TMC2209
#define Y_DRIVER_TYPE TMC2209
#define Z_DRIVER_TYPE TMC2209
//#define X2_DRIVER_TYPE A4988
//#define Y2_DRIVER_TYPE A4988
//#define Z2_DRIVER_TYPE A4988
//#define Z3_DRIVER_TYPE A4988
//#define Z4_DRIVER_TYPE A4988
#define E0_DRIVER_TYPE TMC2209
//#define E1_DRIVER_TYPE A4988
//#define E2_DRIVER_TYPE A4988
//#define E3_DRIVER_TYPE A4988
//#define E4_DRIVER_TYPE A4988
//#define E5_DRIVER_TYPE A4988
//#define E6_DRIVER_TYPE A4988
//#define E7_DRIVER_TYPE A4988

// Enable this feature if all enabled endstop pins are interrupt-capable.
// This will remove the need to poll the interrupt pins, saving many CPU cycles.
#define ENDSTOP_INTERRUPTS_FEATURE

```



```

/**
 * Endstop Noise Threshold
 *
 * Enable if your probe or endstops falsely trigger due to noise.
 *
 * - Higher values may affect repeatability or accuracy of some bed probes.
 * - To fix noise install a 100nF ceramic capacitor in parallel with the switch.
 * - This feature is not required for common micro-switches mounted on PCBs
 *   based on the Makerbot design, which already have the 100nF capacitor.
 *
 * :[2,3,4,5,6,7]
 */
#define ENDSTOP_NOISE_THRESHOLD 2

// Check for stuck or disconnected endstops during homing moves.
#define DETECT_BROKEN_ENDSTOP

//=====
//===== Movement Settings
//=====
// @section motion

/**
 * Default Settings
 *
 * These settings can be reset by M502
 *
 * Note that if EEPROM is enabled, saved values will override these.
 */

/**
 * With this option each E stepper can have its own factors for the
 * following movement settings. If fewer factors are given than the
 * total number of extruders, the last value applies to the rest.
 */
#define DISTINCT_E_FACTORS

/**
 * Default Axis Steps Per Unit (steps/mm)
 * Override with M92
 *
 * X, Y, Z, E0 [, E1[, E2...]]
 */
#define DEFAULT_AXIS_STEPS_PER_UNIT { 80, 80, 400, 382.2 }

/**
 * Default Max Feed Rate (mm/s)
 * Override with M203
 *
 * X, Y, Z, E0 [, E1[, E2...]]
 */
#define DEFAULT_MAX_FEEDRATE { 300, 300, 30, 100 }

//define LIMITED_MAX_FR_EDITING // Limit edit via M203 or LCD to
//DEFAULT_MAX_FEEDRATE * 2
#if ENABLED(LIMITED_MAX_FR_EDITING)
  #define MAX_FEEDRATE_EDIT_VALUES { 600, 600, 10, 50 } // ...or, set your own edit
  limits
#endif

/**
 * Default Max Acceleration (change/s) change = mm/s
 * (Maximum start speed for accelerated moves)
 * Override with M201
 *
 * X, Y, Z, E0 [, E1[, E2...]]
 */

```

```

#define DEFAULT_MAX_ACCELERATION { 3000, 3000, 100, 10000 }

//define LIMITED_MAX_ACCEL_EDITING // Limit edit via M201 or LCD to
DEFAULT_MAX_ACCELERATION * 2
#if ENABLED(LIMITED_MAX_ACCEL_EDITING)
  #define MAX_ACCEL_EDIT_VALUES { 6000, 6000, 200, 20000 } // ...or, set your own edit
limits
#endif

/**
 * Default Acceleration (change/s) change = mm/s
 * Override with M204
 *
 * M204 P Acceleration
 * M204 R Retract Acceleration
 * M204 T Travel Acceleration
 */
#define DEFAULT_ACCELERATION 1000 // X, Y, Z and E acceleration for printing
moves
#define DEFAULT_RETRACT_ACCELERATION 3000 // E acceleration for retracts
#define DEFAULT_TRAVEL_ACCELERATION 1500 // X, Y, Z acceleration for travel (non
printing) moves

/**
 * Default Jerk limits (mm/s)
 * Override with M205 X Y Z E
 *
 * "Jerk" specifies the minimum speed change that requires acceleration.
 * When changing speed and direction, if the difference is less than the
 * value set here, it may happen instantaneously.
 */
//define CLASSIC_JERK
#if ENABLED(CLASSIC_JERK)
  #define DEFAULT_XJERK 10.0
  #define DEFAULT_YJERK 10.0
  #define DEFAULT_ZJERK 0.3

  //define TRAVEL_EXTRA_XYJERK 0.0 // Additional jerk allowance for all travel moves

  //define LIMITED_JERK_EDITING // Limit edit via M205 or LCD to DEFAULT_aJERK *
2
  #if ENABLED(LIMITED_JERK_EDITING)
    #define MAX_JERK_EDIT_VALUES { 20, 20, 0.6, 10 } // ...or, set your own edit limits
  #endif
#endif

#define DEFAULT_EJERK 5.0 // May be used by Linear Advance

/**
 * Junction Deviation Factor
 *
 * See:
 * https://reprap.org/forum/read.php?1,739819
 * https://blog.kyneticcnc.com/2018/10/computing-junction-deviation-for-marlin.html
 */
#if DISABLED(CLASSIC_JERK)
  #define JUNCTION_DEVIATION_MM 0.013 // (mm) Distance from real junction edge
  #define JD_HANDLE_SMALL_SEGMENTS // Use curvature estimation instead of just the
junction angle
// for small segments (< 1mm) with large junction angles (> 135°).
#endif

/**
 * S-Curve Acceleration
 *
 * This option eliminates vibration during printing by fitting a Bézier
 * curve to move acceleration, producing much smoother direction changes.

```

```

*
* See https://github.com/synthetos/TinyG/wiki/Jerk-Controlled-Motion-Explained
*/
#define S_CURVE_ACCELERATION

//=====
//===== Z Probe Options
//=====
//=====
// @section probes

//
// See https://marlinfw.org/docs/configuration/probes.html
//

/**
 * Enable this option for a probe connected to the Z-MIN pin.
 * The probe replaces the Z-MIN endstop and is used for Z homing.
 * (Automatically enables USE_PROBE_FOR_Z_HOMING.)
 */
#define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN

// Force the use of the probe for Z-axis homing
#define USE_PROBE_FOR_Z_HOMING

/**
 * Z_MIN_PROBE_PIN
 *
 * Define this pin if the probe is not connected to Z_MIN_PIN.
 * If not defined the default pin for the selected MOTHERBOARD
 * will be used. Most of the time the default is what you want.
 *
 * - The simplest option is to use a free endstop connector.
 * - Use 5V for powered (usually inductive) sensors.
 *
 * - RAMPS 1.3/1.4 boards may use the 5V, GND, and Aux4->D32 pin:
 *   - For simple switches connect...
 *     - normally-closed switches to GND and D32.
 *     - normally-open switches to 5V and D32.
 */
#define Z_MIN_PROBE_PIN 32 // Pin 32 is the RAMPS default

/**
 * Probe Type
 *
 * Allen Key Probes, Servo Probes, Z-Sled Probes, FIX_MOUNTED_PROBE, etc.
 * Activate one of these to use Auto Bed Leveling below.
 */

/**
 * The "Manual Probe" provides a means to do "Auto" Bed Leveling without a probe.
 * Use G29 repeatedly, adjusting the Z height at each point with movement commands
 * or (with LCD_BED_LEVELING) the LCD controller.
 */
#define PROBE_MANUALLY
#define MANUAL_PROBE_START_Z 0.2

/**
 * A Fix-Mounted Probe either doesn't deploy or needs manual deployment.
 * (e.g., an inductive probe or a nozzle-based probe-switch.)
 */
#define FIX_MOUNTED_PROBE

/**
 * Use the nozzle as the probe, as with a conductive

```

```

* nozzle system or a piezo-electric smart effector.
*/
#define NOZZLE_AS_PROBE

/**
 * Z Servo Probe, such as an endstop switch on a rotating arm.
 */
#define Z_PROBE_SERVO_NR 0 // Defaults to SERVO 0 connector.
#define Z_SERVO_ANGLES { 70, 0 } // Z Servo Deploy and Stow angles

/**
 * The BLTouch probe uses a Hall effect sensor and emulates a servo.
 */
#define BLTOUCH

/**
 * Pressure sensor with a BLTouch-like interface
 */
#define CREALITY_TOUCH

/**
 * Touch-MI Probe by hotends.fr
 *
 * This probe is deployed and activated by moving the X-axis to a magnet at the edge of the
 * bed.
 * By default, the magnet is assumed to be on the left and activated by a home. If the magnet is
 * on the right, enable and set TOUCH_MI_DEPLOY_XPOS to the deploy position.
 *
 * Also requires: BABYSTEPPING, BABYSTEP_ZPROBE_OFFSET, Z_SAFE_HOMING,
 * and a minimum Z_HOMING_HEIGHT of 10.
 */
#define TOUCH_MI_PROBE
#if ENABLED(TOUCH_MI_PROBE)
  #define TOUCH_MI_RETRACT_Z 0.5 // Height at which the probe retracts
  #define TOUCH_MI_DEPLOY_XPOS (X_MAX_BED + 2) // For a magnet on the right side
  of the bed
  #define TOUCH_MI_MANUAL_DEPLOY // For manual deploy (LCD menu)
#endif

// A probe that is deployed and stowed with a solenoid pin (SOL1_PIN)
#define SOLENOID_PROBE

// A sled-mounted probe like those designed by Charles Bell.
#define Z_PROBE_SLED
#define SLED_DOCKING_OFFSET 5 // The extra distance the X axis must travel to pickup
the sled. 0 should be fine but you can push it further if you'd like.

// A probe deployed by moving the x-axis, such as the Wilson II's rack-and-pinion probe
designed by Marty Rice.
#define RACK_AND_PINION_PROBE
#if ENABLED(RACK_AND_PINION_PROBE)
  #define Z_PROBE_DEPLOY_X X_MIN_POS
  #define Z_PROBE_RETRACT_X X_MAX_POS
#endif

// Duet Smart Effector (for delta printers) - https://bit.ly/2ul5U7J
// When the pin is defined you can use M672 to set/reset the probe sensitivity.
#define DUET_SMART_EFFECTOR
#if ENABLED(DUET_SMART_EFFECTOR)
  #define SMART_EFFECTOR_MOD_PIN -1 // Connect a GPIO pin to the Smart Effector
  MOD pin
#endif

/**
 * Use StallGuard2 to probe the bed with the nozzle.
 * Requires stallGuard-capable Trinamic stepper drivers.
 * CAUTION: This can damage machines with Z lead screws.

```

```

*      Take extreme care when setting up this feature.
*/
//define SENSORLESS_PROBING

//
// For Z_PROBE_ALLEN_KEY see the Delta example configurations.
//

/**
 * Nozzle-to-Probe offsets { X, Y, Z }
 *
 * - Use a caliper or ruler to measure the distance from the tip of
 *   the Nozzle to the center-point of the Probe in the X and Y axes.
 * - For the Z offset use your best known value and adjust at runtime.
 * - Probe Offsets can be tuned at runtime with 'M851', LCD menus, babystepping, etc.
 *
 * Assuming the typical work area orientation:
 * - Probe to RIGHT of the Nozzle has a Positive X offset
 * - Probe to LEFT of the Nozzle has a Negative X offset
 * - Probe in BACK of the Nozzle has a Positive Y offset
 * - Probe in FRONT of the Nozzle has a Negative Y offset
 *
 * Some examples:
 * #define NOZZLE_TO_PROBE_OFFSET { 10, 10, -1 } // Example "1"
 * #define NOZZLE_TO_PROBE_OFFSET {-10, 5, -1 } // Example "2"
 * #define NOZZLE_TO_PROBE_OFFSET { 5, -5, -1 } // Example "3"
 * #define NOZZLE_TO_PROBE_OFFSET {-15,-10, -1 } // Example "4"
 *
 * +-- BACK ---+
 * |  [+]  |
 * L|    1 | R <-- Example "1" (right+, back+)
 * E|  2   | I <-- Example "2" ( left-, back+)
 * F|[-] N [+] | G <-- Nozzle
 * T|    3 | H <-- Example "3" (right+, front-)
 * |  4   | T <-- Example "4" ( left-, front-)
 * |  [-]  |
 * O-- FRONT --+
 */
#define NOZZLE_TO_PROBE_OFFSET { 0, 0, 0 }

// Most probes should stay away from the edges of the bed, but
// with NOZZLE_AS_PROBE this can be negative for a wider probing area.
#define PROBING_MARGIN 10

// X and Y axis travel speed (mm/min) between probes
#define XY_PROBE_SPEED (133*60)

// Feedrate (mm/min) for the first approach when double-probing (MULTIPLE_PROBING == 2)
#define Z_PROBE_SPEED_FAST HOMING_FEEDRATE_Z

// Feedrate (mm/min) for the "accurate" probe of each point
#define Z_PROBE_SPEED_SLOW (Z_PROBE_SPEED_FAST / 2)

/**
 * Multiple Probing
 *
 * You may get improved results by probing 2 or more times.
 * With EXTRA_PROBING the more atypical reading(s) will be disregarded.
 *
 * A total of 2 does fast/slow probes with a weighted average.
 * A total of 3 or more adds more slow probes, taking the average.
 */
//define MULTIPLE_PROBING 2
//define EXTRA_PROBING 1

/**
 * Z probes require clearance when deploying, stowing, and moving between

```

```

* probe points to avoid hitting the bed and other hardware.
* Servo-mounted probes require extra space for the arm to rotate.
* Inductive probes need space to keep from triggering early.
*
* Use these settings to specify the distance (mm) to raise the probe (or
* lower the bed). The values set here apply over and above any (negative)
* probe Z Offset set with NOZZLE_TO_PROBE_OFFSET, M851, or the LCD.
* Only integer values >= 1 are valid here.
*
* Example: `M851 Z-5` with a CLEARANCE of 4 => 9mm from bed to nozzle.
* But: `M851 Z+1` with a CLEARANCE of 2 => 2mm from bed to nozzle.
*/
#define Z_CLEARANCE_DEPLOY_PROBE 10 // Z Clearance for Deploy/Stow
#define Z_CLEARANCE_BETWEEN_PROBES 5 // Z Clearance between probe points
#define Z_CLEARANCE_MULTI_PROBE 5 // Z Clearance between multiple probes
//#define Z_AFTER_PROBING 5 // Z position after probing is done

#define Z_PROBE_LOW_POINT -2 // Farthest distance below the trigger-point to go
before stopping

// For M851 give a range for adjusting the Z probe offset
#define Z_PROBE_OFFSET_RANGE_MIN -20
#define Z_PROBE_OFFSET_RANGE_MAX 20

// Enable the M48 repeatability test to test probe accuracy
//#define Z_MIN_PROBE_REPEATABILITY_TEST

// Before deploy/stow pause for user confirmation
//#define PAUSE_BEFORE_DEPLOY_STOW
#if ENABLED(PAUSE_BEFORE_DEPLOY_STOW)
  //#define PAUSE_PROBE_DEPLOY_WHEN_TRIGGERED // For Manual Deploy Allenkey
  Probe
#endif

/**
* Enable one or more of the following if probing seems unreliable.
* Heaters and/or fans can be disabled during probing to minimize electrical
* noise. A delay can also be added to allow noise and vibration to settle.
* These options are most useful for the BLTouch probe, but may also improve
* readings with inductive probes and piezo sensors.
*/
//#define PROBING_HEATERS_OFF // Turn heaters off when probing
#if ENABLED(PROBING_HEATERS_OFF)
  //#define WAIT_FOR_BED_HEATER // Wait for bed to heat back up between probes (to
  improve accuracy)
#endif
#define PROBING_FANS_OFF // Turn fans off when probing
#define PROBING_STEPPERS_OFF // Turn steppers off (unless needed to hold position)
when probing
#define DELAY_BEFORE_PROBING 200 // (ms) To prevent vibrations from triggering piezo
sensors

// For Inverting Stepper Enable Pins (Active Low) use 0, Non Inverting (Active High) use 1
// :{ 0:'Low', 1:'High' }
#define X_ENABLE_ON 0
#define Y_ENABLE_ON 0
#define Z_ENABLE_ON 0
#define E_ENABLE_ON 0 // For all extruders

// Disable axis steppers immediately when they're not being stepped.
// WARNING: When motors turn off there is a chance of losing position accuracy!
#define DISABLE_X false
#define DISABLE_Y false
#define DISABLE_Z false

// Turn off the display blinking that warns about possible accuracy reduction
//#define DISABLE_REDUCED_ACCURACY_WARNING

```

```

// @section extruder

#define DISABLE_E true          // Disable the extruder when not stepping
#define DISABLE_INACTIVE_EXTRUDER // Keep only the active extruder enabled

// @section machine

// Invert the stepper direction. Change (or reverse the motor connector) if an axis goes the
// wrong way.
#define INVERT_X_DIR true
#define INVERT_Y_DIR true
#define INVERT_Z_DIR true

// @section extruder

// For direct drive extruder v9 set to true, for geared extruder set to (false).
#define INVERT_E0_DIR true
#define INVERT_E1_DIR false
#define INVERT_E2_DIR false
#define INVERT_E3_DIR false
#define INVERT_E4_DIR false
#define INVERT_E5_DIR false
#define INVERT_E6_DIR false
#define INVERT_E7_DIR false

// @section homing

// #define NO_MOTION_BEFORE_HOMING // Inhibit movement until all axes have been
// homed

// #define UNKNOWN_Z_NO_RAISE // Don't raise Z (lower the bed) if Z is "unknown." For
// beds that fall when Z is powered off.

// #define Z_HOMING_HEIGHT 4 // (mm) Minimal Z height before homing (G28) for Z
// clearance above the bed, clamps, ...
// Be sure to have this much clearance over your Z_MAX_POS to prevent
// grinding.

// #define Z_AFTER_HOMING 10 // (mm) Height to move to after homing Z

// Direction of endstops when homing; 1=MAX, -1=MIN
// :[-1,1]
#define X_HOME_DIR -1
#define Y_HOME_DIR -1
#define Z_HOME_DIR -1

// @section machine

// The size of the print bed
#define X_BED_SIZE 220
#define Y_BED_SIZE 220

// Travel limits (mm) after homing, corresponding to endstop positions.
#define X_MIN_POS 0
#define Y_MIN_POS 0
#define Z_MIN_POS 0
#define X_MAX_POS X_BED_SIZE+5
#define Y_MAX_POS Y_BED_SIZE+5
#define Z_MAX_POS 235

/**
 * Software Endstops
 *
 * - Prevent moves outside the set machine bounds.
 * - Individual axes can be disabled, if desired.
 * - X and Y only apply to Cartesian robots.

```

```

* - Use 'M211' to set software endstops on/off or report current state
*/

// Min software endstops constrain movement within minimum coordinate bounds
#define MIN_SOFTWARE_ENDSTOPS
#if ENABLED(MIN_SOFTWARE_ENDSTOPS)
  #define MIN_SOFTWARE_ENDSTOP_X
  #define MIN_SOFTWARE_ENDSTOP_Y
  #define MIN_SOFTWARE_ENDSTOP_Z
#endif

// Max software endstops constrain movement within maximum coordinate bounds
#define MAX_SOFTWARE_ENDSTOPS
#if ENABLED(MAX_SOFTWARE_ENDSTOPS)
  #define MAX_SOFTWARE_ENDSTOP_X
  #define MAX_SOFTWARE_ENDSTOP_Y
  #define MAX_SOFTWARE_ENDSTOP_Z
#endif

#if EITHER(MIN_SOFTWARE_ENDSTOPS, MAX_SOFTWARE_ENDSTOPS)
  // #define SOFT_ENDSTOPS_MENU_ITEM // Enable/Disable software endstops from the LCD
#endif

/**
 * Filament Runout Sensors
 * Mechanical or opto endstops are used to check for the presence of filament.
 *
 * RAMPS-based boards use SERVO3_PIN for the first runout sensor.
 * For other boards you may need to define FIL_RUNOUT_PIN, FIL_RUNOUT2_PIN, etc.
 */
#define FILAMENT_RUNOUT_SENSOR
#if ENABLED(FILAMENT_RUNOUT_SENSOR)
  #define FIL_RUNOUT_ENABLED_DEFAULT true // Enable the sensor on startup. Override
  with M412 followed by M500.
  #define NUM_RUNOUT_SENSORS 1 // Number of sensors, up to one per extruder.
  Define a FIL_RUNOUT#_PIN for each.
  #define FIL_RUNOUT_STATE LOW // Pin state indicating that filament is NOT
  present.
  #define FIL_RUNOUT_PULLUP // Use internal pullup for filament runout pins.
  // #define FIL_RUNOUT_PULLEDOWN // Use internal pulldown for filament runout pins.

  // Set one or more commands to execute on filament runout.
  // (After 'M412 H' Marlin will ask the host to handle the process.)
  #define FILAMENT_RUNOUT_SCRIPT "M600"

  // After a runout is detected, continue printing this length of filament
  // before executing the runout script. Useful for a sensor at the end of
  // a feed tube. Requires 4 bytes SRAM per sensor, plus 4 bytes overhead.
  // #define FILAMENT_RUNOUT_DISTANCE_MM 25

  #ifdef FILAMENT_RUNOUT_DISTANCE_MM
    // Enable this option to use an encoder disc that toggles the runout pin
    // as the filament moves. (Be sure to set FILAMENT_RUNOUT_DISTANCE_MM
    // large enough to avoid false positives.)
    // #define FILAMENT_MOTION_SENSOR
  #endif
#endif

//=====
//===== Bed Leveling
//=====
//=====
// @section calibrate

```



```

/**
 * Choose one of the options below to enable G29 Bed Leveling. The parameters
 * and behavior of G29 will change depending on your selection.
 *
 * If using a Probe for Z Homing, enable Z_SAFE_HOMING also!
 *
 * - AUTO_BED_LEVELING_3POINT
 *   Probe 3 arbitrary points on the bed (that aren't collinear)
 *   You specify the XY coordinates of all 3 points.
 *   The result is a single tilted plane. Best for a flat bed.
 *
 * - AUTO_BED_LEVELING_LINEAR
 *   Probe several points in a grid.
 *   You specify the rectangle and the density of sample points.
 *   The result is a single tilted plane. Best for a flat bed.
 *
 * - AUTO_BED_LEVELING_BILINEAR
 *   Probe several points in a grid.
 *   You specify the rectangle and the density of sample points.
 *   The result is a mesh, best for large or uneven beds.
 *
 * - AUTO_BED_LEVELING_UBL (Unified Bed Leveling)
 *   A comprehensive bed leveling system combining the features and benefits
 *   of other systems. UBL also includes integrated Mesh Generation, Mesh
 *   Validation and Mesh Editing systems.
 *
 * - MESH_BED_LEVELING
 *   Probe a grid manually
 *   The result is a mesh, suitable for large or uneven beds. (See BILINEAR.)
 *   For machines without a probe, Mesh Bed Leveling provides a method to perform
 *   leveling in steps so you can manually adjust the Z height at each grid-point.
 *   With an LCD controller the process is guided step-by-step.
 */
#define AUTO_BED_LEVELING_3POINT
#define AUTO_BED_LEVELING_LINEAR
#define AUTO_BED_LEVELING_BILINEAR
#define AUTO_BED_LEVELING_UBL
#define MESH_BED_LEVELING

/**
 * Normally G28 leaves leveling disabled on completion. Enable
 * this option to have G28 restore the prior leveling state.
 */
#define RESTORE_LEVELING_AFTER_G28

/**
 * Enable detailed logging of G28, G29, M48, etc.
 * Turn on with the command 'M111 S32'.
 * NOTE: Requires a lot of PROGMEM!
 */
#define DEBUG_LEVELING_FEATURE

#if ANY(MESH_BED_LEVELING, AUTO_BED_LEVELING_BILINEAR,
AUTO_BED_LEVELING_UBL)
  // Gradually reduce leveling correction until a set height is reached,
  // at which point movement will be level to the machine's XY plane.
  // The height can be set with M420 Z<height>
  #define ENABLE_LEVELING_FADE_HEIGHT

  // For Cartesian machines, instead of dividing moves on mesh boundaries,
  // split up moves into short segments like a Delta. This follows the
  // contours of the bed more closely than edge-to-edge straight moves.
  #define SEGMENT_LEVELED_MOVES
  #define LEVELED_SEGMENT_LENGTH 5.0 // (mm) Length of all segments (except the last
one)

/**

```

```

* Enable the G26 Mesh Validation Pattern tool.
*/
//#define G26_MESH_VALIDATION
#if ENABLED(G26_MESH_VALIDATION)
  #define MESH_TEST_NOZZLE_SIZE 0.4 // (mm) Diameter of primary nozzle.
  #define MESH_TEST_LAYER_HEIGHT 0.2 // (mm) Default layer height for the G26 Mesh
  Validation Tool.
  #define MESH_TEST_HOTEND_TEMP 205 // (°C) Default nozzle temperature for the
  G26 Mesh Validation Tool.
  #define MESH_TEST_BED_TEMP 60 // (°C) Default bed temperature for the G26 Mesh
  Validation Tool.
  #define G26_XY_FEEDRATE 20 // (mm/s) Feedrate for XY Moves for the G26 Mesh
  Validation Tool.
  #define G26_RETRACT_MULTIPLIER 1.0 // G26 Q (retraction) used by default between
  mesh test elements.
#endif

#endif

#if EITHER(AUTO_BED_LEVELING_LINEAR, AUTO_BED_LEVELING_BILINEAR)

  // Set the number of grid points per dimension.
  #define GRID_MAX_POINTS_X 3
  #define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X

  // Probe along the Y axis, advancing X after each column
  // #define PROBE_Y_FIRST

  #if ENABLED(AUTO_BED_LEVELING_BILINEAR)

    // Beyond the probed grid, continue the implied tilt?
    // Default is to maintain the height of the nearest edge.
    // #define EXTRAPOLATE_BEYOND_GRID

    //
    // Experimental Subdivision of the grid by Catmull-Rom method.
    // Synthesizes intermediate points to produce a more detailed mesh.
    //
    // #define ABL_BILINEAR_SUBDIVISION
    #if ENABLED(ABL_BILINEAR_SUBDIVISION)
      // Number of subdivisions between probe points
      #define BILINEAR_SUBDIVISIONS 3
    #endif

  #endif

  #elif ENABLED(AUTO_BED_LEVELING_UBL)

    //=====
    //=====
    //===== Unified Bed Leveling =====
    //=====
    //=====

    // #define MESH_EDIT_GFX_OVERLAY // Display a graphics overlay while editing the mesh

    #define MESH_INSET 1 // Set Mesh bounds as an inset region of the bed
    #define GRID_MAX_POINTS_X 10 // Don't use more than 15 points per axis,
    implementation limited.
    #define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X

    #define UBL_MESH_EDIT_MOVES_Z // Sophisticated users prefer no movement of nozzle
    #define UBL_SAVE_ACTIVE_ON_M500 // Save the currently active mesh in the current slot
    on M500

    // #define UBL_Z_RAISE_WHEN_OFF_MESH 2.5 // When the nozzle is off the mesh, this
    value is used

```

```

// as the Z-Height correction value.

#elif ENABLED(MESH_BED_LEVELING)

//=====
=====
//===== Mesh
=====
//=====
=====

#define MESH_INSET 10 // Set Mesh bounds as an inset region of the bed
#define GRID_MAX_POINTS_X 3 // Don't use more than 7 points per axis, implementation
limited.
#define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X

// #define MESH_G28_REST_ORIGIN // After homing all axes ('G28' or 'G28 XYZ') rest Z at
Z_MIN_POS

#endif // BED_LEVELING

/**
 * Add a bed leveling sub-menu for ABL or MBL.
 * Include a guided procedure if manual probing is enabled.
 */
#define LCD_BED_LEVELING

#if ENABLED(LCD_BED_LEVELING)
  #define MESH_EDIT_Z_STEP 0.025 // (mm) Step size while manually probing Z axis.
  #define LCD_PROBE_Z_RANGE 4 // (mm) Z Range centered on Z_MIN_POS for LCD Z
  adjustment
  // #define MESH_EDIT_MENU // Add a menu to edit mesh points
#endif

// Add a menu item to move between bed corners for manual bed adjustment
#define LEVEL_BED_CORNERS

#if ENABLED(LEVEL_BED_CORNERS)
  #define LEVEL_CORNERS_INSET_LFRB { 30, 30, 30, 30 } // (mm) Left, Front, Right, Back
  insets
  #define LEVEL_CORNERS_HEIGHT 0.0 // (mm) Z height of nozzle at leveling points
  #define LEVEL_CORNERS_Z_HOP 4.0 // (mm) Z height of nozzle between leveling
  points
  // #define LEVEL_CENTER_TOO // Move to the center after the last corner
#endif

/**
 * Commands to execute at the end of G29 probing.
 * Useful to retract or move the Z probe out of the way.
 */
// #define Z_PROBE_END_SCRIPT "G1 Z10 F12000\nG1 X15 Y330\nG1 Z0.5\nG1 Z10"

// @section homing

// The center of the bed is at (X=0, Y=0)
// #define BED_CENTER_AT_0_0

// Manually set the home position. Leave these undefined for automatic settings.
// For DELTA this is the top-center of the Cartesian print volume.
// #define MANUAL_X_HOME_POS 0
// #define MANUAL_Y_HOME_POS 0
// #define MANUAL_Z_HOME_POS 0

// Use "Z Safe Homing" to avoid homing with a Z probe outside the bed area.
//
// With this feature enabled:
//

```

```
// - Allow Z homing only after X and Y homing AND stepper drivers still enabled.
// - If stepper drivers time out, it will need X and Y homing again before Z homing.
// - Move the Z probe (or nozzle) to a defined XY point before Z Homing.
// - Prevent Z homing when the Z probe is outside bed area.
```

```
//
```

```
##define Z_SAFE_HOMING
```

```
#if ENABLED(Z_SAFE_HOMING)
```

```
#define Z_SAFE_HOMING_X_POINT X_CENTER // X point for Z homing
```

```
#define Z_SAFE_HOMING_Y_POINT Y_CENTER // Y point for Z homing
```

```
#endif
```

```
// Homing speeds (mm/min)
```

```
#define HOMING_FEEDRATE_XY (50*60)
```

```
#define HOMING_FEEDRATE_Z (8*60)
```

```
// Validate that endstops are triggered on homing moves
```

```
#define VALIDATE_HOMING_ENDSTOPS
```

```
// @section calibrate
```

```
/**
```

```
 * Bed Skew Compensation
```

```
 *
```

```
 * This feature corrects for misalignment in the XYZ axes.
```

```
 *
```

```
 * Take the following steps to get the bed skew in the XY plane:
```

```
 * 1. Print a test square (e.g., https://www.thingiverse.com/thing:2563185)
```

```
 * 2. For XY_DIAG_AC measure the diagonal A to C
```

```
 * 3. For XY_DIAG_BD measure the diagonal B to D
```

```
 * 4. For XY_SIDE_AD measure the edge A to D
```

```
 *
```

```
 * Marlin automatically computes skew factors from these measurements.
```

```
 * Skew factors may also be computed and set manually:
```

```
 *
```

```
 * - Compute AB : SQRT(2*AC*AC+2*BD*BD-4*AD*AD)/2
```

```
 * - XY_SKEW_FACTOR : TAN(PI/2-ACOS((AC*AC-AB*AB-AD*AD)/(2*AB*AD)))
```

```
 *
```

```
 * If desired, follow the same procedure for XZ and YZ.
```

```
 * Use these diagrams for reference:
```

```
 *
```

```
 *   Y           Z           Z
 *   ^   B-----C   ^   B-----C   ^   B-----C
 *   | / / / / /   | / / / / /   | / / / / /
 *   | / / / / /   | / / / / /   | / / / / /
 *   | A-----D   | A-----D   | A-----D
 *   +----->X   +----->X   +----->Y
```

```
 *   XY_SKEW_FACTOR   XZ_SKEW_FACTOR   YZ_SKEW_FACTOR
```

```
 */
```

```
##define SKEW_CORRECTION
```

```
#if ENABLED(SKEW_CORRECTION)
```

```
 // Input all length measurements here:
```

```
#define XY_DIAG_AC 282.8427124746
```

```
#define XY_DIAG_BD 282.8427124746
```

```
#define XY_SIDE_AD 200
```

```
 // Or, set the default skew factors directly here
```

```
 // to override the above measurements:
```

```
#define XY_SKEW_FACTOR 0.0
```

```
##define SKEW_CORRECTION_FOR_Z
```

```
#if ENABLED(SKEW_CORRECTION_FOR_Z)
```

```
#define XZ_DIAG_AC 282.8427124746
```

```
#define XZ_DIAG_BD 282.8427124746
```

```
#define YZ_DIAG_AC 282.8427124746
```

```
#define YZ_DIAG_BD 282.8427124746
```

```

#define YZ_SIDE_AD 200
#define XZ_SKEW_FACTOR 0.0
#define YZ_SKEW_FACTOR 0.0
#endif

// Enable this option for M852 to set skew at runtime
// #define SKEW_CORRECTION_GCODE
#endif

//=====
=====
//===== Additional Features
=====
//=====
=====

// @section extras

/**
 * EEPROM
 *
 * Persistent storage to preserve configurable settings across reboots.
 *
 * M500 - Store settings to EEPROM.
 * M501 - Read settings from EEPROM. (i.e., Throw away unsaved changes)
 * M502 - Revert settings to "factory" defaults. (Follow with M500 to init the EEPROM.)
 */
#define EEPROM_SETTINGS // Persistent storage with M500 and M501
// #define DISABLE_M503 // Saves ~2700 bytes of PROGMEM. Disable for release!
#define EEPROM_CHITCHAT // Give feedback on EEPROM commands. Disable to save
PROGMEM.
#define EEPROM_BOOT_SILENT // Keep M503 quiet and only give errors during first load
#if ENABLED(EEPROM_SETTINGS)
  // #define EEPROM_AUTO_INIT // Init EEPROM automatically on any errors.
#endif

//
// Host Keepalive
//
// When enabled Marlin will send a busy status message to the host
// every couple of seconds when it can't accept commands.
//
#define HOST_KEEPAIVE_FEATURE // Disable this if your host doesn't like keepalive
messages
#define DEFAULT_KEEPAIVE_INTERVAL 2 // Number of seconds between "busy"
messages. Set with M113.
#define BUSY_WHILE_HEATING // Some hosts require "busy" messages even during
heating

//
// G20/G21 Inch mode support
//
// #define INCH_MODE_SUPPORT

//
// M149 Set temperature units support
//
// #define TEMPERATURE_UNITS_SUPPORT

// @section temperature

// Preheat Constants
#define PREHEAT_1_LABEL "PLA"
#define PREHEAT_1_TEMP_HOTEND 200
#define PREHEAT_1_TEMP_BED 50
#define PREHEAT_1_FAN_SPEED 0 // Value from 0 to 255

```

```

#define PREHEAT_2_LABEL    "PETG"
#define PREHEAT_2_TEMP_HOTEND 240
#define PREHEAT_2_TEMP_BED 70
#define PREHEAT_2_FAN_SPEED 0 // Value from 0 to 255

/**
 * Nozzle Park
 *
 * Park the nozzle at the given XYZ position on idle or G27.
 *
 * The "P" parameter controls the action applied to the Z axis:
 *
 * P0 (Default) If Z is below park Z raise the nozzle.
 * P1 Raise the nozzle always to Z-park height.
 * P2 Raise the nozzle by Z-park amount, limited to Z_MAX_POS.
 */
#define NOZZLE_PARK_FEATURE

#if ENABLED(NOZZLE_PARK_FEATURE)
  // Specify a park position as { X, Y, Z_raise }
  #define NOZZLE_PARK_POINT { (X_MIN_POS + 10), (Y_MAX_POS - 10), 20 }
  //#define NOZZLE_PARK_X_ONLY        // X move only is required to park
  //#define NOZZLE_PARK_Y_ONLY        // Y move only is required to park
  #define NOZZLE_PARK_Z_RAISE_MIN 2 // (mm) Always raise Z by at least this distance
  #define NOZZLE_PARK_XY_FEEDRATE 100 // (mm/s) X and Y axes feedrate (also used
for delta Z axis)
  #define NOZZLE_PARK_Z_FEEDRATE 5 // (mm/s) Z axis feedrate (not used for delta
printers)
#endif

/**
 * Clean Nozzle Feature -- EXPERIMENTAL
 *
 * Adds the G12 command to perform a nozzle cleaning process.
 *
 * Parameters:
 * P Pattern
 * S Strokes / Repetitions
 * T Triangles (P1 only)
 *
 * Patterns:
 * P0 Straight line (default). This process requires a sponge type material
  at a fixed bed location. "S" specifies strokes (i.e. back-forth motions)
  between the start / end points.
 *
 * P1 Zig-zag pattern between (X0, Y0) and (X1, Y1), "T" specifies the
  number of zig-zag triangles to do. "S" defines the number of strokes.
  Zig-zags are done in whichever is the narrower dimension.
  For example, "G12 P1 S1 T3" will execute:
  --
  | (X0, Y1) |  \   \   \   | (X1, Y1)
  |           | / \ / \ / \ |
  A |         | / \ / \ / \ |
  |           | / \ / \ / \ |
  | (X0, Y0) | /   V   V   \ | (X1, Y0)
  --+-----+-----+-----+
      |         |         |         |
      T1        T2        T3
  --
 *
 * P2 Circular pattern with middle at NOZZLE_CLEAN_CIRCLE_MIDDLE.
  "R" specifies the radius. "S" specifies the stroke count.
  Before starting, the nozzle moves to NOZZLE_CLEAN_START_POINT.
 *
 * Caveats: The ending Z should be the same as starting Z.
 * Attention: EXPERIMENTAL. G-code arguments may change.
 */

```

```

// #define NOZZLE_CLEAN_FEATURE

#if ENABLED(NOZZLE_CLEAN_FEATURE)
  // Default number of pattern repetitions
  #define NOZZLE_CLEAN_STROKES 12

  // Default number of triangles
  #define NOZZLE_CLEAN_TRIANGLES 3

  // Specify positions for each tool as { { X, Y, Z }, { X, Y, Z } }
  // Dual hotend system may use { { -20, (Y_BED_SIZE / 2), (Z_MIN_POS + 1) }, { 420,
  (Y_BED_SIZE / 2), (Z_MIN_POS + 1) } }
  #define NOZZLE_CLEAN_START_POINT { { 30, 30, (Z_MIN_POS + 1) } }
  #define NOZZLE_CLEAN_END_POINT { { 100, 60, (Z_MIN_POS + 1) } }

  // Circular pattern radius
  #define NOZZLE_CLEAN_CIRCLE_RADIUS 6.5
  // Circular pattern circle fragments number
  #define NOZZLE_CLEAN_CIRCLE_FN 10
  // Middle point of circle
  #define NOZZLE_CLEAN_CIRCLE_MIDDLE NOZZLE_CLEAN_START_POINT

  // Move the nozzle to the initial position after cleaning
  #define NOZZLE_CLEAN_GOBACK

  // For a purge/clean station that's always at the gantry height (thus no Z move)
  // #define NOZZLE_CLEAN_NO_Z

  // For a purge/clean station mounted on the X axis
  // #define NOZZLE_CLEAN_NO_Y

  // Explicit wipe G-code script applies to a G12 with no arguments.
  // #define WIPE_SEQUENCE_COMMANDS "G1 X-17 Y25 Z10 F4000\nG1 Z1\nM114\nG1 X-
  17 Y25\nG1 X-17 Y95\nG1 X-17 Y25\nG1 X-17 Y95\nG1 X-17 Y25\nG1 X-17 Y95\nG1 X-17
  Y25\nG1 X-17 Y95\nG1 X-17 Y25\nG1 X-17 Y95\nG1 X-17 Y25\nG1 X-17 Y95\nG1
  Z15\nM400\nG0 X-10.0 Y-9.0"

#endif

/**
 * Print Job Timer
 *
 * Automatically start and stop the print job timer on M104/M109/M190.
 *
 * M104 (hotend, no wait) - high temp = none, low temp = stop timer
 * M109 (hotend, wait) - high temp = start timer, low temp = stop timer
 * M190 (bed, wait) - high temp = start timer, low temp = none
 *
 * The timer can also be controlled with the following commands:
 *
 * M75 - Start the print job timer
 * M76 - Pause the print job timer
 * M77 - Stop the print job timer
 */
#define PRINTJOB_TIMER_AUTOSTART

/**
 * Print Counter
 *
 * Track statistical data such as:
 *
 * - Total print jobs
 * - Total successful print jobs
 * - Total failed print jobs
 * - Total time printing
 *
 * View the current statistics with M78.

```

```

*/
##define PRINTCOUNTER

/**
 * Password
 *
 * Set a numerical password for the printer which can be requested:
 *
 * - When the printer boots up
 * - Upon opening the 'Print from Media' Menu
 * - When SD printing is completed or aborted
 *
 * The following G-codes can be used:
 *
 * M510 - Lock Printer. Blocks all commands except M511.
 * M511 - Unlock Printer.
 * M512 - Set, Change and Remove Password.
 *
 * If you forget the password and get locked out you'll need to re-flash
 * the firmware with the feature disabled, reset EEPROM, and (optionally)
 * re-flash the firmware again with this feature enabled.
 */
##define PASSWORD_FEATURE
#if ENABLE(PASSWORD_FEATURE)
  #define PASSWORD_LENGTH 4 // (#) Number of digits (1-9). 3 or 4 is
recommended
  #define PASSWORD_ON_STARTUP
  #define PASSWORD_UNLOCK_GCODE // Unlock with the M511 P<password>
command. Disable to prevent brute-force attack.
  #define PASSWORD_CHANGE_GCODE // Change the password with M512 P<old>
S<new>.
  ##define PASSWORD_ON_SD_PRINT_MENU // This does not prevent gcodes from
running
  ##define PASSWORD_AFTER_SD_PRINT_END
  ##define PASSWORD_AFTER_SD_PRINT_ABORT
  ##include "Configuration_Secure.h" // External file with PASSWORD_DEFAULT_VALUE
#endif

//=====
=====
//===== LCD and SD support
=====
//=====
=====

// @section lcd

/**
 * LCD LANGUAGE
 *
 * Select the language to display on the LCD. These languages are available:
 *
 * en, an, bg, ca, cz, da, de, el, el_gr, es, eu, fi, fr, gl, hr, hu, it,
 * jp_kana, ko_KR, nl, pl, pt, pt_br, ro, ru, sk, tr, uk, vi, zh_CN, zh_TW, test
 *
 * :{ 'en':'English', 'an':'Aragonese', 'bg':'Bulgarian', 'ca':'Catalan', 'cz':'Czech', 'da':'Danish',
'de':'German', 'el':'Greek', 'el_gr':'Greek (Greece)', 'es':'Spanish', 'eu':'Basque-Euskera',
'fi':'Finnish', 'fr':'French', 'gl':'Galician', 'hr':'Croatian', 'hu':'Hungarian', 'it':'Italian',
'jp_kana':'Japanese', 'ko_KR':'Korean (South Korea)', 'nl':'Dutch', 'pl':'Polish', 'pt':'Portuguese',
'pt_br':'Portuguese (Brazilian)', 'ro':'Romanian', 'ru':'Russian', 'sk':'Slovak', 'tr':'Turkish',
'uk':'Ukrainian', 'vi':'Vietnamese', 'zh_CN':'Chinese (Simplified)', 'zh_TW':'Chinese (Traditional)',
'test':'TEST' }
 */
#define LCD_LANGUAGE en

/**
 * LCD Character Set

```



```

*
* Note: This option is NOT applicable to Graphical Displays.
*
* All character-based LCDs provide ASCII plus one of these
* language extensions:
*
* - JAPANESE ... the most common
* - WESTERN ... with more accented characters
* - CYRILLIC ... for the Russian language
*
* To determine the language extension installed on your controller:
*
* - Compile and upload with LCD_LANGUAGE set to 'test'
* - Click the controller to view the LCD menu
* - The LCD will display Japanese, Western, or Cyrillic text
*
* See https://marlinfw.org/docs/development/lcd\_language.html
*
* :['JAPANESE', 'WESTERN', 'CYRILLIC']
*/
#define DISPLAY_CHARSET_HD44780 JAPANESE

/**
 * Info Screen Style (0:Classic, 1:Průša)
 *
 * :[0:'Classic', 1:'Průša']
 */
#define LCD_INFO_SCREEN_STYLE 1

/**
 * SD CARD
 *
 * SD Card support is disabled by default. If your controller has an SD slot,
 * you must uncomment the following option or it won't work.
 */
#define SDSUPPORT

/**
 * SD CARD: SPI SPEED
 *
 * Enable one of the following items for a slower SPI transfer speed.
 * This may be required to resolve "volume init" errors.
 */
// #define SPI_SPEED SPI_HALF_SPEED
// #define SPI_SPEED SPI_QUARTER_SPEED
// #define SPI_SPEED SPI_EIGHTH_SPEED

/**
 * SD CARD: ENABLE CRC
 *
 * Use CRC checks and retries on the SD communication.
 */
// #define SD_CHECK_AND_RETRY

/**
 * LCD Menu Items
 *
 * Disable all menus and only display the Status Screen, or
 * just remove some extraneous menu items to recover space.
 */
// #define NO_LCD_MENUS
// #define SLIM_LCD_MENUS

//
// ENCODER SETTINGS
//
// This option overrides the default number of encoder pulses needed to

```

```

// produce one step. Should be increased for high-resolution encoders.
//
#define ENCODER_PULSES_PER_STEP 4

//
// Use this option to override the number of step signals required to
// move between next/prev menu items.
//
#define ENCODER_STEPS_PER_MENU_ITEM 1

/**
 * Encoder Direction Options
 *
 * Test your encoder's behavior first with both options disabled.
 *
 * Reversed Value Edit and Menu Nav? Enable REVERSE_ENCODER_DIRECTION.
 * Reversed Menu Navigation only? Enable REVERSE_MENU_DIRECTION.
 * Reversed Value Editing only? Enable BOTH options.
 */

//
// This option reverses the encoder direction everywhere.
//
// Set this option if CLOCKWISE causes values to DECREASE
//
#define REVERSE_ENCODER_DIRECTION

//
// This option reverses the encoder direction for navigating LCD menus.
//
// If CLOCKWISE normally moves DOWN this makes it go UP.
// If CLOCKWISE normally moves UP this makes it go DOWN.
//
#define REVERSE_MENU_DIRECTION

//
// This option reverses the encoder direction for Select Screen.
//
// If CLOCKWISE normally moves LEFT this makes it go RIGHT.
// If CLOCKWISE normally moves RIGHT this makes it go LEFT.
//
#define REVERSE_SELECT_DIRECTION

//
// Individual Axis Homing
//
// Add individual axis homing items (Home X, Home Y, and Home Z) to the LCD menu.
//
#define INDIVIDUAL_AXIS_HOMING_MENU

//
// SPEAKER/BUZZER
//
// If you have a speaker that can produce tones, enable it here.
// By default Marlin assumes you have a buzzer with a fixed frequency.
//
#define SPEAKER

//
// The duration and frequency for the UI feedback sound.
// Set these to 0 to disable audio feedback in the LCD menus.
//
// Note: Test audio output with the G-Code:
// M300 S<frequency Hz> P<duration ms>
//
#define LCD_FEEDBACK_FREQUENCY_DURATION_MS 2
#define LCD_FEEDBACK_FREQUENCY_HZ 5000

```

```

//=====
=====
//===== LCD / Controller Selection =====
//===== (Character-based LCDs) =====
//=====
=====

//
// RepRapDiscount Smart Controller.
// https://reprap.org/wiki/RepRapDiscount_Smart_Controller
//
// Note: Usually sold with a white PCB.
//
#define REPRAP_DISCOUNT_SMART_CONTROLLER

//
// Original RADDs LCD Display+Encoder+SDCardReader
// http://doku.radds.org/dokumentation/lcd-display/
//
//define RADDs_DISPLAY

//
// ULTIMAKER Controller.
//
//define ULTIMAKERCONTROLLER

//
// ULTIPANEL as seen on Thingiverse.
//
//define ULTIPANEL

//
// PanelOne from T3P3 (via RAMPS 1.4 AUX2/AUX3)
// https://reprap.org/wiki/PanelOne
//
//define PANEL_ONE

//
// GADGETS3D G3D LCD/SD Controller
// https://reprap.org/wiki/RAMPS_1.3/1.4_GADGETS3D_Shield_with_Panel
//
// Note: Usually sold with a blue PCB.
//
//define G3D_PANEL

//
// RigidBot Panel V1.0
// http://www.inventapart.com/
//
//define RIGIDBOT_PANEL

//
// Makeboard 3D Printer Parts 3D Printer Mini Display 1602 Mini Controller
// https://www.aliexpress.com/item/32765887917.html
//
//define MAKEBOARD_MINI_2_LINE_DISPLAY_1602

//
// ANET and Tronxy 20x4 Controller
//
//define ZONESTAR_LCD // Requires ADC_KEYPAD_PIN to be assigned to an analog
pin.
// This LCD is known to be susceptible to electrical interference
// which scrambles the display. Pressing any button clears it up.
// This is a LCD2004 display with 5 analog buttons.

```

```

//
// Generic 16x2, 16x4, 20x2, or 20x4 character-based LCD.
//
#define ULTRA_LCD

//=====
//===== LCD / Controller Selection =====
//===== (I2C and Shift-Register LCDs) =====
//=====
//=====

//
// CONTROLLER TYPE: I2C
//
// Note: These controllers require the installation of Arduino's LiquidCrystal_I2C
// library. For more info: https://github.com/kiyoshigawa/LiquidCrystal_I2C
//

//
// Elefu RA Board Control Panel
// http://www.elefu.com/index.php?route=product/product&product_id=53
//
#define RA_CONTROL_PANEL

//
// Sainsmart (YwRobot) LCD Displays
//
// These require F.Malpartida's LiquidCrystal_I2C library
// https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home
//
#define LCD_SAINSMART_I2C_1602
#define LCD_SAINSMART_I2C_2004

//
// Generic LCM1602 LCD adapter
//
#define LCM1602

//
// PANELOLU2 LCD with status LEDs,
// separate encoder and click inputs.
//
// Note: This controller requires Arduino's LiquidTWI2 library v1.2.3 or later.
// For more info: https://github.com/lincomatic/LiquidTWI2
//
// Note: The PANELOLU2 encoder click input can either be directly connected to
// a pin (if BTN_ENC defined to != -1) or read through I2C (when BTN_ENC == -1).
//
#define LCD_I2C_PANOLU2

//
// Panucatt VIKI LCD with status LEDs,
// integrated click & L/R/U/D buttons, separate encoder inputs.
//
#define LCD_I2C_VIKI

//
// CONTROLLER TYPE: Shift register panels
//

//
// 2-wire Non-latching LCD SR from https://goo.gl/aJJ4sH
// LCD configuration: https://reprap.org/wiki/SAV_3D_LCD
//
#define SAV_3DLCD

```

```

//
// 3-wire SR LCD with strobe using 74HC4094
// https://github.com/mikeshub/SailfishLCD
// Uses the code directly from Sailfish
//
//define FF_INTERFACEBOARD

//
// TFT GLCD Panel with Marlin UI
// Panel connected to main board by SPI or I2C interface.
// See https://github.com/Serhiy-K/TFTGLCDAdapter
//
//define TFTGLCD_PANEL_SPI
//define TFTGLCD_PANEL_I2C

//=====
=====
//===== LCD / Controller Selection =====
//===== (Graphical LCDs) =====
//=====
=====

//
// CONTROLLER TYPE: Graphical 128x64 (DOGM)
//
// IMPORTANT: The U8glib library is required for Graphical Display!
// https://github.com/olikraus/U8glib_Arduino
//
// NOTE: If the LCD is unresponsive you may need to reverse the plugs.
//

//
// RepRapDiscount FULL GRAPHIC Smart Controller
// https://reprap.org/wiki/RepRapDiscount_Full_Graphic_Smart_Controller
//
//define REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER

//
// ReprapWorld Graphical LCD
// https://reprapworld.com/?products_details&products_id/1218
//
//define REPRAPWORLD_GRAPHICAL_LCD

//
// Activate one of these if you have a Panucatt Devices
// Viki 2.0 or mini Viki with Graphic LCD
// https://www.panucatt.com
//
//define VIKI2
//define miniVIKI

//
// MakerLab Mini Panel with graphic
// controller and SD support - https://reprap.org/wiki/Mini_panel
//
//define MINIPANEL

//
// MaKr3d MaKr-Panel with graphic controller and SD support.
// https://reprap.org/wiki/MaKr3d_MaKrPanel
//
//define MAKRPANEL

//
// Adafruit ST7565 Full Graphic Controller.
// https://github.com/eboston/Adafruit-ST7565-Full-Graphic-Controller/
//

```

```

// #define ELB_FULL_GRAPHIC_CONTROLLER

//
// BQ LCD Smart Controller shipped by
// default with the BQ Hephestos 2 and Witbox 2.
//
// #define BQ_LCD_SMART_CONTROLLER

//
// Cartesio UI
// http://mauk.cc/webshop/cartesio-shop/electronics/user-interface
//
// #define CARTESIO_UI

//
// LCD for Melzi Card with Graphical LCD
//
// #define LCD_FOR_MELZI

//
// Original Ulticontroller from Ultimaker 2 printer with SSD1309 I2C display and encoder
// https://github.com/Ultimaker/Ultimaker2/tree/master/1249_Ulticontroller_Board_(x1)
//
// #define ULTI_CONTROLLER

//
// MKS MINI12864 with graphic controller and SD support
// https://reprap.org/wiki/MKS_MINI_12864
//
// #define MKS_MINI_12864

//
// MKS LCD12864A/B with graphic controller and SD support. Follows MKS_MINI_12864
// pinout.
// https://www.aliexpress.com/item/33018110072.html
//
// #define MKS_LCD12864

//
// FYSETC variant of the MINI12864 graphic controller with SD support
// https://wiki.fysetc.com/Mini12864_Panel/
//
// #define FYSETC_MINI_12864_X_X // Type C/D/E/F. No tunable RGB Backlight by default
// #define FYSETC_MINI_12864_1_2 // Type C/D/E/F. Simple RGB Backlight (always on)
// #define FYSETC_MINI_12864_2_0 // Type A/B. Discreet RGB Backlight
// #define FYSETC_MINI_12864_2_1 // Type A/B. NeoPixel RGB Backlight
// #define FYSETC_GENERIC_12864_1_1 // Larger display with basic ON/OFF backlight.

//
// Factory display for Creality CR-10
// https://www.aliexpress.com/item/32833148327.html
//
// This is RAMPS-compatible using a single 10-pin connector.
// (For CR-10 owners who want to replace the Melzi Creality board but retain the display)
//
// #define CR10_STOCKDISPLAY

//
// Ender-2 OEM display, a variant of the MKS_MINI_12864
//
// #define ENDER2_STOCKDISPLAY

//
// ANET and Tronxy Graphical Controller
//
// Anet 128x64 full graphics lcd with rotary encoder as used on Anet A6
// A clone of the RepRapDiscount full graphics display but with

```

```

// different pins/wiring (see pins_ANET_10.h).
//
// #define ANET_FULL_GRAPHICS_LCD

//
// AZSMZ 12864 LCD with SD
// https://www.aliexpress.com/item/32837222770.html
//
// #define AZSMZ_12864

//
// Silvergate GLCD controller
// https://github.com/android444/Silvergate
//
// #define SILVER_GATE_GLCD_CONTROLLER

//=====
=====
//===== OLED
Displays =====
//=====
=====

//
// SSD1306 OLED full graphics generic display
//
// #define U8GLIB_SSD1306

//
// SAV OLEd LCD module support using either SSD1306 or SH1106 based LCD modules
//
// #define SAV_3DGLCD
// #if ENABLED(SAV_3DGLCD)
//   #define U8GLIB_SSD1306
//   // #define U8GLIB_SH1106
// #endif

//
// TinyBoy2 128x64 OLED / Encoder Panel
//
// #define OLED_PANEL_TINYBOY2

//
// MKS OLED 1.3" 128x64 FULL GRAPHICS CONTROLLER
// https://reprap.org/wiki/MKS_12864OLED
//
// Tiny, but very sharp OLED display
//
// #define MKS_12864OLED // Uses the SH1106 controller (default)
// #define MKS_12864OLED_SSD1306 // Uses the SSD1306 controller

//
// Zonestar OLED 128x64 FULL GRAPHICS CONTROLLER
//
// #define ZONESTAR_12864LCD // Graphical (DOGM) with ST7920 controller
// #define ZONESTAR_12864OLED // 1.3" OLED with SH1106 controller (default)
// #define ZONESTAR_12864OLED_SSD1306 // 0.96" OLED with SSD1306 controller

//
// Einstart S OLED SSD1306
//
// #define U8GLIB_SH1106_EINSTART

//
// Overlord OLED display/controller with i2c buzzer and LEDs
//
// #define OVERLORD_OLED

```

```

//
// FYSETC OLED 2.42" 128x64 FULL GRAPHICS CONTROLLER with WS2812 RGB
// Where to find : https://www.aliexpress.com/item/4000345255731.html
// #define FYSETC_242_OLED_12864 // Uses the SSD1309 controller

//=====
//===== Extensible UI Displays =====
//=====
//=====

//
// DGUS Touch Display with DWIN OS. (Choose one.)
// ORIGIN : https://www.aliexpress.com/item/32993409517.html
// FYSETC : https://www.aliexpress.com/item/32961471929.html
//
// #define DGUS_LCD_UI_ORIGIN
// #define DGUS_LCD_UI_FYSETC
// #define DGUS_LCD_UI_HIPRECY

//
// Touch-screen LCD for Malyan M200/M300 printers
//
// #define MALYAN_LCD
// #if ENABLED(MALYAN_LCD)
//   #define LCD_SERIAL_PORT 1 // Default is 1 for Malyan M200
// #endif

//
// Touch UI for FTDI EVE (FT800/FT810) displays
// See Configuration_adv.h for all configuration options.
//
// #define TOUCH_UI_FTDI_EVE

//
// Touch-screen LCD for Anycubic printers
//
// #define ANYCUBIC_LCD_I3MEGA
// #define ANYCUBIC_LCD_CHIRON
// #if EITHER(ANYCUBIC_LCD_I3MEGA, ANYCUBIC_LCD_CHIRON)
//   #define LCD_SERIAL_PORT 3 // Default is 3 for Anycubic
//   #define ANYCUBIC_LCD_DEBUG
// #endif

//
// Third-party or vendor-customized controller interfaces.
// Sources should be installed in 'src/lcd/extensible_ui'.
//
// #define EXTENSIBLE_UI

// #if ENABLED(EXTENSIBLE_UI)
//   #define EXTUI_LOCAL_BEEPER // Enables use of local Beeper pin with external display
// #endif

//=====
//===== Graphical TFTs =====
//=====
//=====

/**
 * TFT Type - Select your Display type
 *
 * Available options are:
 * MKS_TS35_V2_0,

```



```

* MKS_ROBIN_TFT24, MKS_ROBIN_TFT28, MKS_ROBIN_TFT32, MKS_ROBIN_TFT35,
* MKS_ROBIN_TFT43, MKS_ROBIN_TFT_V1_1R
* TFT_TRONXY_X5SA, ANYCUBIC_TFT35, LONGER_LK_TFT28
* TFT_GENERIC
*
* For TFT_GENERIC, you need to configure these 3 options:
* Driver: TFT_DRIVER
* Current Drivers are: AUTO, ST7735, ST7789, ST7796, R61505, ILI9328, ILI9341,
ILI9488
* Resolution: TFT_WIDTH and TFT_HEIGHT
* Interface: TFT_INTERFACE_FSMC or TFT_INTERFACE_SPI
*/
#define TFT_GENERIC

/**
* TFT UI - User Interface Selection. Enable one of the following options:
*
* TFT_CLASSIC_UI - Emulated DOGM - 128x64 Upscaled
* TFT_COLOR_UI - Marlin Default Menus, Touch Friendly, using full TFT capabilities
* TFT_LVGL_UI - A Modern UI using LVGL
*
* For LVGL_UI also copy the 'assets' folder from the build directory to the
* root of your SD card, together with the compiled firmware.
*/
#define TFT_CLASSIC_UI
#define TFT_COLOR_UI
#define TFT_LVGL_UI

/**
* TFT Rotation. Set to one of the following values:
*
* TFT_ROTATE_90, TFT_ROTATE_90_MIRROR_X, TFT_ROTATE_90_MIRROR_Y,
* TFT_ROTATE_180, TFT_ROTATE_180_MIRROR_X, TFT_ROTATE_180_MIRROR_Y,
* TFT_ROTATE_270, TFT_ROTATE_270_MIRROR_X, TFT_ROTATE_270_MIRROR_Y,
* TFT_MIRROR_X, TFT_MIRROR_Y, TFT_NO_ROTATION
*/
#define TFT_ROTATION TFT_NO_ROTATION

//=====
//===== Other Controllers =====
//=====

//
// Ender-3 v2 OEM display. A DWIN display with Rotary Encoder.
//
#define DWIN_CREALITY_LCD

//
// ADS7843/XPT2046 ADC Touchscreen such as ILI9341 2.8
//
#define TOUCH_SCREEN
#if ENABLED(TOUCH_SCREEN)
  #define BUTTON_DELAY_EDIT 50 // (ms) Button repeat delay for edit screens
  #define BUTTON_DELAY_MENU 250 // (ms) Button repeat delay for menus

  #define TOUCH_SCREEN_CALIBRATION

  #define XPT2046_X_CALIBRATION 12316
  #define XPT2046_Y_CALIBRATION -8981
  #define XPT2046_X_OFFSET -43
  #define XPT2046_Y_OFFSET 257
#endif

//
// RepRapWorld REPRAPWORLD_KEYPAD v1.1

```

```

// https://reprapworld.com/products/electronics/ramps/keypad_v1_0_fully_assembled/
//
// #define REPRAPWORLD_KEYPAD
// #define REPRAPWORLD_KEYPAD_MOVE_STEP 10.0 // (mm) Distance to move per key-
// press

// =====
// ===== Extra Features
// =====
// =====

// @section extras

// Set number of user-controlled fans. Disable to use all board-defined fans.
// :[1,2,3,4,5,6,7,8]
// #define NUM_M106_FANS 1

// Increase the FAN PWM frequency. Removes the PWM noise but increases heating in the
// FET/Arduino
// #define FAST_PWM_FAN

// Use software PWM to drive the fan, as for the heaters. This uses a very low frequency
// which is not as annoying as with the hardware PWM. On the other hand, if this frequency
// is too low, you should also increment SOFT_PWM_SCALE.
// #define FAN_SOFT_PWM

// Incrementing this by 1 will double the software PWM frequency,
// affecting heaters, and the fan if FAN_SOFT_PWM is enabled.
// However, control resolution will be halved for each increment;
// at zero value, there are 128 effective control positions.
// :[0,1,2,3,4,5,6,7]
// #define SOFT_PWM_SCALE 0

// If SOFT_PWM_SCALE is set to a value higher than 0, dithering can
// be used to mitigate the associated resolution loss. If enabled,
// some of the PWM cycles are stretched so on average the desired
// duty cycle is attained.
// #define SOFT_PWM_DITHER

// Temperature status LEDs that display the hotend and bed temperature.
// If all hotends, bed temperature, and target temperature are under 54C
// then the BLUE led is on. Otherwise the RED led is on. (1C hysteresis)
// #define TEMP_STAT_LEDS

// Support for the BariCUDA Paste Extruder
// #define BARICUDA

// Support for BlinkM/CyzRgb
// #define BLINKM

// Support for PCA9632 PWM LED driver
// #define PCA9632

// Support for PCA9533 PWM LED driver
// #define PCA9533

/**
 * RGB LED / LED Strip Control
 *
 * Enable support for an RGB LED connected to 5V digital pins, or
 * an RGB Strip connected to MOSFETs controlled by digital pins.
 *
 * Adds the M150 command to set the LED (or LED strip) color.
 * If pins are PWM capable (e.g., 4, 5, 6, 11) then a range of
 * luminance values can be set from 0 to 255.

```

```

* For NeoPixel LED an overall brightness parameter is also available.
*
* *** CAUTION ***
* LED Strips require a MOSFET Chip between PWM lines and LEDs,
* as the Arduino cannot handle the current the LEDs will require.
* Failure to follow this precaution can destroy your Arduino!
* NOTE: A separate 5V power supply is required! The NeoPixel LED needs
* more current than the Arduino 5V linear regulator can produce.
* *** CAUTION ***
*
* LED Type. Enable only one of the following two options.
*/
#define RGB_LED
#define RGBW_LED

#if EITHER(RGB_LED, RGBW_LED)
  #define RGB_LED_R_PIN 34
  #define RGB_LED_G_PIN 43
  #define RGB_LED_B_PIN 35
  #define RGB_LED_W_PIN -1
#endif

// Support for Adafruit NeoPixel LED driver
#define NEOPIXEL_LED
#if ENABLED(NEOPIXEL_LED)
  #define NEOPIXEL_TYPE NEO_GRBW // NEO_GRBW / NEO_GRB - four/three channel
  driver type (defined in Adafruit_NeoPixel.h)
  #define NEOPIXEL_PIN 4 // LED driving pin
  #define NEOPIXEL2_TYPE NEOPIXEL_TYPE
  #define NEOPIXEL2_PIN 5
  #define NEOPIXEL_PIXELS 30 // Number of LEDs in the strip. (Longest strip when
  NEOPIXEL2_SEPARATE is disabled.)
  #define NEOPIXEL_IS_SEQUENTIAL // Sequential display for temperature change - LED by
  LED. Disable to change all LEDs at once.
  #define NEOPIXEL_BRIGHTNESS 127 // Initial brightness (0-255)
  #define NEOPIXEL_STARTUP_TEST // Cycle through colors at startup

  // Support for second Adafruit NeoPixel LED driver controlled with M150 S1 ...
  #define NEOPIXEL2_SEPARATE
  #if ENABLED(NEOPIXEL2_SEPARATE)
    #define NEOPIXEL2_PIXELS 15 // Number of LEDs in the second strip
    #define NEOPIXEL2_BRIGHTNESS 127 // Initial brightness (0-255)
    #define NEOPIXEL2_STARTUP_TEST // Cycle through colors at startup
  #else
    #define NEOPIXEL2_INSERTIES // Default behavior is NeoPixel 2 in parallel
  #endif

  // Use a single NeoPixel LED for static (background) lighting
  #define NEOPIXEL_BKGD_LED_INDEX 0 // Index of the LED to use
  #define NEOPIXEL_BKGD_COLOR { 255, 255, 255, 0 } // R, G, B, W
#endif

/**
* Printer Event LEDs
*
* During printing, the LEDs will reflect the printer status:
*
* - Gradually change from blue to violet as the heated bed gets to target temp
* - Gradually change from violet to red as the hotend gets to temperature
* - Change to white to illuminate work surface
* - Change to green once print has finished
* - Turn off after the print has finished and the user has pushed a button
*/
#if ANY(BLINKM, RGB_LED, RGBW_LED, PCA9632, PCA9533, NEOPIXEL_LED)
  #define PRINTER_EVENT_LEDS
#endif

```

```
/**
 * Number of servos
 *
 * For some servo-related options NUM_SERVOS will be set automatically.
 * Set this manually if there are extra servos needing manual control.
 * Set to 0 to turn off servo support.
 */
#define NUM_SERVOS 3 // Servo index starts with 0 for M280 command

// (ms) Delay before the next move will start, to give the servo time to reach its target angle.
// 300ms is a good value but you can try less delay.
// If the servo can't reach the requested position, increase it.
#define SERVO_DELAY { 300 }

// Only power servos during movement, otherwise leave off to prevent jitter
#define DEACTIVATE_SERVOS_AFTER_MOVE

// Edit servo angles with M281 and save to EEPROM with M500
#define EDITABLE_SERVO_ANGLES
```

APPENDIX E: Marlin Firmware Configuration_adv.h

```

/**
 * Marlin 3D Printer Firmware
 * Copyright (c) 2020 MarlinFirmware [https://github.com/MarlinFirmware/Marlin]
 *
 * Based on Sprinter and grbl.
 * Copyright (c) 2011 Camiel Gubbels / Erik van der Zalm
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <https://www.gnu.org/licenses/>.
 */
#pragma once

/**
 * Configuration_adv.h
 *
 * Advanced settings.
 * Only change these if you know exactly what you're doing.
 * Some of these settings can damage your printer if improperly set!
 *
 * Basic settings can be found in Configuration.h
 */
#define CONFIGURATION_ADV_H_VERSION 020007

//=====
//===== Thermal Settings =====
//=====
// @section temperature

/**
 * Thermocouple sensors are quite sensitive to noise. Any noise induced in
 * the sensor wires, such as by stepper motor wires run in parallel to them,
 * may result in the thermocouple sensor reporting spurious errors. This
 * value is the number of errors which can occur in a row before the error
 * is reported. This allows us to ignore intermittent error conditions while
 * still detecting an actual failure, which should result in a continuous
 * stream of errors from the sensor.
 *
 * Set this value to 0 to fail on the first error to occur.
 */
#define THERMOCOUPLE_MAX_ERRORS 15

//
// Custom Thermistor 1000 parameters
//
#if TEMP_SENSOR_0 == 1000
#define HOTEND0_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
#define HOTEND0_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
#define HOTEND0_BETA 3950 // Beta value
#endif

#if TEMP_SENSOR_1 == 1000
#define HOTEND1_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
#define HOTEND1_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
#define HOTEND1_BETA 3950 // Beta value
#endif

```

```

#if TEMP_SENSOR_2 == 1000
  #define HOTEND2_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
  #define HOTEND2_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
  #define HOTEND2_BETA 3950 // Beta value
#endif

#if TEMP_SENSOR_3 == 1000
  #define HOTEND3_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
  #define HOTEND3_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
  #define HOTEND3_BETA 3950 // Beta value
#endif

#if TEMP_SENSOR_4 == 1000
  #define HOTEND4_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
  #define HOTEND4_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
  #define HOTEND4_BETA 3950 // Beta value
#endif

#if TEMP_SENSOR_5 == 1000
  #define HOTEND5_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
  #define HOTEND5_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
  #define HOTEND5_BETA 3950 // Beta value
#endif

#if TEMP_SENSOR_6 == 1000
  #define HOTEND6_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
  #define HOTEND6_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
  #define HOTEND6_BETA 3950 // Beta value
#endif

#if TEMP_SENSOR_7 == 1000
  #define HOTEND7_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
  #define HOTEND7_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
  #define HOTEND7_BETA 3950 // Beta value
#endif

#if TEMP_SENSOR_BED == 1000
  #define BED_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
  #define BED_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
  #define BED_BETA 3950 // Beta value
#endif

#if TEMP_SENSOR_CHAMBER == 1000
  #define CHAMBER_PULLUP_RESISTOR_OHMS 4700 // Pullup resistor
  #define CHAMBER_RESISTANCE_25C_OHMS 100000 // Resistance at 25C
  #define CHAMBER_BETA 3950 // Beta value
#endif

//
// Hephestos 2 24V heated bed upgrade kit.
// https://store.bq.com/en/heated-bed-kit-hephestos2
//
// #define HEPHESTOS2_HEATED_BED_KIT
// #if ENABLED(HEPHESTOS2_HEATED_BED_KIT)
//   #undef TEMP_SENSOR_BED
//   #define TEMP_SENSOR_BED 70
//   #define HEATER_BED_INVERTING true
// #endif

//
// Heated Bed Bang-Bang options
//
// #if DISABLED(PIDTEMPBED)
//   #define BED_CHECK_INTERVAL 5000 // (ms) Interval between checks in bang-bang control
//   #if ENABLED(BED_LIMIT_SWITCHING)
//     #define BED_HYSTERESIS 2 // (°C) Only set the relevant heater state when ABS(T-target) >
//     BED_HYSTERESIS
//   #endif
// #endif

```

```

//
// Heated Chamber options
//
#if TEMP_SENSOR_CHAMBER
#define CHAMBER_MINTEMP      5
#define CHAMBER_MAXTEMP     60
#define TEMP_CHAMBER_HYSTERESIS 1 // (°C) Temperature proximity considered "close enough"
to the target
// #define CHAMBER_LIMIT_SWITCHING
// #define HEATER_CHAMBER_PIN 44 // Chamber heater on/off pin
// #define HEATER_CHAMBER_INVERTING false

// #define CHAMBER_FAN          // Enable a fan on the chamber
#if ENABLED(CHAMBER_FAN)
#define CHAMBER_FAN_MODE 2 // Fan control mode: 0=Static; 1=Linear increase when temp is
higher than target; 2=V-shaped curve.
#if CHAMBER_FAN_MODE == 0
#define CHAMBER_FAN_BASE 255 // Chamber fan PWM (0-255)
#elif CHAMBER_FAN_MODE == 1
#define CHAMBER_FAN_BASE 128 // Base chamber fan PWM (0-255); turns on when chamber
temperature is above the target
#define CHAMBER_FAN_FACTOR 25 // PWM increase per °C above target
#elif CHAMBER_FAN_MODE == 2
#define CHAMBER_FAN_BASE 128 // Minimum chamber fan PWM (0-255)
#define CHAMBER_FAN_FACTOR 25 // PWM increase per °C difference from target
#endif
#endif

// #define CHAMBER_VENT          // Enable a servo-controlled vent on the chamber
#if ENABLED(CHAMBER_VENT)
#define CHAMBER_VENT_SERVO_NR 1 // Index of the vent servo
#define HIGH_EXCESS_HEAT_LIMIT 5 // How much above target temp to consider there is excess
heat in the chamber
#define LOW_EXCESS_HEAT_LIMIT 3
#define MIN_COOLING_SLOPE_TIME_CHAMBER_VENT 20
#define MIN_COOLING_SLOPE_DEG_CHAMBER_VENT 1.5
#endif
#endif

/**
 * Thermal Protection provides additional protection to your printer from damage
 * and fire. Marlin always includes safe min and max temperature ranges which
 * protect against a broken or disconnected thermistor wire.
 *
 * The issue: If a thermistor falls out, it will report the much lower
 * temperature of the air in the room, and the the firmware will keep
 * the heater on.
 *
 * The solution: Once the temperature reaches the target, start observing.
 * If the temperature stays too far below the target (hysteresis) for too
 * long (period), the firmware will halt the machine as a safety precaution.
 *
 * If you get false positives for "Thermal Runaway", increase
 * THERMAL_PROTECTION_HYSTERESIS and/or THERMAL_PROTECTION_PERIOD
 */
#if ENABLED(THERMAL_PROTECTION_HOTENDS)
#define THERMAL_PROTECTION_PERIOD 60 // Seconds
#define THERMAL_PROTECTION_HYSTERESIS 6 // Degrees Celsius

#define ADAPTIVE_FAN_SLOWING // Slow part cooling fan if temperature drops
#if BOTH(ADAPTIVE_FAN_SLOWING, PIDTEMP)
#define NO_FAN_SLOWING_IN_PID_TUNING // Don't slow fan speed during M303
#endif

/**
 * Whenever an M104, M109, or M303 increases the target temperature, the
 * firmware will wait for the WATCH_TEMP_PERIOD to expire. If the temperature
 * hasn't increased by WATCH_TEMP_INCREASE degrees, the machine is halted and
 * requires a hard reset. This test restarts with any M104/M109/M303, but only
 * if the current temperature is far enough below the target for a reliable
 * test.
 */

```

```

* If you get false positives for "Heating failed", increase WATCH_TEMP_PERIOD
* and/or decrease WATCH_TEMP_INCREASE. WATCH_TEMP_INCREASE should not be set
* below 2.
*/
#define WATCH_TEMP_PERIOD 20          // Seconds
#define WATCH_TEMP_INCREASE 2        // Degrees Celsius
#endif

/**
 * Thermal Protection parameters for the bed are just as above for hotends.
 */
#if ENABLED(THERMAL_PROTECTION_BED)
#define THERMAL_PROTECTION_BED_PERIOD 20 // Seconds
#define THERMAL_PROTECTION_BED_HYSTERESIS 2 // Degrees Celsius

/**
 * As described above, except for the bed (M140/M190/M303).
 */
#define WATCH_BED_TEMP_PERIOD 60 // Seconds
#define WATCH_BED_TEMP_INCREASE 2 // Degrees Celsius
#endif

/**
 * Thermal Protection parameters for the heated chamber.
 */
#if ENABLED(THERMAL_PROTECTION_CHAMBER)
#define THERMAL_PROTECTION_CHAMBER_PERIOD 20 // Seconds
#define THERMAL_PROTECTION_CHAMBER_HYSTERESIS 2 // Degrees Celsius

/**
 * Heated chamber watch settings (M141/M191).
 */
#define WATCH_CHAMBER_TEMP_PERIOD 60 // Seconds
#define WATCH_CHAMBER_TEMP_INCREASE 2 // Degrees Celsius
#endif

#if ENABLED(PIDTEMP)
// Add an experimental additional term to the heater power, proportional to the extrusion speed.
// A well-chosen Kc value should add just enough power to melt the increased material volume.
// #define PID_EXTRUSION_SCALING
#if ENABLED(PID_EXTRUSION_SCALING)
#define DEFAULT_Kc (100) // heating power = Kc * e_speed
#define LPQ_MAX_LEN 50
#endif
#endif

/**
 * Add an experimental additional term to the heater power, proportional to the fan speed.
 * A well-chosen Kf value should add just enough power to compensate for power-loss from the cooling
 fan.
 * You can either just add a constant compensation with the DEFAULT_Kf value
 * or follow the instruction below to get speed-dependent compensation.
 *
 * Constant compensation (use only with fanspeeds of 0% and 100%)
 * -----
 * A good starting point for the Kf-value comes from the calculation:
 *  $k_f = (power\_fan * eff\_fan) / power\_heater * 255$ 
 * where eff_fan is between 0.0 and 1.0, based on fan-efficiency and airflow to the nozzle / heater.
 *
 * Example:
 * Heater: 40W, Fan: 0.1A * 24V = 2.4W, eff_fan = 0.8
 *  $K_f = (2.4W * 0.8) / 40W * 255 = 12.24$ 
 *
 * Fan-speed dependent compensation
 * -----
 * 1. To find a good Kf value, set the hotend temperature, wait for it to settle, and enable the fan (100%).
 * Make sure PID_FAN_SCALING_LIN_FACTOR is 0 and
PID_FAN_SCALING_ALTERNATIVE_DEFINITION is not enabled.
 * If you see the temperature drop repeat the test, increasing the Kf value slowly, until the temperature
 drop goes away. If the temperature overshoots after enabling the fan, the Kf value is too big.
 * 2. Note the Kf-value for fan-speed at 100%
 * 3. Determine a good value for PID_FAN_SCALING_MIN_SPEED, which is around the speed, where
 the fan starts moving.

```



```

* 4. Repeat step 1. and 2. for this fan speed.
* 5. Enable PID_FAN_SCALING_ALTERNATIVE_DEFINITION and enter the two identified Kf-values in
*   PID_FAN_SCALING_AT_FULL_SPEED and PID_FAN_SCALING_AT_MIN_SPEED. Enter the
minimum speed in PID_FAN_SCALING_MIN_SPEED
*/
#define PID_FAN_SCALING
#if ENABLED(PID_FAN_SCALING)
  #define PID_FAN_SCALING_ALTERNATIVE_DEFINITION
  #if ENABLED(PID_FAN_SCALING_ALTERNATIVE_DEFINITION)
    // The alternative definition is used for an easier configuration.
    // Just figure out Kf at fullspeed (255) and PID_FAN_SCALING_MIN_SPEED.
    // DEFAULT_Kf and PID_FAN_SCALING_LIN_FACTOR are calculated accordingly.

    #define PID_FAN_SCALING_AT_FULL_SPEED
13.0    //PID_FAN_SCALING_LIN_FACTOR*255+DEFAULT_Kf
    #define PID_FAN_SCALING_AT_MIN_SPEED
6.0     //PID_FAN_SCALING_LIN_FACTOR*PID_FAN_SCALING_MIN_SPEED+DEFAULT_Kf
    #define PID_FAN_SCALING_MIN_SPEED 10.0    // Minimum fan speed at which to enable
PID_FAN_SCALING

    #define DEFAULT_Kf (255.0*PID_FAN_SCALING_AT_MIN_SPEED-
PID_FAN_SCALING_AT_FULL_SPEED*PID_FAN_SCALING_MIN_SPEED)/(255.0-
PID_FAN_SCALING_MIN_SPEED)
    #define PID_FAN_SCALING_LIN_FACTOR (PID_FAN_SCALING_AT_FULL_SPEED-
DEFAULT_Kf)/255.0

  #else
    #define PID_FAN_SCALING_LIN_FACTOR (0)    // Power loss due to cooling = Kf * (fan_speed)
    #define DEFAULT_Kf 10                    // A constant value added to the PID-tuner
    #define PID_FAN_SCALING_MIN_SPEED 10     // Minimum fan speed at which to enable
PID_FAN_SCALING
  #endif
#endif
#endif

/**
 * Automatic Temperature Mode
 *
 * Dynamically adjust the hotend target temperature based on planned E moves.
 *
 * (Contrast with PID_EXTRUSION_SCALING, which tracks E movement and adjusts PID
 * behavior using an additional kC value.)
 *
 * Autotemp is calculated by (mintemp + factor * mm_per_sec), capped to maxtemp.
 *
 * Enable Autotemp Mode with M104/M109 F<factor> S<mintemp> B<maxtemp>.
 * Disable by sending M104/M109 with no F parameter (or F0 with AUTOTEMP_PROPORTIONAL).
 */
#define AUTOTEMP
#if ENABLED(AUTOTEMP)
  #define AUTOTEMP_OLDWEIGHT 0.98
  // Turn on AUTOTEMP on M104/M109 by default using proportions set here
  #define AUTOTEMP_PROPORTIONAL
  #if ENABLED(AUTOTEMP_PROPORTIONAL)
    #define AUTOTEMP_MIN_P 0 // (°C) Added to the target temperature
    #define AUTOTEMP_MAX_P 5 // (°C) Added to the target temperature
    #define AUTOTEMP_FACTOR_P 1 // Apply this F parameter by default (overridden by M104/M109 F)
  #endif
#endif

// Show Temperature ADC value
// Enable for M105 to include ADC values read from temperature sensors.
#define SHOW_TEMP_ADC_VALUES

/**
 * High Temperature Thermistor Support
 *
 * Thermistors able to support high temperature tend to have a hard time getting
 * good readings at room and lower temperatures. This means HEATER_X_RAW_LO_TEMP
 * will probably be caught when the heating element first turns on during the
 * preheating process, which will trigger a min_temp_error as a safety measure
 * and force stop everything.

```

```

* To circumvent this limitation, we allow for a preheat time (during which,
* min_temp_error won't be triggered) and add a min_temp buffer to handle
* aberrant readings.
*
* If you want to enable this feature for your hotend thermistor(s)
* uncomment and set values > 0 in the constants below
*/

// The number of consecutive low temperature errors that can occur
// before a min_temp_error is triggered. (Shouldn't be more than 10.)
// #define MAX_CONSECUTIVE_LOW_TEMPERATURE_ERROR_ALLOWED 0

// The number of milliseconds a hotend will preheat before starting to check
// the temperature. This value should NOT be set to the time it takes the
// hot end to reach the target temperature, but the time it takes to reach
// the minimum temperature your thermistor can read. The lower the better/safer.
// This shouldn't need to be more than 30 seconds (30000)
// #define MILLISECONDS_PREHEAT_TIME 0

// @section extruder

// Extruder runout prevention.
// If the machine is idle and the temperature over MINTEMP
// then extrude some filament every couple of SECONDS.
// #define EXTRUDER_RUNOUT_PREVENT
// #if ENABLED(EXTRUDER_RUNOUT_PREVENT)
//   #define EXTRUDER_RUNOUT_MINTEMP 190
//   #define EXTRUDER_RUNOUT_SECONDS 30
//   #define EXTRUDER_RUNOUT_SPEED 1500 // (mm/min)
//   #define EXTRUDER_RUNOUT_EXTRUDE 5 // (mm)
// #endif

/**
 * Hotend Idle Timeout
 * Prevent filament in the nozzle from charring and causing a critical jam.
 */
// #define HOTEND_IDLE_TIMEOUT
// #if ENABLED(HOTEND_IDLE_TIMEOUT)
//   #define HOTEND_IDLE_TIMEOUT_SEC (5*60) // (seconds) Time without extruder movement to trigger
//   protection
//   #define HOTEND_IDLE_MIN_TRIGGER 180 // (°C) Minimum temperature to enable hotend
//   protection
//   #define HOTEND_IDLE_NOZZLE_TARGET 0 // (°C) Safe temperature for the nozzle after timeout
//   #define HOTEND_IDLE_BED_TARGET 0 // (°C) Safe temperature for the bed after timeout
// #endif

// @section temperature

// Calibration for AD595 / AD8495 sensor to adjust temperature measurements.
// The final temperature is calculated as (measuredTemp * GAIN) + OFFSET.
// #define TEMP_SENSOR_AD595_OFFSET 0.0
// #define TEMP_SENSOR_AD595_GAIN 1.0
// #define TEMP_SENSOR_AD8495_OFFSET 0.0
// #define TEMP_SENSOR_AD8495_GAIN 1.0

/**
 * Controller Fan
 * To cool down the stepper drivers and MOSFETs.
 *
 * The fan turns on automatically whenever any driver is enabled and turns
 * off (or reduces to idle speed) shortly after drivers are turned off.
 */
// #define USE_CONTROLLER_FAN
// #if ENABLED(USE_CONTROLLER_FAN)
//   // #define CONTROLLER_FAN_PIN -1 // Set a custom pin for the controller fan
//   // #define CONTROLLER_FAN_USE_Z_ONLY // With this option only the Z axis is considered
//   // #define CONTROLLER_FAN_IGNORE_Z // Ignore Z stepper. Useful when stepper timeout is
//   disabled.
//   #define CONTROLLERFAN_SPEED_MIN 0 // (0-255) Minimum speed. (If set below this value the fan
//   is turned off.)

```

```

#define CONTROLLERFAN_SPEED_ACTIVE 255 // (0-255) Active speed, used when any motor is
enabled
#define CONTROLLERFAN_SPEED_IDLE 0 // (0-255) Idle speed, used when motors are disabled
#define CONTROLLERFAN_IDLE_TIME 60 // (seconds) Extra time to keep the fan running after
disabling motors
// #define CONTROLLER_FAN_EDITABLE // Enable M710 configurable settings
// #if ENABLED(CONTROLLER_FAN_EDITABLE)
// #define CONTROLLER_FAN_MENU // Enable the Controller Fan submenu
// #endif
// #endif

// When first starting the main fan, run it at full speed for the
// given number of milliseconds. This gets the fan spinning reliably
// before setting a PWM value. (Does not work with software PWM for fan on Sanguinololu)
// #define FAN_KICKSTART_TIME 100

// Some coolers may require a non-zero "off" state.
// #define FAN_OFF_PWM 1

/**
 * PWM Fan Scaling
 *
 * Define the min/max speeds for PWM fans (as set with M106).
 *
 * With these options the M106 0-255 value range is scaled to a subset
 * to ensure that the fan has enough power to spin, or to run lower
 * current fans with higher current. (e.g., 5V/12V fans with 12V/24V)
 * Value 0 always turns off the fan.
 *
 * Define one or both of these to override the default 0-255 range.
 */
// #define FAN_MIN_PWM 50
// #define FAN_MAX_PWM 128

/**
 * FAST PWM FAN Settings
 *
 * Use to change the FAST FAN PWM frequency (if enabled in Configuration.h)
 * Combinations of PWM Modes, prescale values and TOP resolutions are used internally to produce a
 * frequency as close as possible to the desired frequency.
 *
 * FAST_PWM_FAN_FREQUENCY [undefined by default]
 * Set this to your desired frequency.
 * If left undefined this defaults to F = F_CPU/(2*255*1)
 * i.e., F = 31.4kHz on 16MHz microcontrollers or F = 39.2kHz on 20MHz microcontrollers.
 * These defaults are the same as with the old FAST_PWM_FAN implementation - no migration is
 * required
 * NOTE: Setting very low frequencies (< 10 Hz) may result in unexpected timer behavior.
 *
 * USE_OCR2A_AS_TOP [undefined by default]
 * Boards that use TIMER2 for PWM have limitations resulting in only a few possible frequencies on
 * TIMER2:
 * 16MHz MCUs: [62.5KHz, 31.4KHz (default), 7.8KHz, 3.92KHz, 1.95KHz, 977Hz, 488Hz, 244Hz, 60Hz,
 * 122Hz, 30Hz]
 * 20MHz MCUs: [78.1KHz, 39.2KHz (default), 9.77KHz, 4.9KHz, 2.44KHz, 1.22KHz, 610Hz, 305Hz,
 * 153Hz, 76Hz, 38Hz]
 * A greater range can be achieved by enabling USE_OCR2A_AS_TOP. But note that this option blocks
 * the use of
 * PWM on pin OC2A. Only use this option if you don't need PWM on 0C2A. (Check your schematic.)
 * USE_OCR2A_AS_TOP sacrifices duty cycle control resolution to achieve this broader range of
 * frequencies.
 */
// #if ENABLED(FAST_PWM_FAN)
// #define FAST_PWM_FAN_FREQUENCY 31400
// #define USE_OCR2A_AS_TOP
// #endif

// @section extruder

/**
 * Extruder cooling fans
 *

```

```

* Extruder auto fans automatically turn on when their extruders'
* temperatures go above EXTRUDER_AUTO_FAN_TEMPERATURE.
*
* Your board's pins file specifies the recommended pins. Override those here
* or set to -1 to disable completely.
*
* Multiple extruders can be assigned to the same pin in which case
* the fan will turn on when any selected extruder is above the threshold.
*/
#define E0_AUTO_FAN_PIN -1
#define E1_AUTO_FAN_PIN -1
#define E2_AUTO_FAN_PIN -1
#define E3_AUTO_FAN_PIN -1
#define E4_AUTO_FAN_PIN -1
#define E5_AUTO_FAN_PIN -1
#define E6_AUTO_FAN_PIN -1
#define E7_AUTO_FAN_PIN -1
#define CHAMBER_AUTO_FAN_PIN -1

#define EXTRUDER_AUTO_FAN_TEMPERATURE 50
#define EXTRUDER_AUTO_FAN_SPEED 255 // 255 == full speed
#define CHAMBER_AUTO_FAN_TEMPERATURE 30
#define CHAMBER_AUTO_FAN_SPEED 255

/**
 * Part-Cooling Fan Multiplexer
 *
 * This feature allows you to digitally multiplex the fan output.
 * The multiplexer is automatically switched at tool-change.
 * Set FANMUX[012]_PINs below for up to 2, 4, or 8 multiplexed fans.
 */
#define FANMUX0_PIN -1
#define FANMUX1_PIN -1
#define FANMUX2_PIN -1

/**
 * M355 Case Light on-off / brightness
 */
// #define CASE_LIGHT_ENABLE
// #if ENABLED(CASE_LIGHT_ENABLE)
//   // #define CASE_LIGHT_PIN 4 // Override the default pin if needed
//   #define INVERT_CASE_LIGHT false // Set true if Case Light is ON when pin is LOW
//   #define CASE_LIGHT_DEFAULT_ON true // Set default power-up state on
//   #define CASE_LIGHT_DEFAULT_BRIGHTNESS 105 // Set default power-up brightness (0-255,
// requires PWM pin)
//   // #define CASE_LIGHT_MAX_PWM 128 // Limit pwm
//   // #define CASE_LIGHT_MENU // Add Case Light options to the LCD menu
//   // #define CASE_LIGHT_NO_BRIGHTNESS // Disable brightness control. Enable for non-PWM
// lighting.
//   // #define CASE_LIGHT_USE_NEOPIXEL // Use NeoPixel LED as case light, requires
// NEOPIXEL_LED.
//   #if ENABLED(CASE_LIGHT_USE_NEOPIXEL)
//     #define CASE_LIGHT_NEOPIXEL_COLOR { 255, 255, 255, 255 } // { Red, Green, Blue, White }
//   #endif
// #endif

// @section homing

// If you want endstops to stay on (by default) even when not homing
// enable this option. Override at any time with M120, M121.
// #define ENDSTOPS_ALWAYS_ON_DEFAULT

// @section extras

// #define Z_LATE_ENABLE // Enable Z the last moment. Needed if your Z driver overheats.

// Employ an external closed loop controller. Override pins here if needed.
// #define EXTERNAL_CLOSED_LOOP_CONTROLLER
// #if ENABLED(EXTERNAL_CLOSED_LOOP_CONTROLLER)
//   // #define CLOSED_LOOP_ENABLE_PIN -1
//   // #define CLOSED_LOOP_MOVE_COMPLETE_PIN -1

```

```

#endif

/**
 * Dual Steppers / Dual Endstops
 *
 * This section will allow you to use extra E drivers to drive a second motor for X, Y, or Z axes.
 *
 * For example, set X_DUAL_STEPPER_DRIVERS setting to use a second motor. If the motors need to
 * spin in opposite directions set INVERT_X2_VS_X_DIR. If the second motor needs its own endstop
 * set X_DUAL_ENDSTOPS. This can adjust for "racking." Use X2_USE_ENDSTOP to set the endstop
 * plug
 * that should be used for the second endstop. Extra endstops will appear in the output of 'M119'.
 *
 * Use X_DUAL_ENDSTOP_ADJUSTMENT to adjust for mechanical imperfection. After homing both
 * motors
 * this offset is applied to the X2 motor. To find the offset home the X axis, and measure the error
 * in X2. Dual endstop offsets can be set at runtime with 'M666 X<offset> Y<offset> Z<offset>'.
 */

//define X_DUAL_STEPPER_DRIVERS
#if ENABLED(X_DUAL_STEPPER_DRIVERS)
  #define INVERT_X2_VS_X_DIR true // Set 'true' if X motors should rotate in opposite directions
  //define X_DUAL_ENDSTOPS
  #if ENABLED(X_DUAL_ENDSTOPS)
    #define X2_USE_ENDSTOP _XMAX_
    #define X2_ENDSTOP_ADJUSTMENT 0
  #endif
#endif

//define Y_DUAL_STEPPER_DRIVERS
#if ENABLED(Y_DUAL_STEPPER_DRIVERS)
  #define INVERT_Y2_VS_Y_DIR true // Set 'true' if Y motors should rotate in opposite directions
  //define Y_DUAL_ENDSTOPS
  #if ENABLED(Y_DUAL_ENDSTOPS)
    #define Y2_USE_ENDSTOP _YMAX_
    #define Y2_ENDSTOP_ADJUSTMENT 0
  #endif
#endif

//
// For Z set the number of stepper drivers
//
#define NUM_Z_STEPPER_DRIVERS 1 // (1-4) Z options change based on how many

#if NUM_Z_STEPPER_DRIVERS > 1
  //define Z_MULTI_ENDSTOPS
  #if ENABLED(Z_MULTI_ENDSTOPS)
    #define Z2_USE_ENDSTOP _XMAX_
    #define Z2_ENDSTOP_ADJUSTMENT 0
    #if NUM_Z_STEPPER_DRIVERS >= 3
      #define Z3_USE_ENDSTOP _YMAX_
      #define Z3_ENDSTOP_ADJUSTMENT 0
    #endif
    #if NUM_Z_STEPPER_DRIVERS >= 4
      #define Z4_USE_ENDSTOP _ZMAX_
      #define Z4_ENDSTOP_ADJUSTMENT 0
    #endif
  #endif
#endif

/**
 * Dual X Carriage
 *
 * This setup has two X carriages that can move independently, each with its own hotend.
 * The carriages can be used to print an object with two colors or materials, or in
 * "duplication mode" it can print two identical or X-mirrored objects simultaneously.
 * The inactive carriage is parked automatically to prevent oozing.
 * X1 is the left carriage, X2 the right. They park and home at opposite ends of the X axis.
 * By default the X2 stepper is assigned to the first unused E plug on the board.
 *
 * The following Dual X Carriage modes can be selected with M605 S<mode>:
 */

```

```

* 0 : (FULL_CONTROL) The slicer has full control over both X-carriages and can achieve optimal travel
* results as long as it supports dual X-carriages. (M605 S0)
*
* 1 : (AUTO_PARK) The firmware automatically parks and unparks the X-carriages on tool-change so
* that additional slicer support is not required. (M605 S1)
*
* 2 : (DUPLICATION) The firmware moves the second X-carriage and extruder in synchronization with
* the first X-carriage and extruder, to print 2 copies of the same object at the same time.
* Set the constant X-offset and temperature differential with M605 S2 X[offs] R[deg] and
* follow with M605 S2 to initiate duplicated movement.
*
* 3 : (MIRRORED) Formbot/Vivedino-inspired mirrored mode in which the second extruder duplicates
* the movement of the first except the second extruder is reversed in the X axis.
* Set the initial X offset and temperature differential with M605 S2 X[offs] R[deg] and
* follow with M605 S3 to initiate mirrored movement.
*/
#define DUAL_X_CARRIAGE
#if ENABLED(DUAL_X_CARRIAGE)
  #define X1_MIN_POS X_MIN_POS // Set to X_MIN_POS
  #define X1_MAX_POS X_BED_SIZE // Set a maximum so the first X-carriage can't hit the parked second
X-carriage
  #define X2_MIN_POS 80 // Set a minimum to ensure the second X-carriage can't hit the parked first
X-carriage
  #define X2_MAX_POS 353 // Set this to the distance between toolheads when both heads are
homed
  #define X2_HOME_DIR 1 // Set to 1. The second X-carriage always homes to the maximum
endstop position
  #define X2_HOME_POS X2_MAX_POS // Default X2 home position. Set to X2_MAX_POS.
// However: In this mode the HOTEND_OFFSET_X value for the second extruder provides a
software // override for X2_HOME_POS. This also allow recalibration of the distance between the two
endstops // without modifying the firmware (through the "M218 T1 X???" command).
// Remember: you should set the second extruder x-offset to 0 in your slicer.

// This is the default power-up mode which can be later using M605.
#define DEFAULT_DUAL_X_CARRIAGE_MODE DXC_AUTO_PARK_MODE

// Default x offset in duplication mode (typically set to half print bed width)
#define DEFAULT_DUPLICATION_X_OFFSET 100
#endif

// Activate a solenoid on the active extruder with M380. Disable all with M381.
// Define SOL0_PIN, SOL1_PIN, etc., for each extruder that has a solenoid.
// #define EXT_SOLENOID

// @section homing

/**
 * Homing Procedure
 * Homing (G28) does an indefinite move towards the endstops to establish
 * the position of the toolhead relative to the workspace.
 */

#define SENSORLESS_BACKOFF_MM { 2, 2 } // (mm) Backoff from endstops before sensorless
homing

#define HOMING_BUMP_MM { 5, 5, 2 } // (mm) Backoff from endstops after first bump
#define HOMING_BUMP_DIVISOR { 2, 2, 2 } // Re-Bump Speed Divisor (Divides the Homing Feedrate)

#define HOMING_BACKOFF_POST_MM { 2, 2, 2 } // (mm) Backoff from endstops after homing

#define QUICK_HOME // If G28 contains XY do a diagonal move first
#define HOME_Y_BEFORE_X // If G28 contains XY home Y before X
#define CODEPENDENT_XY_HOMING // If X/Y can't home without homing Y/X first

// @section bltouch

#if ENABLED(BLTOUCH)

```

```

/**
 * Either: Use the defaults (recommended) or: For special purposes, use the following DEFINES
 * Do not activate settings that the probe might not understand. Clones might misunderstand
 * advanced commands.
 *
 * Note: If the probe is not deploying, do a "Reset" and "Self-Test" and then check the
 * wiring of the BROWN, RED and ORANGE wires.
 *
 * Note: If the trigger signal of your probe is not being recognized, it has been very often
 * because the BLACK and WHITE wires needed to be swapped. They are not "interchangeable"
 * like they would be with a real switch. So please check the wiring first.
 *
 * Settings for all BLTouch and clone probes:
 */

// Safety: The probe needs time to recognize the command.
// Minimum command delay (ms). Enable and increase if needed.
// #define BLTOUCH_DELAY 500

/**
 * Settings for BLTOUCH Classic 1.2, 1.3 or BLTouch Smart 1.0, 2.0, 2.2, 3.0, 3.1, and most clones:
 */

// Feature: Switch into SW mode after a deploy. It makes the output pulse longer. Can be useful
// in special cases, like noisy or filtered input configurations.
// #define BLTOUCH_FORCE_SW_MODE

/**
 * Settings for BLTouch Smart 3.0 and 3.1
 * Summary:
 * - Voltage modes: 5V and OD (open drain - "logic voltage free") output modes
 * - High-Speed mode
 * - Disable LCD voltage options
 */

/**
 * Danger: Don't activate 5V mode unless attached to a 5V-tolerant controller!
 * V3.0 or 3.1: Set default mode to 5V mode at Marlin startup.
 * If disabled, OD mode is the hard-coded default on 3.0
 * On startup, Marlin will compare its eeprom to this value. If the selected mode
 * differs, a mode set eeprom write will be completed at initialization.
 * Use the option below to force an eeprom write to a V3.1 probe regardless.
 */
// #define BLTOUCH_SET_5V_MODE

/**
 * Safety: Activate if connecting a probe with an unknown voltage mode.
 * V3.0: Set a probe into mode selected above at Marlin startup. Required for 5V mode on 3.0
 * V3.1: Force a probe with unknown mode into selected mode at Marlin startup ( = Probe EEPROM
write )
 * To preserve the life of the probe, use this once then turn it off and re-flash.
 */
// #define BLTOUCH_FORCE_MODE_SET

/**
 * Use "HIGH SPEED" mode for probing.
 * Danger: Disable if your probe sometimes fails. Only suitable for stable well-adjusted systems.
 * This feature was designed for Delta's with very fast Z moves however higher speed cartesians may
function
 * If the machine cannot raise the probe fast enough after a trigger, it may enter a fault state.
 */
// #define BLTOUCH_HS_MODE

// Safety: Enable voltage mode settings in the LCD menu.
// #define BLTOUCH_LCD_VOLTAGE_MENU

#endif // BLTOUCH

// @section extras

```

```

/**
 * Z Steppers Auto-Alignment
 * Add the G34 command to align multiple Z steppers using a bed probe.
 */
#define Z_STEPPER_AUTO_ALIGN
#if ENABLED(Z_STEPPER_AUTO_ALIGN)
  // Define probe X and Y positions for Z1, Z2 [, Z3 [, Z4]]
  // If not defined, probe limits will be used.
  // Override with 'M422 S<index> X<pos> Y<pos>'
  // #define Z_STEPPER_ALIGN_XY { { 10, 190 }, { 100, 10 }, { 190, 190 } }

  /**
   * Orientation for the automatically-calculated probe positions.
   * Override Z stepper align points with 'M422 S<index> X<pos> Y<pos>'
   *
   * 2 Steppers: (0) (1)
   *      |   | 2 |
   *      | 1 2 |
   *      |   | 1 |
   *
   * 3 Steppers: (0) (1) (2) (3)
   *      | 3 | 1 | 2 | 1 | 2 |
   *      |   | 3 |   | 3 |
   *      | 1 2 | 2 | 3 | 1 |
   *
   * 4 Steppers: (0) (1) (2) (3)
   *      | 4 3 | 1 4 | 2 1 | 3 2 |
   *      |   |   |   |   |
   *      | 1 2 | 2 3 | 3 4 | 4 1 |
   */
  #ifndef Z_STEPPER_ALIGN_XY
    // #define Z_STEPPERS_ORIENTATION 0
  #endif

  // Provide Z stepper positions for more rapid convergence in bed alignment.
  // Requires triple stepper drivers (i.e., set NUM_Z_STEPPER_DRIVERS to 3)
  // #define Z_STEPPER_ALIGN_KNOWN_STEPPER_POSITIONS
  #if ENABLED(Z_STEPPER_ALIGN_KNOWN_STEPPER_POSITIONS)
    // Define Stepper XY positions for Z1, Z2, Z3 corresponding to
    // the Z screw positions in the bed carriage.
    // Define one position per Z stepper in stepper driver order.
    #define Z_STEPPER_ALIGN_STEPPER_XY { { 210.7, 102.5 }, { 152.6, 220.0 }, { 94.5, 102.5 } }
  #else
    // Amplification factor. Used to scale the correction step up or down in case
    // the stepper (spindle) position is farther out than the test point.
    #define Z_STEPPER_ALIGN_AMP 1.0 // Use a value > 1.0 NOTE: This may cause instability!
  #endif

  // On a 300mm bed a 5% grade would give a misalignment of ~1.5cm
  #define G34_MAX_GRADE 5 // (%) Maximum incline that G34 will handle
  #define Z_STEPPER_ALIGN_ITERATIONS 5 // Number of iterations to apply during alignment
  #define Z_STEPPER_ALIGN_ACC 0.02 // Stop iterating early if the accuracy is better than this
  #define RESTORE_LEVELING_AFTER_G34 // Restore leveling after G34 is done?
  // After G34, re-home Z (G28 Z) or just calculate it from the last probe heights?
  // Re-homing might be more precise in reproducing the actual 'G28 Z' homing height, especially on an
  // uneven bed.
  #define HOME_AFTER_G34
  #endif

  //
  // Add the G35 command to read bed corners to help adjust screws. Requires a bed probe.
  //
  // #define ASSISTED_TRAMMING
  #if ENABLED(ASSISTED_TRAMMING)

    // Define positions for probing points, use the hotend as reference not the sensor.
    #define TRAMMING_POINT_XY { { 20, 20 }, { 200, 20 }, { 200, 200 }, { 20, 200 } }

    // Define positions names for probing points.
    #define TRAMMING_POINT_NAME_1 "Front-Left"
  #endif

```



```

#define TRAMMING_POINT_NAME_2 "Front-Right"
#define TRAMMING_POINT_NAME_3 "Back-Right"
#define TRAMMING_POINT_NAME_4 "Back-Left"

#define RESTORE_LEVELING_AFTER_G35 // Enable to restore leveling setup after operation
// #define REPORT_TRAMMING_MM // Report Z deviation (mm) for each point relative to the first
// #define ASSISTED_TRAMMING_MENU_ITEM // Add a menu item for Assisted Trimming

/**
 * Screw thread:
 * M3: 30 = Clockwise, 31 = Counter-Clockwise
 * M4: 40 = Clockwise, 41 = Counter-Clockwise
 * M5: 50 = Clockwise, 51 = Counter-Clockwise
 */
#define TRAMMING_SCREW_THREAD 30

#endif

// @section motion

#define AXIS_RELATIVE_MODES { false, false, false, false }

// Add a Duplicate option for well-separated conjoined nozzles
// #define MULTI_NOZZLE_DUPLICATION

// By default pololu step drivers require an active high signal. However, some high power drivers require an
// active low signal as step.
#define INVERT_X_STEP_PIN false
#define INVERT_Y_STEP_PIN false
#define INVERT_Z_STEP_PIN false
#define INVERT_E_STEP_PIN false

/**
 * Idle Stepper Shutdown
 * Set DISABLE_INACTIVE_? 'true' to shut down axis steppers after an idle period.
 * The Deactive Time can be overridden with M18 and M84. Set to 0 for No Timeout.
 */
#define DEFAULT_STEPPER_DEACTIVE_TIME 120
#define DISABLE_INACTIVE_X true
#define DISABLE_INACTIVE_Y true
#define DISABLE_INACTIVE_Z true // Set 'false' if the nozzle could fall onto your printed part!
#define DISABLE_INACTIVE_E true

// If the Nozzle or Bed falls when the Z stepper is disabled, set its resting position here.
// #define Z_AFTER_DEACTIVATE Z_HOME_POS

// #define HOME_AFTER_DEACTIVATE // Require rehomming after steppers are deactivated

// Default Minimum Feedrates for printing and travel moves
#define DEFAULT_MINIMUMFEEDRATE 0.0 // (mm/s) Minimum feedrate. Set with M205 S.
#define DEFAULT_MINTRAVELFEEDRATE 0.0 // (mm/s) Minimum travel feedrate. Set with M205 T.

// Minimum time that a segment needs to take as the buffer gets emptied
#define DEFAULT_MINSEGMENTTIME 20000 // (µs) Set with M205 B.

// Slow down the machine if the lookahead buffer is (by default) half full.
// Increase the slowdown divisor for larger buffer sizes.
#define SLOWDOWN
#if ENABLED(SLOWDOWN)
  #define SLOWDOWN_DIVISOR 2
#endif

/**
 * XY Frequency limit
 * Reduce resonance by limiting the frequency of small zigzag infill moves.
 * See https://hydraraptor.blogspot.com/2010/12/frequency-limit.html
 * Use M201 F<freq> G<min%> to change limits at runtime.
 */

```

```

//define XY_FREQUENCY_LIMIT    10 // (Hz) Maximum frequency of small zigzag infill moves. Set with
M201 F<hertz>.
#ifdef XY_FREQUENCY_LIMIT
    #define XY_FREQUENCY_MIN_PERCENT 5 // (percent) Minimum FR percentage to apply. Set with
M201 G<min%>.
#endif

// Minimum planner junction speed. Sets the default minimum speed the planner plans for at the end
// of the buffer and all stops. This should not be much greater than zero and should only be changed
// if unwanted behavior is observed on a user's machine when running at very slow speeds.
#define MINIMUM_PLANNER_SPEED 0.05 // (mm/s)

//
// Backlash Compensation
// Adds extra movement to axes on direction-changes to account for backlash.
//
//define BACKLASH_COMPENSATION
#if ENABLED(BACKLASH_COMPENSATION)
    // Define values for backlash distance and correction.
    // If BACKLASH_GCODE is enabled these values are the defaults.
    #define BACKLASH_DISTANCE_MM { 0, 0, 0 } // (mm)
    #define BACKLASH_CORRECTION    0.0 // 0.0 = no correction; 1.0 = full correction

    // Set BACKLASH_SMOOTHING_MM to spread backlash correction over multiple segments
    // to reduce print artifacts. (Enabling this is costly in memory and computation!)
    //define BACKLASH_SMOOTHING_MM 3 // (mm)

    // Add runtime configuration and tuning of backlash values (M425)
    //define BACKLASH_GCODE

    #if ENABLED(BACKLASH_GCODE)
        // Measure the Z backlash when probing (G29) and set with "M425 Z"
        #define MEASURE_BACKLASH_WHEN_PROBING

        #if ENABLED(MEASURE_BACKLASH_WHEN_PROBING)
            // When measuring, the probe will move up to BACKLASH_MEASUREMENT_LIMIT
            // mm away from point of contact in BACKLASH_MEASUREMENT_RESOLUTION
            // increments while checking for the contact to be broken.
            #define BACKLASH_MEASUREMENT_LIMIT    0.5 // (mm)
            #define BACKLASH_MEASUREMENT_RESOLUTION 0.005 // (mm)
            #define BACKLASH_MEASUREMENT_FEEDRATE  Z_PROBE_SPEED_SLOW // (mm/min)
        #endif
    #endif
#endif

/**
 * Automatic backlash, position and hotend offset calibration
 *
 * Enable G425 to run automatic calibration using an electrically-
 * conductive cube, bolt, or washer mounted on the bed.
 *
 * G425 uses the probe to touch the top and sides of the calibration object
 * on the bed and measures and/or correct positional offsets, axis backlash
 * and hotend offsets.
 *
 * Note: HOTEND_OFFSET and CALIBRATION_OBJECT_CENTER must be set to within
 * ±5mm of true values for G425 to succeed.
 */
//define CALIBRATION_GCODE
#if ENABLED(CALIBRATION_GCODE)

    //define CALIBRATION_SCRIPT_PRE "M117 Starting Auto-Calibration\nT0\nG28\nG12\nM117
    Calibrating..."
    //define CALIBRATION_SCRIPT_POST "M500\nM117 Calibration data saved"

    #define CALIBRATION_MEASUREMENT_RESOLUTION    0.01 // mm

    #define CALIBRATION_FEEDRATE_SLOW             60 // mm/min
    #define CALIBRATION_FEEDRATE_FAST            1200 // mm/min
    #define CALIBRATION_FEEDRATE_TRAVEL          3000 // mm/min

```

```

// The following parameters refer to the conical section of the nozzle tip.
#define CALIBRATION_NOZZLE_TIP_HEIGHT      1.0 // mm
#define CALIBRATION_NOZZLE_OUTER_DIAMETER  2.0 // mm

// Uncomment to enable reporting (required for "G425 V", but consumes PROGMEM).
// #define CALIBRATION_REPORTING

// The true location and dimension the cube/bolt/washer on the bed.
#define CALIBRATION_OBJECT_CENTER { 264.0, -22.0, -2.0 } // mm
#define CALIBRATION_OBJECT_DIMENSIONS { 10.0, 10.0, 10.0 } // mm

// Comment out any sides which are unreachable by the probe. For best
// auto-calibration results, all sides must be reachable.
#define CALIBRATION_MEASURE_RIGHT
#define CALIBRATION_MEASURE_FRONT
#define CALIBRATION_MEASURE_LEFT
#define CALIBRATION_MEASURE_BACK

// Probing at the exact top center only works if the center is flat. If
// probing on a screwhead or hollow washer, probe near the edges.
// #define CALIBRATION_MEASURE_AT_TOP_EDGES

// Define the pin to read during calibration
#ifndef CALIBRATION_PIN
  // #define CALIBRATION_PIN -1 // Define here to override the default pin
  #define CALIBRATION_PIN_INVERTING false // Set to true to invert the custom pin
  // #define CALIBRATION_PIN_PULLDOWN
  #define CALIBRATION_PIN_PULLUP
#endif
#endif

/**
 * Adaptive Step Smoothing increases the resolution of multi-axis moves, particularly at step frequencies
 * below 1kHz (for AVR) or 10kHz (for ARM), where aliasing between axes in multi-axis moves causes
 * audible
 * vibration and surface artifacts. The algorithm adapts to provide the best possible step smoothing at the
 * lowest stepping frequencies.
 */
#define ADAPTIVE_STEP_SMOOTHING

/**
 * Custom Microstepping
 * Override as-needed for your setup. Up to 3 MS pins are supported.
 */
// #define MICROSTEP1 LOW,LOW,LOW
// #define MICROSTEP2 HIGH,LOW,LOW
// #define MICROSTEP4 LOW,HIGH,LOW
// #define MICROSTEP8 HIGH,HIGH,LOW
// #define MICROSTEP16 LOW,LOW,HIGH
// #define MICROSTEP32 HIGH,LOW,HIGH

// Microstep settings (Requires a board with pins named X_MS1, X_MS2, etc.)
#define MICROSTEP_MODES { 16, 16, 16, 16, 16, 16 } // [1,2,4,8,16]

/**
 * @section stepper motor current
 *
 * Some boards have a means of setting the stepper motor current via firmware.
 *
 * The power on motor currents are set by:
 * PWM_MOTOR_CURRENT - used by MINIRAMBO & ULTIMAIN_2
 *   known compatible chips: A4982
 * DIGIPOT_MOTOR_CURRENT - used by BQ_ZUM_MEGA_3D, RAMBO & SCOOVO_X9H
 *   known compatible chips: AD5206
 * DAC_MOTOR_CURRENT_DEFAULT - used by PRINTRBOARD_REVF & RIGIDBOARD_V2
 *   known compatible chips: MCP4728
 * DIGIPOT_I2C_MOTOR_CURRENTS - used by 5DPRINT, AZTEEG_X3_PRO,
 * AZTEEG_X5_MINI_WIFI, MIGHTYBOARD_REVE
 *   known compatible chips: MCP4451, MCP4018

```

```

*
* Motor currents can also be set by M907 - M910 and by the LCD.
* M907 - applies to all.
* M908 - BQ_ZUM_MEGA_3D, RAMBO, PRINTRBOARD_REVF, RIGIDBOARD_V2 & SCOOVO_X9H
* M909, M910 & LCD - only PRINTRBOARD_REVF & RIGIDBOARD_V2
*/
#define PWM_MOTOR_CURRENT { 1300, 1300, 1250 } // Values in milliamps
#define DIGIPOT_MOTOR_CURRENT { 135,135,135,135,135 } // Values 0-255 (RAMBO 135 = ~0.75A,
185 = ~1A)
#define DAC_MOTOR_CURRENT_DEFAULT { 70, 80, 90, 80 } // Default drive percent - X, Y, Z, E axis

/**
 * I2C-based DIGIPOTs (e.g., Azteeg X3 Pro)
 */
#define DIGIPOT_MCP4018 // Requires https://github.com/stawel/SlowSoftI2CMaster
#define DIGIPOT_MCP4451
#if EITHER(DIGIPOT_MCP4018, DIGIPOT_MCP4451)
  #define DIGIPOT_I2C_NUM_CHANNELS 8 //
  SDPRINT:4 AZTEEG_X3_PRO:8 MKS_SBASE:5 MIGHTYBOARD_REVE:5

  // Actual motor currents in Amps. The number of entries must match DIGIPOT_I2C_NUM_CHANNELS.
  // These correspond to the physical drivers, so be mindful if the order is changed.
  #define DIGIPOT_I2C_MOTOR_CURRENTS { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 } // AZTEEG_X3_PRO

  // #define DIGIPOT_USE_RAW_VALUES // Use DIGIPOT_MOTOR_CURRENT raw wiper values
  // (instead of A4988 motor currents)

  /**
   * Common slave addresses:
   *
   * A (A shifted) B (B shifted) IC
   * Smoothie 0x2C (0x58) 0x2D (0x5A) MCP4451
   * AZTEEG_X3_PRO 0x2C (0x58) 0x2E (0x5C) MCP4451
   * AZTEEG_X5_MINI 0x2C (0x58) 0x2E (0x5C) MCP4451
   * AZTEEG_X5_MINI_WIFI 0x58 0x5C MCP4451
   * MIGHTYBOARD_REVE 0x2F (0x5E) MCP4018
   */
  #define DIGIPOT_I2C_ADDRESS_A 0x2C // Unshifted slave address for first DIGIPOT
  #define DIGIPOT_I2C_ADDRESS_B 0x2D // Unshifted slave address for second DIGIPOT
#endif

//=====
//=====Additional Features=====
//=====

// @section lcd

#if EITHER(ULTIPANEL, EXTENSIBLE_UI)
  #define MANUAL_FEEDRATE { 50*60, 50*60, 4*60, 2*60 } // (mm/min) Feedrates for manual moves
  along X, Y, Z, E from panel
  #define SHORT_MANUAL_Z_MOVE 0.025 // (mm) Smallest manual Z move (< 0.1mm)
  #if ENABLED(ULTIPANEL)
    #define MANUAL_E_MOVES_RELATIVE // Display extruder move distance rather than "position"
    #define ULTIPANEL_FEEDMULTIPLY // Encoder sets the feedrate multiplier on the Status Screen
  #endif
#endif

// Change values more rapidly when the encoder is rotated faster
#define ENCODER_RATE_MULTIPLIER
#if ENABLED(ENCODER_RATE_MULTIPLIER)
  #define ENCODER_10X_STEPS_PER_SEC 30 // (steps/s) Encoder rate for 10x speed
  #define ENCODER_100X_STEPS_PER_SEC 80 // (steps/s) Encoder rate for 100x speed
#endif

// Play a beep when the feedrate is changed from the Status Screen
#define BEEP_ON_FEEDRATE_CHANGE
#if ENABLED(BEEP_ON_FEEDRATE_CHANGE)
  #define FEEDRATE_CHANGE_BEEP_DURATION 10
  #define FEEDRATE_CHANGE_BEEP_FREQUENCY 440
#endif

```

```

#if HAS_LCD_MENU

// Add Probe Z Offset calibration to the Z Probe Offsets menu
#if HAS_BED_PROBE
  // #define PROBE_OFFSET_WIZARD
  #if ENABLED(PROBE_OFFSET_WIZARD)
    #define PROBE_OFFSET_START -4.0 // Estimated nozzle-to-probe Z offset, plus a little extra
  #endif
#endif

// Include a page of printer information in the LCD Main Menu
#define LCD_INFO_MENU
#if ENABLED(LCD_INFO_MENU)
  // #define LCD_PRINTER_INFO_IS_BOOTSCREEN // Show bootscreen(s) instead of Printer Info pages
#endif

// BACK menu items keep the highlight at the top
// #define TURBO_BACK_MENU_ITEM

/**
 * LED Control Menu
 * Add LED Control to the LCD menu
 */
// #define LED_CONTROL_MENU
#if ENABLED(LED_CONTROL_MENU)
  #define LED_COLOR_PRESETS // Enable the Preset Color menu option
  // #define NEO2_COLOR_PRESETS // Enable a second NeoPixel Preset Color menu option
  #if ENABLED(LED_COLOR_PRESETS)
    #define LED_USER_PRESET_RED 255 // User defined RED value
    #define LED_USER_PRESET_GREEN 128 // User defined GREEN value
    #define LED_USER_PRESET_BLUE 0 // User defined BLUE value
    #define LED_USER_PRESET_WHITE 255 // User defined WHITE value
    #define LED_USER_PRESET_BRIGHTNESS 255 // User defined intensity
    // #define LED_USER_PRESET_STARTUP // Have the printer display the user preset color on
  startup
  #endif
  #if ENABLED(NEO2_COLOR_PRESETS)
    #define NEO2_USER_PRESET_RED 255 // User defined RED value
    #define NEO2_USER_PRESET_GREEN 128 // User defined GREEN value
    #define NEO2_USER_PRESET_BLUE 0 // User defined BLUE value
    #define NEO2_USER_PRESET_WHITE 255 // User defined WHITE value
    #define NEO2_USER_PRESET_BRIGHTNESS 255 // User defined intensity
    // #define NEO2_USER_PRESET_STARTUP // Have the printer display the user preset color on
  startup for the second strip
  #endif
  #endif

#endif // HAS_LCD_MENU

// Scroll a longer status message into view
#define STATUS_MESSAGE_SCROLLING

// On the Info Screen, display XY with one decimal place when possible
// #define LCD_DECIMAL_SMALL_XY

// The timeout (in ms) to return to the status screen from sub-menus
#define LCD_TIMEOUT_TO_STATUS 15000

// Add an 'M73' G-code to set the current percentage
// #define LCD_SET_PROGRESS_MANUALLY

// Show the E position (filament used) during printing
#define LCD_SHOW_E_TOTAL

#if ENABLED(SHOW_BOOTSCREEN)
  #define BOOTSCREEN_TIMEOUT 4000 // (ms) Total Duration to display the boot screen(s)
#endif

```

```

#if EITHER(SDSUPPORT, LCD_SET_PROGRESS_MANUALLY) && ANY(HAS_MARLINUI_U8GLIB,
HAS_MARLINUI_HD44780, IS_TFTGLCD_PANEL)
  //define SHOW_REMAINING_TIME // Display estimated time to completion
  #if ENABLED(SHOW_REMAINING_TIME)
    //define USE_M73_REMAINING_TIME // Use remaining time from M73 command instead of
estimation
    //define ROTATE_PROGRESS_DISPLAY // Display (P)rogress, (E)lapsed, and (R)emaining time
  #endif

  #if HAS_MARLINUI_U8GLIB
    //define PRINT_PROGRESS_SHOW_DECIMALS // Show progress with decimal digits
  #endif

  #if EITHER(HAS_MARLINUI_HD44780, IS_TFTGLCD_PANEL)
    //define LCD_PROGRESS_BAR // Show a progress bar on HD44780 LCDs for SD printing
    #if ENABLED(LCD_PROGRESS_BAR)
      #define PROGRESS_BAR_BAR_TIME 2000 // (ms) Amount of time to show the bar
      #define PROGRESS_BAR_MSG_TIME 3000 // (ms) Amount of time to show the status message
      #define PROGRESS_MSG_EXPIRE 0 // (ms) Amount of time to retain the status message
(0=forever)
      //define PROGRESS_MSG_ONCE // Show the message for MSG_TIME then clear it
      //define LCD_PROGRESS_BAR_TEST // Add a menu item to test the progress bar
    #endif
  #endif
#endif

#if ENABLED(SDSUPPORT)

  // The standard SD detect circuit reads LOW when media is inserted and HIGH when empty.
  // Enable this option and set to HIGH if your SD cards are incorrectly detected.
  //define SD_DETECT_STATE HIGH

  //define SDCARD_READONLY // Read-only SD card (to save over 2K of flash)

  #define SD_PROCEDURE_DEPTH 1 // Increase if you need more nested M32 calls

  #define SD_FINISHED_STEPPERRELEASE true // Disable steppers when SD Print is finished
  #define SD_FINISHED_RELEASECOMMAND "M84" // Use "M84XYE" to keep Z enabled so your bed
stays in place

  // Reverse SD sort to show "more recent" files first, according to the card's FAT.
  // Since the FAT gets out of order with usage, SDCARD_SORT_ALPHA is recommended.
  #define SDCARD_RATHERRECENTFIRST

  #define SD_MENU_CONFIRM_START // Confirm the selected SD file before printing

  //define MENU_ADDAUTOSTART // Add a menu option to run auto#.g files

  #define EVENT_GCODE_SD_ABORT "G28XY" // G-code to run on SD Abort Print (e.g., "G28XY" or
"G27")

  #if ENABLED(PRINTER_EVENT_LEDS)
    #define PE_LEDS_COMPLETED_TIME (30*60) // (seconds) Time to keep the LED "done" color before
restoring normal illumination
  #endif

  /**
   * Continue after Power-Loss (Creality3D)
   *
   * Store the current state to the SD Card at the start of each layer
   * during SD printing. If the recovery file is found at boot time, present
   * an option on the LCD screen to continue the print from the last-known
   * point in the file.
   */
  #define POWER_LOSS_RECOVERY
  #if ENABLED(POWER_LOSS_RECOVERY)

```

```

#define PLR_ENABLED_DEFAULT true // Power Loss Recovery enabled by default. (Set with 'M413
Sn' & M500)
// #define BACKUP_POWER_SUPPLY // Backup power / UPS to move the steppers on power loss
// #define POWER_LOSS_RECOVER_ZHOME // Z homing is needed for proper recovery. 99.9% of the
time this should be disabled!
// #define POWER_LOSS_ZRAISE 2 // (mm) Z axis raise on resume (on power loss with UPS)
// #define POWER_LOSS_PIN 44 // Pin to detect power loss. Set to -1 to disable default pin on
boards without module.
// #define POWER_LOSS_STATE HIGH // State of pin indicating power loss
// #define POWER_LOSS_PULL // Set pullup / pulldown as appropriate
// #define POWER_LOSS_PURGE_LEN 20 // (mm) Length of filament to purge on resume
// #define POWER_LOSS_RETRACT_LEN 10 // (mm) Length of filament to retract on fail. Requires
backup power.

// Without a POWER_LOSS_PIN the following option helps reduce wear on the SD card,
// especially with "vase mode" printing. Set too high and vases cannot be continued.
#define POWER_LOSS_MIN_Z_CHANGE 0.05 // (mm) Minimum Z change before saving power-loss
data
#endif

/**
 * Sort SD file listings in alphabetical order.
 *
 * With this option enabled, items on SD cards will be sorted
 * by name for easier navigation.
 *
 * By default...
 *
 * - Use the slowest -but safest- method for sorting.
 * - Folders are sorted to the top.
 * - The sort key is statically allocated.
 * - No added G-code (M34) support.
 * - 40 item sorting limit. (Items after the first 40 are unsorted.)
 *
 * SD sorting uses static allocation (as set by SDSORT_LIMIT), allowing the
 * compiler to calculate the worst-case usage and throw an error if the SRAM
 * limit is exceeded.
 *
 * - SDSORT_USES_RAM provides faster sorting via a static directory buffer.
 * - SDSORT_USES_STACK does the same, but uses a local stack-based buffer.
 * - SDSORT_CACHE_NAMES will retain the sorted file listing in RAM. (Expensive!)
 * - SDSORT_DYNAMIC_RAM only uses RAM when the SD menu is visible. (Use with caution!)
 */
// #define SDCARD_SORT_ALPHA

// SD Card Sorting options
// #if ENABLE(SDCARD_SORT_ALPHA)
// #define SDSORT_LIMIT 40 // Maximum number of sorted items (10-256). Costs 27 bytes each.
// #define FOLDER_SORTING -1 // -1=above 0=none 1=below
// #define SDSORT_GCODE false // Allow turning sorting on/off with LCD and M34 G-code.
// #define SDSORT_USES_RAM false // Pre-allocate a static array for faster pre-sorting.
// #define SDSORT_USES_STACK false // Prefer the stack for pre-sorting to give back some SRAM.
// (Negated by next 2 options.)
// #define SDSORT_CACHE_NAMES false // Keep sorted items in RAM longer for speedy performance.
// Most expensive option.
// #define SDSORT_DYNAMIC_RAM false // Use dynamic allocation (within SD menus). Least expensive
// option. Set SDSORT_LIMIT before use!
// #define SDSORT_CACHE_VFATS 2 // Maximum number of 13-byte VFAT entries to use for sorting.
// // Note: Only affects SCROLL_LONG_FILENAMES with
SDSORT_CACHE_NAMES but not SDSORT_DYNAMIC_RAM.
// #endif

// This allows hosts to request long names for files and folders with M33
#define LONG_FILENAME_HOST_SUPPORT

// Enable this option to scroll long filenames in the SD card menu
#define SCROLL_LONG_FILENAMES

// Leave the heaters on after Stop Print (not recommended!)
// #define SD_ABORT_NO_COOLDOWN

```

```

/**
 * This option allows you to abort SD printing when any endstop is triggered.
 * This feature must be enabled with "M540 S1" or from the LCD menu.
 * To have any effect, endstops must be enabled during SD printing.
 */
#define SD_ABORT_ON_ENDSTOP_HIT

/**
 * This option makes it easier to print the same SD Card file again.
 * On print completion the LCD Menu will open with the file selected.
 * You can just click to start the print, or navigate elsewhere.
 */
// #define SD_REPRINT_LAST_SELECTED_FILE

/**
 * Auto-report SdCard status with M27 S<seconds>
 */
// #define AUTO_REPORT_SD_STATUS

/**
 * Support for USB thumb drives using an Arduino USB Host Shield or
 * equivalent MAX3421E breakout board. The USB thumb drive will appear
 * to Marlin as an SD card.
 *
 * The MAX3421E can be assigned the same pins as the SD card reader, with
 * the following pin mapping:
 *
 * SCLK, MOSI, MISO --> SCLK, MOSI, MISO
 * INT      --> SD_DETECT_PIN [1]
 * SS       --> SDSS
 *
 * [1] On AVR an interrupt-capable pin is best for UHS3 compatibility.
 */
// #define USB_FLASH_DRIVE_SUPPORT
// #if ENABLED(USB_FLASH_DRIVE_SUPPORT)
//   #define USB_CS_PIN  SDSS
//   #define USB_INTR_PIN SD_DETECT_PIN
// #endif

/**
 * USB Host Shield Library
 *
 * - UHS2 uses no interrupts and has been production-tested
 *   on a LulzBot TAZ Pro with a 32-bit Archim board.
 *
 * - UHS3 is newer code with better USB compatibility. But it
 *   is less tested and is known to interfere with Servos.
 *   [1] This requires USB_INTR_PIN to be interrupt-capable.
 */
// #define USE_UHS3_USB
// #endif

/**
 * When using a bootloader that supports SD-Firmware-Flashing,
 * add a menu item to activate SD-FW-Update on the next reboot.
 *
 * Requires ATMEGA2560 (Arduino Mega)
 *
 * Tested with this bootloader:
 * https://github.com/FleetProbe/MicroBridge-Arduino-ATMega2560
 */
// #define SD_FIRMWARE_UPDATE
// #if ENABLED(SD_FIRMWARE_UPDATE)
//   #define SD_FIRMWARE_UPDATE_EEPROM_ADDR 0x1FF
//   #define SD_FIRMWARE_UPDATE_ACTIVE_VALUE 0xF0
//   #define SD_FIRMWARE_UPDATE_INACTIVE_VALUE 0xFF
// #endif

// Add an optimized binary file transfer mode, initiated with 'M28 B1'
// #define BINARY_FILE_TRANSFER

/**

```



```

* Set this option to one of the following (or the board's defaults apply):
*
*   LCD - Use the SD drive in the external LCD controller.
*   ONBOARD - Use the SD drive on the control board. (No SD_DETECT_PIN. M21 to init.)
*   CUSTOM_CABLE - Use a custom cable to access the SD (as defined in a pins file).
*
* :[ 'LCD', 'ONBOARD', 'CUSTOM_CABLE' ]
*/
#define SDCARD_CONNECTION LCD

#endif // SDSUPPORT

/**
 * By default an onboard SD card reader may be shared as a USB mass-
 * storage device. This option hides the SD card from the host PC.
 */
#define NO_SD_HOST_DRIVE // Disable SD Card access over USB (for security).

/**
 * Additional options for Graphical Displays
 *
 * Use the optimizations here to improve printing performance,
 * which can be adversely affected by graphical display drawing,
 * especially when doing several short moves, and when printing
 * on DELTA and SCARA machines.
 *
 * Some of these options may result in the display lagging behind
 * controller events, as there is a trade-off between reliable
 * printing performance versus fast display updates.
 */
#if HAS_MARLINUI_U8GLIB
// Show SD percentage next to the progress bar
#define DOGM_SD_PERCENT

// Save many cycles by drawing a hollow frame or no frame on the Info Screen
#define XYZ_NO_FRAME
#define XYZ_HOLLOW_FRAME

// Enable to save many cycles by drawing a hollow frame on Menu Screens
#define MENU_HOLLOW_FRAME

// A bigger font is available for edit items. Costs 3120 bytes of PROGMEM.
// Western only. Not available for Cyrillic, Kana, Turkish, Greek, or Chinese.
#define USE_BIG_EDIT_FONT

// A smaller font may be used on the Info Screen. Costs 2434 bytes of PROGMEM.
// Western only. Not available for Cyrillic, Kana, Turkish, Greek, or Chinese.
#define USE_SMALL_INFOFONT

// Swap the CW/CCW indicators in the graphics overlay
#define OVERLAY_GFX_REVERSE

/**
 * ST7920-based LCDs can emulate a 16 x 4 character display using
 * the ST7920 character-generator for very fast screen updates.
 * Enable LIGHTWEIGHT_UI to use this special display mode.
 *
 * Since LIGHTWEIGHT_UI has limited space, the position and status
 * message occupy the same line. Set STATUS_EXPIRE_SECONDS to the
 * length of time to display the status message before clearing.
 *
 * Set STATUS_EXPIRE_SECONDS to zero to never clear the status.
 * This will prevent position updates from being displayed.
 */
#if ENABLED(U8GLIB_ST7920)
// Enable this option and reduce the value to optimize screen updates.
// The normal delay is 10µs. Use the lowest value that still gives a reliable display.
#define DOGM_SPI_DELAY_US 5

#define LIGHTWEIGHT_UI

```

```

    #if ENABLED(LIGHTWEIGHT_UI)
      #define STATUS_EXPIRE_SECONDS 20
    #endif
  #endif

  /**
   * Status (Info) Screen customizations
   * These options may affect code size and screen render time.
   * Custom status screens can forcibly override these settings.
   */
  // #define STATUS_COMBINE_HEATERS // Use combined heater images instead of separate ones
  // #define STATUS_HOTEND_NUMBERLESS // Use plain hotend icons instead of numbered ones (with
2+ hotends)
  #define STATUS_HOTEND_INVERTED // Show solid nozzle bitmaps when heating (Requires
STATUS_HOTEND_ANIM)
  #define STATUS_HOTEND_ANIM // Use a second bitmap to indicate hotend heating
  #define STATUS_BED_ANIM // Use a second bitmap to indicate bed heating
  #define STATUS_CHAMBER_ANIM // Use a second bitmap to indicate chamber heating
  // #define STATUS_CUTTER_ANIM // Use a second bitmap to indicate spindle / laser active
  // #define STATUS_ALT_BED_BITMAP // Use the alternative bed bitmap
  // #define STATUS_ALT_FAN_BITMAP // Use the alternative fan bitmap
  // #define STATUS_FAN_FRAMES 3 // :[0,1,2,3,4] Number of fan animation frames
  // #define STATUS_HEAT_PERCENT // Show heating in a progress bar
  // #define BOOT_MARLIN_LOGO_SMALL // Show a smaller Marlin logo on the Boot Screen (saving 399
bytes of flash)
  // #define BOOT_MARLIN_LOGO_ANIMATED // Animated Marlin logo. Costs ~3260 (or ~940) bytes of
PROGMEM.

  // Frivolous Game Options
  // #define MARLIN_BRICKOUT
  // #define MARLIN_INVADERS
  // #define MARLIN_SNAKE
  // #define GAMES_EASTER_EGG // Add extra blank lines above the "Games" sub-menu

#endif // HAS_MARLINUI_U8GLIB

//
// Additional options for DGUS / DWIN displays
//
#if HAS_DGUS_LCD
  #define LCD_SERIAL_PORT 3
  #define LCD_BAUDRATE 115200

  #define DGUS_RX_BUFFER_SIZE 128
  #define DGUS_TX_BUFFER_SIZE 48
  // #define SERIAL_STATS_RX_BUFFER_OVERRUNS // Fix Rx overrun situation (Currently only for
AVR)

  #define DGUS_UPDATE_INTERVAL_MS 500 // (ms) Interval between automatic screen updates

  #if EITHER(DGUS_LCD_UI_FYSETC, DGUS_LCD_UI_HIPRECY)
    #define DGUS_PRINT_FILENAME // Display the filename during printing
    #define DGUS_PREHEAT_UI // Display a preheat screen during heatup

    #if ENABLED(DGUS_LCD_UI_FYSETC)
      // #define DGUS_UI_MOVE_DIS_OPTION // Disabled by default for UI_FYSETC
    #else
      #define DGUS_UI_MOVE_DIS_OPTION // Enabled by default for UI_HIPRECY
    #endif

    #define DGUS_FILAMENT_LOADUNLOAD
    #if ENABLED(DGUS_FILAMENT_LOADUNLOAD)
      #define DGUS_FILAMENT_PURGE_LENGTH 10
      #define DGUS_FILAMENT_LOAD_LENGTH_PER_TIME 0.5 // (mm) Adjust in proportion to
DGUS_UPDATE_INTERVAL_MS
    #endif

    #define DGUS_UI_WAITING // Show a "waiting" screen between some screens
    #if ENABLED(DGUS_UI_WAITING)
      #define DGUS_UI_WAITING_STATUS 10
    #endif
  #endif

```

```

    #define DGUS_UI_WAITING_STATUS_PERIOD 8 // Increase to slower waiting status looping
#endif
#endif
#endif // HAS_DGUS_LCD

//
// Touch UI for the FTDI Embedded Video Engine (EVE)
//
#if ENABLED(TOUCH_UI_FTDI_EVE)
  // Display board used
  //#define LCD_FTDI_VM800B35A    // FTDI 3.5" with FT800 (320x240)
  //#define LCD_4DSYSTEMS_4DLCD_FT843 // 4D Systems 4.3" (480x272)
  //#define LCD_HAOYU_FT800CB      // Haoyu with 4.3" or 5" (480x272)
  //#define LCD_HAOYU_FT810CB      // Haoyu with 5" (800x480)
  //#define LCD_ALEPHOBJECTS_CLCD_UI // Aleph Objects Color LCD UI
  //#define LCD_FYSETC_TFT81050    // FYSETC with 5" (800x480)

  // Correct the resolution if not using the stock TFT panel.
  //#define TOUCH_UI_320x240
  //#define TOUCH_UI_480x272
  //#define TOUCH_UI_800x480

  // Mappings for boards with a standard RepRapDiscount Display connector
  //#define AO_EXP1_PINMAP      // AlephObjects CLCD UI EXP1 mapping
  //#define AO_EXP2_PINMAP      // AlephObjects CLCD UI EXP2 mapping
  //#define CR10_TFT_PINMAP     // Rudolph Riedel's CR10 pin mapping
  //#define S6_TFT_PINMAP       // FYSETC S6 pin mapping
  //#define F6_TFT_PINMAP       // FYSETC F6 pin mapping

  //#define OTHER_PIN_LAYOUT // Define pins manually below
  #if ENABLED(OTHER_PIN_LAYOUT)
    // Pins for CS and MOD_RESET (PD) must be chosen
    #define CLCD_MOD_RESET 9
    #define CLCD_SPI_CS 10

    // If using software SPI, specify pins for SCLK, MOSI, MISO
    //#define CLCD_USE_SOFT_SPI
    #if ENABLED(CLCD_USE_SOFT_SPI)
      #define CLCD_SOFT_SPI_MOSI 11
      #define CLCD_SOFT_SPI_MISO 12
      #define CLCD_SOFT_SPI_SCLK 13
    #endif
  #endif
#endif

  // Display Orientation. An inverted (i.e. upside-down) display
  // is supported on the FT800. The FT810 and beyond also support
  // portrait and mirrored orientations.
  //#define TOUCH_UI_INVERTED
  //#define TOUCH_UI_PORTRAIT
  //#define TOUCH_UI_MIRRORED

  // UTF8 processing and rendering.
  // Unsupported characters are shown as '?'.
  //#define TOUCH_UI_USE_UTF8
  #if ENABLED(TOUCH_UI_USE_UTF8)
    // Western accents support. These accented characters use
    // combined bitmaps and require relatively little storage.
    #define TOUCH_UI_UTF8_WESTERN_CHARSET
    #if ENABLED(TOUCH_UI_UTF8_WESTERN_CHARSET)
      // Additional character groups. These characters require
      // full bitmaps and take up considerable storage:
      //#define TOUCH_UI_UTF8_SUPERSCRIPTS // ¹ ² ³
      //#define TOUCH_UI_UTF8_COPYRIGHT // © ®
      //#define TOUCH_UI_UTF8_GERMANIC // ß
      //#define TOUCH_UI_UTF8_SCANDINAVIAN // Æ Ð Ø Þ æ ð ø þ
      //#define TOUCH_UI_UTF8_PUNCTUATION // « » ¿ ¡
      //#define TOUCH_UI_UTF8_CURRENCY // ¢ £ ¤ ¥
      //#define TOUCH_UI_UTF8_ORDINALS // ° º
      //#define TOUCH_UI_UTF8_MATHEMATICS // ± × ÷
      //#define TOUCH_UI_UTF8_FRACTIONS // ¼ ½ ¾
      //#define TOUCH_UI_UTF8_SYMBOLS // µ ¶ · ¸ ¹ º »

```

```

#endif
#endif

// Use a smaller font when labels don't fit buttons
#define TOUCH_UI_FIT_TEXT

// Allow language selection from menu at run-time (otherwise use LCD_LANGUAGE)
//#define LCD_LANGUAGE_1 en
//#define LCD_LANGUAGE_2 fr
//#define LCD_LANGUAGE_3 de
//#define LCD_LANGUAGE_4 es
//#define LCD_LANGUAGE_5 it

// Use a numeric passcode for "Screen lock" keypad.
// (recommended for smaller displays)
//#define TOUCH_UI_PASSCODE

// Output extra debug info for Touch UI events
//#define TOUCH_UI_DEBUG

// Developer menu (accessed by touching "About Printer" copyright text)
//#define TOUCH_UI_DEVELOPER_MENU
#endif

//
// Classic UI Options
//
#if TFT_SCALED_DOGLCD
  #define TFT_MARLINUI_COLOR 0xFFFF // White
  #define TFT_MARLINBG_COLOR 0x0000 // Black
  #define TFT_DISABLED_COLOR 0x0003 // Almost black
  #define TFT_BTCCANCEL_COLOR 0xF800 // Red
  #define TFT_BTARROWS_COLOR 0xDEE6 // 11011 110111 00110 Yellow
  #define TFT_BTOKMENU_COLOR 0x145F // 00010 100010 11111 Cyan
#endif

//
// ADC Button Debounce
//
#if HAS_ADC_BUTTONS
  #define ADC_BUTTON_DEBOUNCE_DELAY 16 // Increase if buttons bounce or repeat too fast
#endif

// @section safety

/**
 * The watchdog hardware timer will do a reset and disable all outputs
 * if the firmware gets too overloaded to read the temperature sensors.
 *
 * If you find that watchdog reboot causes your AVR board to hang forever,
 * enable WATCHDOG_RESET_MANUAL to use a custom timer instead of WDTO.
 * NOTE: This method is less reliable as it can only catch hangups while
 * interrupts are enabled.
 */
#define USE_WATCHDOG
#if ENABLED(USE_WATCHDOG)
  #define WATCHDOG_RESET_MANUAL
#endif

// @section lcd

/**
 * Babystepping enables movement of the axes by tiny increments without changing
 * the current position values. This feature is used primarily to adjust the Z
 * axis in the first layer of a print in real-time.
 *
 * Warning: Does not respect endstops!
 */
#define BABYSTEPPING

```

```

#if ENABLED(BABystepping)
  //define INTEGRATED_BABystepping    // EXPERIMENTAL integration of babystepping into the
  Stepper ISR
  //define BABystepping_WITHOUT_HOMING
  //define BABystepping_ALWAYS_AVAILABLE // Allow babystepping at all times (not just during
  movement).
  //define BABystepping_XY             // Also enable X/Y Babystepping. Not supported on DELTA!
  #define BABystepping_INVERT_Z false // Change if Z babystepping should go the other way
  //define BABystepping_MILLIMETER_UNITS // Specify BABystepping_MULTIPLICATOR_(XY|Z) in mm
  instead of micro-steps
  #define BABystepping_MULTIPLICATOR_Z 8 // (steps or mm) Steps or millimeter distance for each Z
  babystep
  #define BABystepping_MULTIPLICATOR_XY 1 // (steps or mm) Steps or millimeter distance for each
  XY babystep

  #define DOUBLECLICK_FOR_Z_BABystepping // Double-click on the Status Screen for Z
  Babystepping.
  #if ENABLED(DOUBLECLICK_FOR_Z_BABystepping)
    #define DOUBLECLICK_MAX_INTERVAL 1250 // Maximum interval between clicks, in milliseconds.
    // Note: Extra time may be added to mitigate controller latency.
    //define MOVE_Z_WHEN_IDLE // Jump to the move Z menu on doubleclick when printer is
    idle.
    #if ENABLED(MOVE_Z_WHEN_IDLE)
      #define MOVE_Z_IDLE_MULTIPLICATOR 1 // Multiply 1mm by this factor for the move step size.
    #endif
  #endif

  //define BABystepping_DISPLAY_TOTAL // Display total babystepping since last G28

  //define BABystepping_ZPROBE_OFFSET // Combine M851 Z and Babystepping
  #if ENABLED(BABystepping_ZPROBE_OFFSET)
    //define BABystepping_HOTEND_Z_OFFSET // For multiple hotends, babystep relative Z offsets
    //define BABystepping_ZPROBE_GFX_OVERLAY // Enable graphical overlay on Z-offset editor
  #endif
#endif

// @section extruder

/**
 * Linear Pressure Control v1.5
 *
 * * Assumption: advance [steps] = k * (delta velocity [steps/s])
 * * K=0 means advance disabled.
 *
 * * NOTE: K values for LIN_ADVANCE 1.5 differ from earlier versions!
 *
 * * Set K around 0.22 for 3mm PLA Direct Drive with ~6.5cm between the drive gear and heatbreak.
 * * Larger K values will be needed for flexible filament and greater distances.
 * * If this algorithm produces a higher speed offset than the extruder can handle (compared to E jerk)
 * * print acceleration will be reduced during the affected moves to keep within the limit.
 *
 * * See https://marlinfw.org/docs/features/lin\_advance.html for full instructions.
 */
#define LIN_ADVANCE
#if ENABLED(LIN_ADVANCE)
  //define EXTRA_LIN_ADVANCE_K // Enable for second linear advance constants
  #define LIN_ADVANCE_K 0.22 // Unit: mm compression per 1mm/s extruder speed
  //define LA_DEBUG // If enabled, this will generate debug information output over USB.
  #define EXPERIMENTAL_SCURVE // Enable this option to permit S-Curve Acceleration
#endif

// @section leveling

/**
 * Points to probe for all 3-point Leveling procedures.
 * Override if the automatically selected points are inadequate.
 */
#if EITHER(AUTO_BED_LEVELING_3POINT, AUTO_BED_LEVELING_UBL)
  //define PROBE_PT_1_X 15
  //define PROBE_PT_1_Y 180
  //define PROBE_PT_2_X 15

```

```

#define PROBE_PT_2_Y 20
#define PROBE_PT_3_X 170
#define PROBE_PT_3_Y 20
#endif

/**
 * Probing Margins
 *
 * Override PROBING_MARGIN for each side of the build plate
 * Useful to get probe points to exact positions on targets or
 * to allow leveling to avoid plate clamps on only specific
 * sides of the bed. With NOZZLE_AS_PROBE negative values are
 * allowed, to permit probing outside the bed.
 *
 * If you are replacing the prior *_PROBE_BED_POSITION options,
 * LEFT and FRONT values in most cases will map directly over
 * RIGHT and REAR would be the inverse such as
 * (X/Y_BED_SIZE - RIGHT/BACK_PROBE_BED_POSITION)
 *
 * This will allow all positions to match at compilation, however
 * should the probe position be modified with M851XY then the
 * probe points will follow. This prevents any change from causing
 * the probe to be unable to reach any points.
 */
#if PROBE_SELECTED && !IS_KINEMATIC
#define PROBING_MARGIN_LEFT PROBING_MARGIN
#define PROBING_MARGIN_RIGHT PROBING_MARGIN
#define PROBING_MARGIN_FRONT PROBING_MARGIN
#define PROBING_MARGIN_BACK PROBING_MARGIN
#endif

#if EITHER(MESH_BED_LEVELING, AUTO_BED_LEVELING_UBL)
// Override the mesh area if the automatic (max) area is too large
#define MESH_MIN_X MESH_INSET
#define MESH_MIN_Y MESH_INSET
#define MESH_MAX_X X_BED_SIZE - (MESH_INSET)
#define MESH_MAX_Y Y_BED_SIZE - (MESH_INSET)
#endif

/**
 * Repeatedly attempt G29 leveling until it succeeds.
 * Stop after G29_MAX_RETRIES attempts.
 */
#define G29_RETRY_AND_RECOVER
#if ENABLED(G29_RETRY_AND_RECOVER)
#define G29_MAX_RETRIES 3
#define G29_HALT_ON_FAILURE
/**
 * Specify the GCODE commands that will be executed when leveling succeeds,
 * between attempts, and after the maximum number of retries have been tried.
 */
#define G29_SUCCESS_COMMANDS "M117 Bed leveling done."
#define G29_RECOVER_COMMANDS "M117 Probe failed. Rewiping.\nG28\nG12 P0 S12 T0"
#define G29_FAILURE_COMMANDS "M117 Bed leveling failed.\nG0 Z10\nM300 P25 S880\nM300 P50 S0\nM300 P25 S880\nM300 P50 S0\nM300 P25 S880\nM300 P50 S0\nG4 S1"

#endif

/**
 * Thermal Probe Compensation
 * Probe measurements are adjusted to compensate for temperature distortion.
 * Use G76 to calibrate this feature. Use M871 to set values manually.
 * For a more detailed explanation of the process see G76_M871.cpp.
 */
#if HAS_BED_PROBE && TEMP_SENSOR_PROBE && TEMP_SENSOR_BED
// Enable thermal first layer compensation using bed and probe temperatures
#define PROBE_TEMP_COMPENSATION

// Add additional compensation depending on hotend temperature
// Note: this values cannot be calibrated and have to be set manually
#if ENABLED(PROBE_TEMP_COMPENSATION)
// Park position to wait for probe cooldown

```

```

#define PTC_PARK_POS { 0, 0, 100 }

// Probe position to probe and wait for probe to reach target temperature
#define PTC_PROBE_POS { 90, 100 }

// Enable additional compensation using hotend temperature
// Note: this values cannot be calibrated automatically but have to be set manually
// #define USE_TEMP_EXT_COMPENSATION

// Probe temperature calibration generates a table of values starting at PTC_SAMPLE_START
// (e.g. 30), in steps of PTC_SAMPLE_RES (e.g. 5) with PTC_SAMPLE_COUNT (e.g. 10) samples.

// #define PTC_SAMPLE_START 30.0f
// #define PTC_SAMPLE_RES 5.0f
// #define PTC_SAMPLE_COUNT 10U

// Bed temperature calibration builds a similar table.

// #define BTC_SAMPLE_START 60.0f
// #define BTC_SAMPLE_RES 5.0f
// #define BTC_SAMPLE_COUNT 10U

// The temperature the probe should be at while taking measurements during bed temperature
// calibration.
// #define BTC_PROBE_TEMP 30.0f

// Height above Z=0.0f to raise the nozzle. Lowering this can help the probe to heat faster.
// Note: the Z=0.0f offset is determined by the probe offset which can be set using M851.
// #define PTC_PROBE_HEATING_OFFSET 0.5f

// Height to raise the Z-probe between heating and taking the next measurement. Some probes
// may fail to untrigger if they have been triggered for a long time, which can be solved by
// increasing the height the probe is raised to.
// #define PTC_PROBE_RAISE 15U

// If the probe is outside of the defined range, use linear extrapolation using the closest
// point and the PTC_LINEAR_EXTRAPOLATION'th next point. E.g. if set to 4 it will use data[0]
// and data[4] to perform linear extrapolation for values below PTC_SAMPLE_START.
// #define PTC_LINEAR_EXTRAPOLATION 4
#endif
#endif

// @section extras

//
// G60/G61 Position Save and Return
//
// #define SAVED_POSITIONS 1 // Each saved position slot costs 12 bytes

//
// G2/G3 Arc Support
//
// #define ARC_SUPPORT // Disable this feature to save ~3226 bytes
// #if ENABLED(ARC_SUPPORT)
//   #define MM_PER_ARC_SEGMENT 1 // (mm) Length (or minimum length) of each arc segment
//   #define ARC_SEGMENTS_PER_R 1 // Max segment length, MM_PER = Min
//   #define MIN_ARC_SEGMENTS 24 // Minimum number of segments in a complete circle
//   #define ARC_SEGMENTS_PER_SEC 50 // Use feedrate to choose segment length (with
//   MM_PER_ARC_SEGMENT as the minimum)
//   #define N_ARC_CORRECTION 25 // Number of interpolated segments between corrections
//   #define ARC_P_CIRCLES // Enable the 'P' parameter to specify complete circles
//   #define CNC_WORKSPACE_PLANES // Allow G2/G3 to operate in XY, ZX, or YZ planes
//   #define SF_ARC_FIX // Enable only if using SkeinForge with "Arc Point" fillet procedure
// #endif

// Support for G5 with XYZE destination and IJPQ offsets. Requires ~2666 bytes.
// #define BEZIER_CURVE_SUPPORT

```

```

/**
 * Direct Stepping
 *
 * Comparable to the method used by Klipper, G6 direct stepping significantly
 * reduces motion calculations, increases top printing speeds, and results in
 * less step aliasing by calculating all motions in advance.
 * Preparing your G-code: https://github.com/colinrgodsey/step-daemon
 */
#define DIRECT_STEPPING

/**
 * G38 Probe Target
 *
 * This option adds G38.2 and G38.3 (probe towards target)
 * and optionally G38.4 and G38.5 (probe away from target).
 * Set MULTIPLE_PROBING for G38 to probe more than once.
 */
#define G38_PROBE_TARGET
#if ENABLED(G38_PROBE_TARGET)
  #define G38_PROBE_AWAY // Include G38.4 and G38.5 to probe away from target
  #define G38_MINIMUM_MOVE 0.0275 // (mm) Minimum distance that will produce a move.
#endif

// Moves (or segments) with fewer steps than this will be joined with the next move
#define MIN_STEPS_PER_SEGMENT 6

/**
 * Minimum delay before and after setting the stepper DIR (in ns)
 * 0 : No delay (Expect at least 10µs since one Stepper ISR must transpire)
 * 20 : Minimum for TMC2xxx drivers
 * 200 : Minimum for A4988 drivers
 * 400 : Minimum for A5984 drivers
 * 500 : Minimum for LV8729 drivers (guess, no info in datasheet)
 * 650 : Minimum for DRV8825 drivers
 * 1500 : Minimum for TB6600 drivers (guess, no info in datasheet)
 * 15000 : Minimum for TB6560 drivers (guess, no info in datasheet)
 *
 * Override the default value based on the driver type set in Configuration.h.
 */
#define MINIMUM_STEPPER_POST_DIR_DELAY 650
#define MINIMUM_STEPPER_PRE_DIR_DELAY 650

/**
 * Minimum stepper driver pulse width (in µs)
 * 0 : Smallest possible width the MCU can produce, compatible with TMC2xxx drivers
 * 0 : Minimum 500ns for LV8729, adjusted in stepper.h
 * 1 : Minimum for A4988 and A5984 stepper drivers
 * 2 : Minimum for DRV8825 stepper drivers
 * 3 : Minimum for TB6600 stepper drivers
 * 30 : Minimum for TB6560 stepper drivers
 *
 * Override the default value based on the driver type set in Configuration.h.
 */
#define MINIMUM_STEPPER_PULSE 2

/**
 * Maximum stepping rate (in Hz) the stepper driver allows
 * If undefined, defaults to 1MHz / (2 * MINIMUM_STEPPER_PULSE)
 * 5000000 : Maximum for TMC2xxx stepper drivers
 * 1000000 : Maximum for LV8729 stepper driver
 * 500000 : Maximum for A4988 stepper driver
 * 250000 : Maximum for DRV8825 stepper driver
 * 150000 : Maximum for TB6600 stepper driver
 * 15000 : Maximum for TB6560 stepper driver
 *
 * Override the default value based on the driver type set in Configuration.h.
 */
#define MAXIMUM_STEPPER_RATE 250000

// @section temperature

```



```

// Control heater 0 and heater 1 in parallel.
// #define HEATERS_PARALLEL

//=====
//===== Buffers =====
//=====

// @section motion

// The number of linear moves that can be in the planner at once.
// The value of BLOCK_BUFFER_SIZE must be a power of 2 (e.g. 8, 16, 32)
#if BOTH(SDSUPPORT, DIRECT_STEPPING)
  #define BLOCK_BUFFER_SIZE 8
#elif ENABLED(SDSUPPORT)
  #define BLOCK_BUFFER_SIZE 16
#else
  #define BLOCK_BUFFER_SIZE 16
#endif

// @section serial

// The ASCII buffer for serial input
#define MAX_CMD_SIZE 96
#define BUFSIZE 4

// Transmission to Host Buffer Size
// To save 386 bytes of PROGMEM (and TX_BUFFER_SIZE+3 bytes of RAM) set to 0.
// To buffer a simple "ok" you need 4 bytes.
// For ADVANCED_OK (M105) you need 32 bytes.
// For debug-echo: 128 bytes for the optimal speed.
// Other output doesn't need to be that speedy.
// :[0, 2, 4, 8, 16, 32, 64, 128, 256]
#define TX_BUFFER_SIZE 0

// Host Receive Buffer Size
// Without XON/XOFF flow control (see SERIAL_XON_XOFF below) 32 bytes should be enough.
// To use flow control, set this buffer size to at least 1024 bytes.
// :[0, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048]
// #define RX_BUFFER_SIZE 1024

#if RX_BUFFER_SIZE >= 1024
  // Enable to have the controller send XON/XOFF control characters to
  // the host to signal the RX buffer is becoming full.
  // #define SERIAL_XON_XOFF
#endif

// Add M575 G-code to change the baud rate
// #define BAUD_RATE_GCODE

#if ENABLED(SDSUPPORT)
  // Enable this option to collect and display the maximum
  // RX queue usage after transferring a file to SD.
  // #define SERIAL_STATS_MAX_RX_QUEUED

  // Enable this option to collect and display the number
  // of dropped bytes after a file transfer to SD.
  // #define SERIAL_STATS_DROPPED_RX
#endif

/**
 * Emergency Command Parser
 *
 * * Add a low-level parser to intercept certain commands as they
 * * enter the serial receive buffer, so they cannot be blocked.
 * * Currently handles M108, M112, M410, M876
 * * NOTE: Not yet implemented for all platforms.
 */

```

```

#define EMERGENCY_PARSER

// Bad Serial-connections can miss a received command by sending an 'ok'
// Therefore some clients abort after 30 seconds in a timeout.
// Some other clients start sending commands while receiving a 'wait'.
// This "wait" is only sent when the buffer is empty. 1 second is a good value here.
#define NO_TIMEOUTS 1000 // Milliseconds

// Some clients will have this feature soon. This could make the NO_TIMEOUTS unnecessary.
#define ADVANCED_OK

// Printron may have trouble receiving long strings all at once.
// This option inserts short delays between lines of serial output.
#define SERIAL_OVERRUN_PROTECTION

// For serial echo, the number of digits after the decimal point
#define SERIAL_FLOAT_PRECISION 4

// @section extras

/**
 * Extra Fan Speed
 * Adds a secondary fan speed for each print-cooling fan.
 * 'M106 P<fan> T3-255' : Set a secondary speed for <fan>
 * 'M106 P<fan> T2' : Use the set secondary speed
 * 'M106 P<fan> T1' : Restore the previous fan speed
 */
#define EXTRA_FAN_SPEED

/**
 * Firmware-based and LCD-controlled retract
 *
 * Add G10 / G11 commands for automatic firmware-based retract / recover.
 * Use M207 and M208 to define parameters for retract / recover.
 *
 * Use M209 to enable or disable auto-retract.
 * With auto-retract enabled, all G1 E moves within the set range
 * will be converted to firmware-based retract/recover moves.
 *
 * Be sure to turn off auto-retract during filament change.
 *
 * Note that M207 / M208 / M209 settings are saved to EEPROM.
 */
#define FWRETRACT
#if ENABLED(FWRETRACT)
  #define FWRETRACT_AUTORETRACT // Override slicer retractions
  #if ENABLED(FWRETRACT_AUTORETRACT)
    #define MIN_AUTORETRACT 0.1 // (mm) Don't convert E moves under this length
    #define MAX_AUTORETRACT 10.0 // (mm) Don't convert E moves over this length
  #endif
  #define RETRACT_LENGTH 3 // (mm) Default retract length (positive value)
  #define RETRACT_LENGTH_SWAP 13 // (mm) Default swap retract length (positive value)
  #define RETRACT_FEEDRATE 45 // (mm/s) Default feedrate for retracting
  #define RETRACT_ZRAISE 0 // (mm) Default retract Z-raise
  #define RETRACT_RECOVER_LENGTH 0 // (mm) Default additional recover length (added to retract
length on recover)
  #define RETRACT_RECOVER_LENGTH_SWAP 0 // (mm) Default additional swap recover length
(added to retract length on recover from toolchange)
  #define RETRACT_RECOVER_FEEDRATE 8 // (mm/s) Default feedrate for recovering from retraction
  #define RETRACT_RECOVER_FEEDRATE_SWAP 8 // (mm/s) Default feedrate for recovering from swap
retraction
  #if ENABLED(MIXING_EXTRUDER)
    #define RETRACT_SYNC_MIXING // Retract and restore all mixing steppers simultaneously
  #endif
#endif

/**
 * Universal tool change settings.
 * Applies to all types of extruders except where explicitly noted.
 */
#if HAS_MULTI_EXTRUDER

```

```

// Z raise distance for tool-change, as needed for some extruders
#define TOOLCHANGE_ZRAISE 2 // (mm)
// #define TOOLCHANGE_ZRAISE_BEFORE_RETRACT // Apply raise before swap retraction (if
enabled)
// #define TOOLCHANGE_NO_RETURN // Never return to previous position on tool-change
// if ENABLED(TOOLCHANGE_NO_RETURN)
// #define EVENT_GCODE_AFTER_TOOLCHANGE "G12X" // Extra G-code to run after tool-change
// endif

/**
 * Retract and prime filament on tool-change to reduce
 * ooze and stringing and to get cleaner transitions.
 */
// #define TOOLCHANGE_FILAMENT_SWAP
// if ENABLED(TOOLCHANGE_FILAMENT_SWAP)
// Load / Unload
// #define TOOLCHANGE_FS_LENGTH 12 // (mm) Load / Unload length
// #define TOOLCHANGE_FS_EXTRA_RESUME_LENGTH 0 // (mm) Extra length for better restart, fine
tune by LCD/Gcode)
// #define TOOLCHANGE_FS_RETRACT_SPEED (50*60) // (mm/min) (Unloading)
// #define TOOLCHANGE_FS_UNRETRACT_SPEED (25*60) // (mm/min) (On SINGLENOZZLE or
Bowden loading must be slowed down)

// Longer prime to clean out a SINGLENOZZLE
// #define TOOLCHANGE_FS_EXTRA_PRIME 0 // (mm) Extra priming length
// #define TOOLCHANGE_FS_PRIME_SPEED (4.6*60) // (mm/min) Extra priming feedrate
// #define TOOLCHANGE_FS_WIPE_RETRACT 0 // (mm/min) Retract before cooling for less
stringing, better wipe, etc.

// Cool after prime to reduce stringing
// #define TOOLCHANGE_FS_FAN -1 // Fan index or -1 to skip
// #define TOOLCHANGE_FS_FAN_SPEED 255 // 0-255
// #define TOOLCHANGE_FS_FAN_TIME 10 // (seconds)

// Swap uninitialized extruder with TOOLCHANGE_FS_PRIME_SPEED for all lengths (recover + prime)
// (May break filament if not retracted beforehand.)
// #define TOOLCHANGE_FS_INIT_BEFORE_SWAP

// Prime on the first T0 (If other, TOOLCHANGE_FS_INIT_BEFORE_SWAP applied)
// Enable it (M217 V[0/1]) before printing, to avoid unwanted priming on host connect
// #define TOOLCHANGE_FS_PRIME_FIRST_USED

/**
 * Tool Change Migration
 * This feature provides G-code and LCD options to switch tools mid-print.
 * All applicable tool properties are migrated so the print can continue.
 * Tools must be closely matching and other restrictions may apply.
 * Useful to:
 * - Change filament color without interruption
 * - Switch spools automatically on filament runout
 * - Switch to a different nozzle on an extruder jam
 */
// #define TOOLCHANGE_MIGRATION_FEATURE

// endif

/**
 * Position to park head during tool change.
 * Doesn't apply to SWITCHING_TOOLHEAD, DUAL_X_CARRIAGE, or PARKING_EXTRUDER
 */
// #define TOOLCHANGE_PARK
// if ENABLED(TOOLCHANGE_PARK)
// #define TOOLCHANGE_PARK_XY { X_MIN_POS + 10, Y_MIN_POS + 10 }
// #define TOOLCHANGE_PARK_XY_FEEDRATE 6000 // (mm/min)
// #define TOOLCHANGE_PARK_X_ONLY // X axis only move
// #define TOOLCHANGE_PARK_Y_ONLY // Y axis only move
// endif
// #endif // HAS_MULTI_EXTRUDER

/**

```

```

* Advanced Pause
* Experimental feature for filament change support and for parking the nozzle when paused.
* Adds the GCode M600 for initiating filament change.
* If PARK_HEAD_ON_PAUSE enabled, adds the GCode M125 to pause printing and park the nozzle.
*
* Requires an LCD display.
* Requires NOZZLE_PARK_FEATURE.
* This feature is required for the default FILAMENT_RUNOUT_SCRIPT.
*/
#define ADVANCED_PAUSE_FEATURE
#if ENABLED(ADVANCED_PAUSE_FEATURE)
  #define PAUSE_PARK_RETRACT_FEEDRATE    60 // (mm/s) Initial retract feedrate.
  #define PAUSE_PARK_RETRACT_LENGTH      2 // (mm) Initial retract.
  // This short retract is done immediately, before parking the nozzle.
  #define FILAMENT_CHANGE_UNLOAD_FEEDRATE 10 // (mm/s) Unload filament feedrate. This can
  be pretty fast.
  #define FILAMENT_CHANGE_UNLOAD_ACCEL    25 // (mm/s^2) Lower acceleration may allow a
  faster feedrate.
  #define FILAMENT_CHANGE_UNLOAD_LENGTH  100 // (mm) The length of filament for a complete
  unload.
  // For Bowden, the full length of the tube and nozzle.
  // For direct drive, the full length of the nozzle.
  // Set to 0 for manual unloading.
  #define FILAMENT_CHANGE_SLOW_LOAD_FEEDRATE 6 // (mm/s) Slow move when starting load.
  #define FILAMENT_CHANGE_SLOW_LOAD_LENGTH  0 // (mm) Slow length, to allow time to insert
  material.
  // 0 to disable start loading and skip to fast load only
  #define FILAMENT_CHANGE_FAST_LOAD_FEEDRATE 6 // (mm/s) Load filament feedrate. This can
  be pretty fast.
  #define FILAMENT_CHANGE_FAST_LOAD_ACCEL    25 // (mm/s^2) Lower acceleration may allow a
  faster feedrate.
  #define FILAMENT_CHANGE_FAST_LOAD_LENGTH  0 // (mm) Load length of filament, from extruder
  gear to nozzle.
  // For Bowden, the full length of the tube and nozzle.
  // For direct drive, the full length of the nozzle.
  // #define ADVANCED_PAUSE_CONTINUOUS_PURGE // Purge continuously up to the purge length
  until interrupted.
  #define ADVANCED_PAUSE_PURGE_FEEDRATE    3 // (mm/s) Extrude feedrate (after loading).
  Should be slower than load feedrate.
  #define ADVANCED_PAUSE_PURGE_LENGTH      50 // (mm) Length to extrude after loading.
  // Set to 0 for manual extrusion.
  // Filament can be extruded repeatedly from the Filament Change menu
  // until extrusion is consistent, and to purge old filament.
  #define ADVANCED_PAUSE_RESUME_PRIME      0 // (mm) Extra distance to prime nozzle after
  returning from park.
  // #define ADVANCED_PAUSE_FANS_PAUSE // Turn off print-cooling fans while the machine is
  paused.

  // Filament Unload does a Retract, Delay, and Purge first:
  #define FILAMENT_UNLOAD_PURGE_RETRACT    13 // (mm) Unload initial retract length.
  #define FILAMENT_UNLOAD_PURGE_DELAY      5000 // (ms) Delay for the filament to cool after
  retract.
  #define FILAMENT_UNLOAD_PURGE_LENGTH      8 // (mm) An unretract is done, then this length is
  purged.
  #define FILAMENT_UNLOAD_PURGE_FEEDRATE    25 // (mm/s) feedrate to purge before unload

  #define PAUSE_PARK_NOZZLE_TIMEOUT        45 // (seconds) Time limit before the nozzle is turned
  off for safety.
  #define FILAMENT_CHANGE_ALERT_BEEPS      10 // Number of alert beeps to play when a
  response is needed.
  #define PAUSE_PARK_NO_STEPPER_TIMEOUT // Enable for XYZ steppers to stay powered on
  during filament change.

  // #define PARK_HEAD_ON_PAUSE // Park the nozzle during pause and filament change.
  // #define HOME_BEFORE_FILAMENT_CHANGE // If needed, home before parking for filament
  change

  // #define FILAMENT_LOAD_UNLOAD_GCODES // Add M701/M702 Load/Unload G-codes, plus
  Load/Unload in the LCD Prepare menu.
  // #define FILAMENT_UNLOAD_ALL_EXTRUDERS // Allow M702 to unload all extruders above a
  minimum target temp (as set by M302)
#endif

```

```

// @section tmc

/**
 * TMC26X Stepper Driver options
 *
 * The TMC26XStepper library is required for this stepper driver.
 * https://github.com/trinamic/TMC26XStepper
 */
#if HAS_DRIVER(TMC26X)

  #if AXIS_DRIVER_TYPE_X(TMC26X)
    #define X_MAX_CURRENT 1000 // (mA)
    #define X_SENSE_RESISTOR 91 // (mOhms)
    #define X_MICROSTEPS 16 // Number of microsteps
  #endif

  #if AXIS_DRIVER_TYPE_X2(TMC26X)
    #define X2_MAX_CURRENT 1000
    #define X2_SENSE_RESISTOR 91
    #define X2_MICROSTEPS 16
  #endif

  #if AXIS_DRIVER_TYPE_Y(TMC26X)
    #define Y_MAX_CURRENT 1000
    #define Y_SENSE_RESISTOR 91
    #define Y_MICROSTEPS 16
  #endif

  #if AXIS_DRIVER_TYPE_Y2(TMC26X)
    #define Y2_MAX_CURRENT 1000
    #define Y2_SENSE_RESISTOR 91
    #define Y2_MICROSTEPS 16
  #endif

  #if AXIS_DRIVER_TYPE_Z(TMC26X)
    #define Z_MAX_CURRENT 1000
    #define Z_SENSE_RESISTOR 91
    #define Z_MICROSTEPS 16
  #endif

  #if AXIS_DRIVER_TYPE_Z2(TMC26X)
    #define Z2_MAX_CURRENT 1000
    #define Z2_SENSE_RESISTOR 91
    #define Z2_MICROSTEPS 16
  #endif

  #if AXIS_DRIVER_TYPE_Z3(TMC26X)
    #define Z3_MAX_CURRENT 1000
    #define Z3_SENSE_RESISTOR 91
    #define Z3_MICROSTEPS 16
  #endif

  #if AXIS_DRIVER_TYPE_Z4(TMC26X)
    #define Z4_MAX_CURRENT 1000
    #define Z4_SENSE_RESISTOR 91
    #define Z4_MICROSTEPS 16
  #endif

  #if AXIS_DRIVER_TYPE_E0(TMC26X)
    #define E0_MAX_CURRENT 1000
    #define E0_SENSE_RESISTOR 91
    #define E0_MICROSTEPS 16
  #endif

  #if AXIS_DRIVER_TYPE_E1(TMC26X)
    #define E1_MAX_CURRENT 1000
    #define E1_SENSE_RESISTOR 91
    #define E1_MICROSTEPS 16
  #endif

```

```

#endif

#if AXIS_DRIVER_TYPE_E2(TMC26X)
#define E2_MAX_CURRENT 1000
#define E2_SENSE_RESISTOR 91
#define E2_MICROSTEPS 16
#endif

#if AXIS_DRIVER_TYPE_E3(TMC26X)
#define E3_MAX_CURRENT 1000
#define E3_SENSE_RESISTOR 91
#define E3_MICROSTEPS 16
#endif

#if AXIS_DRIVER_TYPE_E4(TMC26X)
#define E4_MAX_CURRENT 1000
#define E4_SENSE_RESISTOR 91
#define E4_MICROSTEPS 16
#endif

#if AXIS_DRIVER_TYPE_E5(TMC26X)
#define E5_MAX_CURRENT 1000
#define E5_SENSE_RESISTOR 91
#define E5_MICROSTEPS 16
#endif

#if AXIS_DRIVER_TYPE_E6(TMC26X)
#define E6_MAX_CURRENT 1000
#define E6_SENSE_RESISTOR 91
#define E6_MICROSTEPS 16
#endif

#if AXIS_DRIVER_TYPE_E7(TMC26X)
#define E7_MAX_CURRENT 1000
#define E7_SENSE_RESISTOR 91
#define E7_MICROSTEPS 16
#endif

#endif // TMC26X

// @section tmc_smart

/**
 * To use TMC2130, TMC2160, TMC2660, TMC5130, TMC5160 stepper drivers in SPI mode
 * connect your SPI pins to the hardware SPI interface on your board and define
 * the required CS pins in your `pins_MYBOARD.h` file. (e.g., RAMPS 1.4 uses AUX3
 * pins `X_CS_PIN 53`, `Y_CS_PIN 49`, etc.).
 * You may also use software SPI if you wish to use general purpose IO pins.
 *
 * To use TMC2208 stepper UART-configurable stepper drivers connect #_SERIAL_TX_PIN
 * to the driver side PDN_UART pin with a 1K resistor.
 * To use the reading capabilities, also connect #_SERIAL_RX_PIN to PDN_UART without
 * a resistor.
 * The drivers can also be used with hardware serial.
 *
 * TMCStepper library is required to use TMC stepper drivers.
 * https://github.com/teemuatlut/TMCStepper
 */
#if HAS_TRINAMIC_CONFIG

#define HOLD_MULTIPLIER 0.5 // Scales down the holding current from run current
#define INTERPOLATE true // Interpolate X/Y/Z_MICROSTEPS to 256

#if AXIS_IS_TMC(X)
#define X_CURRENT 800 // (mA) RMS current. Multiply by 1.414 for peak current.
#define X_CURRENT_HOME X_CURRENT // (mA) RMS current for sensorless homing
#define X_MICROSTEPS 16 // 0..256
#define X_RSENSE 0.11
#define X_CHAIN_POS -1 // <=0 : Not chained. 1 : MCU MOSI connected. 2 : Next in chain, ...

```

```
#endif
```

```
#if AXIS_IS_TMC(X2)
#define X2_CURRENT 800
#define X2_CURRENT_HOME X2_CURRENT
#define X2_MICROSTEPS 16
#define X2_RSENSE 0.11
#define X2_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(Y)
#define Y_CURRENT 800
#define Y_CURRENT_HOME Y_CURRENT
#define Y_MICROSTEPS 16
#define Y_RSENSE 0.11
#define Y_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(Y2)
#define Y2_CURRENT 800
#define Y2_CURRENT_HOME Y2_CURRENT
#define Y2_MICROSTEPS 16
#define Y2_RSENSE 0.11
#define Y2_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(Z)
#define Z_CURRENT 800
#define Z_CURRENT_HOME Z_CURRENT
#define Z_MICROSTEPS 16
#define Z_RSENSE 0.11
#define Z_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(Z2)
#define Z2_CURRENT 800
#define Z2_CURRENT_HOME Z2_CURRENT
#define Z2_MICROSTEPS 16
#define Z2_RSENSE 0.11
#define Z2_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(Z3)
#define Z3_CURRENT 800
#define Z3_CURRENT_HOME Z3_CURRENT
#define Z3_MICROSTEPS 16
#define Z3_RSENSE 0.11
#define Z3_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(Z4)
#define Z4_CURRENT 800
#define Z4_CURRENT_HOME Z4_CURRENT
#define Z4_MICROSTEPS 16
#define Z4_RSENSE 0.11
#define Z4_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(E0)
#define E0_CURRENT 800
#define E0_MICROSTEPS 16
#define E0_RSENSE 0.11
#define E0_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(E1)
#define E1_CURRENT 800
#define E1_MICROSTEPS 16
#define E1_RSENSE 0.11
#define E1_CHAIN_POS -1
```

```
#endif
```

```
#if AXIS_IS_TMC(E2)
#define E2_CURRENT 800
#define E2_MICROSTEPS 16
#define E2_RSENSE 0.11
#define E2_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(E3)
#define E3_CURRENT 800
#define E3_MICROSTEPS 16
#define E3_RSENSE 0.11
#define E3_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(E4)
#define E4_CURRENT 800
#define E4_MICROSTEPS 16
#define E4_RSENSE 0.11
#define E4_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(E5)
#define E5_CURRENT 800
#define E5_MICROSTEPS 16
#define E5_RSENSE 0.11
#define E5_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(E6)
#define E6_CURRENT 800
#define E6_MICROSTEPS 16
#define E6_RSENSE 0.11
#define E6_CHAIN_POS -1
#endif
```

```
#if AXIS_IS_TMC(E7)
#define E7_CURRENT 800
#define E7_MICROSTEPS 16
#define E7_RSENSE 0.11
#define E7_CHAIN_POS -1
#endif
```

```
/**
```

```
 * Override default SPI pins for TMC2130, TMC2160, TMC2660, TMC5130 and TMC5160 drivers here.
```

```
 * The default pins can be found in your board's pins file.
```

```
 */
```

```
//#define X_CS_PIN -1
//#define Y_CS_PIN -1
//#define Z_CS_PIN -1
//#define X2_CS_PIN -1
//#define Y2_CS_PIN -1
//#define Z2_CS_PIN -1
//#define Z3_CS_PIN -1
//#define E0_CS_PIN -1
//#define E1_CS_PIN -1
//#define E2_CS_PIN -1
//#define E3_CS_PIN -1
//#define E4_CS_PIN -1
//#define E5_CS_PIN -1
//#define E6_CS_PIN -1
//#define E7_CS_PIN -1
```

```
/**
```

```
 * Software option for SPI driven drivers (TMC2130, TMC2160, TMC2660, TMC5130 and TMC5160).
```

```
 * The default SW SPI pins are defined the respective pins files,
```

```
 * but you can override or define them here.
```

```
 */
```

```
//#define TMC_USE_SW_SPI
//#define TMC_SW_MOSI -1
```



```

//#define TMC_SW_MISO -1
//#define TMC_SW_SCK -1

/**
 * Four TMC2209 drivers can use the same HW/SW serial port with hardware configured addresses.
 * Set the address using jumpers on pins MS1 and MS2.
 * Address | MS1 | MS2
 * 0 | LOW | LOW
 * 1 | HIGH | LOW
 * 2 | LOW | HIGH
 * 3 | HIGH | HIGH
 *
 * Set *_SERIAL_TX_PIN and *_SERIAL_RX_PIN to match for all drivers
 * on the same serial port, either here or in your board's pins file.
 */
#define X_SLAVE_ADDRESS 0
#define Y_SLAVE_ADDRESS 0
#define Z_SLAVE_ADDRESS 0
#define X2_SLAVE_ADDRESS 0
#define Y2_SLAVE_ADDRESS 0
#define Z2_SLAVE_ADDRESS 0
#define Z3_SLAVE_ADDRESS 0
#define Z4_SLAVE_ADDRESS 0
#define E0_SLAVE_ADDRESS 0
#define E1_SLAVE_ADDRESS 0
#define E2_SLAVE_ADDRESS 0
#define E3_SLAVE_ADDRESS 0
#define E4_SLAVE_ADDRESS 0
#define E5_SLAVE_ADDRESS 0
#define E6_SLAVE_ADDRESS 0
#define E7_SLAVE_ADDRESS 0

/**
 * Software enable
 *
 * Use for drivers that do not use a dedicated enable pin, but rather handle the same
 * function through a communication line such as SPI or UART.
 */
//#define SOFTWARE_DRIVER_ENABLE

/**
 * TMC2130, TMC2160, TMC2208, TMC2209, TMC5130 and TMC5160 only
 * Use Trinamic's ultra quiet stepping mode.
 * When disabled, Marlin will use spreadCycle stepping mode.
 */
#define STEALTHCHOP_XY
#define STEALTHCHOP_Z
#define STEALTHCHOP_E

/**
 * Optimize spreadCycle chopper parameters by using predefined parameter sets
 * or with the help of an example included in the library.
 * Provided parameter sets are
 * CHOPPER_DEFAULT_12V
 * CHOPPER_DEFAULT_19V
 * CHOPPER_DEFAULT_24V
 * CHOPPER_DEFAULT_36V
 * CHOPPER_09STEP_24V // 0.9 degree steppers (24V)
 * CHOPPER_PRUSAMK3_24V // Imported parameters from the official Průša firmware for MK3 (24V)
 * CHOPPER_MARLIN_119 // Old defaults from Marlin v1.1.9
 *
 * Define you own with
 * { <off_time[1..15]>, <hysteresis_end[-3..12]>, hysteresis_start[1..8] }
 */
#define CHOPPER_TIMING CHOPPER_DEFAULT_24V

/**
 * Monitor Trinamic drivers
 * for error conditions like overtemperature and short to ground.
 * To manage over-temp Marlin can decrease the driver current until the error condition clears.
 * Other detected conditions can be used to stop the current print.
 * Relevant G-codes:

```

* M906 - Set or get motor current in milliamps using axis codes X, Y, Z, E. Report values if no axis codes given.

* M911 - Report stepper driver overtemperature pre-warn condition.

* M912 - Clear stepper driver overtemperature pre-warn condition flag.

* M122 - Report driver parameters (Requires TMC_DEBUG)

*/

#define MONITOR_DRIVER_STATUS

#if ENABLED(MONITOR_DRIVER_STATUS)

#define CURRENT_STEP_DOWN 50 // [mA]

#define REPORT_CURRENT_CHANGE

#define STOP_ON_ERROR

#endif

/**

* TMC2130, TMC2160, TMC2208, TMC2209, TMC5130 and TMC5160 only

* The driver will switch to spreadCycle when stepper speed is over HYBRID_THRESHOLD.

* This mode allows for faster movements at the expense of higher noise levels.

* STEALTHCHOP_(XY|Z|E) must be enabled to use HYBRID_THRESHOLD.

* M913 X/Y/Z/E to live tune the setting

*/

#define HYBRID_THRESHOLD

#define X_HYBRID_THRESHOLD 100 // [mm/s]

#define X2_HYBRID_THRESHOLD 100

#define Y_HYBRID_THRESHOLD 100

#define Y2_HYBRID_THRESHOLD 100

#define Z_HYBRID_THRESHOLD 3

#define Z2_HYBRID_THRESHOLD 3

#define Z3_HYBRID_THRESHOLD 3

#define Z4_HYBRID_THRESHOLD 3

#define E0_HYBRID_THRESHOLD 30

#define E1_HYBRID_THRESHOLD 30

#define E2_HYBRID_THRESHOLD 30

#define E3_HYBRID_THRESHOLD 30

#define E4_HYBRID_THRESHOLD 30

#define E5_HYBRID_THRESHOLD 30

#define E6_HYBRID_THRESHOLD 30

#define E7_HYBRID_THRESHOLD 30

/**

* Use StallGuard to home / probe X, Y, Z.

*

* TMC2130, TMC2160, TMC2209, TMC2660, TMC5130, and TMC5160 only

* Connect the stepper driver's DIAG1 pin to the X/Y endstop pin.

* X, Y, and Z homing will always be done in spreadCycle mode.

*

* X/Y/Z_STALL_SENSITIVITY is the default stall threshold.

* Use M914 X Y Z to set the stall threshold at runtime:

*

* Sensitivity TMC2209 Others

* HIGHEST 255 -64 (Too sensitive => False positive)

* LOWEST 0 63 (Too insensitive => No trigger)

*

* It is recommended to set HOMING_BUMP_MM to { 0, 0, 0 }.

*

* SPI_ENDSTOPS *** Beta feature! *** TMC2130 Only ***

* Poll the driver through SPI to determine load when homing.

* Removes the need for a wire from DIAG1 to an endstop pin.

*

* IMPROVE_HOMING_RELIABILITY tunes acceleration and jerk when

* homing and adds a guard period for endstop triggering.

*

* Comment *_STALL_SENSITIVITY to disable sensorless homing for that axis.

*/

#define SENSORLESS_HOMING // StallGuard capable drivers only

#if EITHER(SENSORLESS_HOMING, SENSORLESS_PROBING)

// TMC2209: 0...255. TMC2130: -64...63

#define X_STALL_SENSITIVITY 60

// #define X2_STALL_SENSITIVITY X_STALL_SENSITIVITY

#define Y_STALL_SENSITIVITY 65

```

#define Y2_STALL_SENSITIVITY Y_STALL_SENSITIVITY
#define Z_STALL_SENSITIVITY 8
#define Z2_STALL_SENSITIVITY Z_STALL_SENSITIVITY
#define Z3_STALL_SENSITIVITY Z_STALL_SENSITIVITY
#define Z4_STALL_SENSITIVITY Z_STALL_SENSITIVITY
#define SPI_ENDSTOPS // TMC2130 only
#define IMPROVE_HOMING_RELIABILITY
#endif

/**
 * TMC Homing stepper phase.
 *
 * Improve homing repeatability by homing to stepper coil's nearest absolute
 * phase position. Trinamic drivers use a stepper phase table with 1024 values
 * spanning 4 full steps with 256 positions each (ergo, 1024 positions).
 * Full step positions (128, 384, 640, 896) have the highest holding torque.
 *
 * Values from 0..1023, -1 to disable homing phase for that axis.
 */
#define TMC_HOME_PHASE { 896, 896, 896 }

/**
 * Beta feature!
 * Create a 50/50 square wave step pulse optimal for stepper drivers.
 */
#define SQUARE_WAVE_STEPPING

/**
 * Enable M122 debugging command for TMC stepper drivers.
 * M122 S0/1 will enable continous reporting.
 */
#define TMC_DEBUG

/**
 * You can set your own advanced settings by filling in predefined functions.
 * A list of available functions can be found on the library github page
 * https://github.com/teemuatlut/TMCStepper
 *
 * Example:
 * #define TMC_ADV() { \
 *   stepperX.diag0_otpw(1); \
 *   stepperY.intpol(0); \
 * }
 */
#define TMC_ADV() { }

#endif // HAS_TRINAMIC_CONFIG

// @section L64XX

/**
 * L64XX Stepper Driver options
 *
 * Arduino-L6470 library (0.8.0 or higher) is required.
 * https://github.com/ameyer/Arduino-L6470
 *
 * Requires the following to be defined in your pins_YOUR_BOARD file
 * L6470_CHAIN_SCK_PIN
 * L6470_CHAIN_MISO_PIN
 * L6470_CHAIN_MOSI_PIN
 * L6470_CHAIN_SS_PIN
 * ENABLE_RESET_L64XX_CHIPS(Q) where Q is 1 to enable and 0 to reset
 */

#if HAS_L64XX

#define L6470_CHITCHAT // Display additional status info

#if AXIS_IS_L64XX(X)

```

```

#define X_MICROSTEPS    128 // Number of microsteps (VALID: 1, 2, 4, 8, 16, 32, 128) - L6474 max
is 16
#define X_OVERCURRENT    2000 // (mA) Current where the driver detects an over current
    // L6470 & L6474 - VALID: 375 x (1 - 16) - 6A max - rounds down
    // POWERSTEP01: VALID: 1000 x (1 - 32) - 32A max - rounds down
#define X_STALLCURRENT    1500 // (mA) Current where the driver detects a stall (VALID: 31.25 * (1-
128) - 4A max - rounds down)
    // L6470 & L6474 - VALID: 31.25 * (1-128) - 4A max - rounds down
    // POWERSTEP01: VALID: 200 x (1 - 32) - 6.4A max - rounds down
    // L6474 - STALLCURRENT setting is used to set the nominal (TVAL) current
#define X_MAX_VOLTAGE    127 // 0-255, Maximum effective voltage seen by stepper - not used by
L6474
#define X_CHAIN_POS      -1 // Position in SPI chain, 0=Not in chain, 1=Nearest MOSI
#define X_SLEW_RATE      1 // 0-3, Slew 0 is slowest, 3 is fastest
#endif

#if AXIS_IS_L64XX(X2)
#define X2_MICROSTEPS    128
#define X2_OVERCURRENT    2000
#define X2_STALLCURRENT    1500
#define X2_MAX_VOLTAGE    127
#define X2_CHAIN_POS      -1
#define X2_SLEW_RATE      1
#endif

#if AXIS_IS_L64XX(Y)
#define Y_MICROSTEPS    128
#define Y_OVERCURRENT    2000
#define Y_STALLCURRENT    1500
#define Y_MAX_VOLTAGE    127
#define Y_CHAIN_POS      -1
#define Y_SLEW_RATE      1
#endif

#if AXIS_IS_L64XX(Y2)
#define Y2_MICROSTEPS    128
#define Y2_OVERCURRENT    2000
#define Y2_STALLCURRENT    1500
#define Y2_MAX_VOLTAGE    127
#define Y2_CHAIN_POS      -1
#define Y2_SLEW_RATE      1
#endif

#if AXIS_IS_L64XX(Z)
#define Z_MICROSTEPS    128
#define Z_OVERCURRENT    2000
#define Z_STALLCURRENT    1500
#define Z_MAX_VOLTAGE    127
#define Z_CHAIN_POS      -1
#define Z_SLEW_RATE      1
#endif

#if AXIS_IS_L64XX(Z2)
#define Z2_MICROSTEPS    128
#define Z2_OVERCURRENT    2000
#define Z2_STALLCURRENT    1500
#define Z2_MAX_VOLTAGE    127
#define Z2_CHAIN_POS      -1
#define Z2_SLEW_RATE      1
#endif

#if AXIS_IS_L64XX(Z3)
#define Z3_MICROSTEPS    128
#define Z3_OVERCURRENT    2000
#define Z3_STALLCURRENT    1500
#define Z3_MAX_VOLTAGE    127
#define Z3_CHAIN_POS      -1
#define Z3_SLEW_RATE      1
#endif

#if AXIS_IS_L64XX(Z4)

```

```

#define Z4_MICROSTEPS 128
#define Z4_OVERCURRENT 2000
#define Z4_STALLCURRENT 1500
#define Z4_MAX_VOLTAGE 127
#define Z4_CHAIN_POS -1
#define Z4_SLEW_RATE 1
#endif

```

```

#if AXIS_IS_L64XX(E0)
#define E0_MICROSTEPS 128
#define E0_OVERCURRENT 2000
#define E0_STALLCURRENT 1500
#define E0_MAX_VOLTAGE 127
#define E0_CHAIN_POS -1
#define E0_SLEW_RATE 1
#endif

```

```

#if AXIS_IS_L64XX(E1)
#define E1_MICROSTEPS 128
#define E1_OVERCURRENT 2000
#define E1_STALLCURRENT 1500
#define E1_MAX_VOLTAGE 127
#define E1_CHAIN_POS -1
#define E1_SLEW_RATE 1
#endif

```

```

#if AXIS_IS_L64XX(E2)
#define E2_MICROSTEPS 128
#define E2_OVERCURRENT 2000
#define E2_STALLCURRENT 1500
#define E2_MAX_VOLTAGE 127
#define E2_CHAIN_POS -1
#define E2_SLEW_RATE 1
#endif

```

```

#if AXIS_IS_L64XX(E3)
#define E3_MICROSTEPS 128
#define E3_OVERCURRENT 2000
#define E3_STALLCURRENT 1500
#define E3_MAX_VOLTAGE 127
#define E3_CHAIN_POS -1
#define E3_SLEW_RATE 1
#endif

```

```

#if AXIS_IS_L64XX(E4)
#define E4_MICROSTEPS 128
#define E4_OVERCURRENT 2000
#define E4_STALLCURRENT 1500
#define E4_MAX_VOLTAGE 127
#define E4_CHAIN_POS -1
#define E4_SLEW_RATE 1
#endif

```

```

#if AXIS_IS_L64XX(E5)
#define E5_MICROSTEPS 128
#define E5_OVERCURRENT 2000
#define E5_STALLCURRENT 1500
#define E5_MAX_VOLTAGE 127
#define E5_CHAIN_POS -1
#define E5_SLEW_RATE 1
#endif

```

```

#if AXIS_IS_L64XX(E6)
#define E6_MICROSTEPS 128
#define E6_OVERCURRENT 2000
#define E6_STALLCURRENT 1500
#define E6_MAX_VOLTAGE 127
#define E6_CHAIN_POS -1
#define E6_SLEW_RATE 1
#endif

```

```

#if AXIS_IS_L64XX(E7)
  #define E7_MICROSTEPS 128
  #define E7_OVERCURRENT 2000
  #define E7_STALLCURRENT 1500
  #define E7_MAX_VOLTAGE 127
  #define E7_CHAIN_POS -1
  #define E7_SLEW_RATE 1
#endif

/**
 * Monitor L6470 drivers for error conditions like over temperature and over current.
 * In the case of over temperature Marlin can decrease the drive until the error condition clears.
 * Other detected conditions can be used to stop the current print.
 * Relevant G-codes:
 * M906 - I1/2/3/4/5 Set or get motor drive level using axis codes X, Y, Z, E. Report values if no axis
codes given.
 * I not present or I0 or I1 - X, Y, Z or E0
 * I2 - X2, Y2, Z2 or E1
 * I3 - Z3 or E3
 * I4 - Z4 or E4
 * I5 - E5
 * M916 - Increase drive level until get thermal warning
 * M917 - Find minimum current thresholds
 * M918 - Increase speed until max or error
 * M122 S0/1 - Report driver parameters
 */
// #define MONITOR_L6470_DRIVER_STATUS

#if ENABLED(MONITOR_L6470_DRIVER_STATUS)
  #define KVAL_HOLD_STEP_DOWN 1
  // #define L6470_STOP_ON_ERROR
#endif

#endif // HAS_L64XX

// @section i2cbus

//
// I2C Master ID for LPC176x LCD and Digital Current control
// Does not apply to other peripherals based on the Wire library.
//
// #define I2C_MASTER_ID 1 // Set a value from 0 to 2

/**
 * TWI/I2C BUS
 *
 * This feature is an EXPERIMENTAL feature so it shall not be used on production
 * machines. Enabling this will allow you to send and receive I2C data from slave
 * devices on the bus.
 *
 * ; Example #1
 * ; This macro send the string "Marlin" to the slave device with address 0x63 (99)
 * ; It uses multiple M260 commands with one B<base 10> arg
 * M260 A99 ; Target slave address
 * M260 B77 ; M
 * M260 B97 ; a
 * M260 B114 ; r
 * M260 B108 ; l
 * M260 B105 ; i
 * M260 B110 ; n
 * M260 S1 ; Send the current buffer
 *
 * ; Example #2
 * ; Request 6 bytes from slave device with address 0x63 (99)
 * M261 A99 B5
 *
 * ; Example #3
 * ; Example serial output of a M261 request
 * echo:i2c-reply: from:99 bytes:5 data:hello
 */

```

```

// #define EXPERIMENTAL_I2CBUS
// #if ENABLED(EXPERIMENTAL_I2CBUS)
//   #define I2C_SLAVE_ADDRESS 0 // Set a value from 8 to 127 to act as a slave
// #endif

// @section extras

/**
 * Photo G-code
 * Add the M240 G-code to take a photo.
 * The photo can be triggered by a digital pin or a physical movement.
 */
// #define PHOTO_GCODE
// #if ENABLED(PHOTO_GCODE)
//   // A position to move to (and raise Z) before taking the photo
//   // #define PHOTO_POSITION { X_MAX_POS - 5, Y_MAX_POS, 0 } // { xpos, ypos, zraise } (M240 X Y Z)
//   // #define PHOTO_DELAY_MS 100 // (ms) Duration to pause before moving back
//   (M240 P)
//   // #define PHOTO_RETRACT_MM 6.5 // (mm) E retract/recover for the photo move
//   (M240 R S)

// Canon RC-1 or homebrew digital camera trigger
// Data from: https://www.doc-diy.net/photo/rc-1\_hacked/
// #define PHOTOGRAPH_PIN 23

// Canon Hack Development Kit
// https://captain-slow.dk/2014/03/09/3d-printing-timelapses/
// #define CHDK_PIN 4

// Optional second move with delay to trigger the camera shutter
// #define PHOTO_SWITCH_POSITION { X_MAX_POS, Y_MAX_POS } // { xpos, ypos } (M240 I J)

// Duration to hold the switch or keep CHDK_PIN high
// #define PHOTO_SWITCH_MS 50 // (ms) (M240 D)

/**
 * PHOTO_PULSES_US may need adjustment depending on board and camera model.
 * Pin must be running at 48.4kHz.
 * Be sure to use a PHOTOGRAPH_PIN which can rise and fall quick enough.
 * (e.g., MKS SBase temp sensor pin was too slow, so used P1.23 on J8.)
 *
 * Example pulse data for Nikon: https://bit.ly/2FKD0Aq
 * IR Wiring: https://git.io/JvJf7
 */
// #define PHOTO_PULSES_US { 2000, 27850, 400, 1580, 400, 3580, 400 } // (µs) Durations for each
// 48.4kHz oscillation
// #ifdef PHOTO_PULSES_US
//   #define PHOTO_PULSE_DELAY_US 13 // (µs) Approximate duration of each HIGH and LOW pulse in
// the oscillation
// #endif
// #endif

/**
 * Spindle & Laser control
 *
 * Add the M3, M4, and M5 commands to turn the spindle/laser on and off, and
 * to set spindle speed, spindle direction, and laser power.
 *
 * SuperPid is a router/spindle speed controller used in the CNC milling community.
 * Marlin can be used to turn the spindle on and off. It can also be used to set
 * the spindle speed from 5,000 to 30,000 RPM.
 *
 * You'll need to select a pin for the ON/OFF function and optionally choose a 0-5V
 * hardware PWM pin for the speed control and a pin for the rotation direction.
 *
 * See https://marlinfw.org/docs/configuration/laser\_spindle.html for more config details.
 */
// #define SPINDLE_FEATURE
// #define LASER_FEATURE
// #if EITHER(SPINDLE_FEATURE, LASER_FEATURE)
//   #define SPINDLE_LASER_ACTIVE_STATE LOW // Set to "HIGH" if the on/off function is active HIGH

```

```

#define SPINDLE_LASER_PWM          true // Set to "true" if your controller supports setting the
speed/power
#define SPINDLE_LASER_PWM_INVERT    false // Set to "true" if the speed/power goes up when you
want it to go slower

#define SPINDLE_LASER_FREQUENCY    2500 // (Hz) Spindle/laser frequency (only on supported
HALs: AVR and LPC)

/**
 * Speed / Power can be set ('M3 S') and displayed in terms of:
 * - PWM255 (S0 - S255)
 * - PERCENT (S0 - S100)
 * - RPM (S0 - S50000) Best for use with a spindle
 */
#define CUTTER_POWER_UNIT PWM255

/**
 * Relative Cutter Power
 * Normally, 'M3 O<power>' sets
 * OCR power is relative to the range SPEED_POWER_MIN...SPEED_POWER_MAX.
 * so input powers of 0...255 correspond to SPEED_POWER_MIN...SPEED_POWER_MAX
 * instead of normal range (0 to SPEED_POWER_MAX).
 * Best used with (e.g.) SuperPID router controller: S0 = 5,000 RPM and S255 = 30,000 RPM
 */
// #define CUTTER_POWER_RELATIVE // Set speed proportional to
// [SPEED_POWER_MIN...SPEED_POWER_MAX]

#if ENABLED(SPINDLE_FEATURE)
  // #define SPINDLE_CHANGE_DIR // Enable if your spindle controller can change spindle
  // direction
  // #define SPINDLE_CHANGE_DIR_STOP // Enable if the spindle should stop before changing spin
  // direction
  // #define SPINDLE_INVERT_DIR false // Set to "true" if the spin direction is reversed

  #define SPINDLE_LASER_POWERUP_DELAY 5000 // (ms) Delay to allow the spindle/laser to come
  up to speed/power
  #define SPINDLE_LASER_POWERDOWN_DELAY 5000 // (ms) Delay to allow the spindle to stop

  /**
   * M3/M4 Power Equation
   *
   * Each tool uses different value ranges for speed / power control.
   * These parameters are used to convert between tool power units and PWM.
   *
   * Speed/Power = (PWMDC / 255 * 100 - SPEED_POWER_INTERCEPT) / SPEED_POWER_SLOPE
   * PWMDC = (spdpwr - SPEED_POWER_MIN) / (SPEED_POWER_MAX - SPEED_POWER_MIN) /
   SPEED_POWER_SLOPE
   */
  #define SPEED_POWER_INTERCEPT 0 // (%) 0-100 i.e., Minimum power percentage
  #define SPEED_POWER_MIN 5000 // (RPM)
  #define SPEED_POWER_MAX 30000 // (RPM) SuperPID router controller 0 - 30,000 RPM
  #define SPEED_POWER_STARTUP 25000 // (RPM) M3/M4 speed/power default (with no
  arguments)

#else

  #define SPEED_POWER_INTERCEPT 0 // (%) 0-100 i.e., Minimum power percentage
  #define SPEED_POWER_MIN 0 // (%) 0-100
  #define SPEED_POWER_MAX 100 // (%) 0-100
  #define SPEED_POWER_STARTUP 80 // (%) M3/M4 speed/power default (with no arguments)

  /**
   * Enable inline laser power to be handled in the planner / stepper routines.
   * Inline power is specified by the I (inline) flag in an M3 command (e.g., M3 S20 I)
   * or by the 'S' parameter in G0/G1/G2/G3 moves (see LASER_MOVE_POWER).
   *
   * This allows the laser to keep in perfect sync with the planner and removes
   * the powerup/down delay since lasers require negligible time.
   */
  #define LASER_POWER_INLINE

```



```

#if ENABLED(LASER_POWER_INLINE)
/**
 * Scale the laser's power in proportion to the movement rate.
 *
 * - Sets the entry power proportional to the entry speed over the nominal speed.
 * - Ramps the power up every N steps to approximate the speed trapezoid.
 * - Due to the limited power resolution this is only approximate.
 */
#define LASER_POWER_INLINE_TRAPEZOID

/**
 * Continuously calculate the current power (nominal_power * current_rate / nominal_rate).
 * Required for accurate power with non-trapezoidal acceleration (e.g., S_CURVE_ACCELERATION).
 * This is a costly calculation so this option is discouraged on 8-bit AVR boards.
 *
 * LASER_POWER_INLINE_TRAPEZOID_CONT_PER defines how many step cycles there are
between power updates. If your
 * board isn't able to generate steps fast enough (and you are using
LASER_POWER_INLINE_TRAPEZOID_CONT), increase this.
 * Note that when this is zero it means it occurs every cycle; 1 means a delay wait one cycle then run,
etc.
 */
//#define LASER_POWER_INLINE_TRAPEZOID_CONT

/**
 * Stepper iterations between power updates. Increase this value if the board
 * can't keep up with the processing demands of LASER_POWER_INLINE_TRAPEZOID_CONT.
 * Disable (or set to 0) to recalculate power on every stepper iteration.
 */
//#define LASER_POWER_INLINE_TRAPEZOID_CONT_PER 10

/**
 * Include laser power in G0/G1/G2/G3/G5 commands with the 'S' parameter
 */
//#define LASER_MOVE_POWER

#if ENABLED(LASER_MOVE_POWER)
// Turn off the laser on G0 moves with no power parameter.
// If a power parameter is provided, use that instead.
//#define LASER_MOVE_G0_OFF

// Turn off the laser on G28 homing.
//#define LASER_MOVE_G28_OFF
#endif

/**
 * Inline flag inverted
 *
 * WARNING: M5 will NOT turn off the laser unless another move
 * is done (so G-code files must end with 'M5 I').
 */
//#define LASER_POWER_INLINE_INVERT

/**
 * Continuously apply inline power. ('M3 S3' == 'G1 S3' == 'M3 S3 I')
 *
 * The laser might do some weird things, so only enable this
 * feature if you understand the implications.
 */
//#define LASER_POWER_INLINE_CONTINUOUS

#else

#define SPINDLE_LASER_POWERUP_DELAY 50 // (ms) Delay to allow the spindle/laser to come
up to speed/power
#define SPINDLE_LASER_POWERDOWN_DELAY 50 // (ms) Delay to allow the spindle to stop

#endif

```

```

#endif
#endif

/**
 * Coolant Control
 *
 * Add the M7, M8, and M9 commands to turn mist or flood coolant on and off.
 *
 * Note: COOLANT_MIST_PIN and/or COOLANT_FLOOD_PIN must also be defined.
 */
#define COOLANT_CONTROL
#if ENABLED(COOLANT_CONTROL)
  #define COOLANT_MIST          // Enable if mist coolant is present
  #define COOLANT_FLOOD         // Enable if flood coolant is present
  #define COOLANT_MIST_INVERT false // Set "true" if the on/off function is reversed
  #define COOLANT_FLOOD_INVERT false // Set "true" if the on/off function is reversed
#endif

/**
 * Filament Width Sensor
 *
 * Measures the filament width in real-time and adjusts
 * flow rate to compensate for any irregularities.
 *
 * Also allows the measured filament diameter to set the
 * extrusion rate, so the slicer only has to specify the
 * volume.
 *
 * Only a single extruder is supported at this time.
 *
 * 34 RAMPS_14 : Analog input 5 on the AUX2 connector
 * 81 PRINTRBOARD : Analog input 2 on the Exp1 connector (version B,C,D,E)
 * 301 RAMBO : Analog input 3
 *
 * Note: May require analog pins to be defined for other boards.
 */
#define FILAMENT_WIDTH_SENSOR

#if ENABLED(FILAMENT_WIDTH_SENSOR)
  #define FILAMENT_SENSOR_EXTRUDER_NUM 0 // Index of the extruder that has the filament
  sensor. :[0,1,2,3,4]
  #define MEASUREMENT_DELAY_CM 14 // (cm) The distance from the filament sensor to the
  melting chamber

  #define FILWIDTH_ERROR_MARGIN 1.0 // (mm) If a measurement differs too much from nominal
  width ignore it
  #define MAX_MEASUREMENT_DELAY 20 // (bytes) Buffer size for stored measurements (1 byte
  per cm). Must be larger than MEASUREMENT_DELAY_CM.

  #define DEFAULT_MEASURED_FILAMENT_DIA DEFAULT_NOMINAL_FILAMENT_DIA // Set
  measured to nominal initially

  // Display filament width on the LCD status line. Status messages will expire after 5 seconds.
  #define FILAMENT_LCD_DISPLAY
#endif

/**
 * Power Monitor
 * Monitor voltage (V) and/or current (A), and -when possible- power (W)
 *
 * Read and configure with M430
 *
 * The current sensor feeds DC voltage (relative to the measured current) to an analog pin
 * The voltage sensor feeds DC voltage (relative to the measured voltage) to an analog pin
 */
#define POWER_MONITOR_CURRENT // Monitor the system current
#define POWER_MONITOR_VOLTAGE // Monitor the system voltage
#if EITHER(POWER_MONITOR_CURRENT, POWER_MONITOR_VOLTAGE)
  #define POWER_MONITOR_VOLTS_PER_AMP 0.05000 // Input voltage to the MCU analog pin per
  amp - DO NOT apply more than ADC_VREF!

```

```

#define POWER_MONITOR_CURRENT_OFFSET -1      // Offset value for current sensors with linear
function output
#define POWER_MONITOR_VOLTS_PER_VOLT 0.11786 // Input voltage to the MCU analog pin per
volt - DO NOT apply more than ADC_VREF!
#define POWER_MONITOR_FIXED_VOLTAGE 13.6     // Voltage for a current sensor with no voltage
sensor (for power display)
#endif

/**
 * CNC Coordinate Systems
 *
 * Enables G53 and G54-G59.3 commands to select coordinate systems
 * and G92.1 to reset the workspace to native machine space.
 */
#define CNC_COORDINATE_SYSTEMS

/**
 * Auto-report temperatures with M155 S<seconds>
 */
#define AUTO_REPORT_TEMPERATURES

/**
 * Include capabilities in M115 output
 */
#define EXTENDED_CAPABILITIES_REPORT
#if ENABLED(EXTENDED_CAPABILITIES_REPORT)
  // #define M115_GEOMETRY_REPORT
#endif

/**
 * Expected Printer Check
 * Add the M16 G-code to compare a string to the MACHINE_NAME.
 * M16 with a non-matching string causes the printer to halt.
 */
#define EXPECTED_PRINTER_CHECK

/**
 * Disable all Volumetric extrusion options
 */
#define NO_VOLUMETRICS

#if DISABLED(NO_VOLUMETRICS)
  /**
   * Volumetric extrusion default state
   * Activate to make volumetric extrusion the default method,
   * with DEFAULT_NOMINAL_FILAMENT_DIA as the default diameter.
   *
   * M200 D0 to disable, M200 Dn to set a new diameter (and enable volumetric).
   * M200 S0/S1 to disable/enable volumetric extrusion.
   */
  // #define VOLUMETRIC_DEFAULT_ON

  // #define VOLUMETRIC_EXTRUDER_LIMIT
  #if ENABLED(VOLUMETRIC_EXTRUDER_LIMIT)
    /**
     * Default volumetric extrusion limit in cubic mm per second (mm^3/sec).
     * This factory setting applies to all extruders.
     * Use 'M200 [T<extruder>] L<limit>' to override and 'M502' to reset.
     * A non-zero value activates Volume-based Extrusion Limiting.
     */
    #define DEFAULT_VOLUMETRIC_EXTRUDER_LIMIT 0.00 // (mm^3/sec)
  #endif
#endif

/**
 * Enable this option for a leaner build of Marlin that removes all
 * workspace offsets, simplifying coordinate transformations, leveling, etc.
 *
 * - M206 and M428 are disabled.
 * - G92 will revert to its behavior from Marlin 1.0.
 */

```

```

#define NO_WORKSPACE_OFFSETS

// Extra options for the M114 "Current Position" report
#define M114_DETAIL // Use 'M114` for details to check planner calculations
#define M114_REALTIME // Real current position based on forward kinematics
#define M114_LEGACY // M114 used to synchronize on every call. Enable if needed.

#define REPORT_FAN_CHANGE // Report the new fan speed when changed by M106 (and others)

/**
 * Set the number of proportional font spaces required to fill up a typical character space.
 * This can help to better align the output of commands like `G29 O` Mesh Output.
 *
 * For clients that use a fixed-width font (like OctoPrint), leave this set to 1.0.
 * Otherwise, adjust according to your client and font.
 */
#define PROPORTIONAL_FONT_RATIO 1.0

/**
 * Spend 28 bytes of SRAM to optimize the GCode parser
 */
#define FASTER_GCODE_PARSER

#if ENABLED(FASTER_GCODE_PARSER)
  #define GCODE_QUOTED_STRINGS // Support for quoted string parameters
#endif

#define GCODE_CASE_INSENSITIVE // Accept G-code sent to the firmware in lowercase

#define REPETIER_GCODE_M360 // Add commands originally from Repetier FW

/**
 * CNC G-code options
 * Support CNC-style G-code dialects used by laser cutters, drawing machine cams, etc.
 * Note that G0 feedrates should be used with care for 3D printing (if used at all).
 * High feedrates may cause ringing and harm print quality.
 */
#define PAREN_COMMENTS // Support for parentheses-delimited comments
#define GCODE_MOTION_MODES // Remember the motion mode (G0 G1 G2 G3 G5 G38.X) and apply
for X Y Z E F, etc.

// Enable and set a (default) feedrate for all G0 moves
#define G0_FEEDRATE 3000 // (mm/min)
#ifdef G0_FEEDRATE
  #define VARIABLE_G0_FEEDRATE // The G0 feedrate is set by F in G0 motion mode
#endif

/**
 * Startup commands
 *
 * Execute certain G-code commands immediately after power-on.
 */
#define STARTUP_COMMANDS "M17 Z"

/**
 * G-code Macros
 *
 * Add G-codes M810-M819 to define and run G-code macros.
 * Macros are not saved to EEPROM.
 */
#define GCODE_MACROS
#if ENABLED(GCODE_MACROS)
  #define GCODE_MACROS_SLOTS 5 // Up to 10 may be used
  #define GCODE_MACROS_SLOT_SIZE 50 // Maximum length of a single macro
#endif

/**
 * User-defined menu items that execute custom GCode
 */

```

[illegible]

```

#define I2CPE_ENC_1_ADDR      I2CPE_PRESET_ADDR_X // I2C address of the encoder. 30-200.
#define I2CPE_ENC_1_AXIS      X_AXIS              // Axis the encoder module is installed
on. <X|Y|Z|E>_AXIS.
#define I2CPE_ENC_1_TYPE      I2CPE_ENC_TYPE_LINEAR // Type of
encoder: I2CPE_ENC_TYPE_LINEAR -or-
// I2CPE_ENC_TYPE_ROTARY.
#define I2CPE_ENC_1_TICKS_UNIT 2048              // 1024 for magnetic strips with 2mm poles;
2048 for
// 1mm poles. For linear encoders this is ticks / mm,
// for rotary encoders this is ticks / revolution.
// #define I2CPE_ENC_1_TICKS_REV (16 * 200) // Only needed for rotary encoders; number of
stepper
// steps per full revolution (motor steps/rev * microstepping)
// #define I2CPE_ENC_1_INVERT // Invert the direction of axis travel.
#define I2CPE_ENC_1_EC_METHOD I2CPE_ECM_MICROSTEP // Type of error error correction.
#define I2CPE_ENC_1_EC_THRESH 0.10              // Threshold size for error (in mm) above which
the
// printer will attempt to correct the error; errors
// smaller than this are ignored to minimize effects of
// measurement noise / latency (filter).

#define I2CPE_ENC_2_ADDR      I2CPE_PRESET_ADDR_Y // Same as above, but for encoder 2.
#define I2CPE_ENC_2_AXIS      Y_AXIS
#define I2CPE_ENC_2_TYPE      I2CPE_ENC_TYPE_LINEAR
#define I2CPE_ENC_2_TICKS_UNIT 2048
// #define I2CPE_ENC_2_TICKS_REV (16 * 200)
// #define I2CPE_ENC_2_INVERT
#define I2CPE_ENC_2_EC_METHOD I2CPE_ECM_MICROSTEP
#define I2CPE_ENC_2_EC_THRESH 0.10

#define I2CPE_ENC_3_ADDR      I2CPE_PRESET_ADDR_Z // Encoder 3. Add additional
configuration options
#define I2CPE_ENC_3_AXIS      Z_AXIS              // as above, or use defaults below.

#define I2CPE_ENC_4_ADDR      I2CPE_PRESET_ADDR_E // Encoder 4.
#define I2CPE_ENC_4_AXIS      E_AXIS

#define I2CPE_ENC_5_ADDR      34                  // Encoder 5.
#define I2CPE_ENC_5_AXIS      E_AXIS

// Default settings for encoders which are enabled, but without settings configured above.
#define I2CPE_DEF_TYPE        I2CPE_ENC_TYPE_LINEAR
#define I2CPE_DEF_ENC_TICKS_UNIT 2048
#define I2CPE_DEF_TICKS_REV   (16 * 200)
#define I2CPE_DEF_EC_METHOD    I2CPE_ECM_NONE
#define I2CPE_DEF_EC_THRESH    0.1

// #define I2CPE_ERR_THRESH_ABORT 100.0 // Threshold size for error (in mm) error on
any given
// axis after which the printer will abort. Comment out to
// disable abort behavior.

#define I2CPE_TIME_TRUSTED    10000              // After an encoder fault, there must be no further
fault
// for this amount of time (in ms) before the encoder
// is trusted again.

/**
 * Position is checked every time a new command is executed from the buffer but during long moves,
 * this setting determines the minimum update time between checks. A value of 100 works well with
 * error rolling average when attempting to correct only for skips and not for vibration.
 */
#define I2CPE_MIN_UPD_TIME_MS 4                  // (ms) Minimum time between encoder checks.

// Use a rolling average to identify persistent errors that indicate skips, as opposed to vibration and noise.
#define I2CPE_ERR_ROLLING_AVERAGE

#endif // I2C_POSITION_ENCODERS

```

```

/**
 * Analog Joystick(s)
 */
#define JOYSTICK
#if ENABLED(JOYSTICK)
  #define JOY_X_PIN 5 // RAMPS: Suggested pin A5 on AUX2
  #define JOY_Y_PIN 10 // RAMPS: Suggested pin A10 on AUX2
  #define JOY_Z_PIN 12 // RAMPS: Suggested pin A12 on AUX2
  #define JOY_EN_PIN 44 // RAMPS: Suggested pin D44 on AUX2

  // #define INVERT_JOY_X // Enable if X direction is reversed
  // #define INVERT_JOY_Y // Enable if Y direction is reversed
  // #define INVERT_JOY_Z // Enable if Z direction is reversed

  // Use M119 with JOYSTICK_DEBUG to find reasonable values after connecting:
  #define JOY_X_LIMITS { 5600, 8190-100, 8190+100, 10800 } // min, deadzone start, deadzone end, max
  #define JOY_Y_LIMITS { 5600, 8250-100, 8250+100, 11000 }
  #define JOY_Z_LIMITS { 4800, 8080-100, 8080+100, 11550 }
  // #define JOYSTICK_DEBUG
#endif

/**
 * Mechanical Gantry Calibration
 * Modern replacement for the Prusa TMC_Z_CALIBRATION.
 * Adds capability to work with any adjustable current drivers.
 * Implemented as G34 because M915 is deprecated.
 */
#define MECHANICAL_GANTRY_CALIBRATION
#if ENABLED(MECHANICAL_GANTRY_CALIBRATION)
  #define GANTRY_CALIBRATION_CURRENT 600 // Default calibration current in ma
  #define GANTRY_CALIBRATION_EXTRA_HEIGHT 15 // Extra distance in mm past Z_###_POS to
  move
  #define GANTRY_CALIBRATION_FEEDRATE 500 // Feedrate for correction move
  // #define GANTRY_CALIBRATION_TO_MIN // Enable to calibrate Z in the MIN direction

  // #define GANTRY_CALIBRATION_SAFE_POSITION { X_CENTER, Y_CENTER } // Safe position for
  nozzle
  // #define GANTRY_CALIBRATION_XY_PARK_FEEDRATE 3000 // XY Park Feedrate - MMM
  // #define GANTRY_CALIBRATION_COMMANDS_PRE ""
  #define GANTRY_CALIBRATION_COMMANDS_POST "G28" // G28 highly recommended to ensure
  an accurate position
#endif

/**
 * MAX7219 Debug Matrix
 *
 * Add support for a low-cost 8x8 LED Matrix based on the Max7219 chip as a realtime status display.
 * Requires 3 signal wires. Some useful debug options are included to demonstrate its usage.
 */
#define MAX7219_DEBUG
#if ENABLED(MAX7219_DEBUG)
  #define MAX7219_CLK_PIN 64
  #define MAX7219_DIN_PIN 57
  #define MAX7219_LOAD_PIN 44

  // #define MAX7219_GCODE // Add the M7219 G-code to control the LED matrix
  #define MAX7219_INIT_TEST 2 // Test pattern at startup: 0=none, 1=sweep, 2=spiral
  #define MAX7219_NUMBER_UNITS 1 // Number of Max7219 units in chain.
  #define MAX7219_ROTATE 0 // Rotate the display clockwise (in multiples of +/- 90°)
  // connector at: right=0 bottom=-90 top=90 left=180
  // #define MAX7219_REVERSE_ORDER // The individual LED matrix units may be in reversed order
  // #define MAX7219_SIDE_BY_SIDE // Big chip+matrix boards can be chained side-by-side

  /**
   * Sample debug features
   * If you add more debug displays, be careful to avoid conflicts!
   */
  #define MAX7219_DEBUG_PRINTER_ALIVE // Blink corner LED of 8x8 matrix to show that the
  firmware is functioning

```

```

#define MAX7219_DEBUG_PLANNER_HEAD 3 // Show the planner queue head position on this and
the next LED matrix row
#define MAX7219_DEBUG_PLANNER_TAIL 5 // Show the planner queue tail position on this and the
next LED matrix row

#define MAX7219_DEBUG_PLANNER_QUEUE 0 // Show the current planner queue depth on this and
the next LED matrix row
// If you experience stuttering, reboots, etc. this option can reveal how
// tweaks made to the configuration are affecting the printer in real-time.

#endif

/**
 * NanoDLP Sync support
 *
 * Add support for Synchronized Z moves when using with NanoDLP. G0/G1 axis moves will output
 "Z_move_comp"
 * string to enable synchronization with DLP projector exposure. This change will allow to use
 * [[WaitForDoneMessage]] instead of populating your gcode with M400 commands
 */
// #define NANODLP_Z_SYNC
// if ENABLED(NANODLP_Z_SYNC)
// #define NANODLP_ALL_AXIS // Enables "Z_move_comp" output on any axis move.
// Default behavior is limited to Z axis only.

#endif

/**
 * WiFi Support (Espressif ESP32 WiFi)
 */
// #define WIFISUPPORT // Marlin embedded WiFi management
// #define ESP3D_WIFISUPPORT // ESP3D Library WiFi management (https://github.com/luc-github/ESP3DLib)

// if EITHER(WIFISUPPORT, ESP3D_WIFISUPPORT)
// #define WEBSUPPORT // Start a webserver (which may include auto-discovery)
// #define OTASUPPORT // Support over-the-air firmware updates
// #define WIFI_CUSTOM_COMMAND // Accept feature config commands (e.g., WiFi ESP3D) from the
host

/**
 * To set a default WiFi SSID / Password, create a file called Configuration_Secure.h with
 * the following defines, customized for your network. This specific file is excluded via
 * .gitignore to prevent it from accidentally leaking to the public.
 *
 * #define WIFI_SSID "WiFi SSID"
 * #define WIFI_PWD "WiFi Password"
 */
// #include "Configuration_Secure.h" // External file with WiFi SSID / Password
#endif

/**
 * Průša Multi-Material Unit v2
 * Enable in Configuration.h
 */
// if ENABLED(PRUSA_MMU2)

// Serial port used for communication with MMU2.
// For AVR enable the UART port used for the MMU. (e.g., mmuSerial)
// For 32-bit boards check your HAL for available serial ports. (e.g., Serial2)
#define MMU2_SERIAL_PORT 2
#define MMU2_SERIAL mmuSerial

// Use hardware reset for MMU if a pin is defined for it
// #define MMU2_RST_PIN 23

// Enable if the MMU2 has 12V stepper motors (MMU2 Firmware 1.0.2 and up)
// #define MMU2_MODE_12V

// G-code to execute when MMU2 F.I.N.D.A. probe detects filament runout
#define MMU2_FILAMENT_RUNOUT_SCRIPT "M600"

```



```

// Add an LCD menu for MMU2
// #define MMU2_MENUS
// #if ENABLED(MMU2_MENUS)
//   Settings for filament load / unload from the LCD menu.
//   This is for Průša MK3-style extruders. Customize for your hardware.
// #define MMU2_FILAMENTCHANGE_EJECT_FEED 80.0
// #define MMU2_LOAD_TO_NOZZLE_SEQUENCE \
//   { 7.2, 1145 }, \
//   { 14.4, 871 }, \
//   { 36.0, 1393 }, \
//   { 14.4, 871 }, \
//   { 50.0, 198 }

// #define MMU2_RAMMING_SEQUENCE \
//   { 1.0, 1000 }, \
//   { 1.0, 1500 }, \
//   { 2.0, 2000 }, \
//   { 1.5, 3000 }, \
//   { 2.5, 4000 }, \
//   { -15.0, 5000 }, \
//   { -14.0, 1200 }, \
//   { -6.0, 600 }, \
//   { 10.0, 700 }, \
//   { -10.0, 400 }, \
//   { -50.0, 2000 }
// #endif

/**
 * MMU Extruder Sensor
 *
 * Support for a Průša (or other) IR Sensor to detect filament near the extruder
 * and make loading more reliable. Suitable for an extruder equipped with a filament
 * sensor less than 38mm from the gears.
 *
 * During loading the extruder will stop when the sensor is triggered, then do a last
 * move up to the gears. If no filament is detected, the MMU2 can make some more attempts.
 * If all attempts fail, a filament runout will be triggered.
 */
// #define MMU_EXTRUDER_SENSOR
// #if ENABLED(MMU_EXTRUDER_SENSOR)
//   #define MMU_LOADING_ATTEMPTS_NR 5 // max. number of attempts to load filament if first load fail
// #endif

/**
 * Using a sensor like the MMU2S
 * This mode requires a MK3S extruder with a sensor at the extruder idler, like the MMU2S.
 * See https://help.prusa3d.com/en/guide/3b-mk3s-mk2-5s-extruder-upgrade\_41560, step 11
 */
// #define PRUSA_MMU2_S_MODE
// #if ENABLED(PRUSA_MMU2_S_MODE)
//   #define MMU2_C0_RETRY 5 // Number of retries (total time = timeout*retries)

//   #define MMU2_CAN_LOAD_FEEDRATE 800 // (mm/min)
//   #define MMU2_CAN_LOAD_SEQUENCE \
//     { 0.1, MMU2_CAN_LOAD_FEEDRATE }, \
//     { 60.0, MMU2_CAN_LOAD_FEEDRATE }, \
//     { -52.0, MMU2_CAN_LOAD_FEEDRATE }

//   #define MMU2_CAN_LOAD_RETRACT 6.0 // (mm) Keep under the distance between Load
//   Sequence values
//   #define MMU2_CAN_LOAD_DEVIATION 0.8 // (mm) Acceptable deviation

//   #define MMU2_CAN_LOAD_INCREMENT 0.2 // (mm) To reuse within MMU2 module
//   #define MMU2_CAN_LOAD_INCREMENT_SEQUENCE \
//     { -MMU2_CAN_LOAD_INCREMENT, MMU2_CAN_LOAD_FEEDRATE }

// #endif

// #define MMU2_DEBUG // Write debug info to serial output

```

```

#endif // PRUSA_MMU2

/**
 * Advanced Print Counter settings
 */
#if ENABLED(PRINTCOUNTER)
  #define SERVICE_WARNING_BUZZES 3
  // Activate up to 3 service interval watchdogs
  //#define SERVICE_NAME_1    "Service S"
  //#define SERVICE_INTERVAL_1 100 // print hours
  //#define SERVICE_NAME_2    "Service L"
  //#define SERVICE_INTERVAL_2 200 // print hours
  //#define SERVICE_NAME_3    "Service 3"
  //#define SERVICE_INTERVAL_3 1 // print hours
#endif

// @section develop

//
// M100 Free Memory Watcher to debug memory usage
//
//#define M100_FREE_MEMORY_WATCHER

//
// M42 - Set pin states
//
//#define DIRECT_PIN_CONTROL

//
// M43 - display pin status, toggle pins, watch pins, watch endstops & toggle LED, test servo probe
//
//#define PINS_DEBUGGING

// Enable Marlin dev mode which adds some special commands
//#define MARLIN_DEV_MODE

```

APPENDIX F: CAD Drawings for Enclosure

Note: Scale on the drawings are not accurate due to image scaling.

