DESIGN AND FABRICATE REAL-TIME INTERNET OF THINGS (IoT) MONITORING SYSTEM FOR FLOATING SOLAR PHOTOVOLTAIC (PV) SYSTEM

TING KAI ZIN

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

Faculty of Engineering and Green Technology
Universiti Tunku Abdul Rahman

May 2023

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	:	[illegible]
Name	:	Ting Kai Zin
ID No.	:	18AGB01864
Date	:	30 April 2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DESIGN AND FABRICATE REAL-TIME INTERNET OF THINGS (IoT) MONITORING SYSTEM FOR FLOATING SOLAR PHOTOVOLTAIC (PV) SYSTEM"** was prepared by **TING KAI ZIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons) Industrial Engineering at Universiti Tunku Abdul Rahman.

Signature	: Nicholas
Supervisor	: Ts. Dr. Tan Ming Hui
Date	: 1st May 2023
Signature	: Lim Boon Han
Co-Supervisor	: Ir. Dr. Lim Boon Han
Date	: 1st May 2023

ACKNOWLEDGEMENTS

I would like to thank my lecturers and my friends who contributed to the successful accomplishment of this project. I would like to express my gratitude to my supervisor, Ts. Dr. Tan Ming Hui for his invaluable advice, guidance and patience throughout the progression of this project.

In addition, I would like to extend my gratitude to the personnel and authorities of Universiti Tunku Abdul Rahman for providing me with a great facilities and environment that enable me to conduct and complete my research.

Just as importantly, I would like to thank my parents who have given me encouragement and mental support to face every challenge and come out better and wiser than before.

DESIGN AND FABRICATE REAL-TIME INTERNET OF THINGS (IoT) MONITORING SYSTEM FOR FLOATING SOLAR PHOTOVOLTAIC (PV) SYSTEM

ABSTRACT

Due to increasing global need for renewable energy, photovoltaic systems have grown in popularity. Solar photovoltaic panel is one of the power generator that emits zero greenhouse gases in the process of energy conversion from sunlight to electricity. While there are different solar panel mounting methods, several studies show that floating type photovoltaic systems have improved energy conversion efficiency and power output. This is because conventional photovoltaic systems that are mounted on ground and roof-top can absorb a lot of heat from sunlight. The maximum power generated and efficiency of solar panels decrease as the temperature of the panels rises. Meanwhile, floating type photovoltaic system is one of the alternatives for limited land space for solar panel system installation. To evaluate and analyse the performance of floating type photovoltaic system, and compare the performance with ground mounted photovoltaic system, an IoT data monitoring system is designed and fabricated in this project. The purpose of using an IoT data monitoring system is to obtain real time data. In other words, to acquire the most recent information about the status of the solar panels immediately following data collection. The major function of the IoT data monitoring system includes gathering data from the solar panel and its surroundings, and sending it to a cloud server. In this project, Google Spreadsheet was configured as the cloud server using Google Apps Script. To obtain accurate data readings, the measurement system has been calibrated and validated. As a result, an IoT data collection and data monitoring system with percentage error around 1.5 % was developed and implemented.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATION

PV	Photovoltaic
FPV	Floating photovoltaics
GPV	Ground mounted photovoltaics
IoT	Internet of Things
HDPE	High Density Polyethylene
GRP	Glass Fibre Reinforced Plastic
UV	Ultraviolet
NodeMCU	Node Microcontroller Unit
OLED	Organic Light-Emitting Diodes
RTC	Real Time Clock
SD	Secure Digital
SPDT	Single Pole Double Throw
NPN	Negative-Positive-Negative
I/O	Input/Output
IDE	Integrated Development Environment
PWM	Pulse Width Modulation
SRAM	Static Random-Access Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
USB	Universal Serial Bus

GPIO	General Purpose Input Output
UART	Universal Asynchronous Receiver Transmitter
RX	Receive
TX	Transmit
ADC	Analogue to Digital Converter
NC	Normally Close
NO	Normally Open
COM	Common
I2C	Inter-Integrated Circuit
VCC	Common Collector Voltage
GND	Ground
SCL	Serial Clock Line
SDA	Serial Data Line
MISO	Master In Slave Out
MOSI	Master Out Slave In
SCK	Serial Clock
CS	Chip Select
URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
XML	Extensible Markup Language
JSON	JavaScript Object Notation

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Harvesting energy from the sun has high potential in Malaysia due to its geographical location close to the equatorial line. Around 4000 to 5000 Wh/m^2 of solar radiations can be received in Malaysia daily (Azhari et al., 2008). There are several ways to harvest solar energy, including solar thermal system and solar photovoltaic (PV) system. Solar energy is one of the renewable energies that can be converted into electricity in a green, low-environmental-impact way, as it emits zero Green Houses Gases in the energy conversion process.

Solar PV system, is the technology of generating flow of electricity with the application of compound semiconductor material. The panel is usually structured with layers of crystalline cells made from silicon. Then, the layers of PV cells are covered by transparent glass on the layer facing to the sunlight, a piece of polymer plastic at the bottom, and aluminium frame (Svarc, 2020).

There are several factors affecting the energy conversion efficiency of the solar PV panel, including temperature, installation method, shadings by foreign objects, difference in spectral irradiance and etc. The efficiency of solar PV panel plays an important role as the higher the efficiency of the PV panel is, the higher the energy output generated.

The installation of conventional PV system is typically done on land. This type of installation usually involves larger scale to generate enough electricity. However, the availability of land limits further development of land type PV system. The main reason to this is reservation of land for more important purposes especially food related one. For example, land for agriculture purposes. Once land is used for PV system implementation, the land is occupied and became solely for PV system. The system is often not able to be integrated with any other usage. Therefore, exploration on PV system installation method other than land type is gaining more concerns.

Floating PV system is the design of PV system in a floatable way and is implemented on water bodies. Speaking of large scale PV system, floating PV has the potential to replace land type PV system as water bodies are readily available in Malaysia. Depending on the design, the structure of floating PV system can be as simple as fixing normal PV panel on floating platform, or redesigning PV panel that is waterproof and can directly contact with water. In either way, research shows improvement of performance due to lower PV panel temperature compared to land type one. Other benefits of floating PV system include availability of water for cleaning the surface of PV panel, reduction of water losses from evaporation, and improvement of water quality.

A study of floating PV panel done by Majid et. al. (2013) under Malaysia climate condition shows improvement of floating PV performance. The result shows under different solar irradiance level, the temperature of floating PV is always lower than normal PV, while the power output of floating PV is always higher than that of normal PV. The experiment has been conducted in two hours, with 15.5 % increase of energy gain by the floating PV compared to normal PV.

Data collection and monitoring for floating PV system is more challenging than the conventional PV system. This is because floating PV panel could be located at the middle of water body and is unreachable by human if no walkway is built on the platform. Therefore, floating PV monitoring system integrated with Internet of Things (IoT) is designed in this project to solve the aforementioned problem.

1.2 Problem Statements

Solar PV panels were traditionally built on land. Nowadays, with reduction of land availability and the need for conservation of land for natural reserves, more solutions emerged such as PV panels mounted on rooftop, canal top, and floater. Each method has its own advantages and disadvantages. Ground mounted PV panels allow the power system to be constructed in a larger scale. In contrast, rooftop mounted PV panels usually could only be built on the space of rooftop. Hence, it is only capable to generate electricity for household usage.

Both ground mounted and rooftop PV panels has the common downside, which is reduction of efficiency due to high temperature as a result of long exposure under sunlight. This is caused by temperature coefficient when the temperature of solar panel rises above 25 degrees Celsius. Temperature coefficient is the decrease on the efficiency of solar panel with 1 degree Celsius increase of temperature. PV panels usually have the temperature coefficient ranging from $-0.3\ \%/℃$ to $-0.5\ \%/℃$ (Ost, 2020). Therefore, floating PV panel is one of the solutions to the problem of limited land, while it can also be built into larger scale depending on the size of water body. Hence, floating PV panel could be more competitive than rooftop PV panel when larger surface area is possible. At the same time, the effect of water evaporation at the water body could help cool down heated PV panel, enhancing the energy conversion efficiency.

Investigation on how temperature can be reduced with floating PV panel, and how much energy output can be yield from this method was carried out in this project. Floating PV performance monitoring could be done by storing the data in hardware storage such as SD card. However, the data stored offline could be difficult to be retrieved by human if no walkway is built on the water body. Furthermore, the offline data needs to be post-processed in order to be converted into tables and charts in an Excel spreadsheet. Thus, to improve the safety and productivity of the data collection process, another solution which integrates the data collection system with IoT is applied in this project to obtain data remotely and allow real time monitoring.

1.3 Aims and Objectives

The main purposes of conducting this project entitled **DESIGN AND FABRICATE REAL-TIME INTERNET OF THINGS (IoT) MONITORING SYSTEM FOR FLOATING SOLAR PHOTOVOLTAIC (PV) SYSTEM** are as shown below:

1) To design and construct an IoT data collection and data monitoring system for floating PV panel
2) To calibrate and validate the measurement values of the IoT data collection and data monitoring system for floating PV panel
3) To test the functionality and stability of the IoT data collection and data monitoring system for floating PV panel

1.4 Outline of Report

Chapter 1: Introduction

A brief description on the background, problem statement and purpose of performance evaluation of floating PV panel in Malaysia.

Chapter 2: Literature Review

A general review on the results and discussion obtained from several journals and resources.

Chapter 3: Methodology

Explanation of tasks performed in this project. A summary of tasks concluded in two Gantt Charts with the timelines stated respectively.

Chapter 4: Results and Discussion

Analysis and discussion made based on the results obtained. A summary of problems encountered and solutions implemented were concluded in a table.

Chapter 5: Conclusion and Recommendation

Conclusion made based on the entire report and achievement of the project. A few recommendations were suggested for future improvement of any similar project.

CHAPTER 2

LITERATURE REVIEW

2.1 External Factors Affecting Performance of PV Panels

2.1.1 Temperature

Temperature has negative effect to the efficiency and performance of PV panels. The temperature of a PV panels under operation rises due to only partial number of photons striking on the PV panels being converted into electricity, while the rest of the energy is converted into heat (Pradhan and Panda, 2017). PV panel absorbs this heat energy and thus the temperature of the panel increase. The study done by Pradhan and Panda (2017) shows that when the temperature increase, the negative effects to the module include reduction in maximum power generated, fill factor and efficiency. Furthermore, when temperature of the PV panel exceeded the upper limit point mentioned by the manufacturer, it can result long term damage to the panel (Mathur et al., 1984). Other than increasing temperature, Pradhan and Panda mentioned that ununiform temperature distribution could also reduce the efficiency of PV panel.

2.1.2 Humidity

Pradhan and Panda (2017) conducted an indoor experiment and created humid condition to measure the effect of humidity to PV modules. Reduction of maximum power, fill factor, and efficiency is observed with the rise of humidity percentage. Solar module degradation occurs when water vapor penetrates into the cells. Pradhan and Panda mentioned that PV modules is subjected to delamination damage in moisture condition, as the moisture erode the interfacial adhesion of the cells. During rain periods, the output of the PV module falls as the humidity increases.

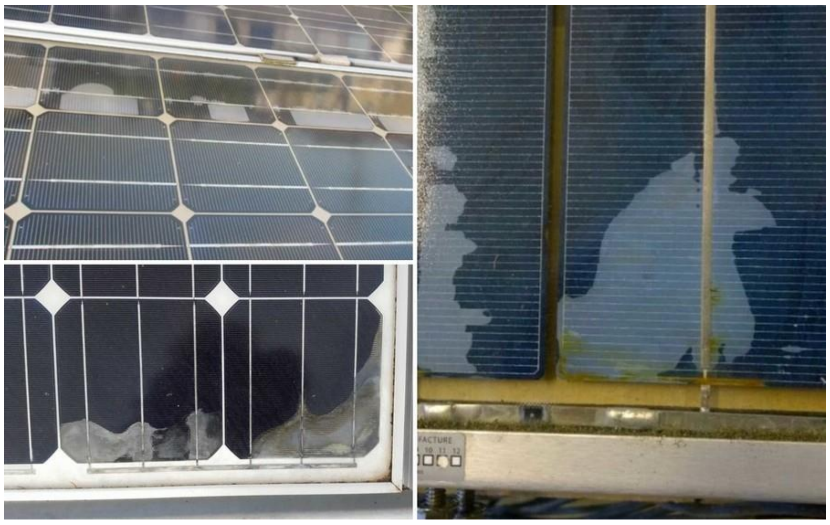


Figure 2.1: Examples of delamination on PV panels (Xia, 2021)

2.1.3 Shading

Shading is blockage of sunlight from striking on PV panels. It can be categorised into hard shading or soft shading. The former refers to complete blockage of sunlight by solid object while the latter refers to partial blockage such as shading by smog (Maghami et al., 2016). Many external objects can cause shading on PV module, whether they are from nature or man-made. For examples, bird droppings, dry leaves, buildings etc. Ununiform shading on solar module causes mismatch of solar cell, which happens when power generated by unshaded cell is dissipated by shaded cell. This can cause over heating on the module, resulting irreversible damage

(PVEducation, n.d.). Pradhan and Panda (2017) conducted the research on the effect of different degrees of shading on PV panels. The result shows that as shading percentage increase, the fill factor and efficiency decrease. To avoid mismatch losses due to shading, bypass diode can be installed in the PV module.

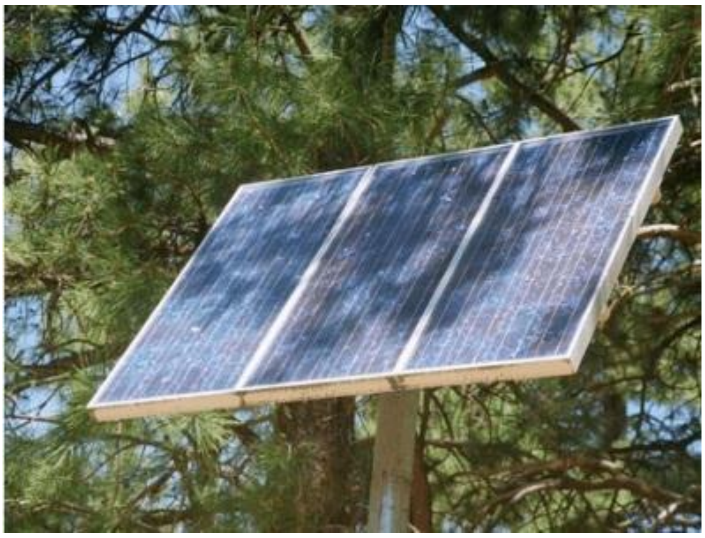


Figure 2.2: Partially shaded PV module (Dwivedi, Yadav and Saket, 2016)

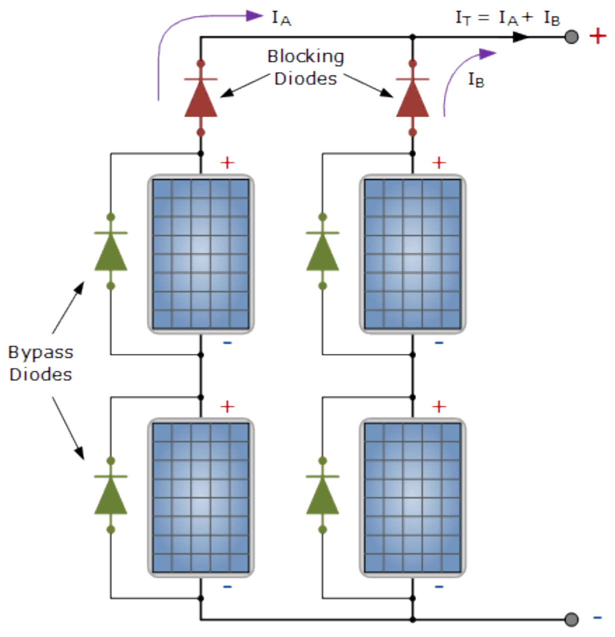


Figure 2.3: Bypass diodes installed parallelly in each PV panel (Electronics Tutorial, n.d.)

2.1.4 Wind Velocity

In the experiment on the effect of wind speed to the performance of PV panel, Pradhan and Panda (2017) found that the combined effect of cooling and dust removal from wind boost the efficiency of PV panels. Adequate amount of air flow helps cooling down PV modules. Furthermore, wind can also help to remove dust accumulated on PV panels. However, wind speed that is too high or too low will not improve the performance of PV panels. The result from Pradhan and Panda shows that the maximum output, fill factor and efficiency are the highest at wind speed of 10 m/s amongst other lower wind speeds. The performance was also lower at wind speed of 18 m/s.

2.2 Floating PV panels

2.2.1 Advantages of Floating PV panels

There are several advantages of implementing Floating PV panel. However, the main reason of implementing floating PV panels is due to its advantage of not utilizing precious land for PV system. Only small amount of land is required for devices such as inverter and electric meter. Depending on the size of water surface, floating PV panels has potential to provide equivalent scale of power generation as ground mounted PV panels. Water bodies that generally not utilized by other activities, is abundantly available such as lakes, ponds, man-made water reservoirs and off-shores.

The next main purpose of implementing floating PV panels is due to the improved efficiency of the energy conversion from sunlight to electricity. The efficiency of floating PV panels can be 11 % higher than ground mounted PV panels (Choi, 2014). Research done by Liu et al., (2017) concluded that floating PV panels has temperature around 3.5 ℃ lower than the ground mounted one. There are some theories supporting this efficiency improvement. Firstly, the evaporation effect of water body makes the surrounding temperature of floating solar panels cooler (Sahu, Yadav and Sudhakar, 2016). Secondly, the ambient temperature of water body is lower due to the reflectivity of water surface (Sahu, Yadav and Sudhakar, 2016). As a result, more sunlight can be reflected by water surface as compared to ground. Soil absorbs most of the heat from sunlight than reflecting them, resulting higher ambient temperature. Thirdly, water has higher specific heat capacity, which means it can hold more heat energy to raise its temperature by one degree Celsius. Moreover, PV panels floating on water body will experience less dust accumulation (Sahu, Yadav and Sudhakar, 2016). Therefore, performance of floating PV panels is better with their cleaner surface.

In terms of cleaning and maintenance job, water is readily available for cleaning the surface of solar panels. This makes cleaning solar panels more cost saving as water do not have to be pumped from other water sources that might be far away. A floating PV panels project designed by MIRARCO does not even have a floating platform to place their solar panels. The solar panels contact with water surface directly so the cooling effect by water can be boosted, and the surface of PV panels remains clean due to the self-cleaning effect (Trapani and Redón Santafé, 2014).

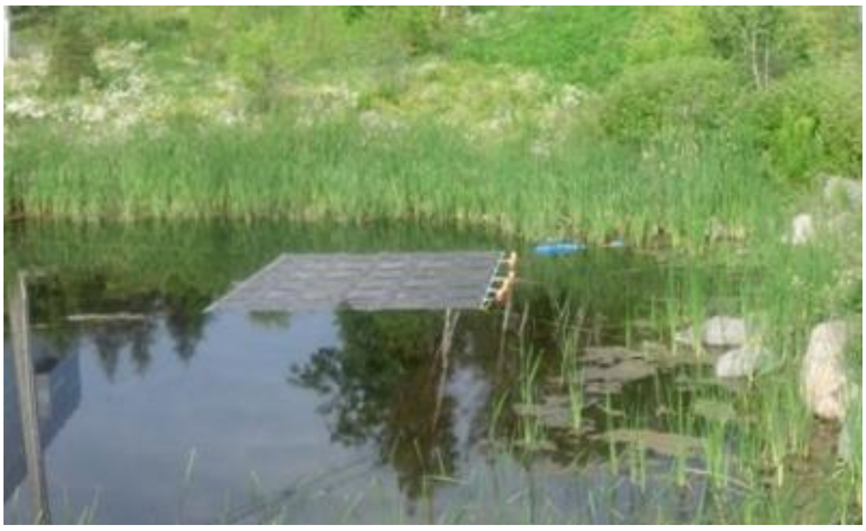


Figure 2.4: Floating PV panel designed by MIRARCO (Trapani and Redón Santafé, 2014)

Other advantages of FPV include reduced water losses from evaporation and improved water quality. These two advantages are due to shading of PV panels on the water surface preventing sunlight from traveling into the water. As water surface being covered by PV panels, less heat from sunlight is absorbed by the water body. Hence, this could result lower water evaporation rate, preventing water from escaping in the form of vapor. Water loses from evaporation can be reduced up to 33 % and 50 % for natural water bodies such as lakes and ponds, and man-made facility respectively (Choi, 2014). Furthermore, less sunlight penetration also reduces photosynthesis process of algae, preventing them from overgrowing.

2.2.2 Challenges of Floating PV Panels

Due to the external factors in natural water bodies, the structure of floating PV panels needs to be carefully designed to handle different conditions. Durability of PV panels may be an issue. Most importantly, the PV panel system needs to be stable enough to float on water most of the time, and strong enough to withstand external forces such as wind load and water tides (Sahu, Yadav and Sudhakar, 2016).

The first challenge is the strength of the system. Well-designed structure of floating PV system is required to withstand external forces caused by strong wind and waves. Due to these external forces, solar panels may experience more stress and vibration then those ground mounted one. This could result formation of cracks on the rigid PV panels, reducing its electricity output and durability (Cazzaniga et al., 2018). Furthermore, right material for the floating structure should be chosen depending on the type of water body. Due to the effect of electrochemical corrosion, corrosion of metal frame could be severe and this reduce the strength of the floating structure. The material may need to be highly anti-corrosion if the system is to be installed in sea water. Secondly, electricity cables used in the system are always contact with water. Hence, they need to be well insulated and durable enough to reduce risk of shocking and other safety issues. Thirdly, the operation of floating PV panels may get interrupted by wildlife such as birds, fish and other local animals. For instance, bird droppings may accumulate on the surface of PV panels if the installation location is near to the habitat of birds. This could result reduced PV efficiency due to the shading on the panel.

2.3 Floating PV designs

2.3.1 Floating PV Panels with Pontoon

Pontoon is a giant floating structure that is made up by smaller sized floats. The modular design allows suitable sizing of the floating platform for PV panels to be built according to the requirement. The materials for the floats could be made up of High Density Polyethylene (HDPE) or Glass Fibre Reinforced Plastic (GRP) (Sahu, Yadav and Sudhakar, 2016). HDPE is characterised by its strength, lightweight, UV and corrosion resistance. Research shows HDPE does not have any sign of degradation after UV exposure (Sahu, Sudhakar and Sarviya, 2019). GRP has similar properties with HDPE and is highly impact resistant (Engineered Composites, n.d.).

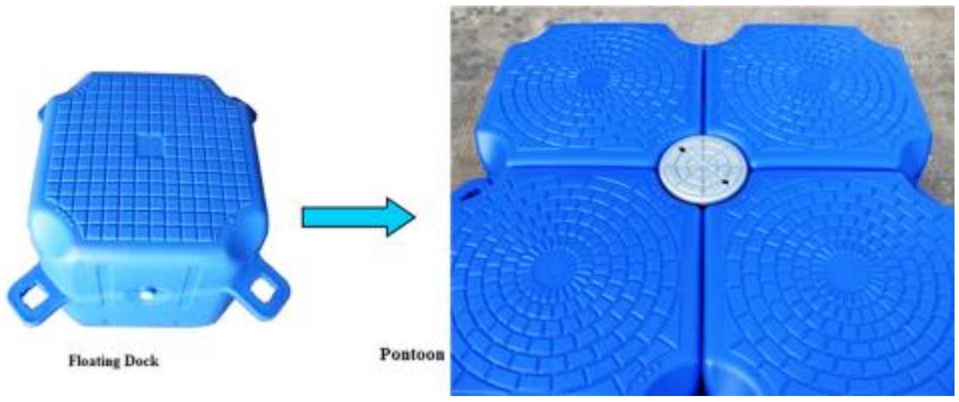


Figure 2.5: Modular structure of pontoon (Sahu, Yadav and Sudhakar, 2016)

The characteristic of this type of floating PV panels is that the panels do not get contacted with water directly. It is a safer way to directly utilize conventional PV panels in floating PV projects, as water resistance of the PV panel is often unsure. An example of large-scale PV power plant done by Kyocera Corporation and Century Tokyo Leasing Corporation is as shown in Figure 2.6. The design of the floater is strong enough to resist typhoon (Sahu, Yadav and Sudhakar, 2016).

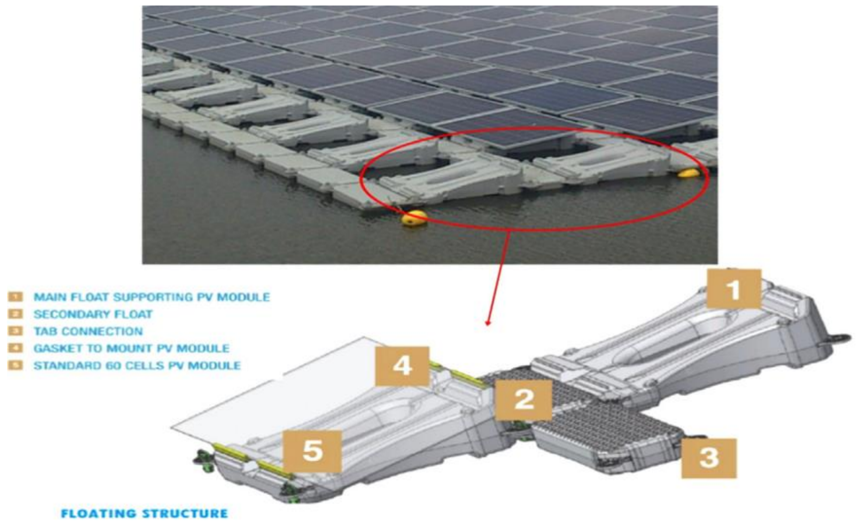

1 MAIN FLOAT SUPPORTING PV MODULE
2 SECONDARY FLOAT
3 TAB CONNECTION
4 GASKET TO MOUNT PV MODULE
5 STANDARD 60 CELLS PV MODULE
FLOATING STRUCTURE

Figure 2.6: Floating solar panel assembly structure (Sahu, Yadav and Sudhakar, 2016)

2.3.2 Flexible Floating PV Panels

The idea behind flexible floating PV is to solve the durability issue of rigid floating PV mentioned beforehand. Flexible PV is able to follow the motion of waves rather than withstanding them (Trapani and Redón Santafé, 2014). Therefore, they encounter less impact from wind loads and waves as compared to the conventional rigid one. This novel design opens the opportunity of implementing floating PV panels in off-shore areas, which are subjected to greater wave movements and wind load. What makes this flexible thin film PV panel differ from conventional PV panel is that they are very lightweight. The design from MIRARCO allows buoyancy force to be integrated with the PV panels, by trapping air within the laminated thin film (Trapani and Redón Santafé, 2014).

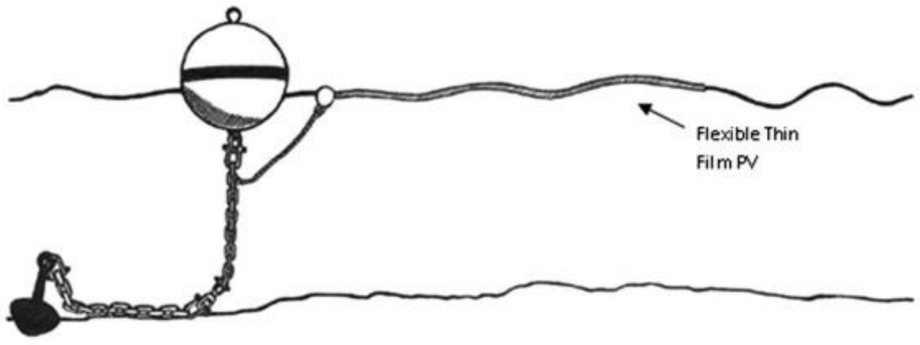


Figure 2.7: Flexible thin film PV (Trapani and Redón Santafé, 2014)

2.3.3 Submerged PV Panels

Submerged PV panels have several advantages. Firstly, due to the entire PV panels being completely submerged in water, the cooling effect of the panels will be enhanced. Secondly, this idea eliminates the need for cleaning PV panels (Ranjbarann et al., 2019). The design for submerged PV panels is not limited for flexible one. For example, the design from SCINTEC applies conventional rigid PV panels, while completely submerging it under water surface (Trapani and Redón Santafé, 2014). Since rigid PV panels cannot withstand strong wave movement, the module is designed to be submerged under water up to 2 m to avoid the waves. When the water surface is calm, the module can be lifted very close to the water surface (0 to 2 mm under water) to receive effective solar radiation without any deter from water (Rosa-Clot et al., 2010).

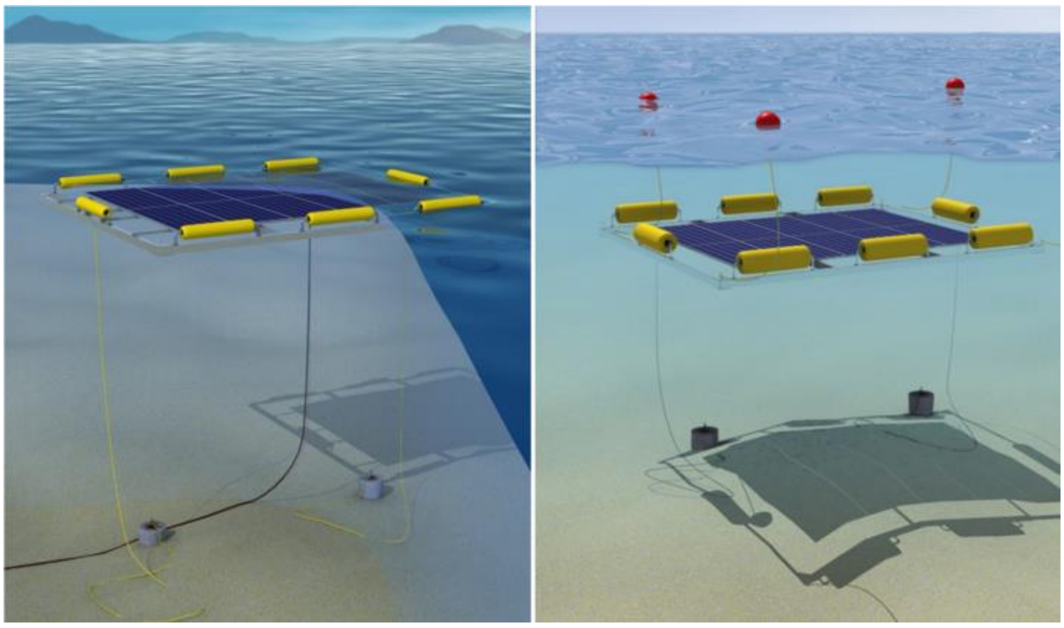


Figure 2.8: Design of floating PV from SCINTEC that is submergible in different water depth (Trapani and Redón Santafé, 2014)

CHAPTER 3

METHODOLOGY

3.1 Project Management

Table 3.1: Final Year Project 1's Gantt Chart

Activity	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Project title selection	▨													
Literature review		▨	▨											
Conceptual design of hardware system				▨	▨									
Selection of hardware components						▨	▨							
Building and testing the circuit and							▨	▨	▨					

the Arduino program							▨	▨	▨					
Exploration of IoT platforms							▨	▨	▨					
Code writing								▨	▨	▨	▨			
Testing and trouble-shooting the IoT data sending process									▨	▨	▨			
Report writing				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Table 3.2: Final Year Project 2's Gantt Chart

Activity	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Finalizing the hardware system	▨	▨												
Soldering the hardware components on PCB			▨	▨										

Calibration and validation of sensor data					▨	▨								
Installation of data monitoring system on FPV and GPV							▨							
Data collection of FPV and GPV								▨	▨	▨	▨	▨	▨	
Data analysis											▨	▨	▨	▨
Report writing				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

3.2 Project Flowchart

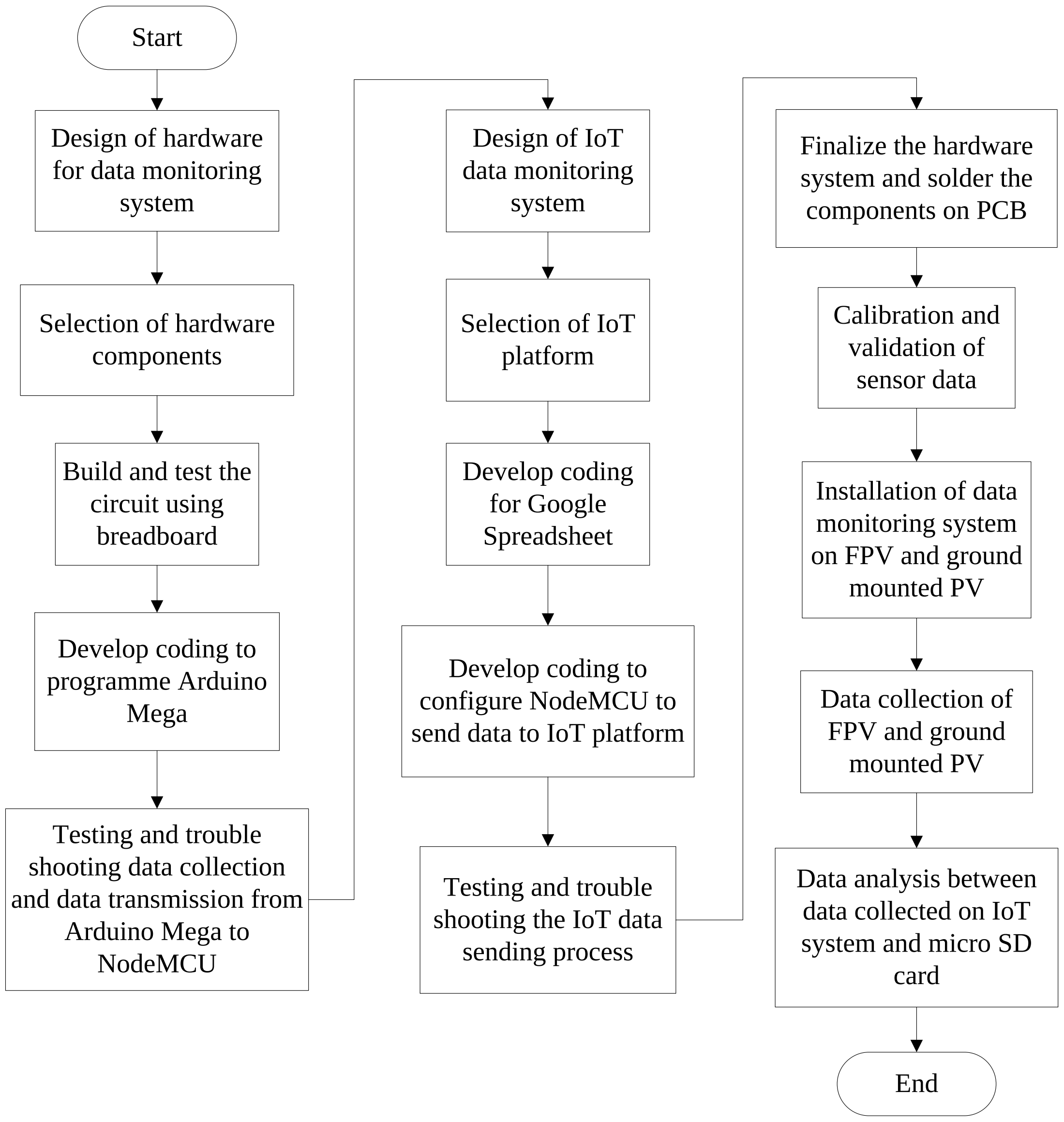

Start
Design of hardware for data monitoring system
Selection of hardware components
Build and test the circuit using breadboard
Develop coding to programme Arduino Mega
Testing and trouble shooting data collection and data transmission from Arduino Mega to NodeMCU
Design of IoT data monitoring system
Selection of IoT platform
Develop coding for Google Spreadsheet
Develop coding to configure NodeMCU to send data to IoT platform
Testing and trouble shooting the IoT data sending process
Finalize the hardware system and solder the components on PCB
Calibration and validation of sensor data
Installation of data monitoring system on FPV and ground mounted PV
Data collection of FPV and ground mounted PV
Data analysis between data collected on IoT system and micro SD card
End

Figure 3.1: Project flow chart

3.3 Hardware Configuration

Table 3.3: List of hardware components

Category	**Component**
Microcontrollers	Arduino Mega 2560
	ESP8266 NodeMCU
Sensors	Voltage sensor (A voltage divider that consists of 30 kΩ and 7.5 kΩ resistors)
	ACS712 current sensor
	DS18B20 temperature sensor
	DHT11 humidity sensor module
Display	OLED display
Time and date remembering system	RTC module
Offline data recording device	SD card module
Other components	SPDT Relay
	2N2222 NPN transistor
	Diode
	4.7 kΩ resistor

3.3.1 Arduino Mega 2560

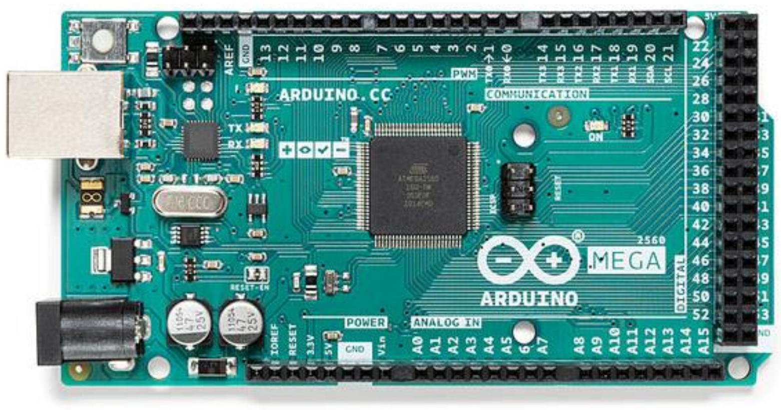


Figure 3.2: Arduino Mega 2560 (Arduino Mega 2560 Rev3, n.d.)

In this project, microcontroller is required to control other hardware components and process the collected sensor data. Arduino Mega 2560 was chosen due to several reasons. Firstly, Arduino Mega has more digital input and output (I/O) pins. This allows more sensors and components to be connected to the microcontroller. Secondly, comparing to its smaller counterparts, Arduino UNO, Arduino Mega has larger flash memory, which allows a larger sketch or code to be uploaded and stored into it (Gudino, 2021). Thirdly, the microcontroller can be easily programmed with the Arduino IDE software, which is an open-sourced platform created by Arduino company. Moreover, Arduino Mega has bigger Static Random-Access Memory (SRAM) space, allowing more variables to be created and manipulated from the code (Gudino, 2021). Table 3.4 below shows the specification of Arduino Mega 2560.

Table 3.4: General specification of Arduino Mega 2560 (Arduino Mega 2560 Rev3, n.d.)

Microcontroller	Arduino Mega 2560
Operating voltage	5 V
Input voltage (Recommended)	7-12 V
Input voltage (Limit)	6-20 V
Digital I/O pins	54 (including 15 PWM output pin)
Analog input pins	16
DC current per I/O pin	20 mA
DC current for 3.3 V pin	50 mA
Flash memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock speed	16 MHz

3.3.2 ESP8266 NodeMCU V3

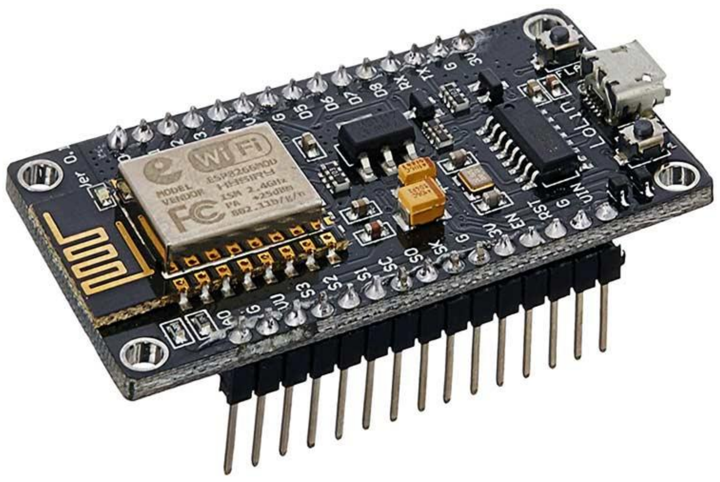


Figure 3.3: NodeMCU V3 (NodeMCU ESP8266, 2020)

This NodeMCU (Node Micro Controller Unit) is a microcontroller that has a built-in WI-FI microchip named ESP8266. Since Arduino Mega cannot send data to the Internet due to absence of WI-FI chip, NodeMCU comes in handy to receive the data from Arduino and send them to IoT platform. It is very similar to Arduino board as it is programmable via the micro-USB port with Arduino IDE software or Lua programming language. In order for PC to successfully detect the NodeMCU board, CH340G driver may need to be installed in the PC.

Table 3.5: General specifications of ESP8266 NodeMCU (NodeMCU ESP8266, 2020)

Microcontroller	NodeMCU V3
Processor	ESP8266 32 bit
Clock speed	80 MHz
USB to Serial	CH340G
Operating voltage	3.3 V
Input voltage	4.5 V – 10 V
Flash memory	4 MB
SRAM	64 KB
GPIO (General Purpose Input Output) pins	16
Analog input pins	1

3.3.3 Serial Communication Between NodeMCU and Arduino Mega 2560

The serial communication is required as sensor data from Arduino Mega need to be sent to NodeMCU, then to the IoT platform. The UART (Universal Asynchronous Receiver Transmitter) pins on NodeMCU labelled as RX (receive pin) and TX (transmit pin) on are used in serial communication. However, for both NodeMCU and Arduino Mega, user can define any digital pin as RX and TX pins for serial communication. This is done by defining the selected pins in the codes and upload the code to the microcontroller. Besides, the baud rate of the Arduino Mega and the NodeMCU is required to be the same for the data to be able to transmit. The baud rate of 115200 is used.

As shown in Figure 3.4, the digital pin 10 and 11 is used for serial communication, with pin 10 representing RX pin while pin 11 as TX pin. The connection of the pins is reciprocal, with RX pin corresponding to TX pin and TX pin corresponding to RX pin. This is because RX pin of NodeMCU will receive data from TX pin of Arduino board, while TX pin of NodeMCU will transmit data to RX pin of Arduino board.

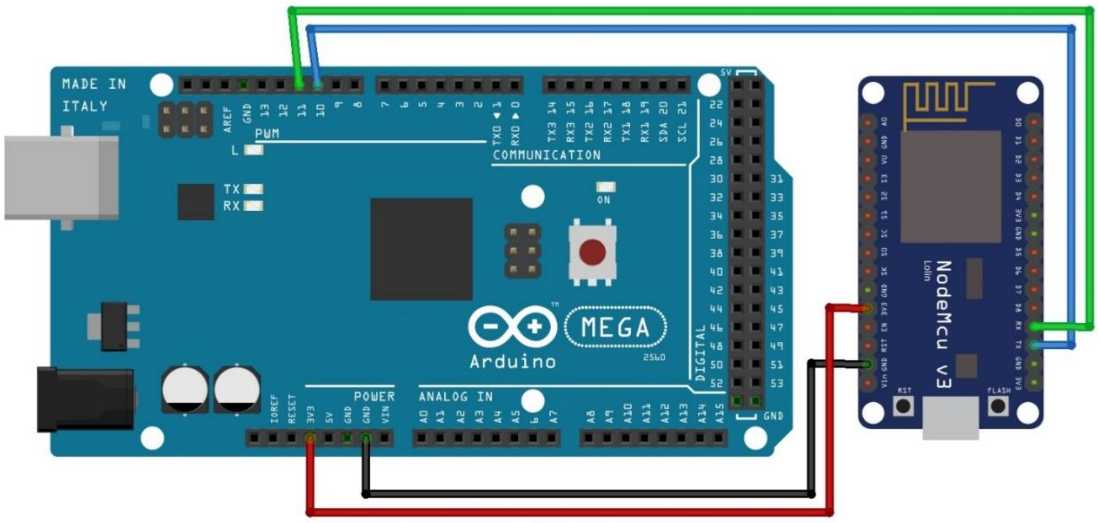


Figure 3.4: Connection between Arduino Mega and NodeMCU

3.3.4 Voltage Sensor

Usually, voltage can be directly measured by Arduino if the input voltage is lower than 5 V. To measure voltage higher than 5 V, voltage divider is required to lower the input voltage so that the board will not be damaged. According to the specification of the solar panel, the open circuit voltage is 21.6 V. Hence, a 5 to 1 ratio of voltage divider was constructed using 30 kΩ and 7.5 kΩ resistors. The construction of voltage divider is based on the formula below.

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

Where

V_{out} = Voltage across R_2 (Ω)

V_{in} = Voltage across R_1 and R_2 (Ω)

R_1 = Resistor 1 voltage (V)

R_2 = Resistor 2 voltage (V)

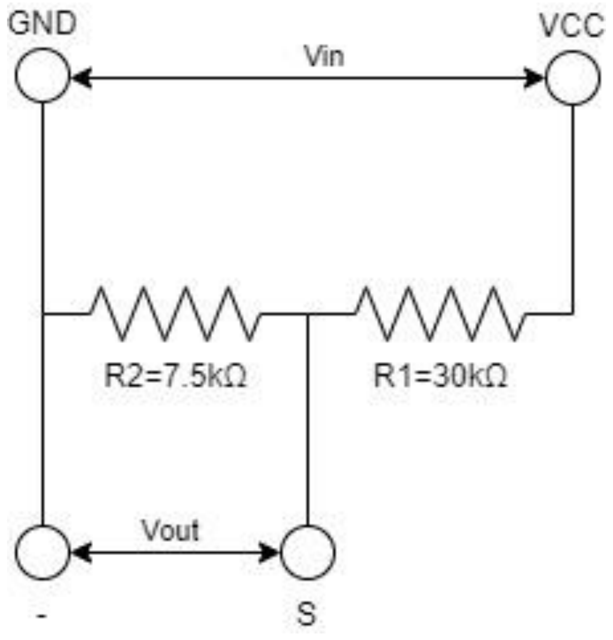


Figure 3.5: Schematic diagram of voltage divider

Referring to Figure 3.5, V_{in} terminals will be connected to solar panel, while V_{out} terminals will be connected to analogue pin and ground pin of Arduino Mega board. Since the voltage value received is in analogue form, the value needs to be processed with the formula below to convert it into digital form.

$$V_{out} = \frac{ADC\ value}{1024} \times V_{ref}$$

Where

V_{out} = Output voltage of voltage sensor (mV)

$ADC\ value$ = Analog to Digital Converter value obtained from analogue pin that the voltage divider is connected to

V_{ref} = Reference voltage (mV) (In this case it is 5000mV)

V_{out} is the value that represents the final reading of the voltage of the solar panel. ADC value is actually the V_{in} from the voltage divider formula. The number "1024" is the number of discrete analogue levels the analogue value can be detected. Since Arduino Mega's analogue pin has built-in ADC (Analog to Digital Converter) of 10 bits, the value 1024 is obtained from 2 to the power of 10. V_{ref} is referred from the voltage supplied to Arduino board, which is usually 5 V. However this value is not always constant. The accuracy of the data is influenced by the number of bits of ADC pin and the reference voltage (Measure DC Voltage and Current with an Arduino, 2021).

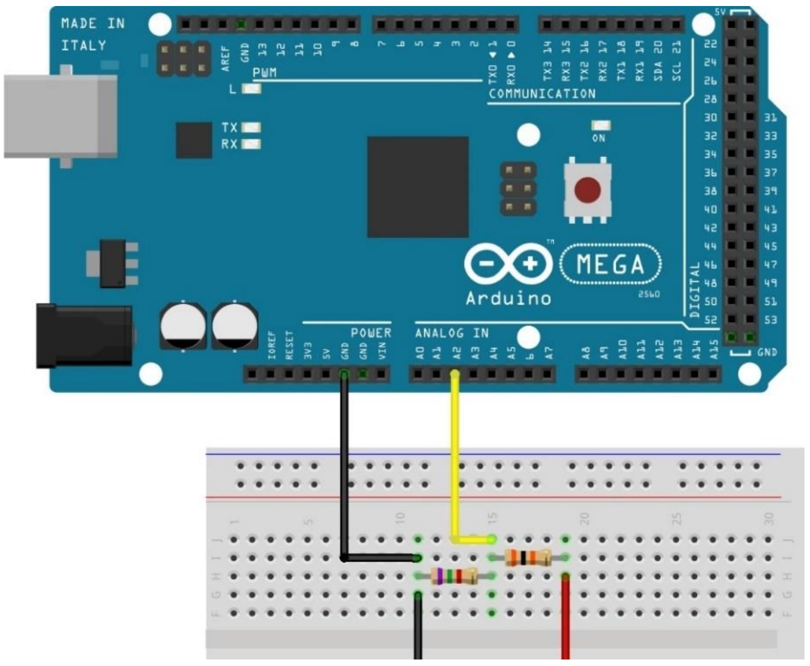


Figure 3.6: Connection between voltage divider and Arduino Mega

3.3.5 Current Sensor

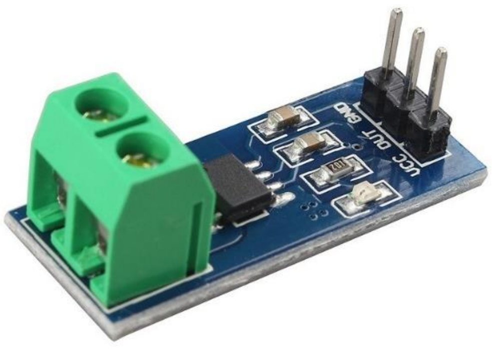


Figure 3.7: ACS712 current sensor (Current Sensor Module ACS712 (30A), n.d.)

The ACS712 is a type of non-invasive current sensor, which means the sensor has no direct connection with the load circuit. This type of sensors works on the principle of Hall-effect. The sensor measures the magnetic field generated by the built-in conductor and converts the value into corresponding analogue output. The ACS712 current sensor has 5 A, 20 A and 30 A variations with sensitivity of 185 mV/A, 100 mV/A and 66 mV/A respectively. Based on the specification of the solar panel, it will generate maximum current of 4.65 A. Hence, a 5 A current sensor is selected due to the in range current measurement and the highest sensitivity among all variations.

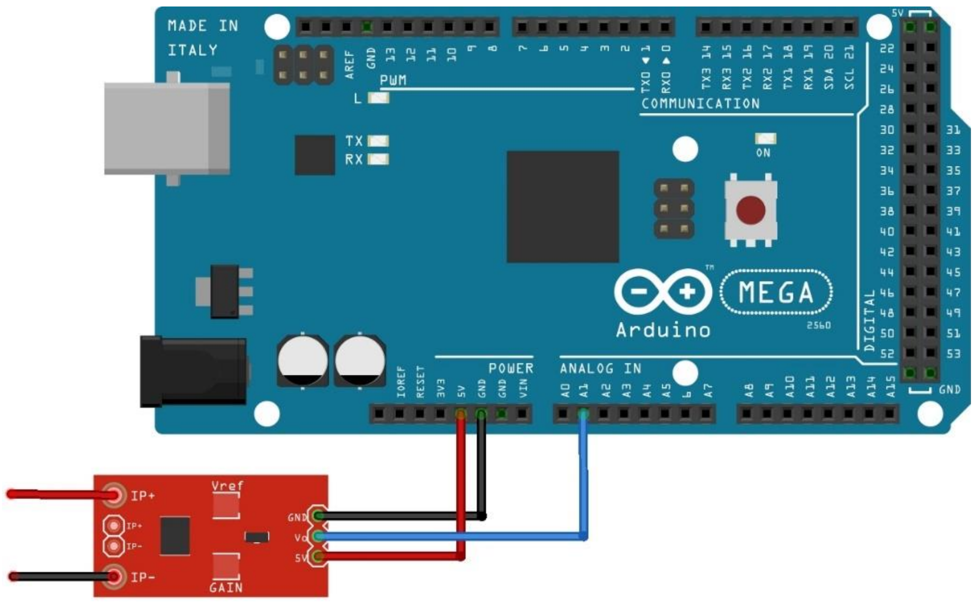


Figure 3.8: Connection between current sensor and Arduino Mega

The value received from the analogue pin on Arduino Mega is an output voltage corresponding to the current it is measuring. According to the datasheet of ACS712, the zero current voltage output is 2.5 V. This means that when it is not connected to any circuit, the sensor will still output 2.5 V. Hence, the calculation of current value is as follow.

$$V_{out} = \frac{ADC\ value}{1024} \times V_{ref}$$

$$I_{out} = \frac{V_{out} - V_{offset}}{Sensitivity}$$

Where

ADC value = Value obtained from analogue pin the sensor is connected to

V_{ref} = Reference voltage (mV) (In this case it is 5000mV)

V_{out} = Output voltage of current sensor (mV)

V_{offset} = Offset voltage (V) (in this case is 2500mV)

Sensitivity = 185mV/A

3.3.6 Relay

Figure 3.9: 5 V relay (5V Relay, n.d.)

Short circuit could happen when voltage and current are measured at the same time. Therefore, relay is required in this project to automatically switch between measuring voltage and measuring current. A Single Pole Double Throw (SPDT) relay is used. Relay operates electromagnetically. The armature in relay is originally in the position of connecting Normally Close (NC) pin to Common (COM) pin. When there is current flow through the coil of the relay, electromagnetic field is generated. The force generated by the electromagnetic field pulls the armature to connect to Normally Open (NO) pin. When no current flow through the coil, the armature reverts back to connecting the Normally Close pin. Figure 3.10 below shows the circuit of relay in different condition.

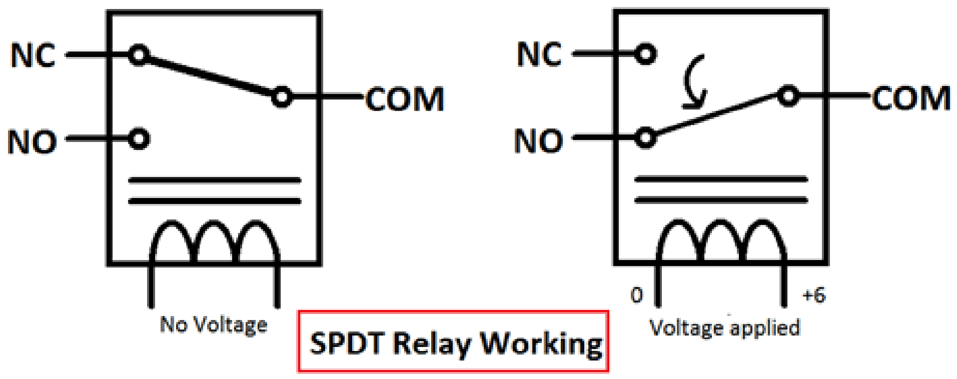


Figure 3.10: Relay circuit when coil is supplied with power or not (Arduino Relay Control Tutorial, 2017)

For this project's application, the voltage of solar panel will be measured first then followed by the current. Hence, the negative terminal of voltage sensor was connected to Normally Closed pin of the relay, while the negative terminal of current sensor was connected to the Normally Open pin. In the code, the relay was controlled to be turned OFF when measuring voltage, and turned ON when measuring current.

An NPN (Negative-Positive-Negative) transistor is required because the output signal pin from Arduino Mega cannot generate current that is high enough to energize the relay coil. The 2N2222 transistor is used in this project. There are three terminals on the transistor, namely Collector (C), Base (B) and Emitter (E). The Collector pin was connected to one of the coil pin of relay. The Base pin was connected to output pin of Arduino Mega. The Emitter pin was connected to ground. A diode is added between the coil pins to avoid large reverse current damaging other components on the circuit when the coil de-energizes. The construction of the switch circuit is as shown in Figure 3.11.

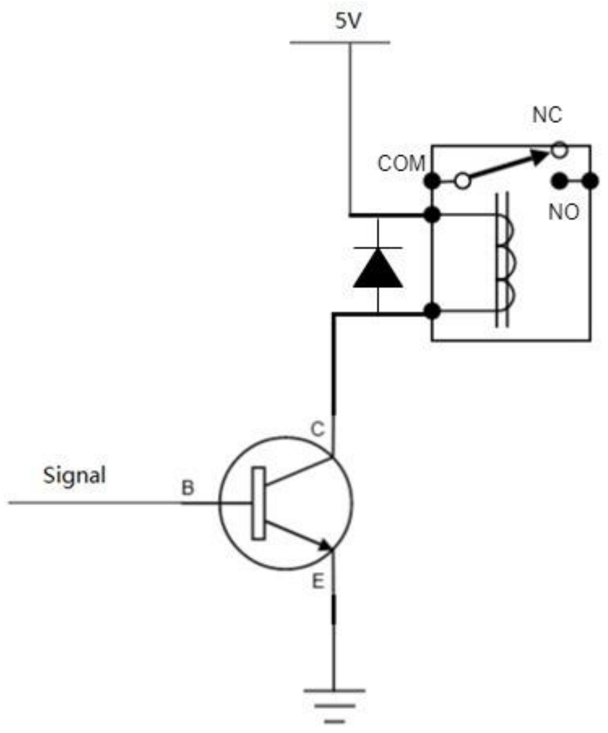


Figure 3.11: Switch circuit of relay with transistor

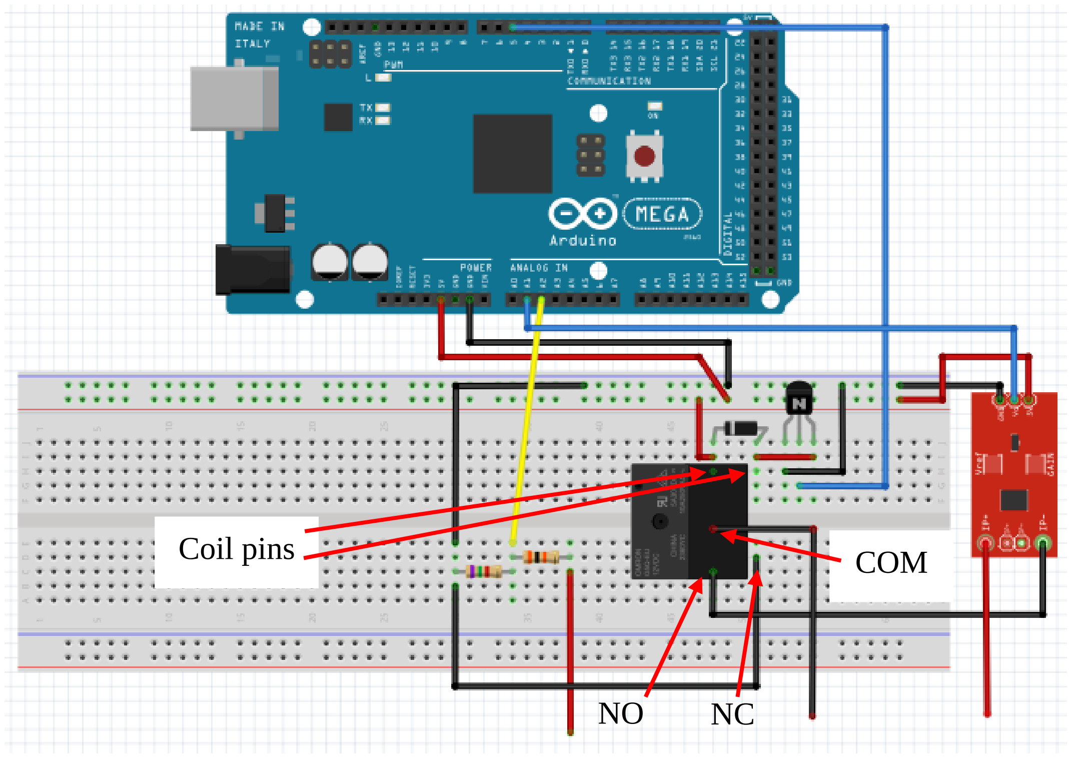


Figure 3.12: Schematic diagram of relay, voltage divider and current sensor connections

3.3.7 Temperature Sensor

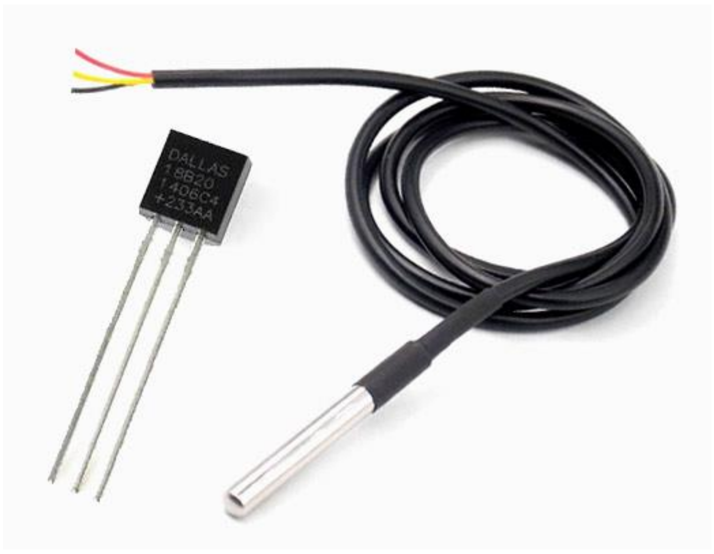


Figure 3.13: DS18B20 temperature sensor (DS18B20 Temperature Sensor, 2018)

The DS18B20 sensor comes with two types-probe type and another one with appearance similar to transistor. The sensor is capable to measure temperature in the range of -55 °C to +125 °C. The accuracy of the sensor is ±0.5 °C when the temperature ranges from -10 °C to +85 °C. Multiple DS18B20 sensors can be controlled by and communicate to one microcontroller with only one data line. This is known as communication over 1-wire bus. The DS18B20 temperature sensor with appearance similar to transistor are used to measure the temperature of the solar panel, to observe the effect of temperature change on the performance of solar panel. To monitor the temperature on different positions of solar panel, three sensors with their pins extended with cables are attached at the back of the solar panel.

Referring to Figure 3.14 below, a 4.7 kΩ resistor is included in the circuit to act as a pull-up resistor. This way it can enhance stability of data transmission between the sensors and Arduino Mega.

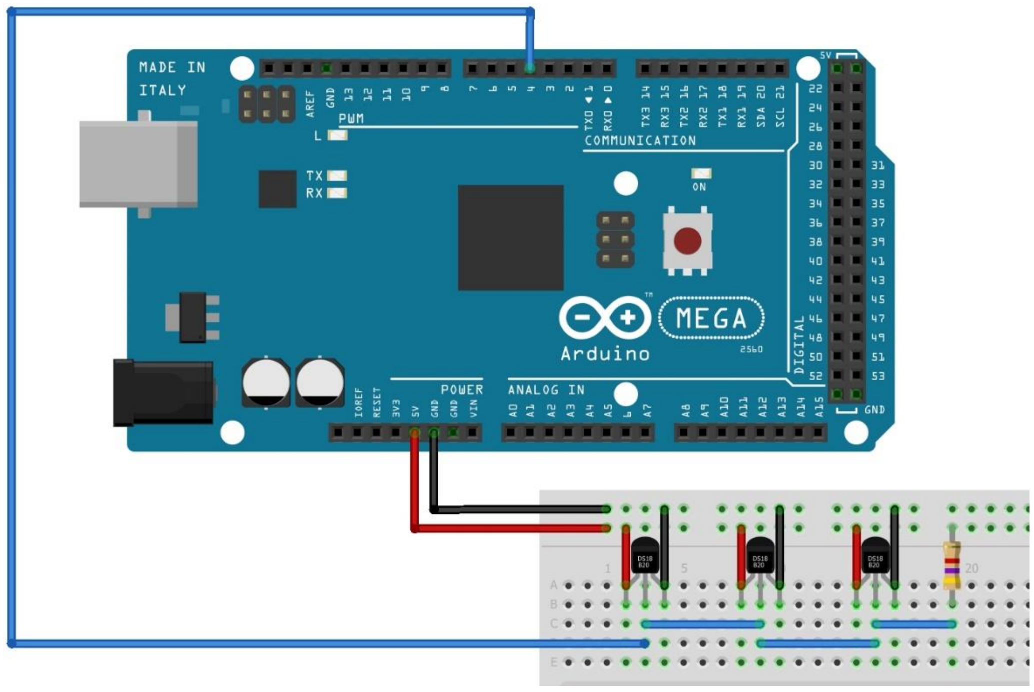


Figure 3.14: Connection between DS18B20 temperature sensor with Arduino Mega

3.3.8 Humidity Sensor

Figure 3.15: DHT11 sensor module (DHT11 sensor module, n.d.)

DHT11 sensor module is used in this project to obtain humidity readings. It measures humidity using capacitive humidity sensor, and also temperature with its thermistor. For this project's purpose, only humidity data of the surrounding air of the PV panel are read. The sensor is able to measure humidity in the range of 20-90 % relative humidity, with accuracy of ± 5 %. The DHT11 sensor can be powered from 3-5.5 V, with 2.5 mA maximum current. The sensor data is slow as it will take 2 seconds to provide a single reading.

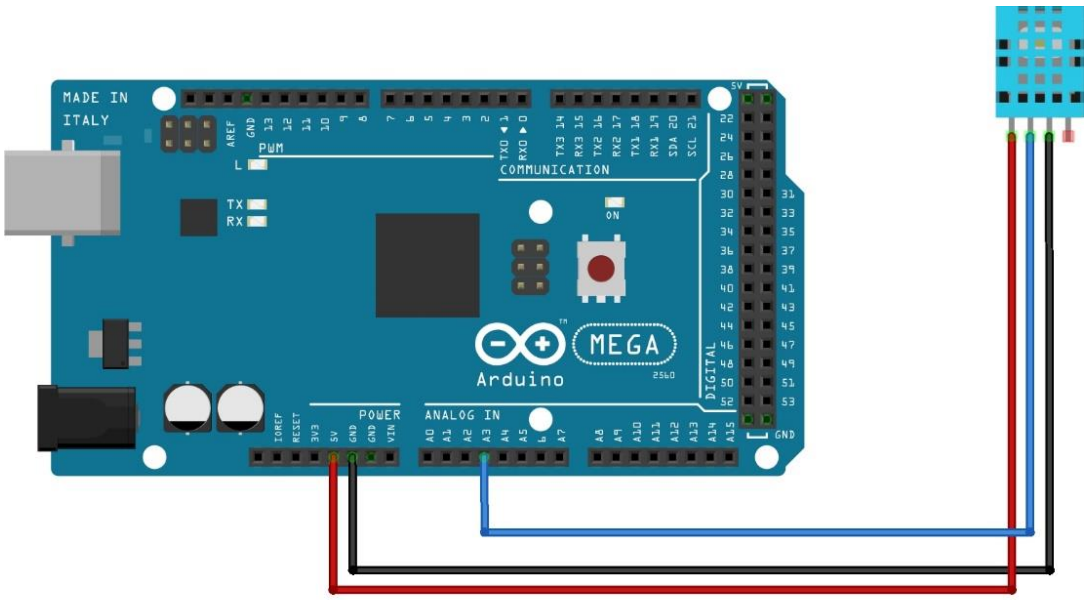


Figure 3.16: Connection between DHT11 sensor and Arduino Mega

3.3.9 OLED Display

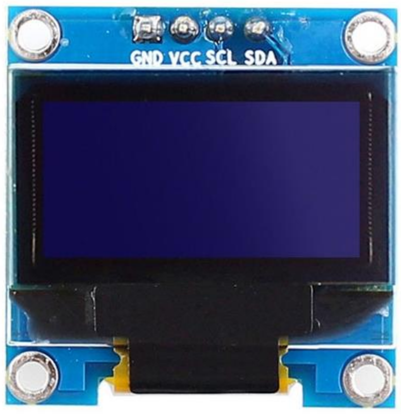


Figure 3.17: OLED display module (I2C 0.96" OLED 128x64, n.d.)

OLED stands for Organic Light-Emitting Diode. A display is required to show the values detected by the sensors. To display all six sensor values and also the date and time, an OLED display that has 128 x 64 blue pixels is used. The OLED display module communicates with Arduino Mega via I2C (Inter-Integrated Circuit).

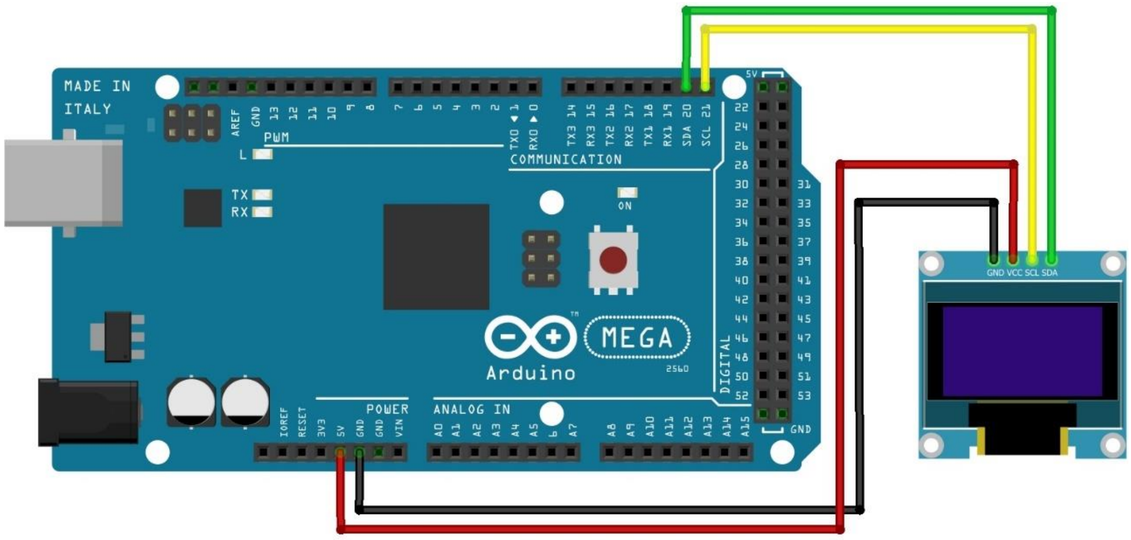


Figure 3.18: Connection between OLED display module and Arduino Mega

3.3.10 RTC Module

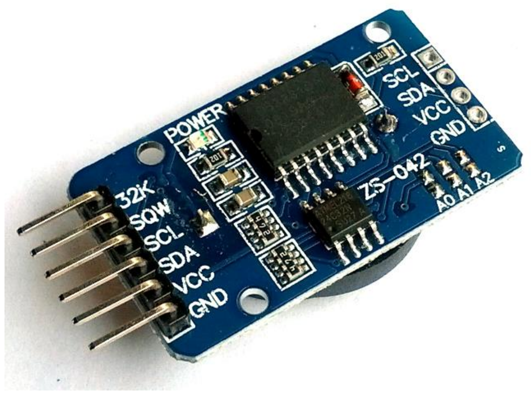


Figure 3.19: RTC module (DS3231 Real Time Clock (RTC) Module, I2C, n.d.)

RTC stands for Real Time Clock. RTC module is a date and time remembering device that is powered by a battery to keep the date and time updated. The RTC module can also be powered by connecting its power pin to external power supply. It is required in this project because Arduino Mega could not obtain real time by itself. Furthermore, recording data offline with timestamps is more helpful for data analysis.

Table 3.6: Pin wiring of RTC module with Arduino Mega

Pin on RTC module	Arduino Mega
VCC	5 V
GND	GND
SCL	Pin 21
SDA	Pin 20

3.3.11 SD Card Module

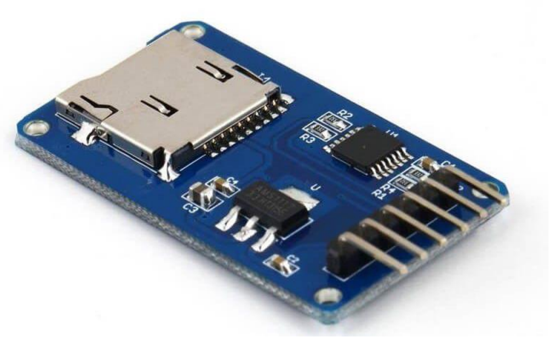


Figure 3.20: SD card module (5V Compatible Micro SD Card Adapter, n.d.)

An SD card module is necessary to save data offline and serve as a backup in the event of an internet service disruption. It allows user to plug in SD card on the module, and read or write data on the card via communication with microcontroller. The SD card that is going to be used in this module needs to formatted to FAT16 or FAT32 format before using it with Arduino board, or else Arduino Mega will not be able to detect it.

Table 3.7: Pin wiring of SD card module with Arduino Mega

Pin on SD card module	Arduino Mega
VCC	5 V
GND	GND
MISO	Pin 50
MOSI	Pin 51
SCK	Pin 52
CS	Pin 53

3.3.12 Complete Hardware System

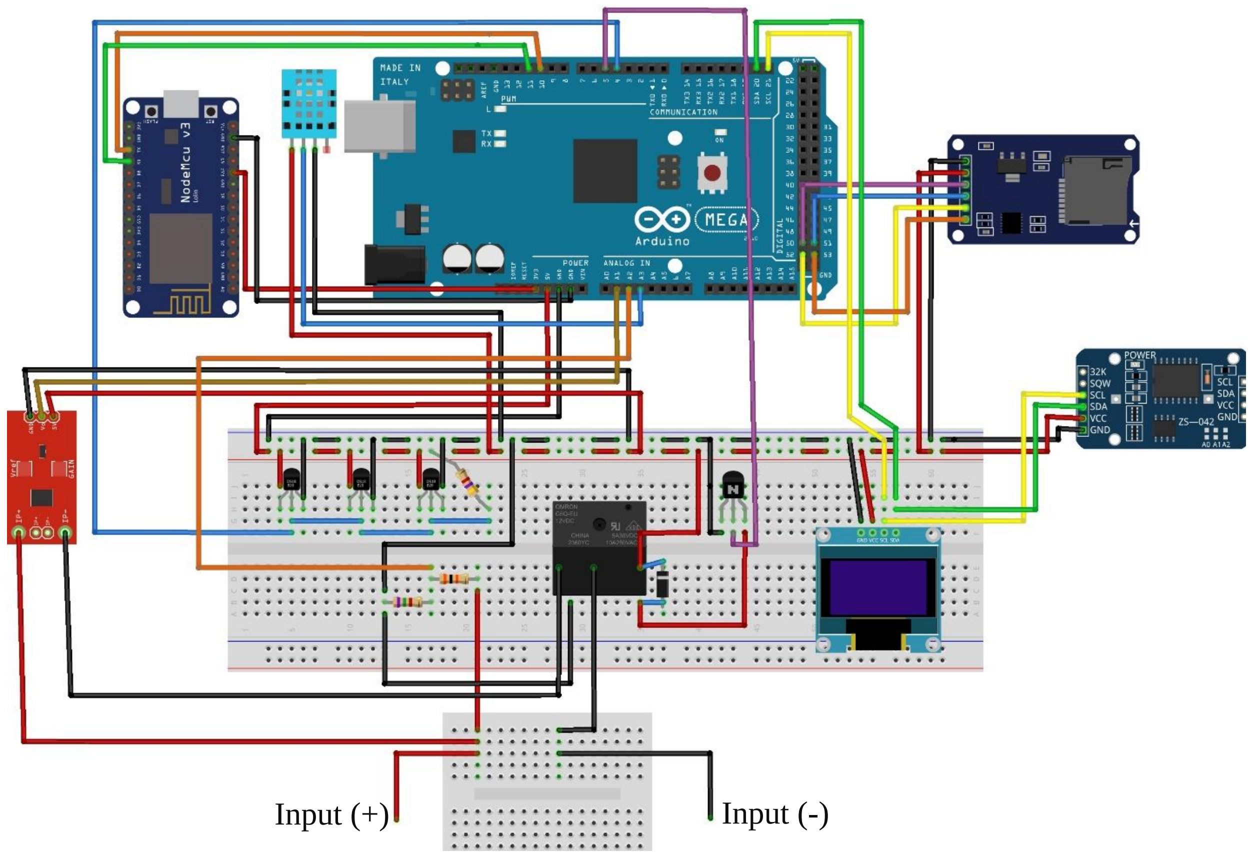


Figure 3.21: Schematic diagram of complete hardware system

The Arduino Mega microcontroller is powered by power bank. The wire labelled as “Input (+)” and “Input (-)” are connected to the positive and negative pole of solar panel respectively. The sensors collect data readings and the reading values are shown on the OLED display. At the same time, the sensor data are saved offline into the SD card. While for online data sending and saving, NodeMCU acts as a communicator to receive sensors data from Arduino Mega microcontroller and send them to IoT platform. The complete schematic diagrams of the hardware system of FPV and GPV are attached in Appendix D and E.

3.3.13 Soldering Electrical Components on PCB Board

Bad electrical contact is a common issue when using breadboard since the small sockets on breadboard can get loose depending on the built quality. Furthermore, the wires and components plugged onto the small sockets can be easily shaken out when there is external force applied on them. For instance, in the case of vibration of PV panel floater due to movement of water waves. To prevent the issue of loose contact, the wires and the electronic components were soldered on Printed Circuit Board (PCB). The layout of PCB was designed using EAGLE PCB drawing software. The finalized PCB layout design for FPV and GPV hardware circuit is attached in Appendix F.

After done soldering the components, the PCB, Arduino board and NodeMCU were placed inside a plastic container with lid to prevent water from damaging the hardware. Some holes were made on the side of the container to allow sensor wires to be taken out. The holes were then sealed with Tack-it elastic clay. Figures 3.22 and 3.23 below show the complete hardware systems for FPV and GPV. Moreover, power banks were used to supply power for both system. The power banks were put in zip bags to make them water-proof.

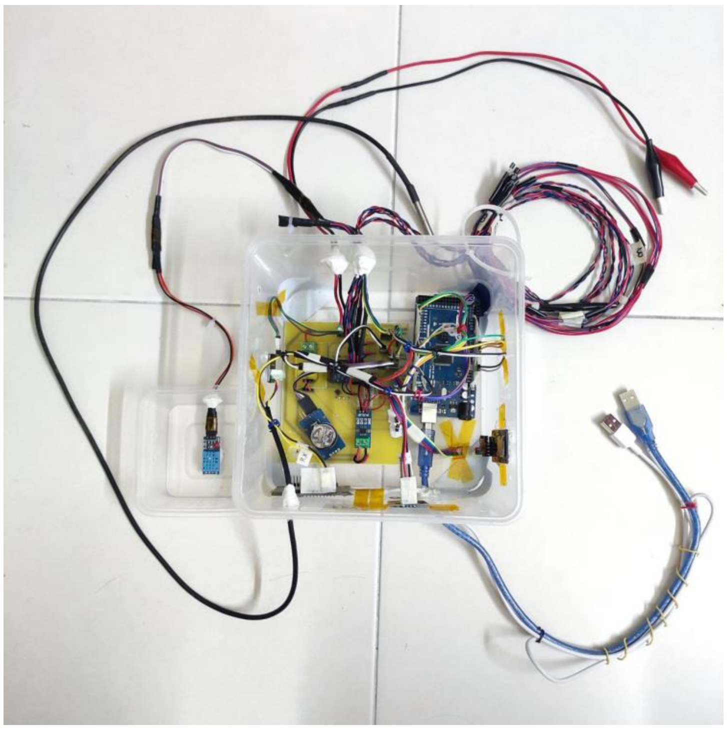


Figure 3.22: Complete hardware system for FPV

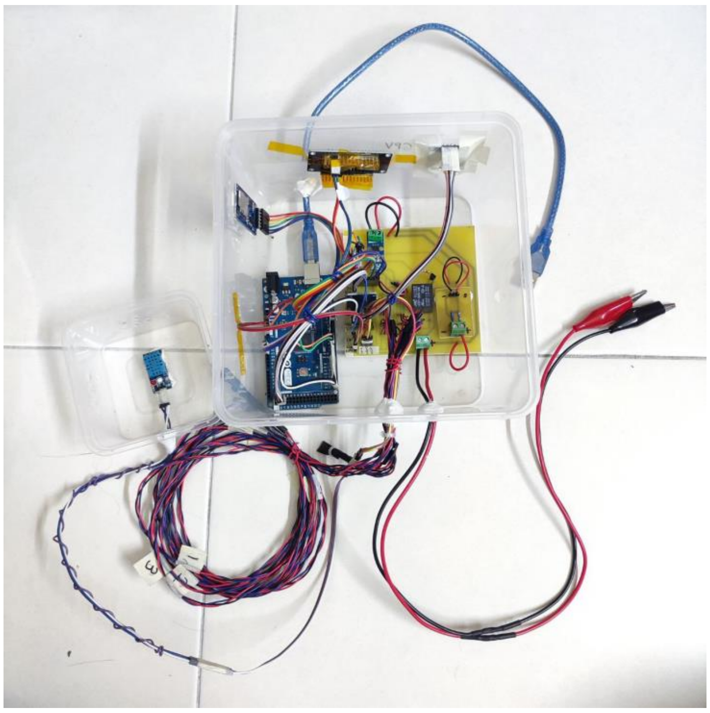


Figure 3.23: Complete hardware system for GPV

3.4 Coding for Arduino Mega Microcontroller

The Arduino IDE sketch or code for Arduino Mega microcontroller can be separated into several sections, which is as shown in the Figure 3.24 below. The flow chart in Figure 3.25 shows actions executed by Arduino Mega in sequence. The detail explanations for the code are written as comments included in the sketch.

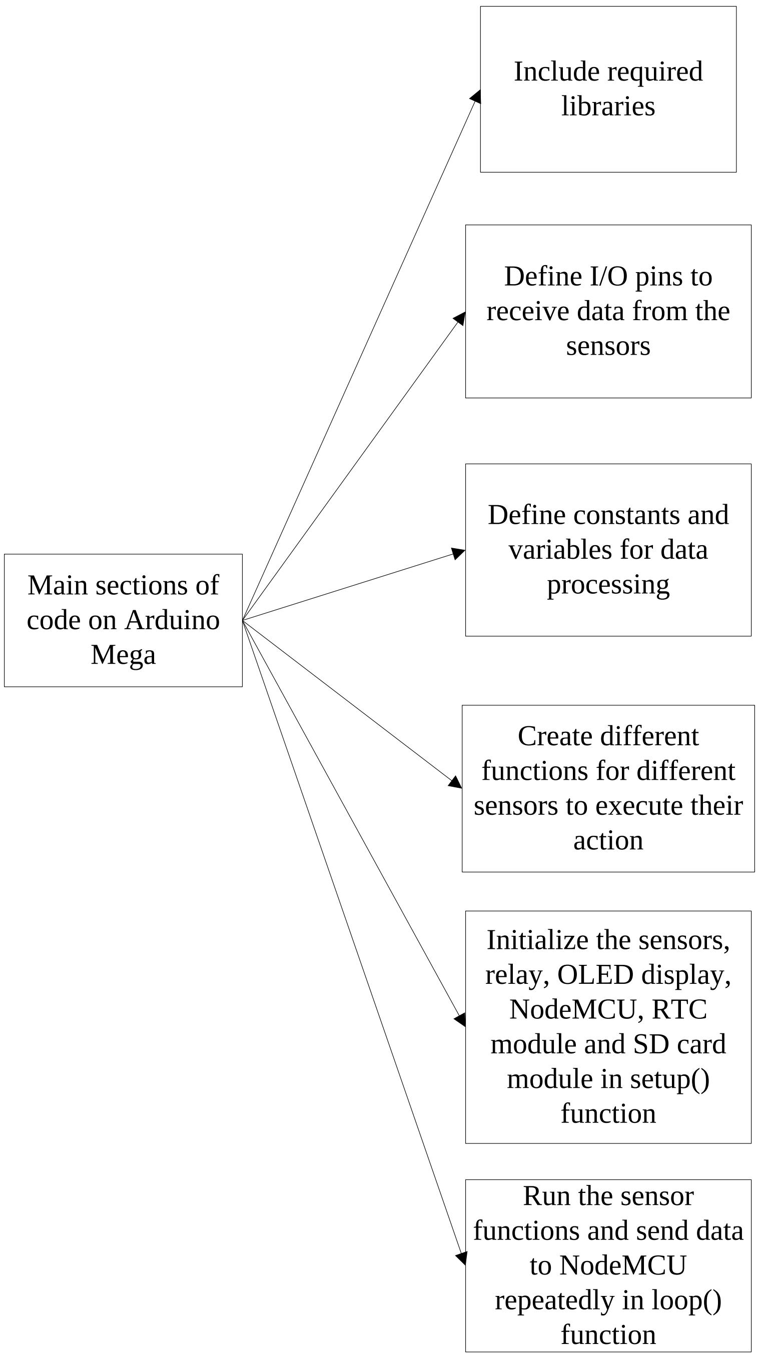

Main sections of code on Arduino Mega
Include required libraries
Define I/O pins to receive data from the sensors
Define constants and variables for data processing
Create different functions for different sensors to execute their action
Initialize the sensors, relay, OLED display, NodeMCU, RTC module and SD card module in setup() function
Run the sensor functions and send data to NodeMCU repeatedly in loop() function

Figure 3.24: Main code for Arduino Mega

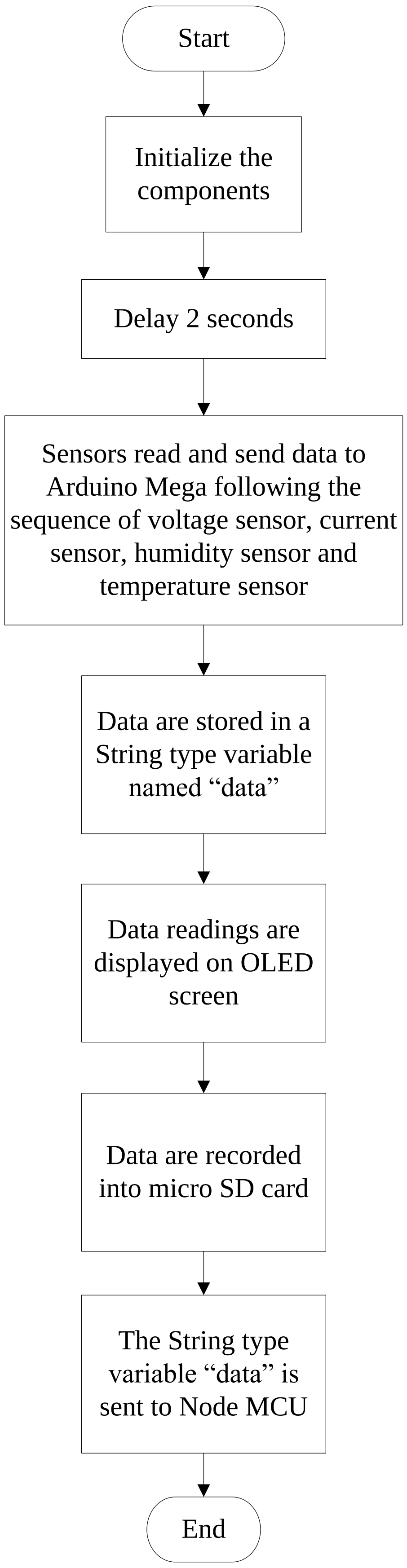
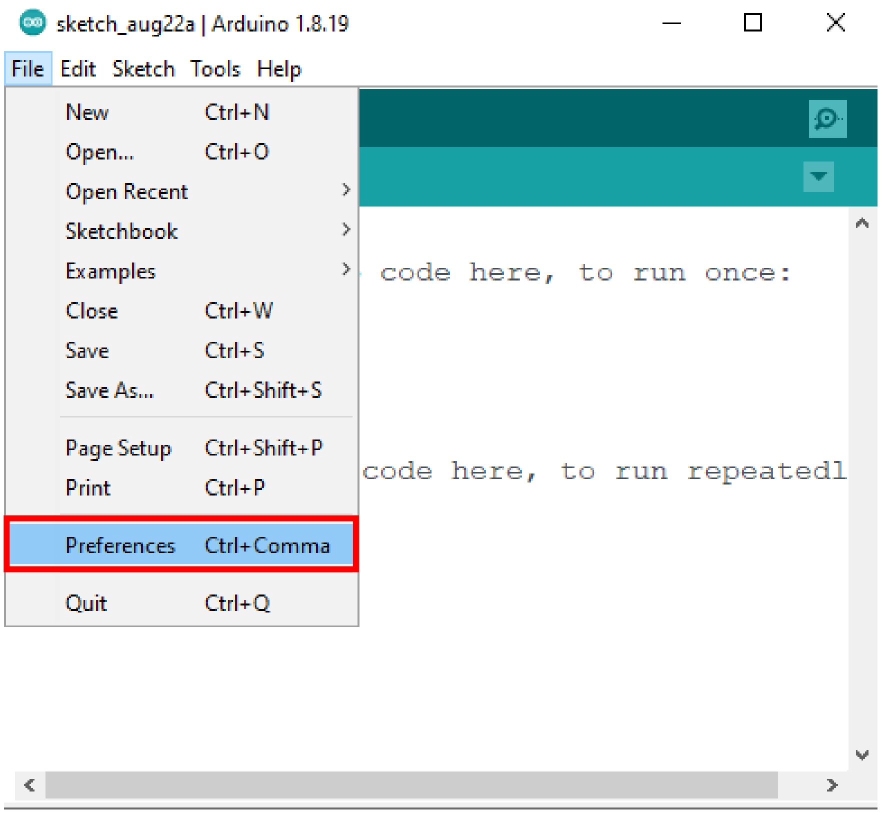

Start
Initialize the components
Delay 2 seconds
Sensors read and send data to Arduino Mega following the sequence of voltage sensor, current sensor, humidity sensor and temperature sensor
Data are stored in a String type variable named “data”
Data readings are displayed on OLED screen
Data are recorded into micro SD card
The String type variable “data” is sent to Node MCU
End

Figure 3.25: Arduino Mega working flowchart

3.5 Installation of Required Libraries on Arduino IDE

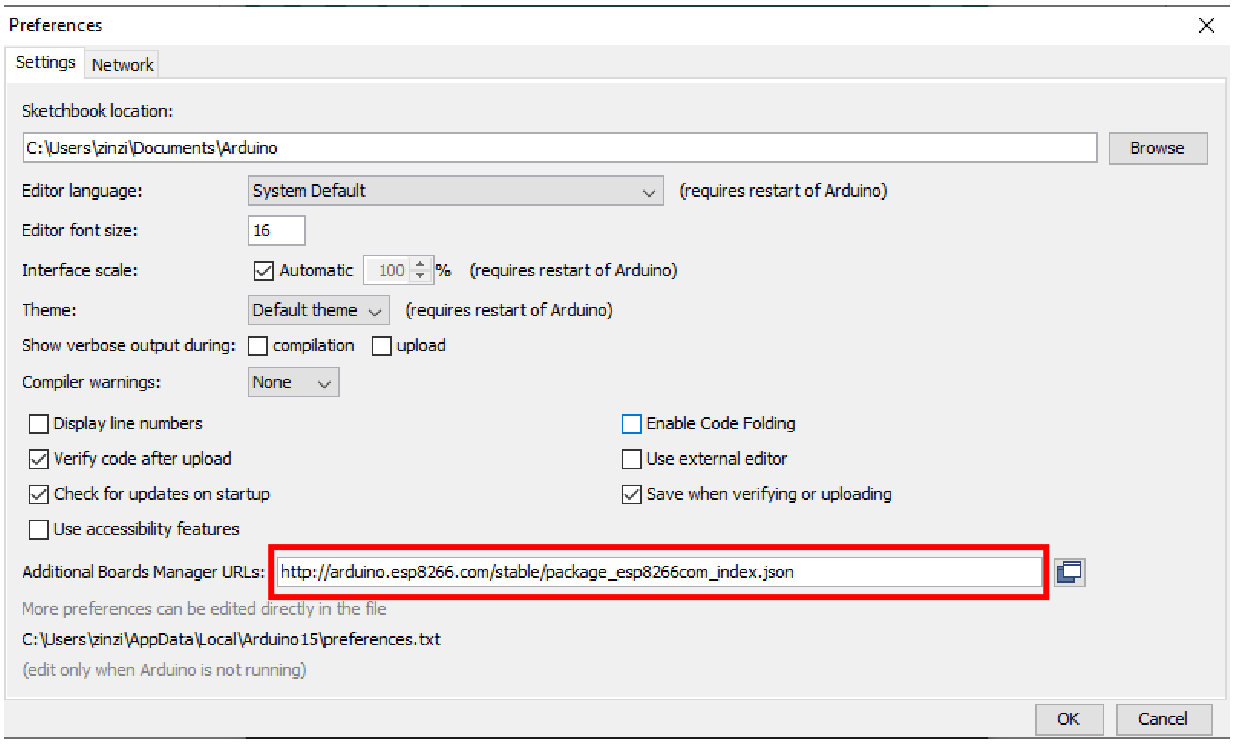
In order for successful detection of ESP8266 NodeMCU board by the Arduino IDE software, the ESP8266 board manager is required to be added. Firstly, user need to open Arduino IDE software, then under the "File" tab, select "Preferences". User then needs to enter the following URL at the blank space of Additional Board URLs:

http://arduino.esp8266.com/stable/package_esp8266com_index.json



```
 code here, to run once:

code here, to run repeatedl
```

Figure 3.26: Location to open "Preferences" window


Preferences

Settings | Network

Sketchbook location:
C:\Users\zinzi\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 16

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [ ] compilation [ ] upload

Compiler warnings: None

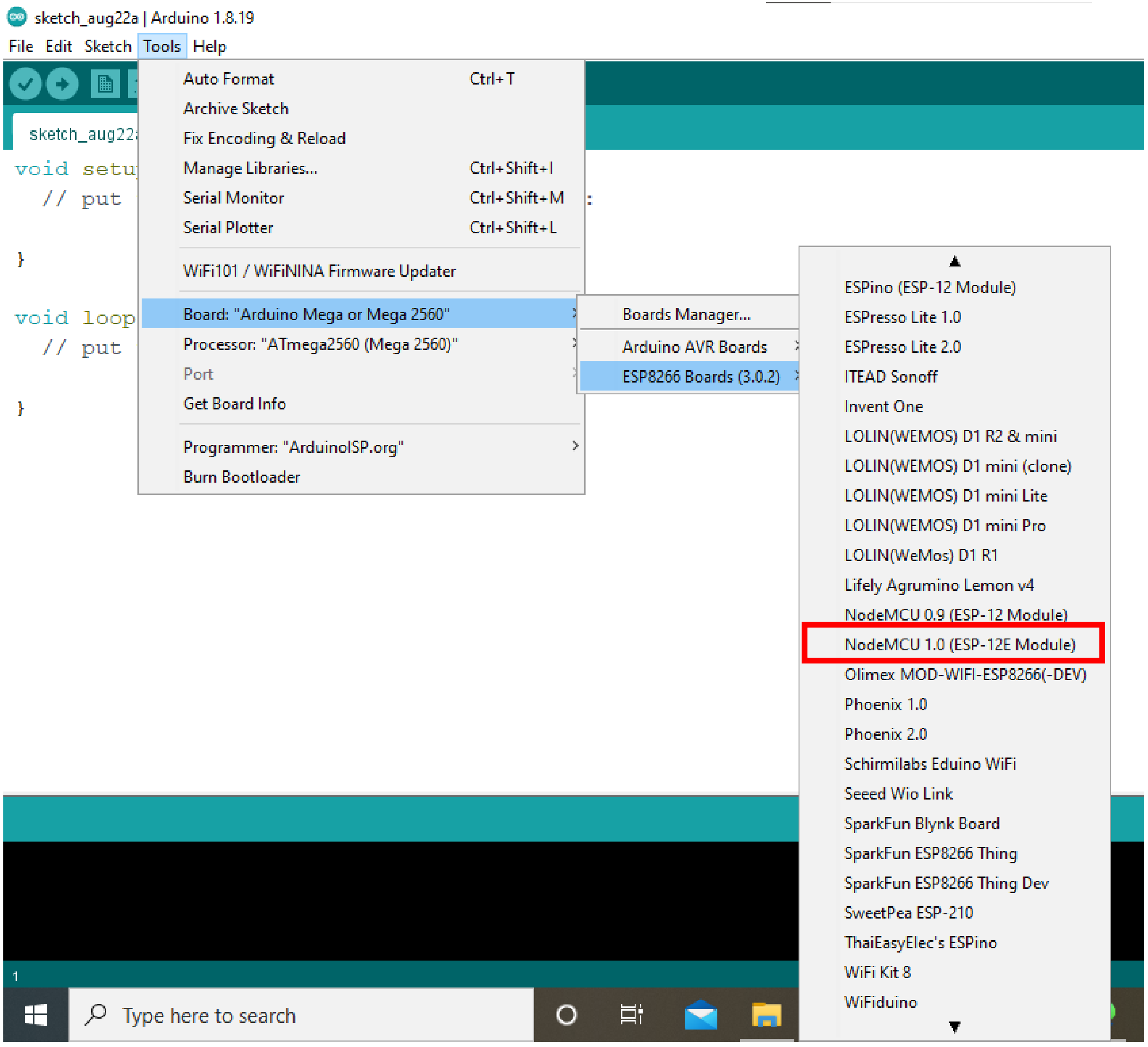
[ ] Display line numbers
[✓] Verify code after upload
[✓] Check for updates on startup
[ ] Use accessibility features
[ ] Enable Code Folding
[ ] Use external editor
[✓] Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

More preferences can be edited directly in the file
C:\Users\zinzi\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Figure 3.27: Adding ESP8266 NodeMCU's board URL

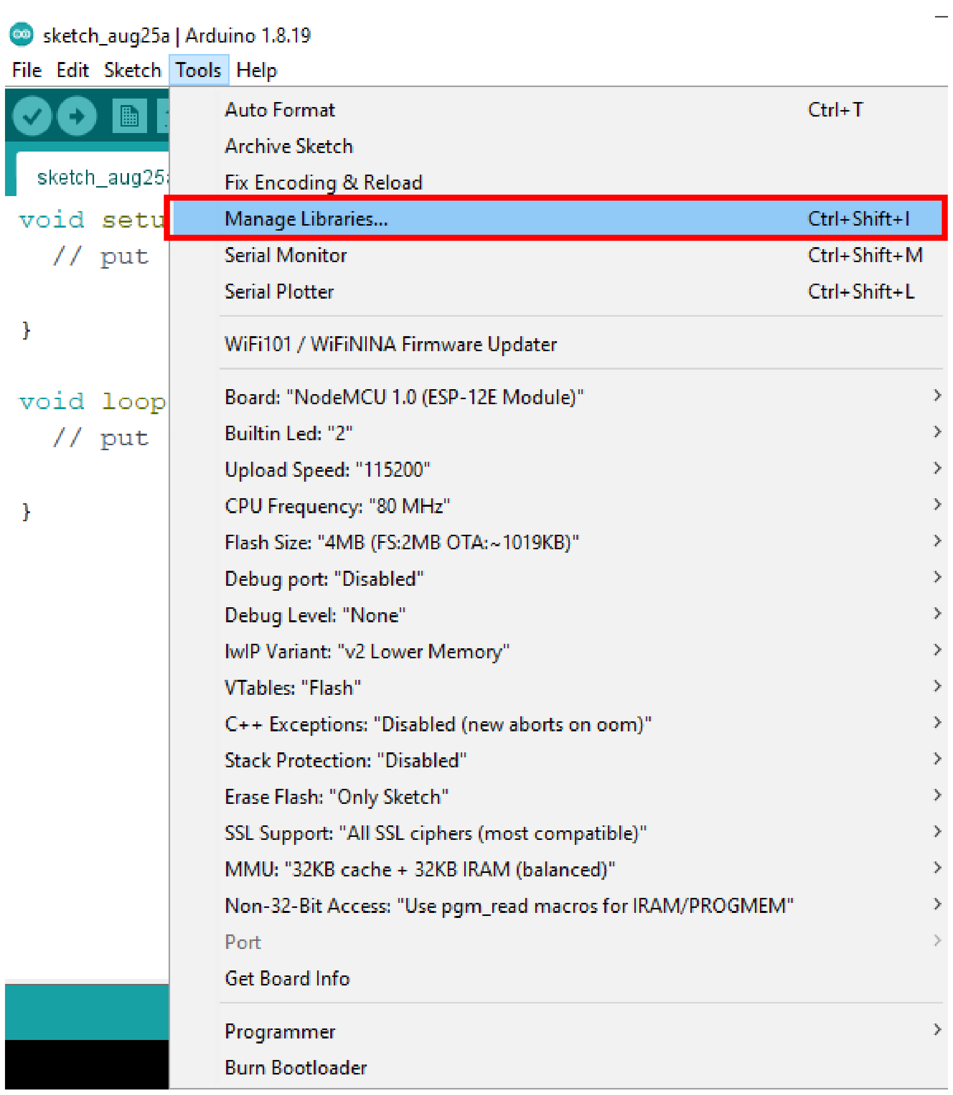
User will be able to select ESP8266 NodeMCU board under “Tools” tab after adding the board URL. “NodeMCU 1.0 (ESP-12E module)” is selected. The detail is as shown in Figure 3.28. With the NodeMCU board added to board manager, user can now write codes that are understandable by the software. When connecting NodeMCU with PC, the Arduino IDE software will be able to detect it.


sketch_aug22a | Arduino 1.8.19
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port
Get Board Info
Programmer: "ArduinoISP.org"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
ESP8266 Boards (3.0.2)
ESPino (ESP-12 Module)
ESPresso Lite 1.0
ESPresso Lite 2.0
ITEAD Sonoff
Invent One
LOLIN(WEMOS) D1 R2 & mini
LOLIN(WEMOS) D1 mini (clone)
LOLIN(WEMOS) D1 mini Lite
LOLIN(WEMOS) D1 mini Pro
LOLIN(WeMos) D1 R1
Lifely Agrumino Lemon v4
NodeMCU 0.9 (ESP-12 Module)
NodeMCU 1.0 (ESP-12E Module)
Olimex MOD-WIFI-ESP8266(-DEV)
Phoenix 1.0
Phoenix 2.0
Schirmilabs Eduino WiFi
Seeed Wio Link
SparkFun Blynk Board
SparkFun ESP8266 Thing
SparkFun ESP8266 Thing Dev
SweetPea ESP-210
ThaiEasyElec's ESPino
WiFi Kit 8
WiFiduino
Type here to search

```
void setu
  // put

}

void loop
  // put

}
```

Figure 3.28: Boards selection under “Tools” tab

To include the libraries for the other hardware components, the similar steps were done for DHT11 humidity sensor, DS18B20 temperature sensors, and RTC module. User needs to go to "Tools" tab and click on "Manage Libraries…" as shown in Figure 3.29 below. After a few seconds, a window named as "Library Manager" should pop out.



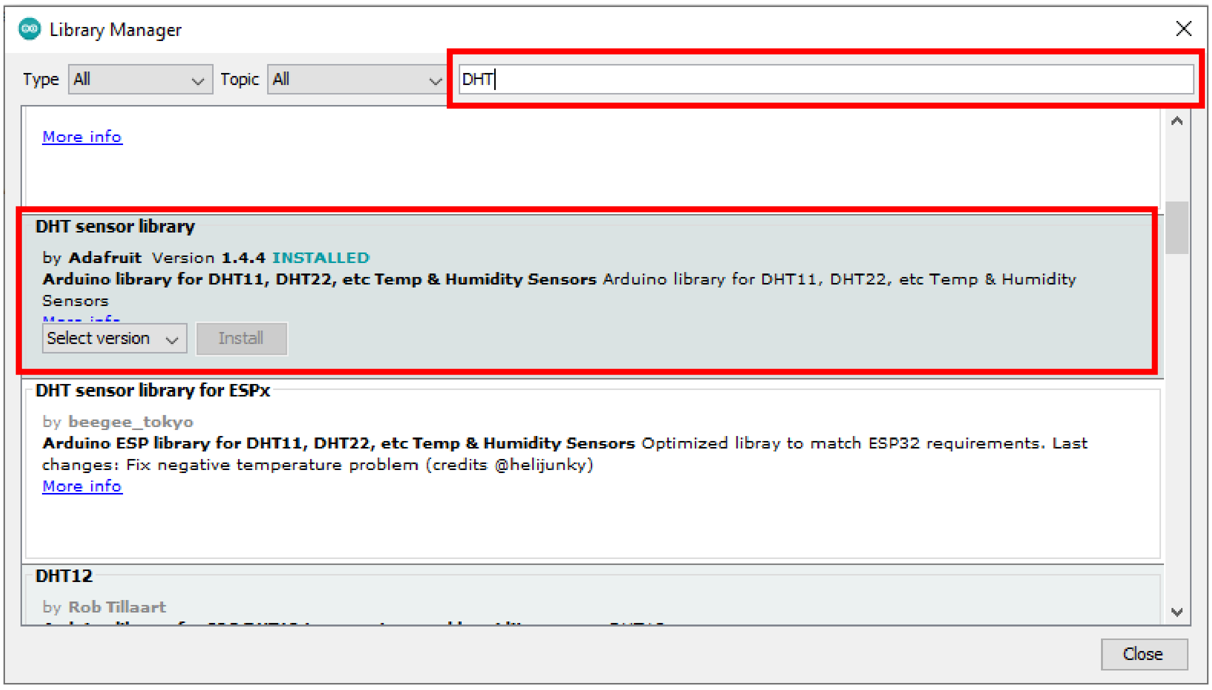
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "NodeMCU 1.0 (ESP-12E Module)"
Builtin Led: "2"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
C++ Exceptions: "Disabled (new aborts on oom)"
Stack Protection: "Disabled"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"
MMU: "32KB cache + 32KB IRAM (balanced)"
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"
Port
Get Board Info
Programmer
Burn Bootloader

Figure 3.29: Location of "Manage Libraries…"

User then need to type the name of the required library on the search bar. The related library should show up. An example is shown in Figure 3.30 to install the DHT sensor library. An install button should appear if user have not installed the library before. Table 3.8 below shows the required libraires to be installed on Arduino IDE.

Table 3.8: Required libraries to be installed on Arduino IDE

Hardware components	Name of the required library
DHT11 humidity sensor	DHT sensor library
DS18B20 temperature sensor	DallasTemperature
	OneWire
OLED display	Adafruit GFX Library
	Adafruit SSD1306
RTC module	RTClib



DHT sensor library
by **Adafruit** Version **1.4.4** INSTALLED
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors

DHT sensor library for ESPx
by **beegee_tokyo**
Arduino ESP library for DHT11, DHT22, etc Temp & Humidity Sensors Optimized libray to match ESP32 requirements. Last changes: Fix negative temperature problem (credits @helijunky)

DHT12
by **Rob Tillaart**

Figure 3.30: Installation of DHT sensor library

3.6 Calibration Process

3.6.1 Voltage and Current Calibration Process

The calibration process is important to obtain accurate voltage and current reading for comparison and analysis between ground type solar PV panels and floating type solar PV panels. The IV plotter device as shown in Figure 3.31 below was used as the reference for open circuit voltage, V_{OC} and short circuit current, I_{SC}. The V_{OC} and I_{SC} readings obtained from Arduino were compared with the readings obtained from the IV plotter device. The calibration process was performed while the sky was clear and sunny. Good weather condition is important in obtaining more consistent voltage and current readings. Since IV plotter device and Arduino system cannot be measuring the voltage and current of PV panel at the same time, the readings from IV plotter device were obtained first, then followed by the Arduino system immediately to avoid change of actual reading values due to change of weather condition.

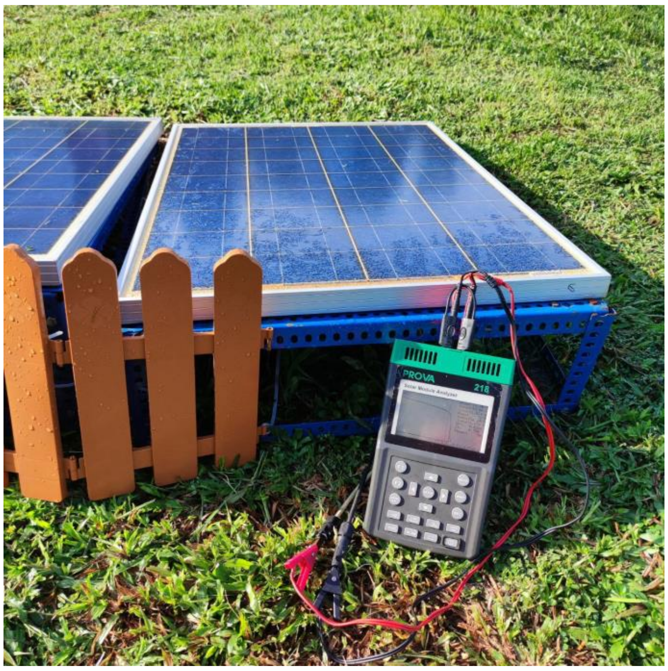


Figure 3.31: IV plotter device

In order to obtain the offset error of the Arduino system, PV panel voltage and current values ranging from the lowest to the highest were obtained. This was done by adjusting the orientation of PV panel, which in another word, by altering the intensity of light hitting on PV panel. For instance, to obtain the highest possible I_{SC} value, the PV panel was positioned directly facing towards the sun. In the other hand, to obtain the lowest possible I_{SC} value, the PV panel was positioned facing down towards ground.

The I_{SC} and V_{OC} values obtained from IV plotter device and Arduino were compared. Firstly, the readings were plotted on scattered graph. Then, linear line and linear line equation were obtained from the graph. The linear line equations are the corrective coefficient for calibrating Arduino reading. Meanwhile, the percentage errors before calibration were calculated. Next, the corrective coefficients were inserted into the Arduino Mega sketch/coding. To verify the calibration, the V_{OC} and I_{SC} readings from IV plotter device and Arduino were obtained again with different PV panel orientation. The values between the two devices were compared and the percentage errors after calibration were calculated. The overall calibration process is as shown in Figure 3.32 below.

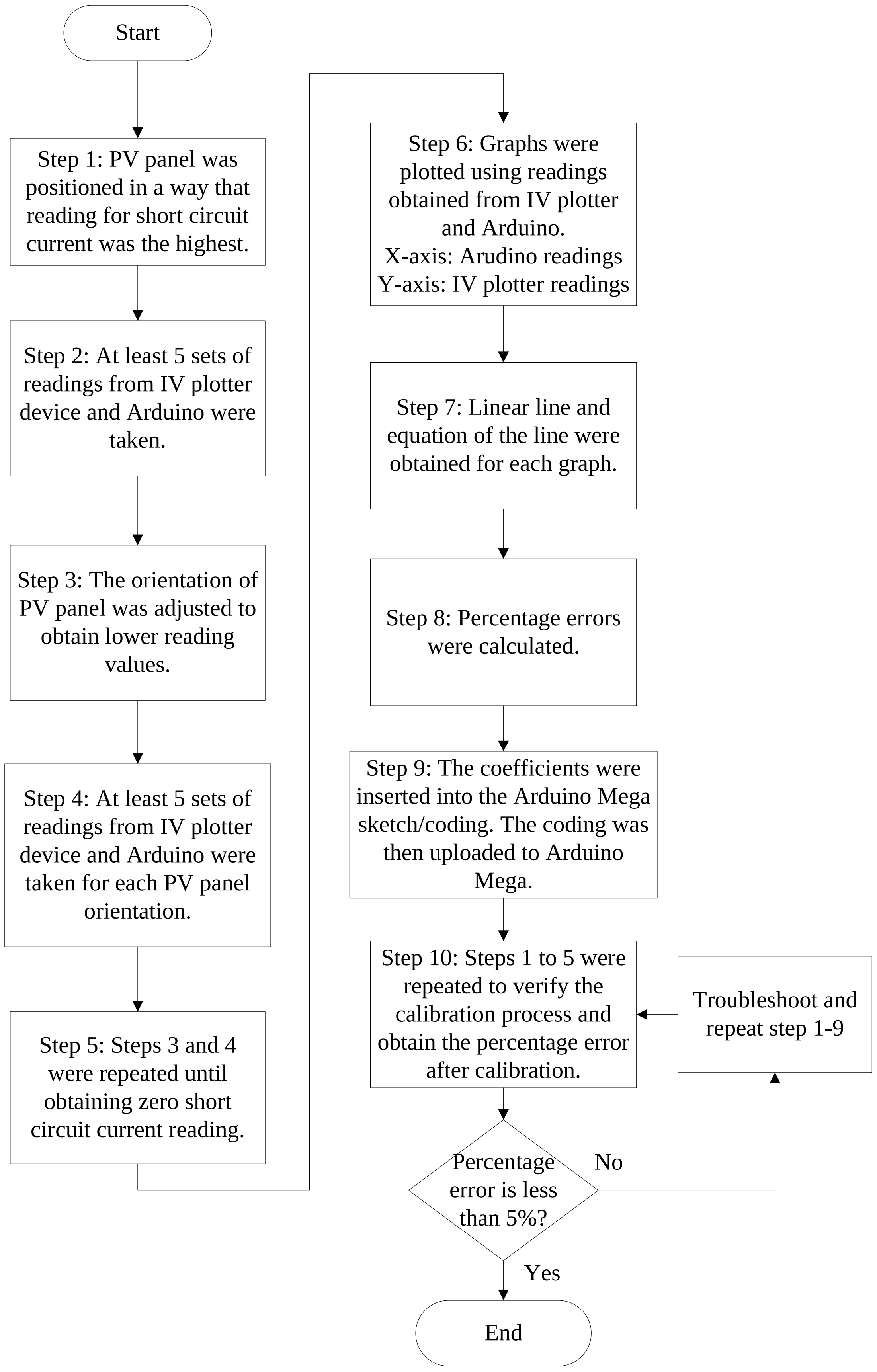

Start
Step 1: PV panel was positioned in a way that reading for short circuit current was the highest.
Step 2: At least 5 sets of readings from IV plotter device and Arduino were taken.
Step 3: The orientation of PV panel was adjusted to obtain lower reading values.
Step 4: At least 5 sets of readings from IV plotter device and Arduino were taken for each PV panel orientation.
Step 5: Steps 3 and 4 were repeated until obtaining zero short circuit current reading.
Step 6: Graphs were plotted using readings obtained from IV plotter and Arduino.
X-axis: Arudino readings
Y-axis: IV plotter readings
Step 7: Linear line and equation of the line were obtained for each graph.
Step 8: Percentage errors were calculated.
Step 9: The coefficients were inserted into the Arduino Mega sketch/coding. The coding was then uploaded to Arduino Mega.
Step 10: Steps 1 to 5 were repeated to verify the calibration process and obtain the percentage error after calibration.
Troubleshoot and repeat step 1-9
Percentage error is less than 5%?
No
Yes
End

Figure 3.32: Voltage and current calibration flowchart

3.6.2 Temperature Sensors Reading Calibration and Verification

To make sure that the DHT11 and DS18B20 sensors are providing accurate readings, they were verified using Fluke's Data Logging Thermometer as shown in Figure 3.33 below.

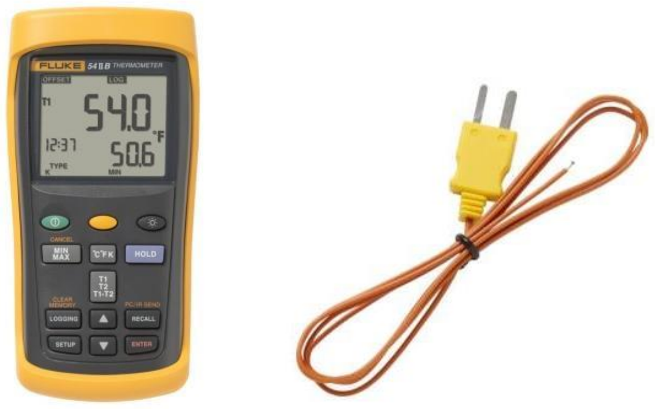


Figure 3.33: Fluke's 54 II B Data Logging Thermometer (FLUKE, n.d.)

About 5 sets of data were taken with readings ranging from room temperature to higher temperature. The sensors readings were taken and compared with Fluke's thermometer at room temperature first. Since the DHT11 sensor and DS18B20 sensors with transistor appearance are not water-proof, they cannot be immersed in water. Therefore, these sensors cannot be used in hot water to obtain higher temperature. Instead, they were held in hand tightly so the readings will increase, since room temperature is lower than human palm temperature. For consistency, the same method was applied on water-proof-type DS18B20 sensor. The readings were taken after both sensor reading and thermometer reading were consistently showing the same value after 5 seconds.

3.7 Installation of Hardware System on FPV and GPV

To measure the temperature of PV panel, the DS18B20 Dallas sensors were attached on three different position of the back of the PV panel. The first sensor was attached on the top right corner of PV panel and the sensor value was named as Temperature 1 or T1. The second sensor was attached on the middle of PV panel and the sensor value was named as Temperature 2 or T2. The last sensor was attached on the bottom left corner of PV panel and the sensor value was named as Temperature 3 or T3. The position of the temperature sensors is as shown in Figure 3.34 and 3.35 below.

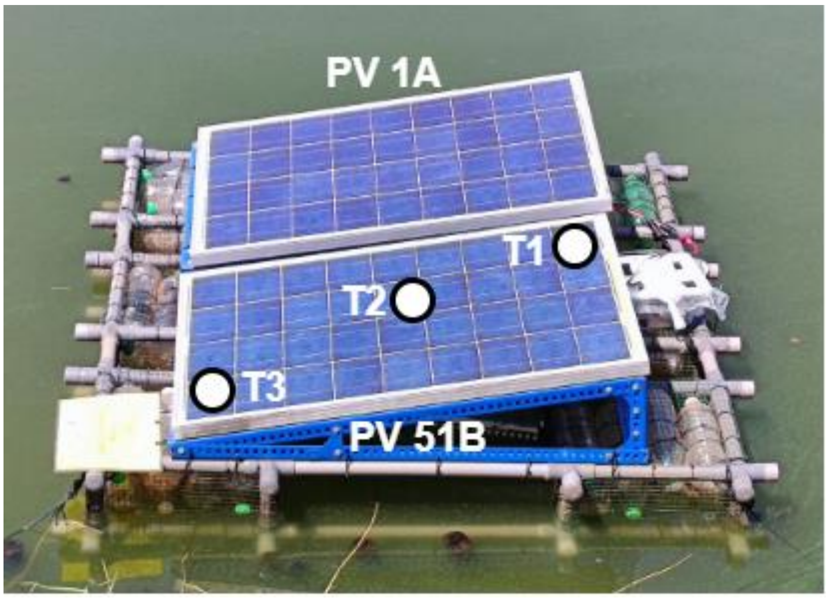


Figure 3.34: Position of temperature sensors on FPV system

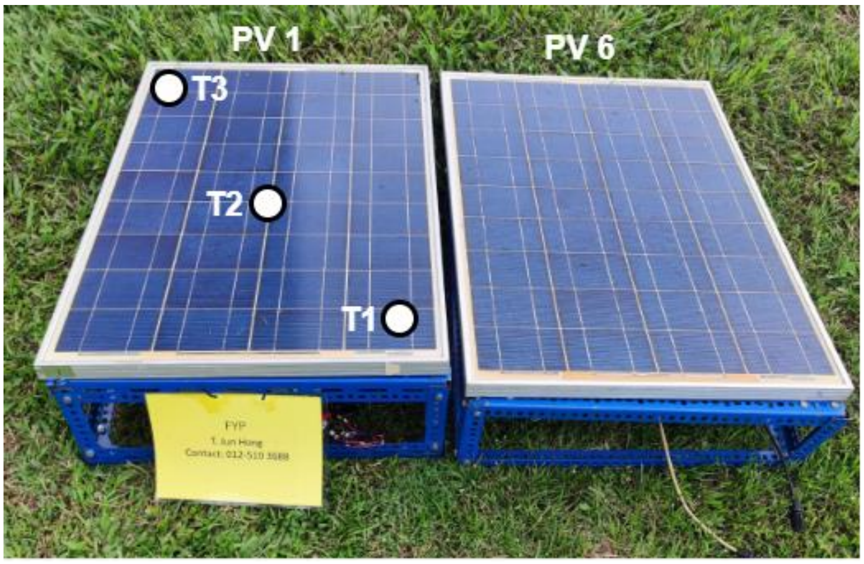


Figure 3.35: Position of temperature sensors on GPV system

To measure the Open Circuit Voltage and Short Circuit Current of PV panel, the positive and negative pole of PV panel were connected to the circuit of Arduino system using cables with crocodile clips. To prevent the crocodile clips from corrosion, the exposed part of the clips was wrapped with electrical insulation tape. Figure 3.36 below shows the side view of the GPV and FPV system. A phone was used to provide internet connection by sharing hotspot. The phone was placed between the GPV and FPV system.

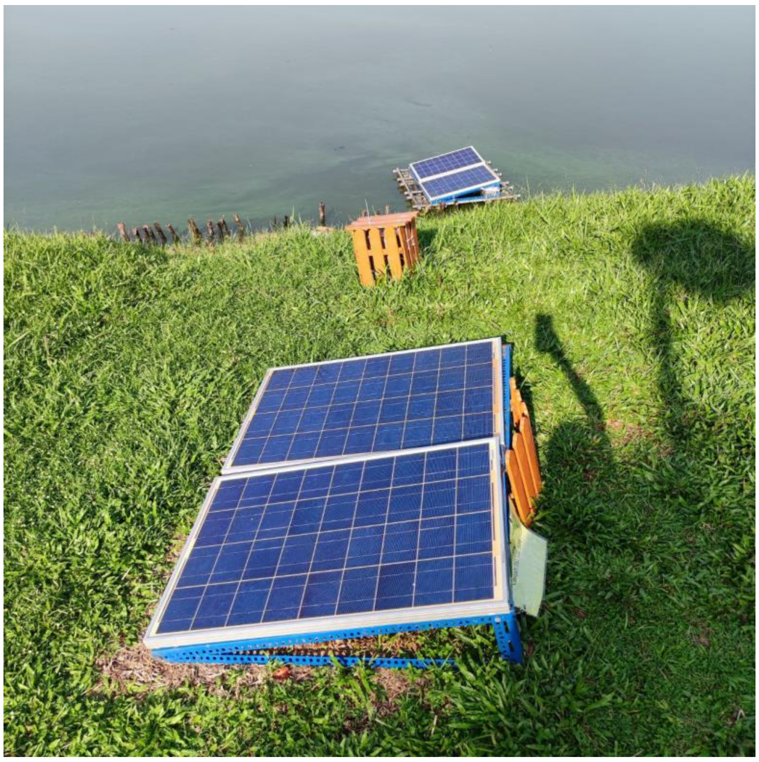


Figure 3.36: Side view of GPV and FPV setup

3.8 Configuration of IoT Data Collection System

The working principle of this IoT data collection system is based on the communication between a client and a server. A client could be any type of web browser, such as Chrome, Safari and Edge. It also can be any program or device. While a server is usually any hardware that has web server software that stores web content.

In this case, NodeMCU connects to the Internet, and acts as the client to send the HTTP (Hypertext Transfer Protocol) request to the Internet. At the same time, Google acts as the web server to receive the request. Since the data transferred to the server is in the form of XHR (XMLHttpRequest) object, the object is transformed into JSON (JavaScript Object Notation) String data type, which is done in the programme written in Apps Script. Figure 3.37 below shows how sensor data are sent and stored in IoT cloud.

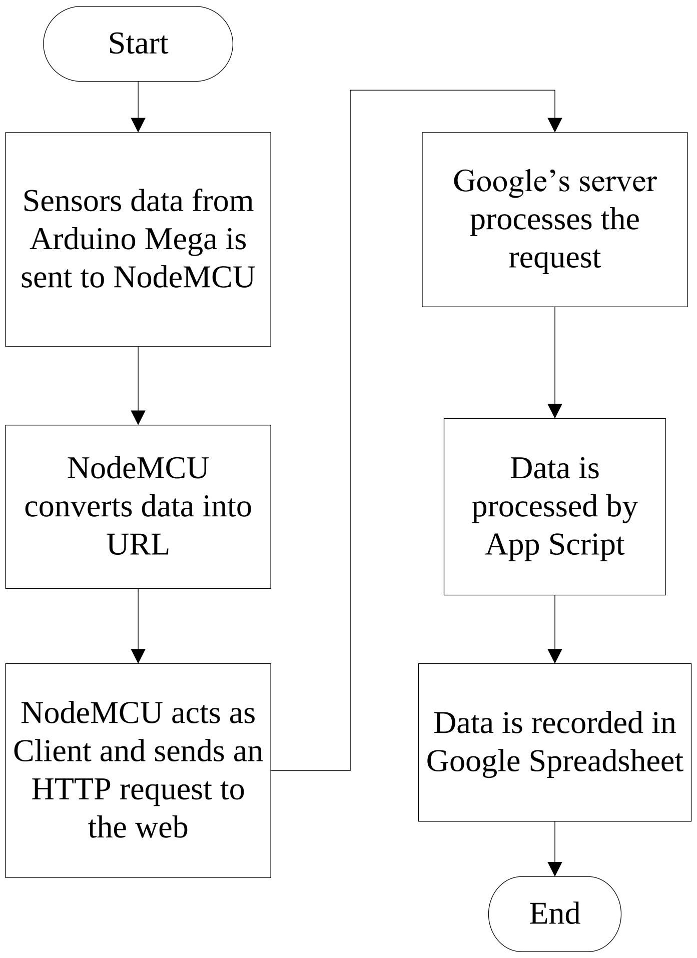

Start
Sensors data from Arduino Mega is sent to NodeMCU
NodeMCU converts data into URL
NodeMCU acts as Client and sends an HTTP request to the web
Google's server processes the request
Data is processed by App Script
Data is recorded in Google Spreadsheet
End

Figure 3.37: Flow of IoT data collection system

3.8.1 Google Spreadsheet as IoT Platform

There are several benefits of using Google Spreadsheet as IoT platform. One of them is that the data are not just displayed on the screen of laptop or smartphone, but directly stored in Google Drive. In this way the data can be easily viewed by the user or other users that the file is shared to. Both user and shared file user can view the data on their personal computer, smartphone or tablet since Google Spreadsheet app is available on these platforms. Furthermore, the data from NodeMCU is directly sent to Google Spreadsheet without needing any third party. To use Google Spreadsheet, user is only required to own or create a Google account, which creating one is completely free.

3.8.2 Configuration of Google Spreadsheet With Apps Script

To enable data receiving and saving feature in Google Spreadsheet, user needs to create an empty spreadsheet, and proceed to the Apps Script extension to programme the spreadsheet. Figure 3.38 below shows the location of Apps Script.

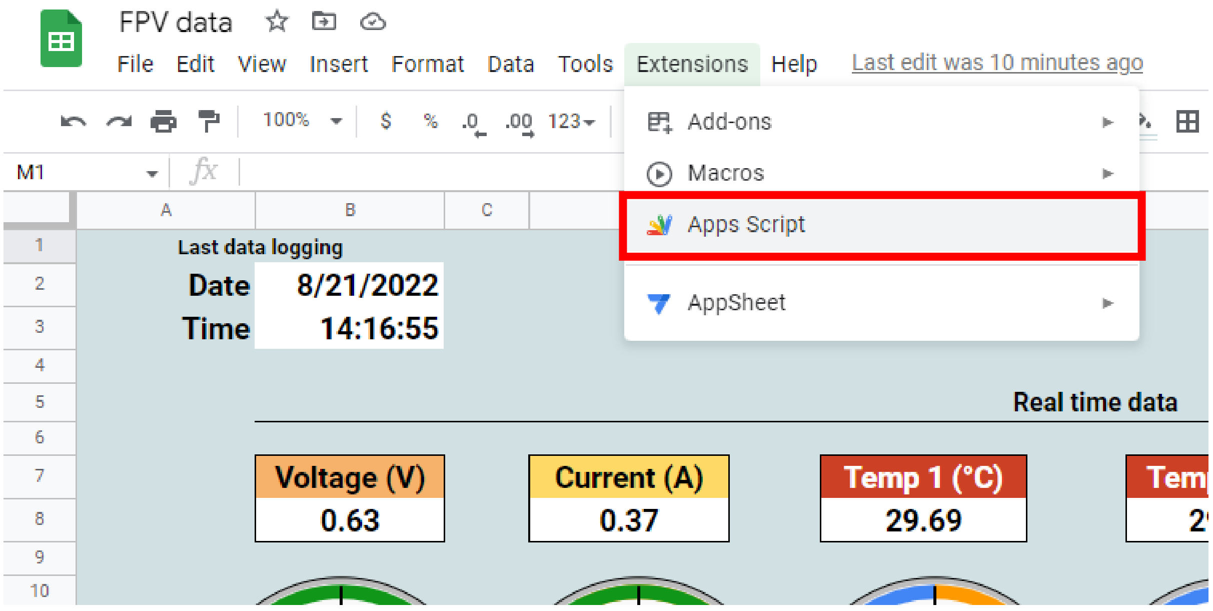
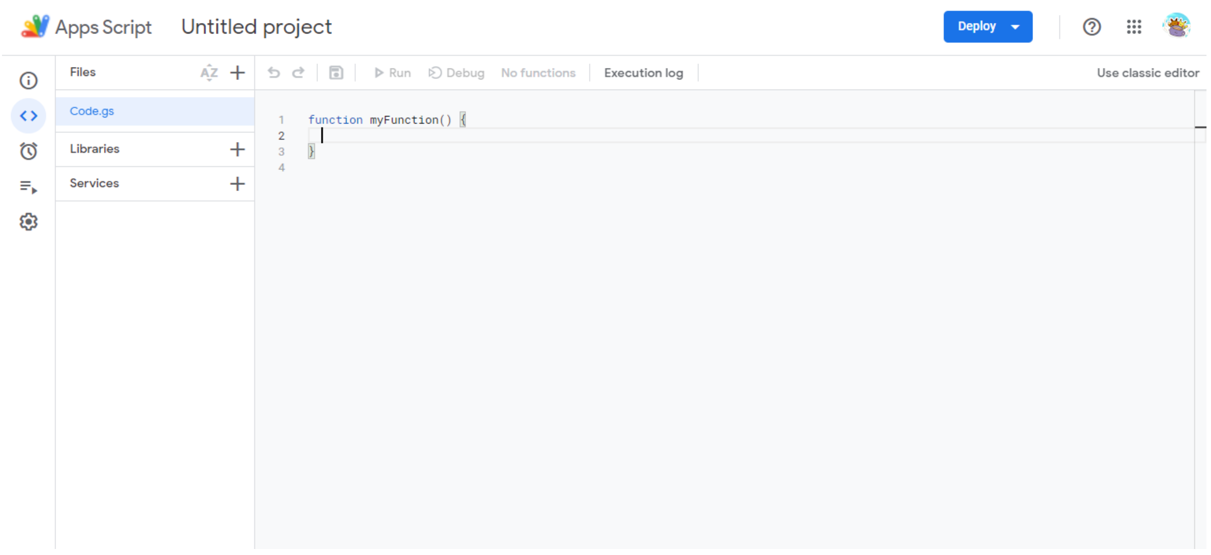


Figure 3.38: Location of Apps Script

A new page will pop out. The page is an Apps Script code editor which allows user to programme their app for automated task execution across Google products. Figure 3.39 shows the Apps Script editor page.



```
function myFunction() {

}
```

Figure 3.39: App Script code editor

The script attached in Appendix K is entered. To let the Apps Script programme recognises the desired spreadsheet that is used to store the data, user needs to copy the Spreadsheet ID from the Spreadsheet URL as shown in Figure 3.40, then enter it into the script as shown in Figure 3.41. The bolded part in the example shown below is the Spreadsheet ID. Spreadsheet ID is a unique code for each Spreadsheet file.

Example of a Spreadsheet ID in the Spreadsheet URL:

https://docs.google.com/spreadsheets/d/**1qKn2gYADXMVUAtuShwHSQmFZ3oVL6GGmEco3ea4-2qI**/edit#gid=1683798591

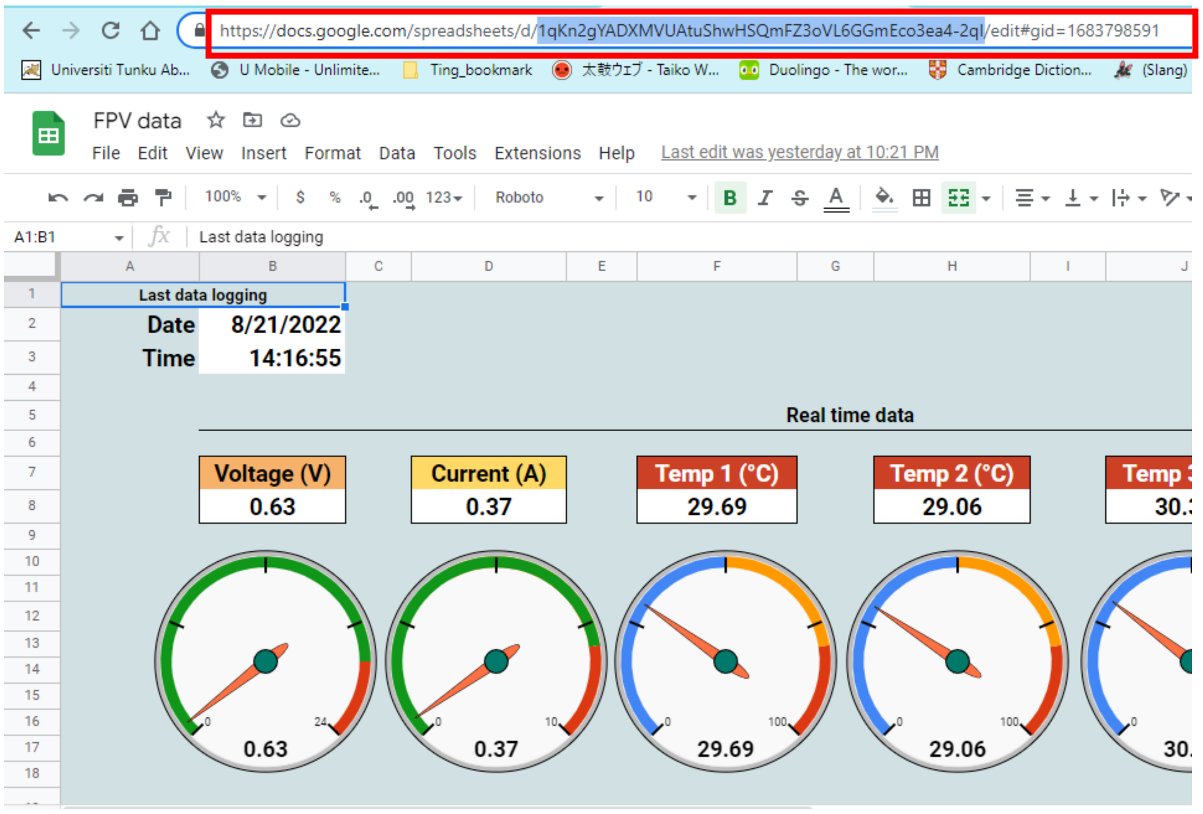
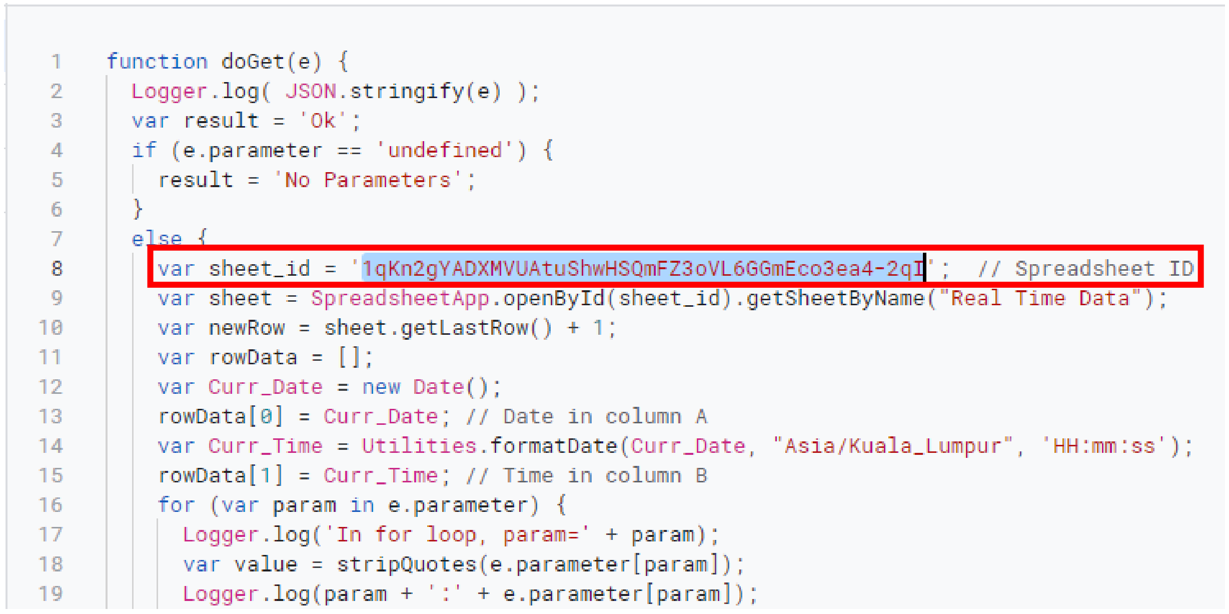

https://docs.google.com/spreadsheets/d/1qKn2gYADXMVUAtuShwHSQmFZ3oVL6GGmEco3ea4-2ql/edit#gid=1683798591
FPV data
File Edit View Insert Format Data Tools Extensions Help
Last edit was yesterday at 10:21 PM
Last data logging
Date 8/21/2022
Time 14:16:55
Real time data
Voltage (V) 0.63
Current (A) 0.37
Temp 1 (°C) 29.69
Temp 2 (°C) 29.06
Temp 3 30.
0.63
0.37
29.69
29.06
30.

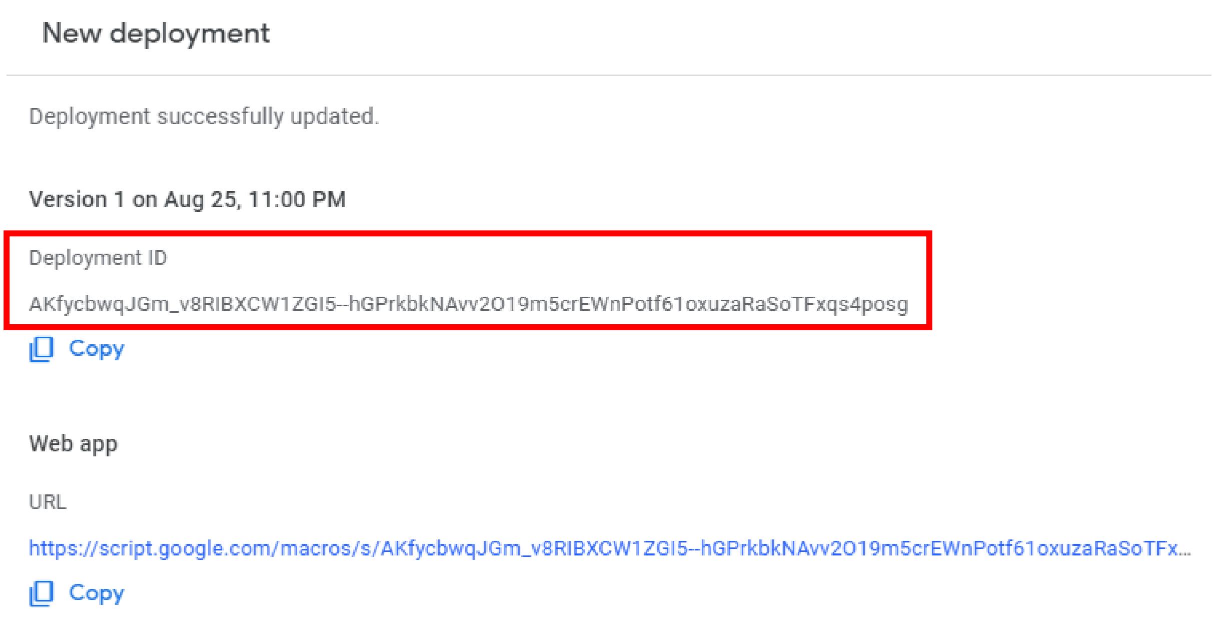
Figure 3.40: Location of Spreadsheet ID



```
function doGet(e) {
  Logger.log( JSON.stringify(e) );
  var result = 'Ok';
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
  }
  else {
    var sheet_id = '1qKn2gYADXMVUAtuShwHSQmFZ3oVL6GGmEco3ea4-2qI';   // Spreadsheet ID
    var sheet = SpreadsheetApp.openById(sheet_id).getSheetByName("Real Time Data");
    var newRow = sheet.getLastRow() + 1;
    var rowData = [];
    var Curr_Date = new Date();
    rowData[0] = Curr_Date; // Date in column A
    var Curr_Time = Utilities.formatDate(Curr_Date, "Asia/Kuala_Lumpur", 'HH:mm:ss');
    rowData[1] = Curr_Time; // Time in column B
    for (var param in e.parameter) {
      Logger.log('In for loop, param=' + param);
      var value = stripQuotes(e.parameter[param]);
      Logger.log(param + ':' + e.parameter[param]);
```

Figure 3.41: Location where Spreadsheet ID is entered

Then the programme is deployed as web app. User needs to allow the authorization for the deployment if this is their first time to deploy a web app. Then, user needs to record the web app deployment ID as shown in Figure 3.42. This ID is important for generating HTTP request from NodeMCU.


New deployment

Deployment successfully updated.

Version 1 on Aug 25, 11:00 PM

Deployment ID

AKfycbwqJGm_v8RIBXCW1ZGI5--hGPrkbkNAvv2O19m5crEWnPotf61oxuzaRaSoTFxqs4posg

Web app

URL

https://script.google.com/macros/s/AKfycbwqJGm_v8RIBXCW1ZGI5--hGPrkbkNAvv2O19m5crEWnPotf61oxuzaRaSoTFx...

Figure 3.42: Deployment ID of the web app

The above mentioned steps in this section are for GPV system. The steps were repeated for FPV system and the script was replaced with the one attached in Appendix L.

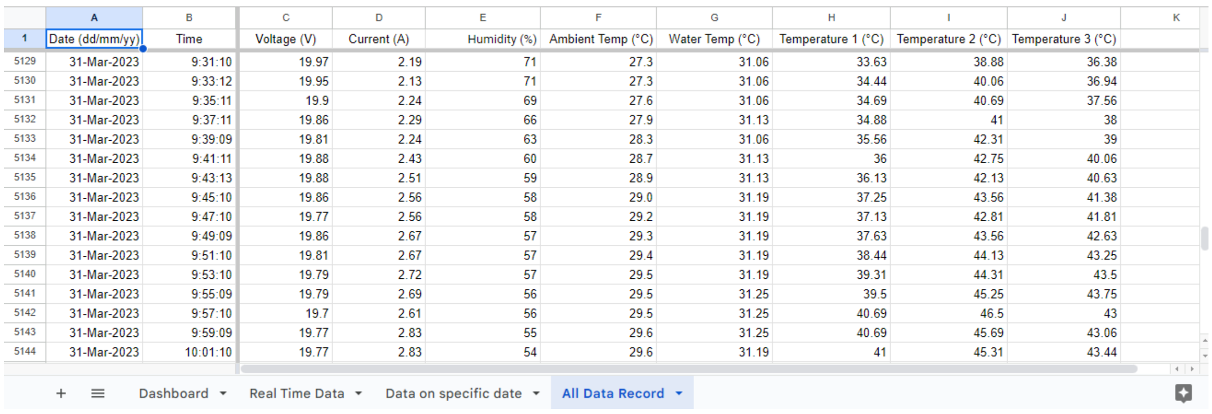
3.8.3 Setting Up Dashboard on Spreadsheet

On the IoT platform, which is Google Spreadsheet, two spreadsheet file named “FPV data” and “GPV data” was created to record and show the real time data collected from the floating PV and ground PV. There are four tabs on each spreadsheet file, namely “Dashboard”, “Real Time Data”, “Data on specific date”, and “All Data Record”. The function of each tabs is explained in Table 3.9 below.

Table 3.9: Function of the tabs in FPV data sheet and GPV data sheet

Tab	**Function**
Dashboard	To show the latest data log. The latest data log is extracted from the last row of “All Data Record” tab.
Real Time Data	To show the line charts of the day for each data type. The line charts are automatically refreshed every day.
Data on specific date	To view past data record and charts by selecting a date
All Data Record	Data from NodeMCU are recorded in this tab. Contains all data from past to present.

The data received from the NodeMCU were recorded in “All Data Record” tab. Newest data were automatically recorded in the last row of the tab. Table 3.10 below shows the column which the data of FPV were recorded in. As for GPV, the columns for the data were from column A to column I as water temperature is not required.

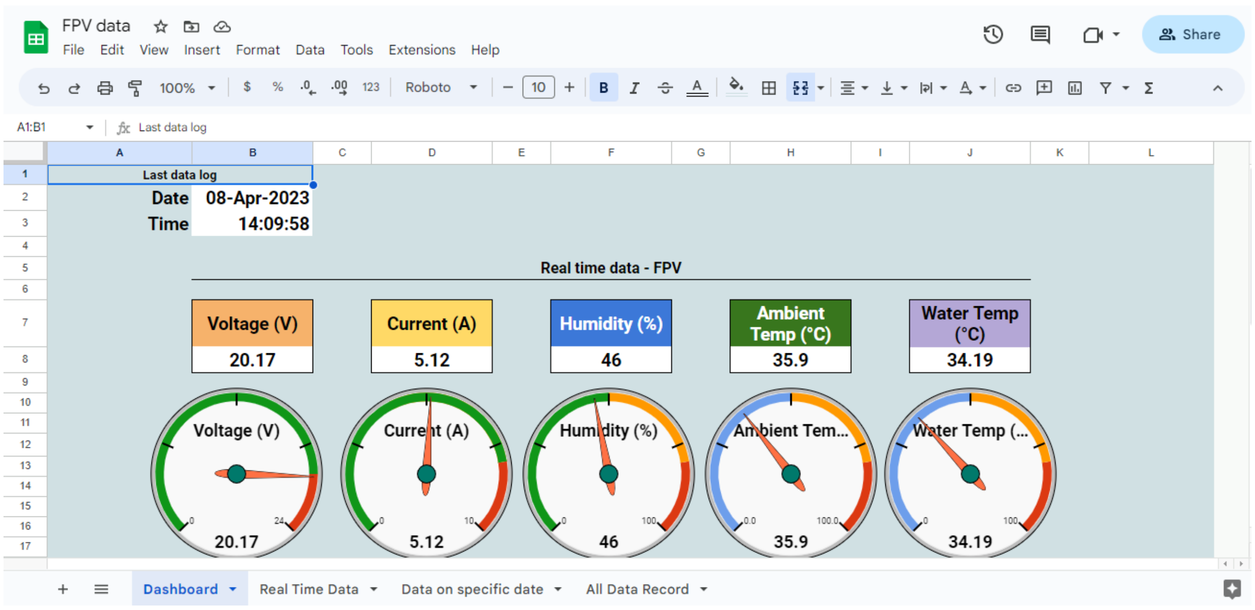


	A	B	C	D	E	F	G	H	I	J	K
1	Date (dd/mm/yy)	Time	Voltage (V)	Current (A)	Humidity (%)	Ambient Temp (°C)	Water Temp (°C)	Temperature 1 (°C)	Temperature 2 (°C)	Temperature 3 (°C)	
5129	31-Mar-2023	9:31:10	19.97	2.19	71	27.3	31.06	33.63	38.88	36.38	
5130	31-Mar-2023	9:33:12	19.95	2.13	71	27.3	31.06	34.44	40.06	36.94	
5131	31-Mar-2023	9:35:11	19.9	2.24	69	27.6	31.06	34.69	40.69	37.56	
5132	31-Mar-2023	9:37:11	19.86	2.29	66	27.9	31.13	34.88	41	38	
5133	31-Mar-2023	9:39:09	19.81	2.24	63	28.3	31.06	35.56	42.31	39	
5134	31-Mar-2023	9:41:11	19.88	2.43	60	28.7	31.13	36	42.75	40.06	
5135	31-Mar-2023	9:43:13	19.88	2.51	59	28.9	31.13	36.13	42.13	40.63	
5136	31-Mar-2023	9:45:10	19.86	2.56	58	29.0	31.19	37.25	43.56	41.38	
5137	31-Mar-2023	9:47:10	19.77	2.56	58	29.2	31.19	37.13	42.81	41.81	
5138	31-Mar-2023	9:49:09	19.86	2.67	57	29.3	31.19	37.63	43.56	42.63	
5139	31-Mar-2023	9:51:10	19.81	2.67	57	29.4	31.19	38.44	44.13	43.25	
5140	31-Mar-2023	9:53:10	19.79	2.72	57	29.5	31.19	39.31	44.31	43.5	
5141	31-Mar-2023	9:55:09	19.79	2.69	56	29.5	31.25	39.5	45.25	43.75	
5142	31-Mar-2023	9:57:10	19.7	2.61	56	29.5	31.25	40.69	46.5	43	
5143	31-Mar-2023	9:59:09	19.77	2.83	55	29.6	31.25	40.69	45.69	43.06	
5144	31-Mar-2023	10:01:10	19.77	2.83	54	29.6	31.19	41	45.31	43.44	

Figure 3.43: All Data Record tab

Table 3.10: Data type and their column location

Data type	Column
Date (dd/mm/yy)	A
Time	B
Voltage (V)	C
Current (A)	D
Humidity (%)	E
Ambient Temp (ºC)	F
Water Temp (ºC)	G
Temperature 1 (ºC)	H
Temperature 2 (ºC)	I
Temperature 3 (ºC)	J



Last data log
Date 08-Apr-2023
Time 14:09:58

Real time data - FPV

Voltage (V)	Current (A)	Humidity (%)	Ambient Temp (°C)	Water Temp (°C)
20.17	5.12	46	35.9	34.19

Figure 3.44: Screenshot of layout of Dashboard tab for FPV (1 of 2)

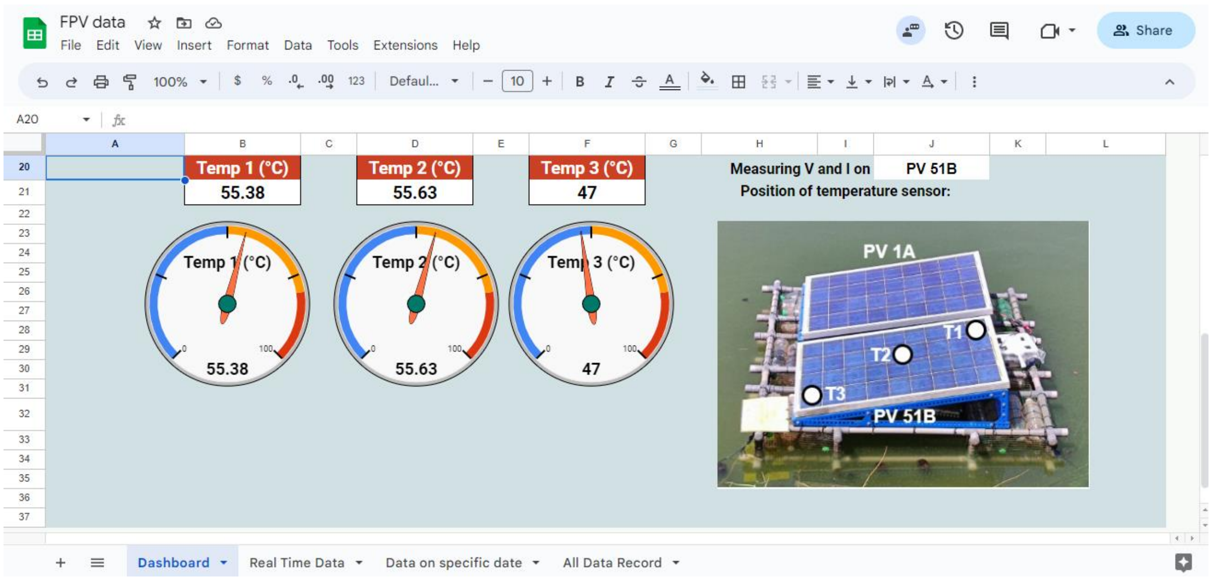


Figure 3.45: Screenshot of layout of Dashboard tab for FPV (2 of 2)

The layout of the FPV dashboard was created as shown in Figure 3.44 and 3.45 above. To insert a gauge chart, user can go to Insert tab on top of the sheet, then click on “Chart type” at the sidebar and select “Gauge chart” as shown in Figure 3.46 and 3.47 below. To let user know the position of temperature sensors, a picture with label was inserted on the Dashboard. Similar layout was created for “GPV data” sheet but with water temperature removed. To show the last data log on the Dashboard, the formulas on the Table 3.11 below were entered into the designated cells. The numbers on the dashboard will be automatically refreshed when there is new data coming in.

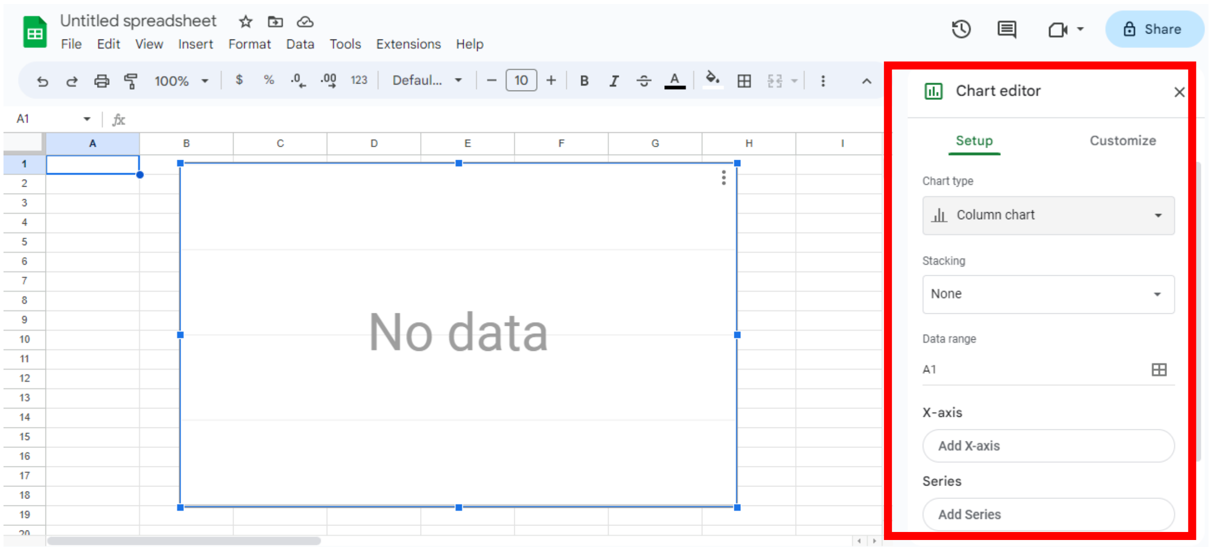


Figure 3.46: Chart editor sidebar

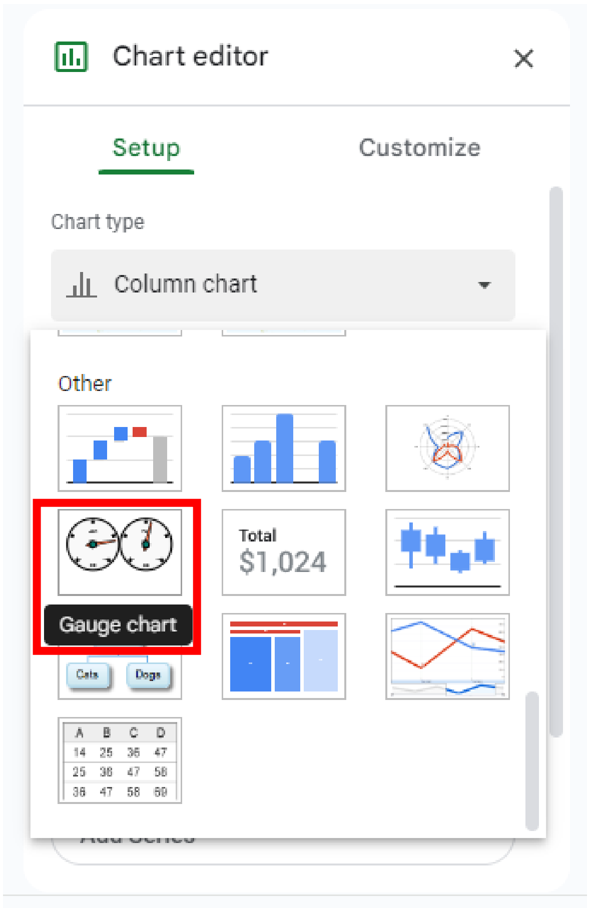


Figure 3.47: Selecting Gauge chart

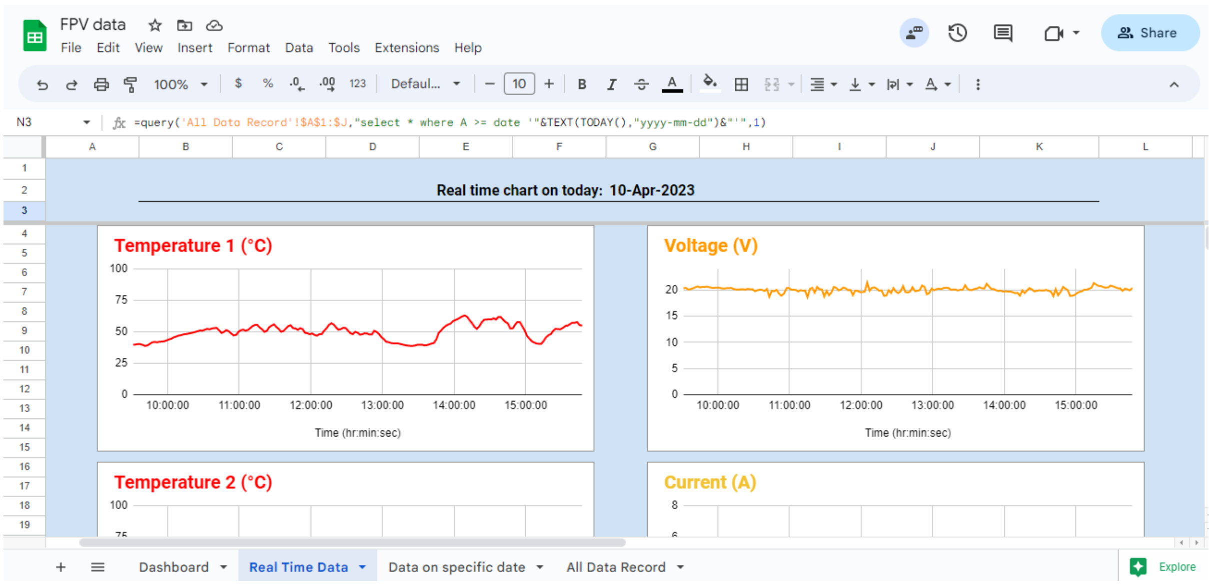
Table 3.11: Formulae to configure the Dashboard tab

Data to be shown on dashboard	**Formula**	**Location of the cell to insert the formula**
Date	=IF(OFFSET('All Data Record'!A1,CountA('All Data Record'!$A:$A)-1,0,1,1)="Date (dd/mm/yy)","-", OFFSET('All Data Record'!A1,CountA('All Data Record'!$A:$A)-1,0,1,1))	B2
Time	=IF(OFFSET('All Data Record'!B1,CountA('All Data Record'!$B:$B)-1,0,1,1)="Time","-", OFFSET('All Data Record'!B1,CountA('All Data Record'!$B:$B)-1,0,1,1))	B3

Voltage (V)	=IF(INDEX('All Data Record'!C1:C,ARRAYFORMULA(MAX((row('All Data Record'!C1:C))*--('All Data Record'!C1:C<>""))))="Voltage (V)","-", INDEX('All Data Record'!C1:C,ARRAYFORMULA(MAX((row('All Data Record'!C1:C))*--('All Data Record'!C1:C<>"")))))	B8
Current (A)	=IF(INDEX('All Data Record'!D1:D,ARRAYFORMULA(MAX((row('All Data Record'!D1:D))*--('All Data Record'!C1:D<>""))))="Current (A)","-", INDEX('All Data Record'!D1:D,ARRAYFORMULA(MAX((row('All Data Record'!D1:D))*--('All Data Record'!D1:D<>"")))))	D8
Humidity (%)	=IF(INDEX('All Data Record'!E1:E,ARRAYFORMULA(MAX((row('All Data Record'!E1:E))*--('All Data Record'!E1:E<>""))))="Humidity (%)","-", INDEX('All Data Record'!E1:E,ARRAYFORMULA(MAX((row('All Data Record'!E1:E))*--('All Data Record'!E1:E<>"")))))	F8
Ambient Temperature (ºC)	=IF(INDEX('All Data Record'!F1:F,ARRAYFORMULA(MAX((row('All Data Record'!F1:F))*--('All Data Record'!F1:F<>""))))="Ambient Temp (°C)","-", INDEX('All Data Record'!F1:F,ARRAYFORMULA(MAX((row('All Data Record'!F1:F))*--('All Data Record'!F1:F<>"")))))	H8

Water temperature (ºC)	=IF(INDEX('All Data Record'!G1:G,ARRAYFORMULA(MAX((row('All Data Record'!G1:G))*--('All Data Record'!G1:G<>""))))="Water Temp (°C)","-", INDEX('All Data Record'!G1:G,ARRAYFORMULA(MAX((row('All Data Record'!G1:G))*--('All Data Record'!G1:G<>"")))))	J8
Temperature 1 (ºC)	=IF(INDEX('All Data Record'!H1:H,ARRAYFORMULA(MAX((row('All Data Record'!H1:H))*--('All Data Record'!H1:H<>""))))="Temperature 1 (°C)","-", INDEX('All Data Record'!H1:H,ARRAYFORMULA(MAX((row('All Data Record'!H1:H))*--('All Data Record'!H1:H<>"")))))	B21
Temperature 2 (ºC)	=IF(INDEX('All Data Record'!I1:I,ARRAYFORMULA(MAX((row('All Data Record'!I1:I))*--('All Data Record'!I1:I<>""))))="Temperature 2 (°C)","-", INDEX('All Data Record'!I1:I,ARRAYFORMULA(MAX((row('All Data Record'!I1:I))*--('All Data Record'!I1:I<>"")))))	D21
Temperature 3 (ºC)	=IF(INDEX('All Data Record'!J1:J,ARRAYFORMULA(MAX((row('All Data Record'!J1:J))*--('All Data Record'!J1:J<>""))))="Temperature 3 (°C)","-", INDEX('All Data Record'!J1:J,ARRAYFORMULA(MAX((row('All Data Record'!J1:J))*--('All Data Record'!J1:J<>"")))))	F21

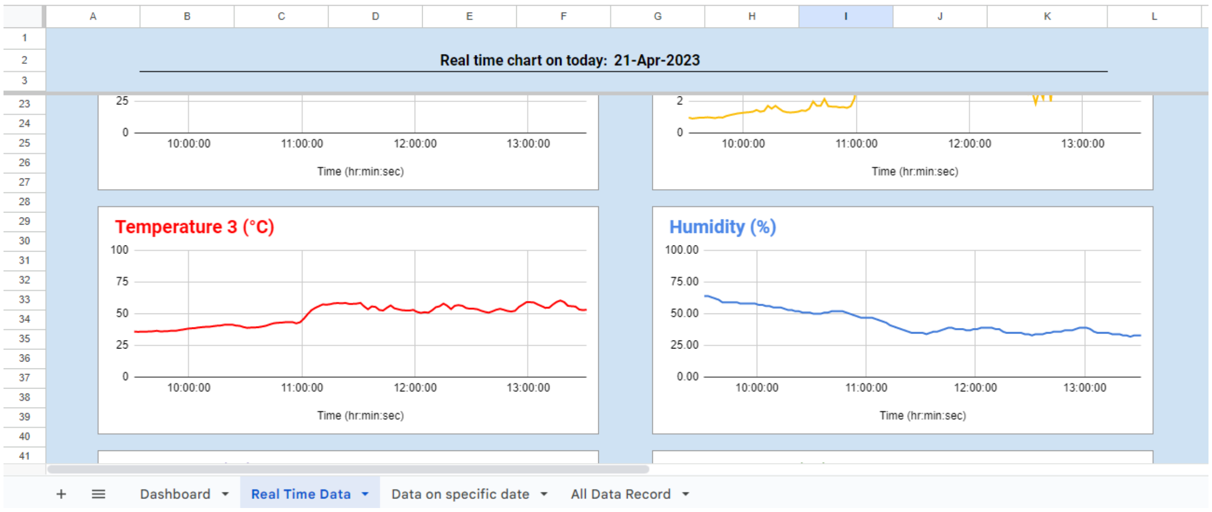
The layout of “Real Time Data” tab was created as shown in Figure 3.48 to 3.50 below. Line charts were used to display the change of the data readings throughout the day.



```
N3   =query('All Data Record'!$A$1:$J,"select * where A >= date '"&TEXT(TODAY(),"yyyy-mm-dd")&"'",1)
```

Real time chart on today: 10-Apr-2023

Figure 3.48: Screenshot of layout of Real Time Data tab (1 of 3)



Real time chart on today: 21-Apr-2023

Figure 3.49: Screenshot of layout of Real Time Data tab (2 of 3)

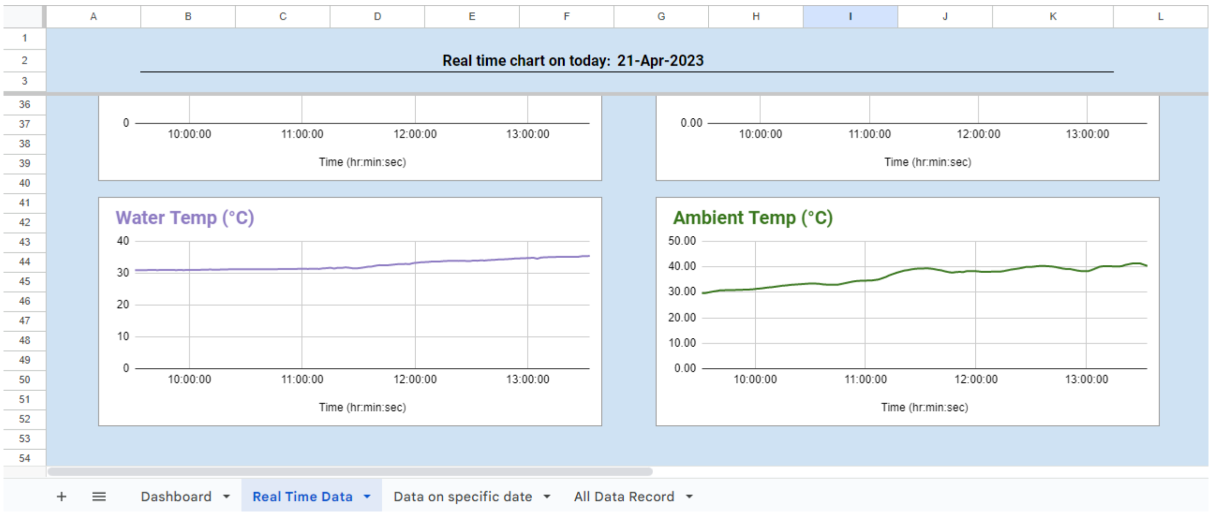
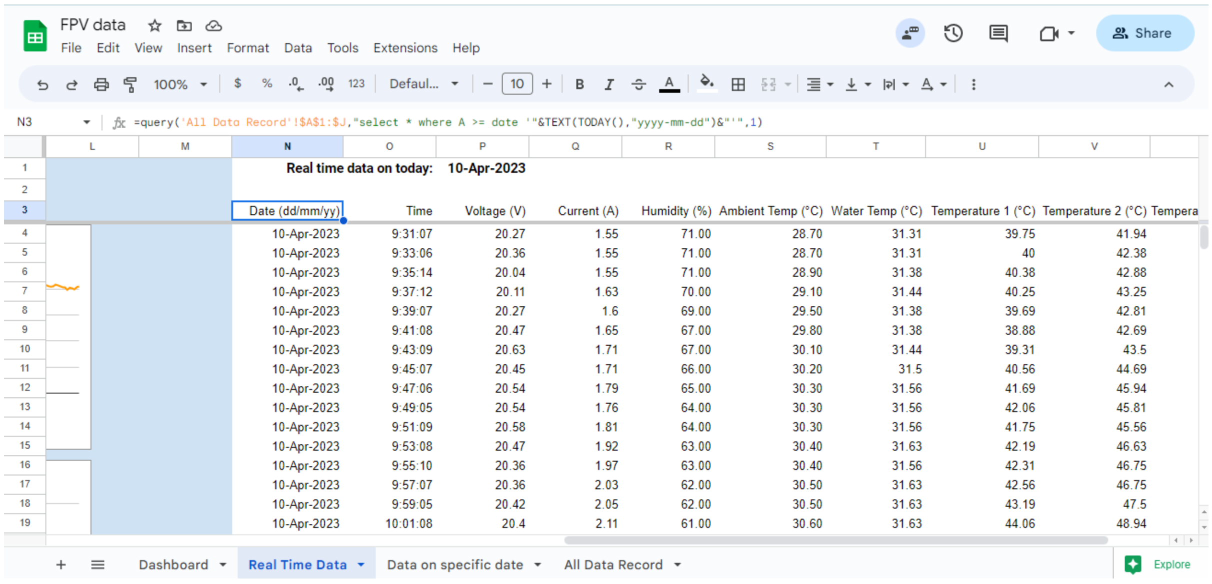


Figure 3.50: Screenshot of layout of Real Time Data tab (3 of 3)

To make the line charts refresh automatically every day, data of "today" need to be extracted out from the tab named "All Data Record". Then, the data to be displayed on the line chart is selected from the range of cells which contains data of "today". The formula used to extract the data is as shown below. This formula is inserted in N3 cell as shown in Figure 3.51.

Formula for extracting data of "today" from All Data tab:

```
=query('All Data Record'!$A$1:$J,"select * where A >= date
'"&TEXT(TODAY(),"yyyy-mm-dd")&"'",1)
```



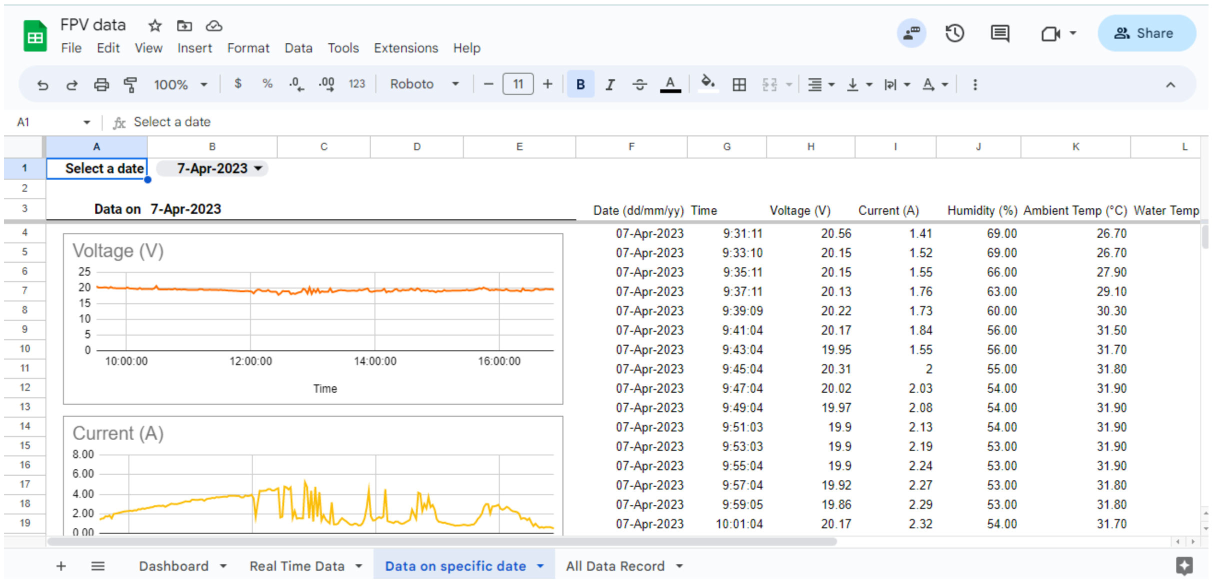
```
=query('All Data Record'!$A$1:$J,"select * where A >= date '"&TEXT(TODAY(),"yyyy-mm-dd")&"'",1)
```

Real time data on today: **10-Apr-2023**

Date (dd/mm/yy)	Time	Voltage (V)	Current (A)	Humidity (%)	Ambient Temp (°C)	Water Temp (°C)	Temperature 1 (°C)	Temperature 2 (°C)	Tempera
10-Apr-2023	9:31:07	20.27	1.55	71.00	28.70	31.31	39.75	41.94	
10-Apr-2023	9:33:06	20.36	1.55	71.00	28.70	31.31	40	42.38	
10-Apr-2023	9:35:14	20.04	1.55	71.00	28.90	31.38	40.38	42.88	
10-Apr-2023	9:37:12	20.11	1.63	70.00	29.10	31.44	40.25	43.25	
10-Apr-2023	9:39:07	20.27	1.6	69.00	29.50	31.38	39.69	42.81	
10-Apr-2023	9:41:08	20.47	1.65	67.00	29.80	31.38	38.88	42.69	
10-Apr-2023	9:43:09	20.63	1.71	67.00	30.10	31.44	39.31	43.5	
10-Apr-2023	9:45:07	20.45	1.71	66.00	30.20	31.5	40.56	44.69	
10-Apr-2023	9:47:06	20.54	1.79	65.00	30.30	31.56	41.69	45.94	
10-Apr-2023	9:49:05	20.54	1.76	64.00	30.30	31.56	42.06	45.81	
10-Apr-2023	9:51:09	20.58	1.81	64.00	30.30	31.56	41.75	45.56	
10-Apr-2023	9:53:08	20.47	1.92	63.00	30.40	31.63	42.19	46.63	
10-Apr-2023	9:55:10	20.36	1.97	63.00	30.40	31.56	42.31	46.75	
10-Apr-2023	9:57:07	20.36	2.03	62.00	30.50	31.63	42.56	46.75	
10-Apr-2023	9:59:05	20.42	2.05	62.00	30.50	31.63	43.19	47.5	
10-Apr-2023	10:01:08	20.4	2.11	61.00	30.60	31.63	44.06	48.94	

Figure 3.51: Data of "today" extracted from All Data Record tab

On the tab named "Data on specific date", user can view back the data history by selecting from a list of dates. The layout of the tab is as shown in Figure 3.52 below. The dropdown list feature on cell B1 in Figure 3.53 was created using Data Validation function as shown in Figure 3.54. Next, user needs to click on add rule, then select dropdown from a range. The range to be selected is the range of cells that record the all the dates in All Data Record tab. The range for this case is **='All Data Record'!$A$2:$A**



Select a date **7-Apr-2023**

Data on **7-Apr-2023**

Date (dd/mm/yy)	Time	Voltage (V)	Current (A)	Humidity (%)	Ambient Temp (°C)	Water Temp
07-Apr-2023	9:31:11	20.56	1.41	69.00	26.70	
07-Apr-2023	9:33:10	20.15	1.52	69.00	26.70	
07-Apr-2023	9:35:11	20.15	1.55	66.00	27.90	
07-Apr-2023	9:37:11	20.13	1.76	63.00	29.10	
07-Apr-2023	9:39:09	20.22	1.73	60.00	30.30	
07-Apr-2023	9:41:04	20.17	1.84	56.00	31.50	
07-Apr-2023	9:43:04	19.95	1.55	56.00	31.70	
07-Apr-2023	9:45:04	20.31	2	55.00	31.80	
07-Apr-2023	9:47:04	20.02	2.03	54.00	31.90	
07-Apr-2023	9:49:04	19.97	2.08	54.00	31.90	
07-Apr-2023	9:51:03	19.9	2.13	54.00	31.90	
07-Apr-2023	9:53:03	19.9	2.19	53.00	31.90	
07-Apr-2023	9:55:04	19.9	2.24	53.00	31.90	
07-Apr-2023	9:57:04	19.92	2.27	53.00	31.80	
07-Apr-2023	9:59:05	19.86	2.29	53.00	31.80	
07-Apr-2023	10:01:04	20.17	2.32	54.00	31.70	

Figure 3.52: Layout of Data on specific date tab

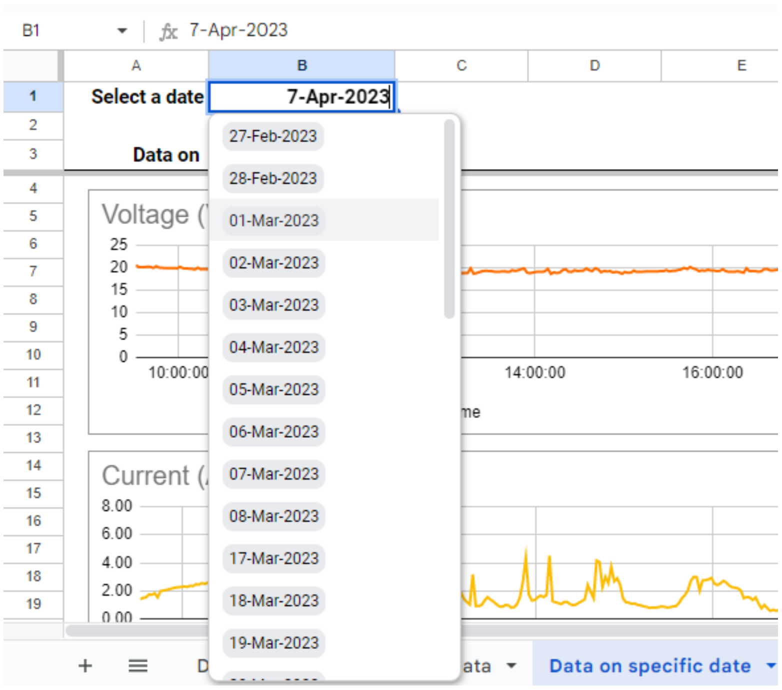


Figure 3.53: Date dropdown list

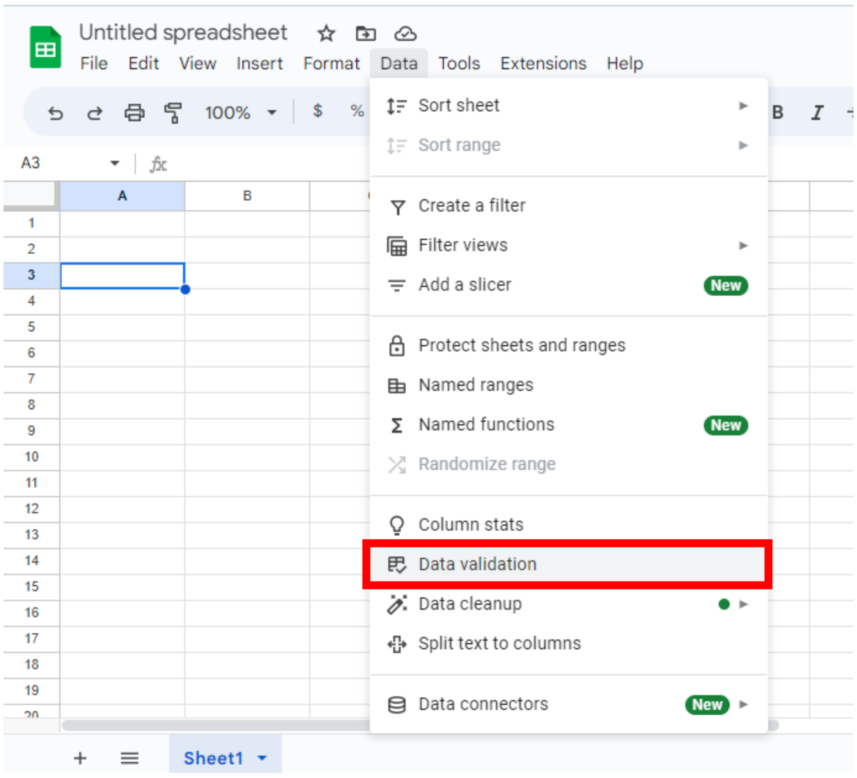


Figure 3.54: Data validation

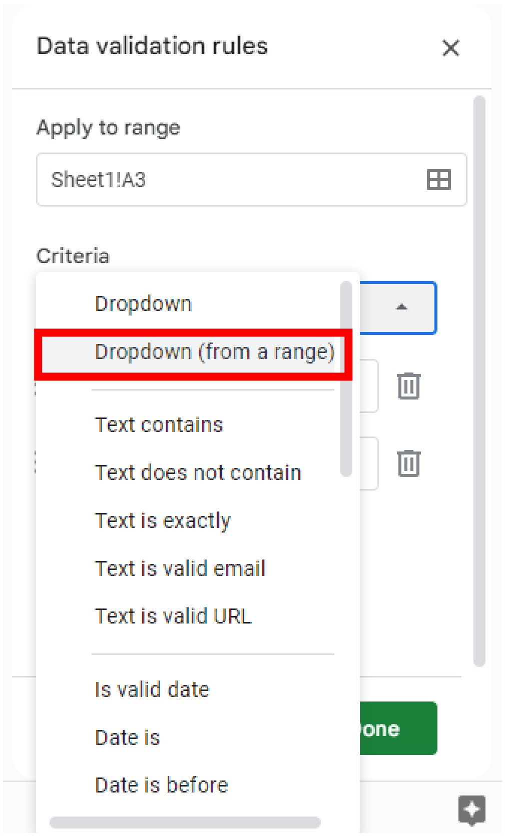


Figure 3.55: Choosing Dropdown (from a range)

Then, the formula below is inserted in F3 cell. The data to be displayed on the line chart is selected from the range of cells from N3 to column O.

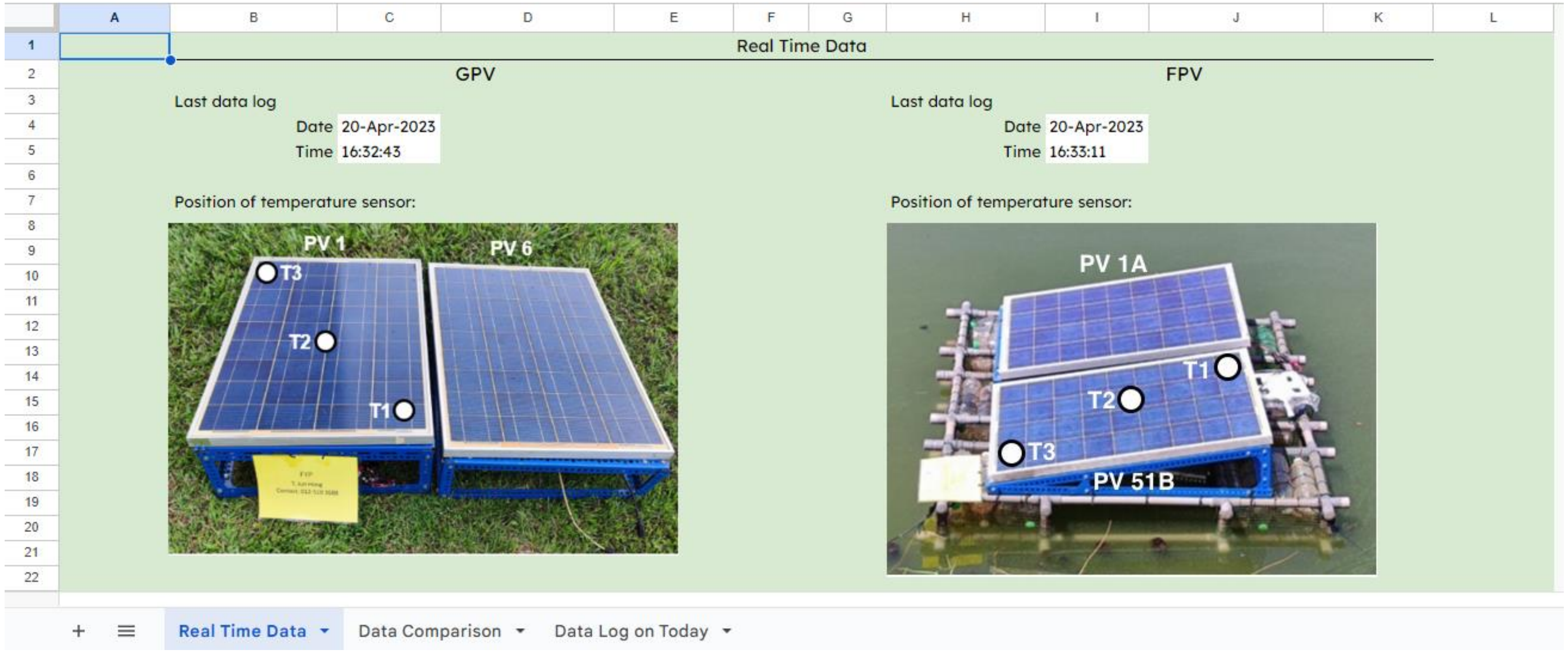
Formula for extracting data based on date selected by user:

```
=query('All Data Record'!$A$1:$J,"select * where A = date
'"&TEXT(DATEVALUE($B$3),"yyyy-mm-dd")&"'",1)
```

To view and compare the real time data of both FPV and GPV in a single dashboard, another spreadsheet file was created. The spreadsheet file was named as “GPV vs FPV”. Three tabs were created in this spreadsheet, namely “Real Time Data”, “Data Comparison” and “Data log on Today”. The function of each tab on this sheet file is explained in Table 3.12 below. The layout of the “Real Time Data” tab is as shown in Figure 3.56 to 3.59 below.

Table 3.12: Function of the tabs in “FPV vs GPV” sheet

Tab	**Function**
Real Time Data	To show the latest data log. The latest data log is extracted from the last row of “Data Log on Today” tab.
Data Comparison	To visualize the data change throughout the day with line charts in bigger size. The graphs are automatically refreshed every day.
Data Log on Today	Data from FPV and GPV sheet files are imported to this tab. Contains only data on today, and the data are automatically refreshed every day.



Real Time Data

GPV

Last data log

Date 20-Apr-2023

Time 16:32:43

Position of temperature sensor:

FPV

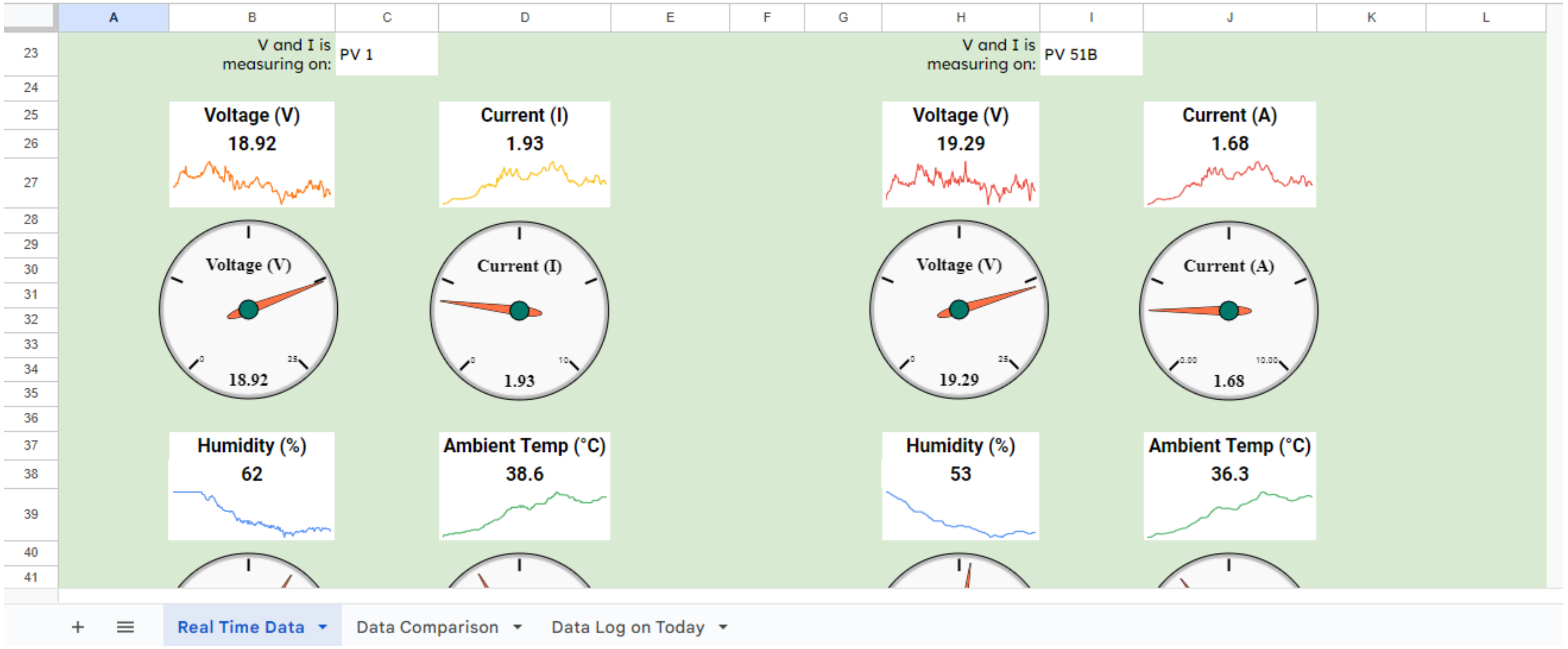
Last data log

Date 20-Apr-2023

Time 16:33:11

Position of temperature sensor:

Figure 3.56: Screenshot of layout of Dashboard tab for GPV vs FPV sheet (1 of 4)


V and I is measuring on: PV 1

V and I is measuring on: PV 51B

Figure 3.57: Screenshot of layout of Dashboard tab for GPV vs FPV sheet (2 of 4)

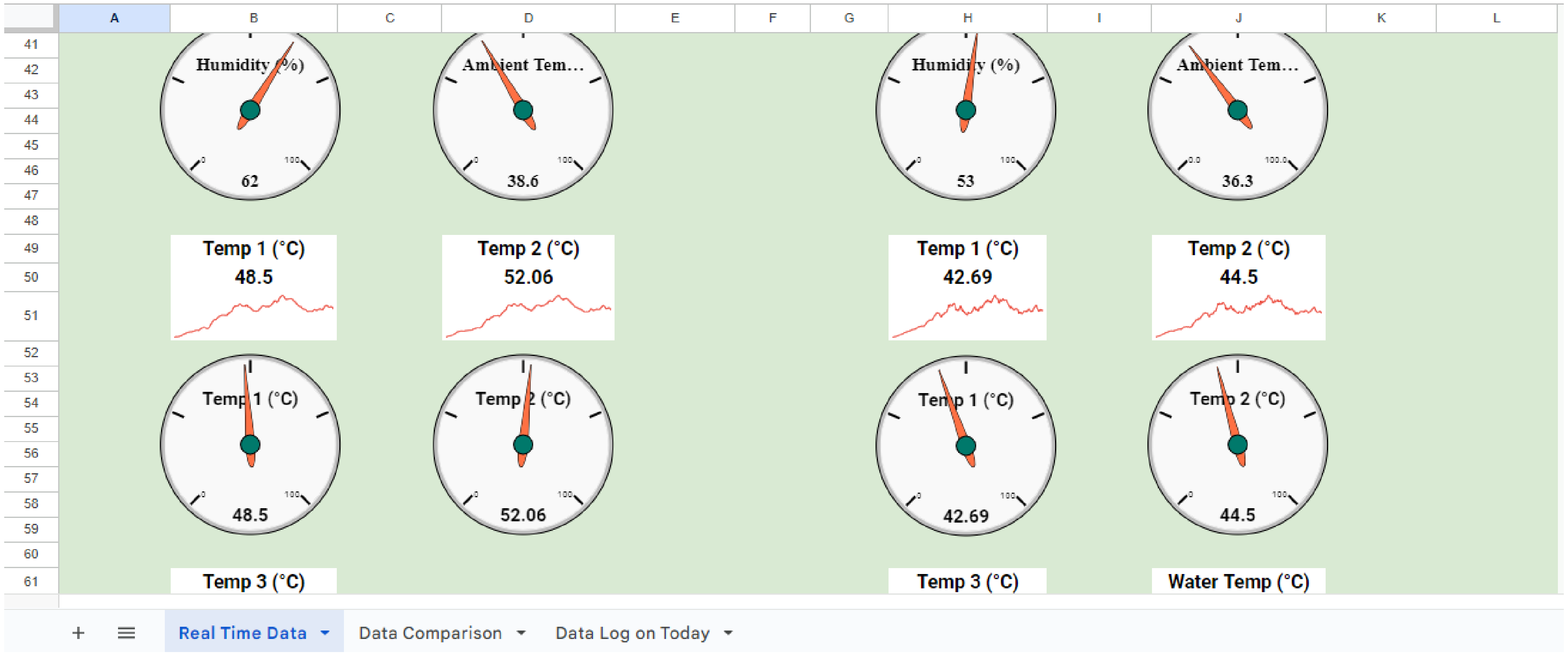


Figure 3.58: Screenshot of layout of Real Time Data tab for GPV vs FPV sheet (3 of 4)

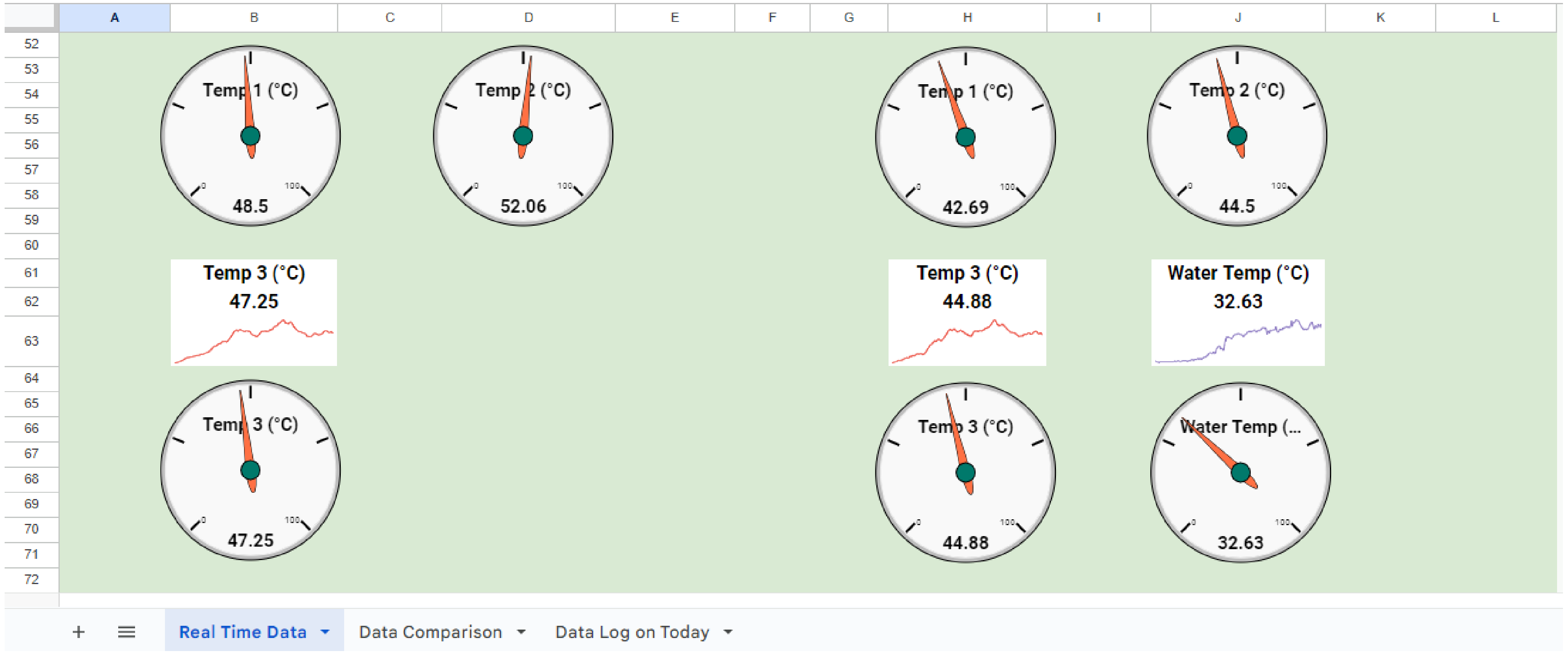


Figure 3.59: Screenshot of layout of Real Time Data tab for GPV vs FPV sheet (4 of 4)

The gauge charts were created using the steps aforementioned. Miniature charts below the data reading were created to visualize the change of the data throughout the day. The formula to create the layout of “Real Time Data” tab is as shown in Table 3.13 and 3.14 below.

Table 3.13: Formulae to configure the Real Time Data tab for GPV

Data to be shown on Real Time Data tab (GPV)	**Formula**	**Location of the cell to insert the formula**
Date	=IF(INDEX('Data Log on Today'!A1:A,ARRAYFORMULA(MAX((row('Data Log on Today'!A1:A))*--('Data Log on Today'!A1:A<>""))))="Date (dd/mm/yy)","-", INDEX('Data Log on Today'!A1:A,ARRAYFORMULA(MAX((row('Data Log on Today'!A1:A))*--('Data Log on Today'!A1:A<>"")))))	C4
Time	=IF(INDEX('Data Log on Today'!B1:B,ARRAYFORMULA(MAX((row('Data Log on Today'!B1:B))*--('Data Log on Today'!B1:B<>""))))="Time","-", INDEX('Data Log on Today'!B1:B,ARRAYFORMULA(MAX((row('Data Log on Today'!B1:B))*--('Data Log on Today'!B1:B<>"")))))	C5
Voltage (V)	=IF(INDEX('Data Log on Today'!C1:C,ARRAYFORMULA(MAX((row('Data Log on Today'!C1:C))*--('Data Log on Today'!C1:C<>""))))="Voltage (V)","-", INDEX('Data Log on Today'!C1:C,ARRAYFORMULA(MAX((row('Data Log on Today'!C1:C))*--('Data Log on Today'!C1:C<>"")))))	B26
Voltage miniature chart	=SPARKLINE('Data Log on Today'!C5:C)	B27

Current (A)	=IF(INDEX('Data Log on Today'!D1:D,ARRAYFORMULA(MAX((row('Data Log on Today'!D1:D))*--('Data Log on Today'!D1:D<>"")))))="Current (A)","-", INDEX('Data Log on Today'!D1:D,ARRAYFORMULA(MAX((row('Data Log on Today'!D1:D))*--('Data Log on Today'!D1:D<>"")))))	D26
Current miniature chart	=SPARKLINE('Data Log on Today'!D5:D)	D27
Humidity (%)	=IF(INDEX('Data Log on Today'!E1:E,ARRAYFORMULA(MAX((row('Data Log on Today'!E1:E))*--('Data Log on Today'!E1:E<>"")))))="Humidity (%)","-", INDEX('Data Log on Today'!E1:E,ARRAYFORMULA(MAX((row('Data Log on Today'!E1:E))*--('Data Log on Today'!E1:E<>"")))))	B38
Humidity miniature chart	=SPARKLINE('Data Log on Today'!E5:E)	B39
Ambient Temperature (ºC)	=IF(INDEX('Data Log on Today'!F1:F,ARRAYFORMULA(MAX((row('Data Log on Today'!F1:F))*--('Data Log on Today'!F1:F<>"")))))="Ambient Temp (°C)","-", INDEX('Data Log on Today'!F1:F,ARRAYFORMULA(MAX((row('Data Log on Today'!F1:F))*--('Data Log on Today'!F1:F<>"")))))	D38
Ambient Temperature miniature chart	=SPARKLINE('Data Log on Today'!F5:F)	D39
Temperature 1 (ºC)	=IF(INDEX('Data Log on Today'!G1:G,ARRAYFORMULA(MAX((row('Data Log on Today'!G1:G))*--('Data Log on Today'!G1:G<>"")))))="Temperature 1 (°C)","-",	B50

	INDEX('Data Log on Today'!G1:G,ARRAYFORMULA(MAX((row('Data Log on Today'!G1:G))*--('Data Log on Today'!G1:G<>"")))))	
Temperature 1 miniature chart	=SPARKLINE('Data Log on Today'!G5:G)	B51
Temperature 2 (ºC)	=IF(INDEX('Data Log on Today'!H1:H,ARRAYFORMULA(MAX((row('Data Log on Today'!H1:H))*--('Data Log on Today'!H1:H<>""))))="Temperature 2 (°C)","-", INDEX('Data Log on Today'!H1:H,ARRAYFORMULA(MAX((row('Data Log on Today'!H1:H))*--('Data Log on Today'!H1:H<>"")))))	D50
Temperature 2 miniature chart	=SPARKLINE('Data Log on Today'!H5:H)	D51
Temperature 3 (ºC)	=IF(INDEX('Data Log on Today'!I1:I,ARRAYFORMULA(MAX((row('Data Log on Today'!I1:I))*--('Data Log on Today'!I1:I<>""))))="Temperature 3 (°C)","-", INDEX('Data Log on Today'!I1:I,ARRAYFORMULA(MAX((row('Data Log on Today'!I1:I))*--('Data Log on Today'!I1:I<>"")))))	B62
Temperature 3 miniature chart	=SPARKLINE('Data Log on Today'!I5:I)	B63

Table 3.14: Formulae to configure the Real Time Data tab for FPV

Data to be shown on Real Time Data tab (FPV)	**Formula**	**Location of the cell to insert the formula**
Date	=IF(INDEX('Data Log on Today'!K1:K,ARRAYFORMULA(MAX((row('Data Log on Today'!K1:K))*--('Data Log on Today'!K1:K<>""))))="Date	I4

	(dd/mm/yy)","-", INDEX('Data Log on Today'!K1:K,ARRAYFORMULA(MAX((row('Data Log on Today'!K1:K))*--('Data Log on Today'!K1:K<>"")))))	
Time	=IF(INDEX('Data Log on Today'!L1:L,ARRAYFORMULA(MAX((row('Data Log on Today'!L1:L))*--('Data Log on Today'!L1:L<>""))))="Time","-", INDEX('Data Log on Today'!L1:L,ARRAYFORMULA(MAX((row('Data Log on Today'!L1:L))*--('Data Log on Today'!L1:L<>"")))))	I5
Voltage (V)	=IF(INDEX('Data Log on Today'!M1:M,ARRAYFORMULA(MAX((row('Data Log on Today'!M1:M))*--('Data Log on Today'!M1:M<>""))))="Voltage (V)","-", INDEX('Data Log on Today'!M1:M,ARRAYFORMULA(MAX((row('Data Log on Today'!M1:M))*--('Data Log on Today'!M1:M<>"")))))	H26
Voltage miniature chart	=SPARKLINE('Data Log on Today'!M5:M)	H27
Current (A)	=IF(INDEX('Data Log on Today'!N1:N,ARRAYFORMULA(MAX((row('Data Log on Today'!N1:N))*--('Data Log on Today'!N1:N<>""))))="Current (A)","-", INDEX('Data Log on Today'!N1:N,ARRAYFORMULA(MAX((row('Data Log on Today'!N1:N))*--('Data Log on Today'!N1:N<>"")))))	J26
Current miniature chart	=SPARKLINE('Data Log on Today'!N5:N)	J27
Humidity (%)	=IF(INDEX('Data Log on Today'!O1:O,ARRAYFORMULA(MAX((row('Data Log on Today'!O1:O))*--('Data Log on Today'!O1:O<>""))))="Humidity (%)","-	H38

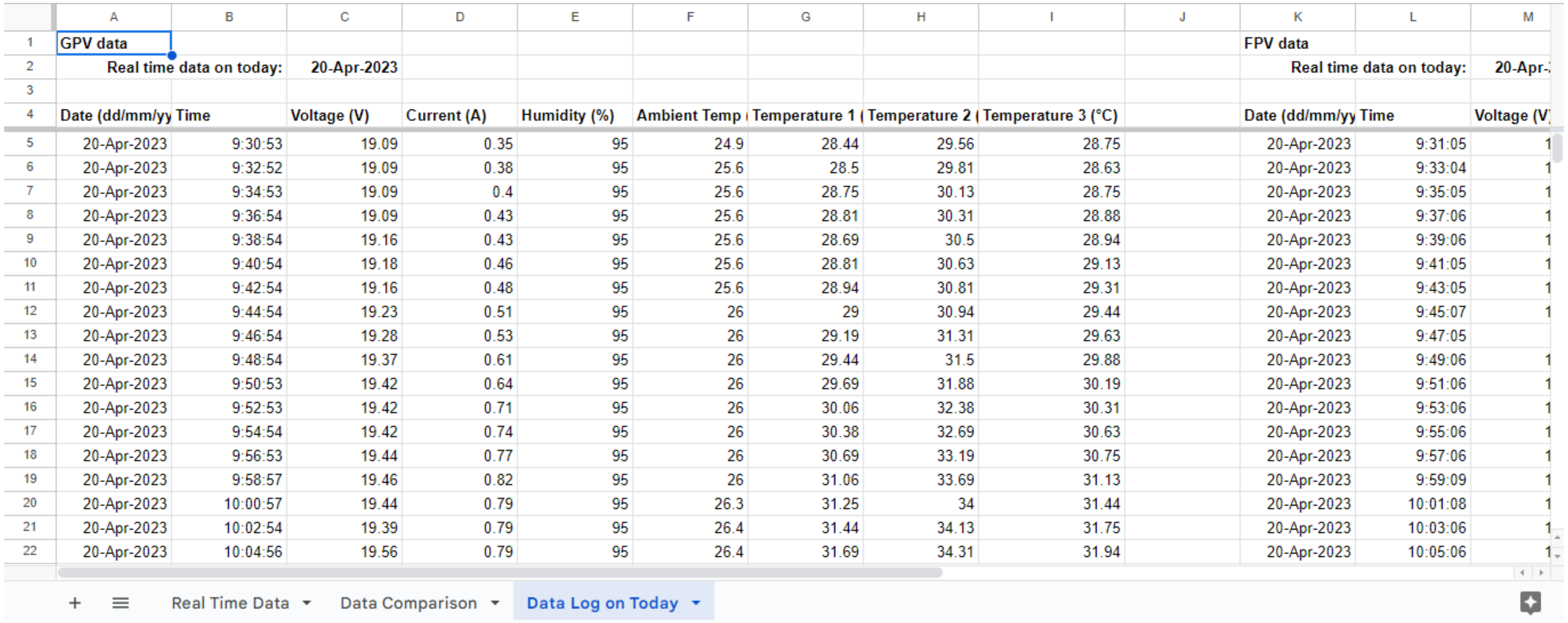
	", INDEX('Data Log on Today'!O1:O,ARRAYFORMULA(MAX((row('Data Log on Today'!O1:O))*--('Data Log on Today'!O1:O<>"")))))	
Humidity miniature chart	=SPARKLINE('Data Log on Today'!O5:O)	H39
Ambient Temperature (ºC)	=IF(INDEX('Data Log on Today'!P1:P,ARRAYFORMULA(MAX((row('Data Log on Today'!P1:P))*--('Data Log on Today'!P1:P<>""))))="Ambient Temp (°C)","-", INDEX('Data Log on Today'!P1:P,ARRAYFORMULA(MAX((row('Data Log on Today'!P1:P))*--('Data Log on Today'!P1:P<>"")))))	J38
Ambient Temperature miniature chart	=SPARKLINE('Data Log on Today'!P5:P)	J39
Temperature 1 (ºC)	=IF(INDEX('Data Log on Today'!R1:R,ARRAYFORMULA(MAX((row('Data Log on Today'!R1:R))*--('Data Log on Today'!R1:R<>""))))="Temperature 1 (°C)","-", INDEX('Data Log on Today'!R1:R,ARRAYFORMULA(MAX((row('Data Log on Today'!R1:R))*--('Data Log on Today'!R1:R<>"")))))	H50
Temperature 1 miniature chart	=SPARKLINE('Data Log on Today'!R5:R)	H51
Temperature 2 (ºC)	=IF(INDEX('Data Log on Today'!S1:S,ARRAYFORMULA(MAX((row('Data Log on Today'!S1:S))*--('Data Log on Today'!S1:S<>""))))="Temperature 2 (°C)","-", INDEX('Data Log on	J50

	Today'!S1:S,ARRAYFORMULA(MAX((row('Data Log on Today'!S1:S))*--('Data Log on Today'!S1:S<>"")))))	
Temperature 2 miniature chart	=SPARKLINE('Data Log on Today'!S5:S)	J51
Temperature 3 (ºC)	=IF(INDEX('Data Log on Today'!T1:T,ARRAYFORMULA(MAX((row('Data Log on Today'!T1:T))*--('Data Log on Today'!T1:T<>""))))="Temperature 3 (°C)","-", INDEX('Data Log on Today'!T1:T,ARRAYFORMULA(MAX((row('Data Log on Today'!T1:T))*--('Data Log on Today'!T1:T<>"")))))	H62
Temperature 3 miniature chart	=SPARKLINE('Data Log on Today'!T5:T)	H63
Water Temperature (ºC)	=IF(INDEX('Data Log on Today'!Q1:Q,ARRAYFORMULA(MAX((row('Data Log on Today'!Q1:Q))*--('Data Log on Today'!Q1:Q<>""))))="Water Temp (°C)","-", INDEX('Data Log on Today'!Q1:Q,ARRAYFORMULA(MAX((row('Data Log on Today'!Q1:Q))*--('Data Log on Today'!Q1:Q<>"")))))	J62
Water Temperature miniature chart	=SPARKLINE('Data Log on Today'!Q5:Q)	J63

The information on this spreadsheet is linked from the previous spreadsheet files-“GPV data” and “FPV data”. This was achieved using the IMPORTRANGE formula as shown in the Table 3.15 below. The text in green colour is the URL link of the “GPV data” and “FPV data” spreadsheet file, and the text in orange is the range of the cells to be copied. This formula was entered in “Data Log on Today” tab. With this formula, the data from the “Real Time Data” tab of “FPV data” and “GPV data” sheet can then be imported to this spreadsheet.

Table 3.15: Formulae for importing data from FPV data sheet and GPV data sheet

Function	**Formula**	**Location of the cell to insert the formula**
Import data from GPV sheet	=IMPORTRANGE("https://docs.google.com/spreadsheets/d/1qKn2gYADXMVUAtuShwHSQmFZ3oVL6GGmEco3ea4-2qI/edit#gid=1673341961", "Real Time Data!O1:W")	K2
Import data from FPV sheet	=IMPORTRANGE("https://docs.google.com/spreadsheets/d/1P2aIakEACvnCPuFFa0lgN8z-CcwsqbvxoWirtJgYu0w/edit#gid=1673341961","Real Time Data!N1:W")	A2



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	GPV data										FPV data		
2		Real time data on today:	20-Apr-2023									Real time data on today:	20-Apr-
3													
4	Date (dd/mm/yy	Time	Voltage (V)	Current (A)	Humidity (%)	Ambient Temp (	Temperature 1 (	Temperature 2 (	Temperature 3 (°C)		Date (dd/mm/yy	Time	Voltage (V
5	20-Apr-2023	9:30:53	19.09	0.35	95	24.9	28.44	29.56	28.75		20-Apr-2023	9:31:05	1
6	20-Apr-2023	9:32:52	19.09	0.38	95	25.6	28.5	29.81	28.63		20-Apr-2023	9:33:04	1
7	20-Apr-2023	9:34:53	19.09	0.4	95	25.6	28.75	30.13	28.75		20-Apr-2023	9:35:05	1
8	20-Apr-2023	9:36:54	19.09	0.43	95	25.6	28.81	30.31	28.88		20-Apr-2023	9:37:06	1
9	20-Apr-2023	9:38:54	19.16	0.43	95	25.6	28.69	30.5	28.94		20-Apr-2023	9:39:06	1
10	20-Apr-2023	9:40:54	19.18	0.46	95	25.6	28.81	30.63	29.13		20-Apr-2023	9:41:05	1
11	20-Apr-2023	9:42:54	19.16	0.48	95	25.6	28.94	30.81	29.31		20-Apr-2023	9:43:05	1
12	20-Apr-2023	9:44:54	19.23	0.51	95	26	29	30.94	29.44		20-Apr-2023	9:45:07	1
13	20-Apr-2023	9:46:54	19.28	0.53	95	26	29.19	31.31	29.63		20-Apr-2023	9:47:05	
14	20-Apr-2023	9:48:54	19.37	0.61	95	26	29.44	31.5	29.88		20-Apr-2023	9:49:06	1
15	20-Apr-2023	9:50:53	19.42	0.64	95	26	29.69	31.88	30.19		20-Apr-2023	9:51:06	1
16	20-Apr-2023	9:52:53	19.42	0.71	95	26	30.06	32.38	30.31		20-Apr-2023	9:53:06	1
17	20-Apr-2023	9:54:54	19.42	0.74	95	26	30.38	32.69	30.63		20-Apr-2023	9:55:06	1
18	20-Apr-2023	9:56:53	19.44	0.77	95	26	30.69	33.19	30.75		20-Apr-2023	9:57:06	1
19	20-Apr-2023	9:58:57	19.46	0.82	95	26	31.06	33.69	31.13		20-Apr-2023	9:59:09	1
20	20-Apr-2023	10:00:57	19.44	0.79	95	26.3	31.25	34	31.44		20-Apr-2023	10:01:08	1
21	20-Apr-2023	10:02:54	19.39	0.79	95	26.4	31.44	34.13	31.75		20-Apr-2023	10:03:06	1
22	20-Apr-2023	10:04:56	19.56	0.79	95	26.4	31.69	34.31	31.94		20-Apr-2023	10:05:06	1

Figure 3.60: Importing data from GPV and FPV spreadsheet

On the “Data Comparison” tab, the line charts were created with data range from “Data Log on Today” tab. The layout of the line chart is as shown in Figure 3.61 to 3.64 below.

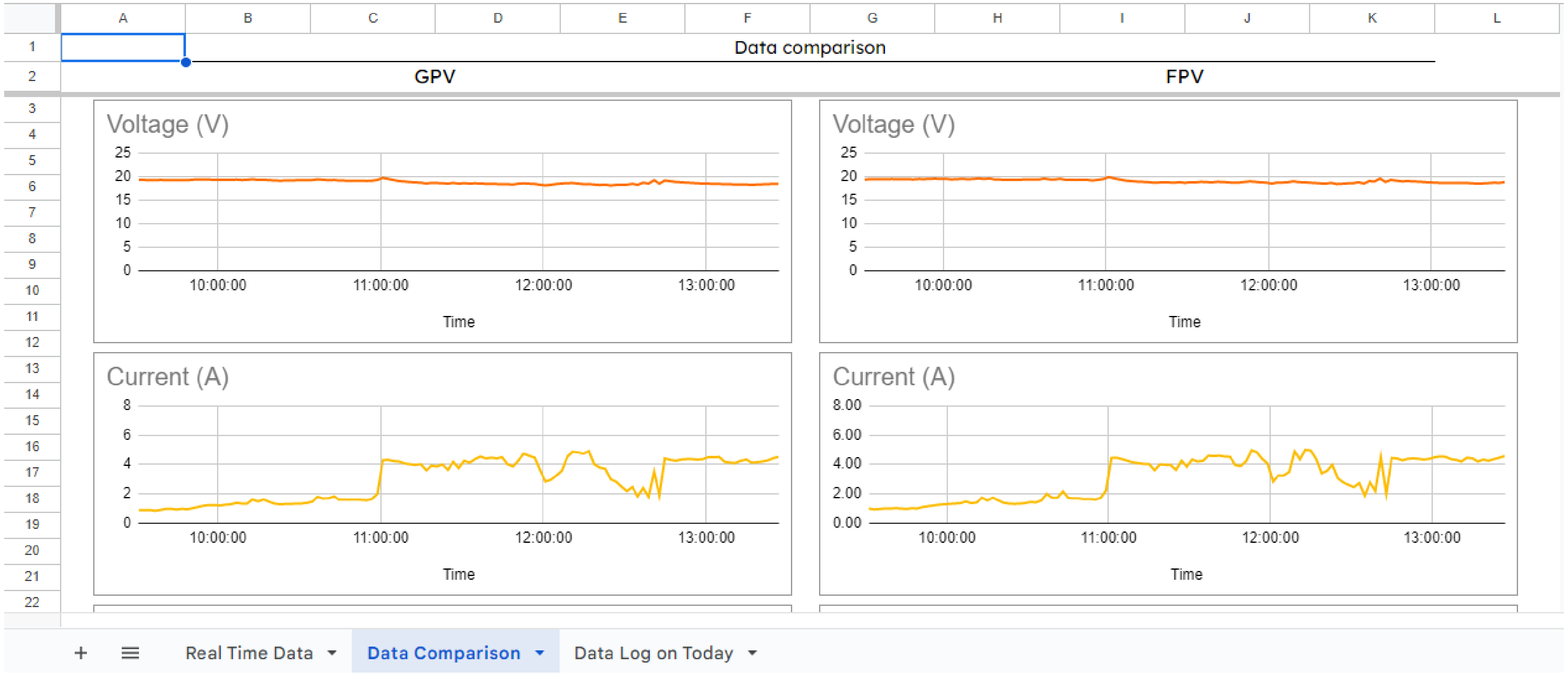


Figure 3.61: Screenshot of line charts in Data Comparison tab (1 of 4)

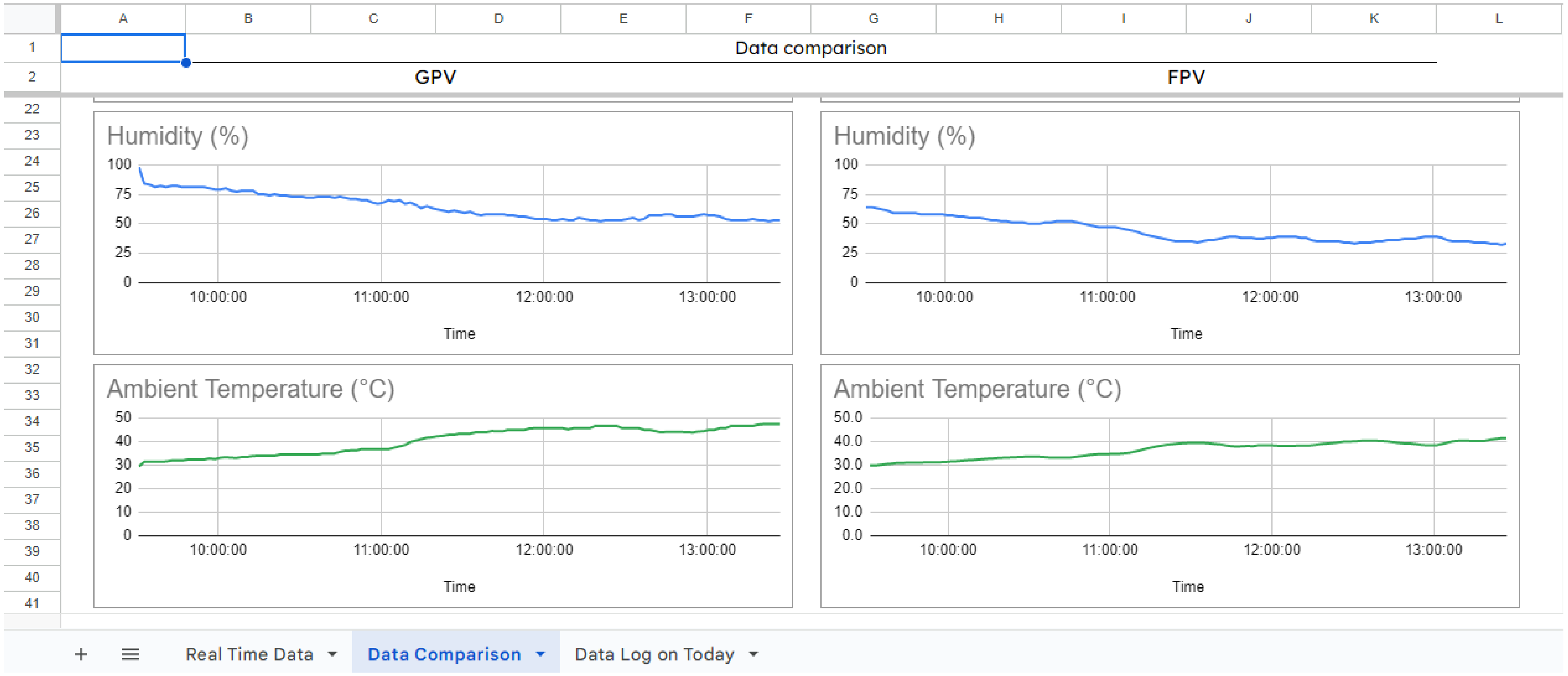


Figure 3.62: Screenshot of line charts in Data Comparison tab (2 of 4)

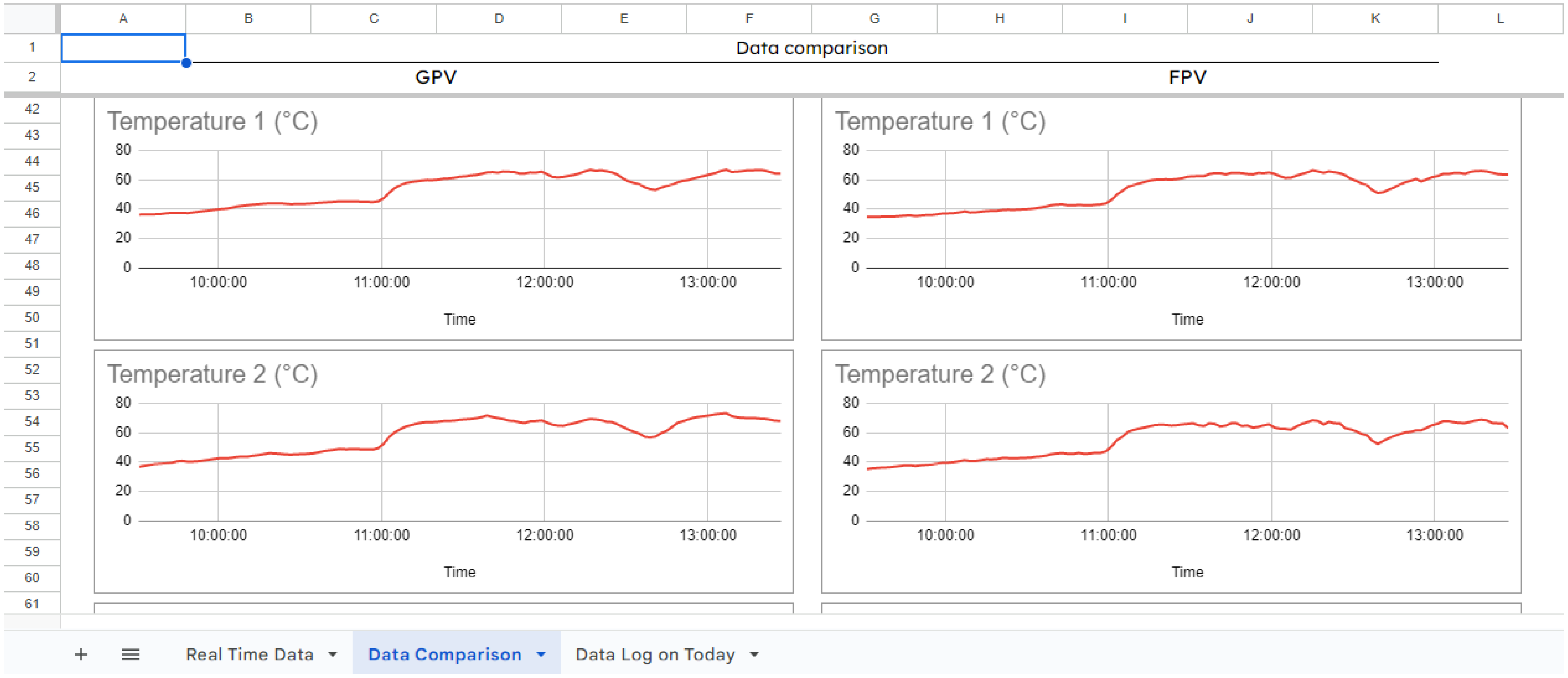


Figure 3.63: Screenshot of line charts in Data Comparison tab (3 of 4)

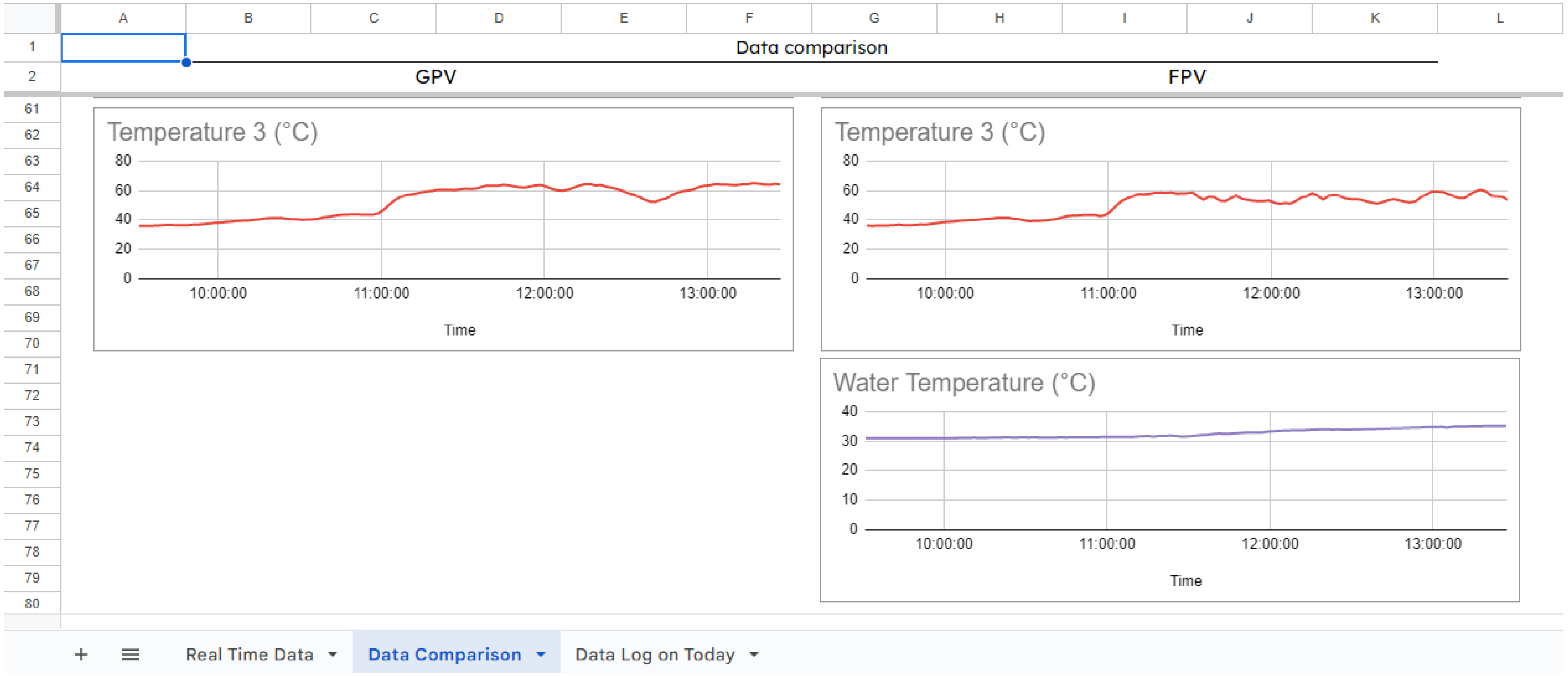
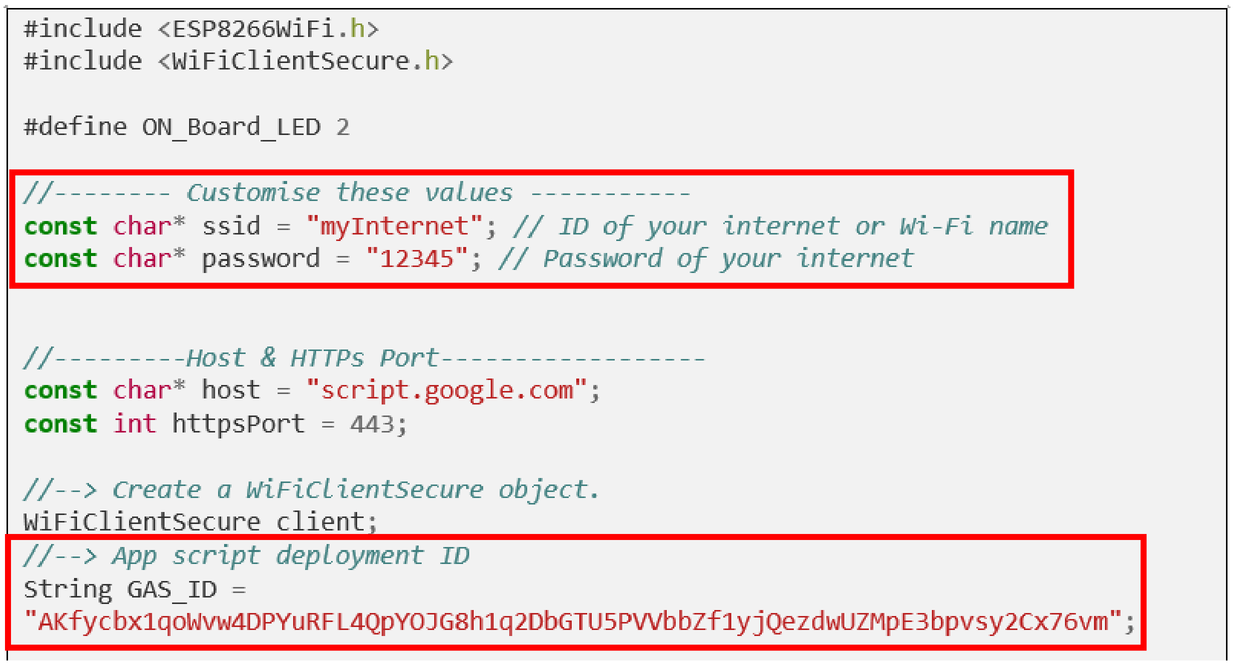


Figure 3.64: Screenshot of line charts in Data Comparison tab (4 of 4)

3.8.4 Configuration of NodeMCU

NodeMCU has to be programmed in Arduino IDE software to connect to the desired WI-FI or Internet. Figure 3.65 below shows how WI-FI name and password added in the code. Next, user needs to add the deployment ID obtained from the Apps script web app deployment. For GPV system, the complete coding for NodeMCU is attached in Appendix I, which user needs to use it to programme NodeMCU in Arduino IDE software. Whereas for FPV system, the complete coding is attached in Appendix J.

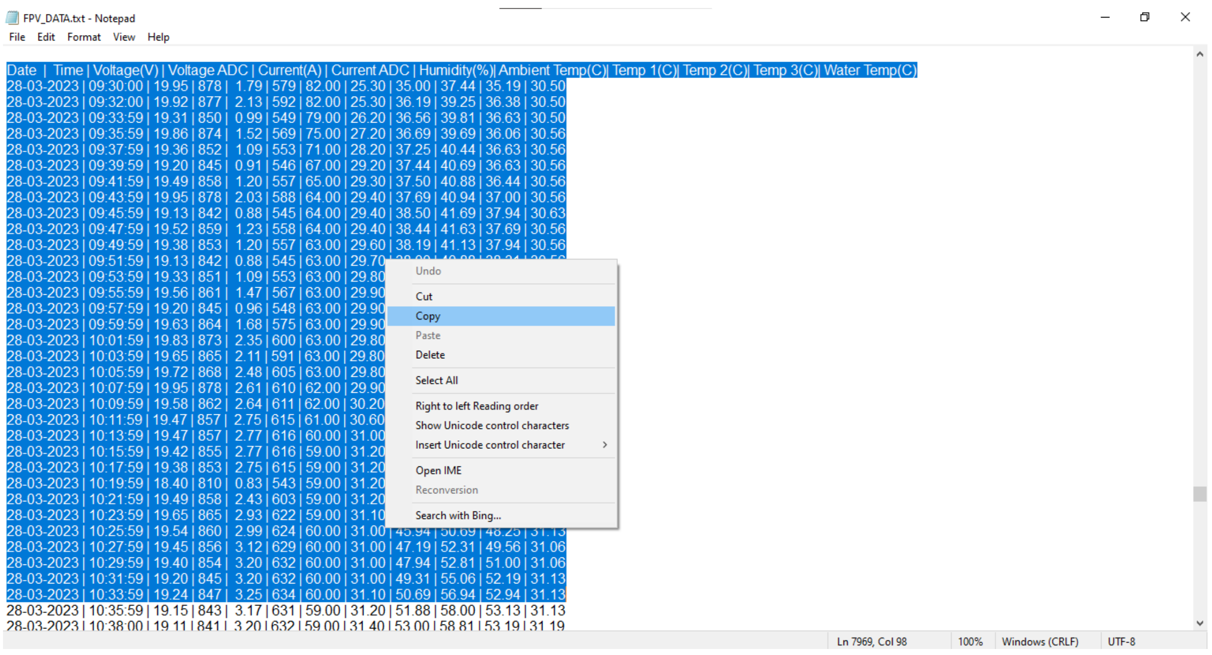


```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

#define ON_Board_LED 2

//-------- Customise these values -----------
const char* ssid = "myInternet"; // ID of your internet or Wi-Fi name
const char* password = "12345"; // Password of your internet


//---------Host & HTTPs Port------------------
const char* host = "script.google.com";
const int httpsPort = 443;

//--> Create a WiFiClientSecure object.
WiFiClientSecure client;
//--> App script deployment ID
String GAS_ID =
"AKfycbx1qoWvw4DPYuRFL4QpYOJG8h1q2DbGTU5PVVbbZf1yjQezdwUZMpE3bpvsy2Cx76vm";
```

Figure 3.65: Locations where WI-FI name, WI-FI password and Apps Script deployment ID are entered

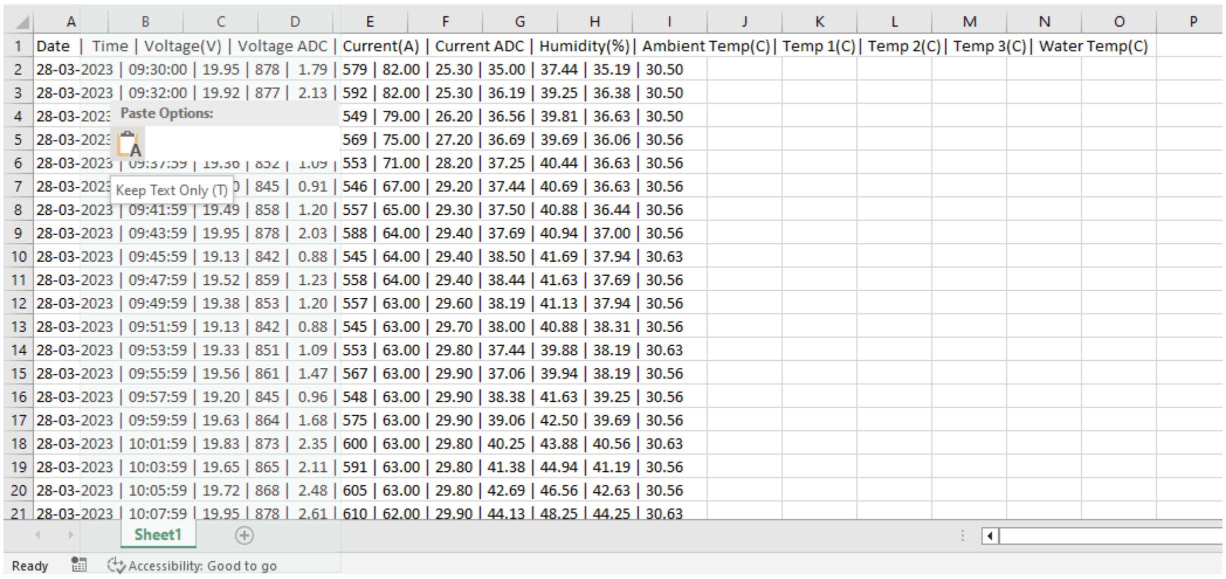
3.9 Conversion of Data Collected on Micro SD Card to Excel

The data collected in micro SD card is in the text file format as shown in Figure 3.66 below. To convert the file to excel format, the data is copied and pasted into an Excel sheet.



```
Date  |  Time | Voltage(V) | Voltage ADC | Current(A) | Current ADC | Humidity(%)| Ambient Temp(C)| Temp 1(C)| Temp 2(C)| Temp 3(C)| Water Temp(C)
28-03-2023 | 09:30:00 | 19.95 | 878 |  1.79 | 579 | 82.00 | 25.30 | 35.00 | 37.44 | 35.19 | 30.50
28-03-2023 | 09:32:00 | 19.92 | 877 |  2.13 | 592 | 82.00 | 25.30 | 36.19 | 39.25 | 36.38 | 30.50
28-03-2023 | 09:33:59 | 19.31 | 850 |  0.99 | 549 | 79.00 | 26.20 | 36.56 | 39.81 | 36.63 | 30.50
28-03-2023 | 09:35:59 | 19.86 | 874 |  1.52 | 569 | 75.00 | 27.20 | 36.69 | 39.69 | 36.06 | 30.56
28-03-2023 | 09:37:59 | 19.36 | 852 |  1.09 | 553 | 71.00 | 28.20 | 37.25 | 40.44 | 36.63 | 30.56
28-03-2023 | 09:39:59 | 19.20 | 845 |  0.91 | 546 | 67.00 | 29.20 | 37.44 | 40.69 | 36.63 | 30.56
28-03-2023 | 09:41:59 | 19.49 | 858 |  1.20 | 557 | 65.00 | 29.30 | 37.50 | 40.88 | 36.44 | 30.56
28-03-2023 | 09:43:59 | 19.95 | 878 |  2.03 | 588 | 64.00 | 29.40 | 37.69 | 40.94 | 37.00 | 30.56
28-03-2023 | 09:45:59 | 19.13 | 842 |  0.88 | 545 | 64.00 | 29.40 | 38.50 | 41.69 | 37.94 | 30.63
28-03-2023 | 09:47:59 | 19.52 | 859 |  1.23 | 558 | 64.00 | 29.40 | 38.44 | 41.63 | 37.69 | 30.56
28-03-2023 | 09:49:59 | 19.38 | 853 |  1.20 | 557 | 63.00 | 29.60 | 38.19 | 41.13 | 37.94 | 30.56
28-03-2023 | 09:51:59 | 19.13 | 842 |  0.88 | 545 | 63.00 | 29.70
28-03-2023 | 09:53:59 | 19.33 | 851 |  1.09 | 553 | 63.00 | 29.80
28-03-2023 | 09:55:59 | 19.56 | 861 |  1.47 | 567 | 63.00 | 29.90
28-03-2023 | 09:57:59 | 19.20 | 845 |  0.96 | 548 | 63.00 | 29.90
28-03-2023 | 09:59:59 | 19.63 | 864 |  1.68 | 575 | 63.00 | 29.90
28-03-2023 | 10:01:59 | 19.83 | 873 |  2.35 | 600 | 63.00 | 29.80
28-03-2023 | 10:03:59 | 19.65 | 865 |  2.11 | 591 | 63.00 | 29.80
28-03-2023 | 10:05:59 | 19.72 | 868 |  2.48 | 605 | 63.00 | 29.80
28-03-2023 | 10:07:59 | 19.95 | 878 |  2.61 | 610 | 62.00 | 29.90
28-03-2023 | 10:09:59 | 19.58 | 862 |  2.64 | 611 | 62.00 | 30.20
28-03-2023 | 10:11:59 | 19.47 | 857 |  2.75 | 615 | 61.00 | 30.60
28-03-2023 | 10:13:59 | 19.47 | 857 |  2.77 | 616 | 60.00 | 31.00
28-03-2023 | 10:15:59 | 19.42 | 855 |  2.77 | 616 | 59.00 | 31.20
28-03-2023 | 10:17:59 | 19.38 | 853 |  2.75 | 615 | 59.00 | 31.20
28-03-2023 | 10:19:59 | 18.40 | 810 |  0.83 | 543 | 59.00 | 31.20
28-03-2023 | 10:21:59 | 19.49 | 858 |  2.43 | 603 | 59.00 | 31.20
28-03-2023 | 10:23:59 | 19.65 | 865 |  2.93 | 622 | 59.00 | 31.10
28-03-2023 | 10:25:59 | 19.54 | 860 |  2.99 | 624 | 60.00 | 31.00
28-03-2023 | 10:27:59 | 19.45 | 856 |  3.12 | 629 | 60.00 | 31.00 | 47.19 | 52.31 | 49.56 | 31.06
28-03-2023 | 10:29:59 | 19.40 | 854 |  3.20 | 632 | 60.00 | 31.00 | 47.94 | 52.81 | 51.00 | 31.06
28-03-2023 | 10:31:59 | 19.20 | 845 |  3.20 | 632 | 60.00 | 31.00 | 49.31 | 55.06 | 52.19 | 31.13
28-03-2023 | 10:33:59 | 19.24 | 847 |  3.25 | 634 | 60.00 | 31.10 | 50.69 | 56.94 | 52.94 | 31.13
28-03-2023 | 10:35:59 | 19.15 | 843 |  3.17 | 631 | 59.00 | 31.20 | 51.88 | 58.00 | 53.13 | 31.13
28-03-2023 | 10:38:00 | 19.11 | 841 |  3.20 | 632 | 59.00 | 31.40 | 53.00 | 58.81 | 53.19 | 31.19
```

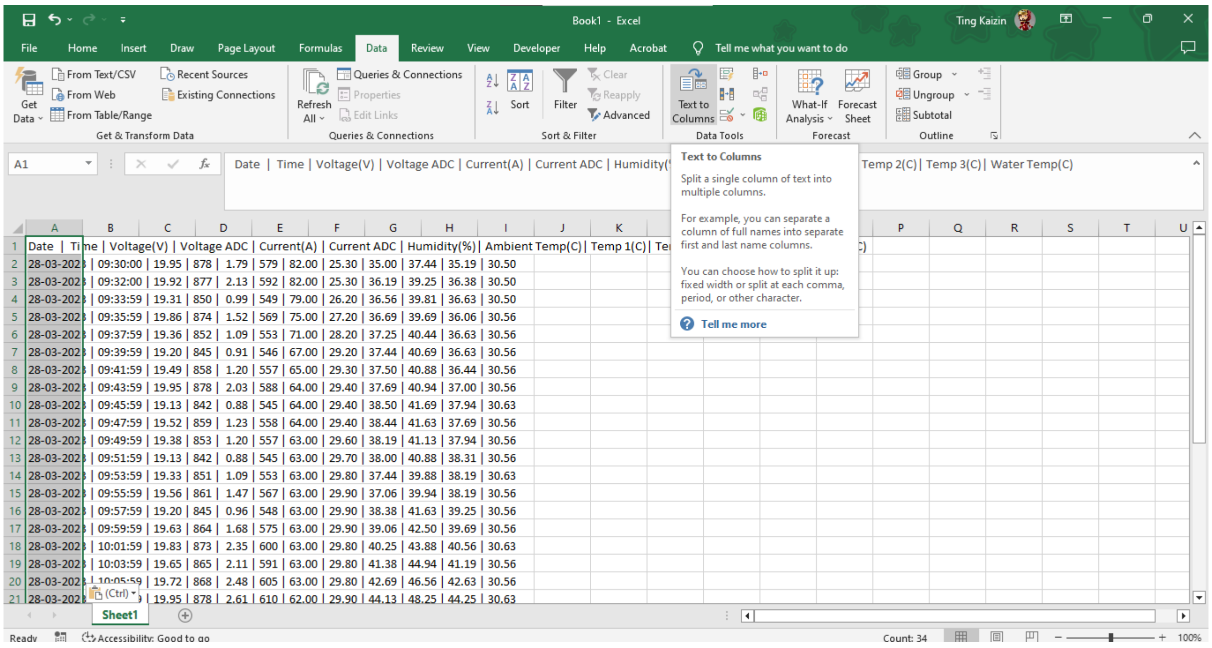
Figure 3.66: Copying data from text file



```
1   Date  |  Time | Voltage(V) | Voltage ADC | Current(A) | Current ADC | Humidity(%)| Ambient Temp(C)| Temp 1(C)| Temp 2(C)| Temp 3(C)| Water Temp(C)
2   28-03-2023 | 09:30:00 | 19.95 | 878 |  1.79 | 579 | 82.00 | 25.30 | 35.00 | 37.44 | 35.19 | 30.50
3   28-03-2023 | 09:32:00 | 19.92 | 877 |  2.13 | 592 | 82.00 | 25.30 | 36.19 | 39.25 | 36.38 | 30.50
4   28-03-202                                    549 | 79.00 | 26.20 | 36.56 | 39.81 | 36.63 | 30.50
5   28-03-202                                    569 | 75.00 | 27.20 | 36.69 | 39.69 | 36.06 | 30.56
6   28-03-2023 | 09:37:59 | 19.36 | 852 |  1.09 | 553 | 71.00 | 28.20 | 37.25 | 40.44 | 36.63 | 30.56
7   28-03-202                    | 845 |  0.91 | 546 | 67.00 | 29.20 | 37.44 | 40.69 | 36.63 | 30.56
8   28-03-2023 | 09:41:59 | 19.49 | 858 |  1.20 | 557 | 65.00 | 29.30 | 37.50 | 40.88 | 36.44 | 30.56
9   28-03-2023 | 09:43:59 | 19.95 | 878 |  2.03 | 588 | 64.00 | 29.40 | 37.69 | 40.94 | 37.00 | 30.56
10  28-03-2023 | 09:45:59 | 19.13 | 842 |  0.88 | 545 | 64.00 | 29.40 | 38.50 | 41.69 | 37.94 | 30.63
11  28-03-2023 | 09:47:59 | 19.52 | 859 |  1.23 | 558 | 64.00 | 29.40 | 38.44 | 41.63 | 37.69 | 30.56
12  28-03-2023 | 09:49:59 | 19.38 | 853 |  1.20 | 557 | 63.00 | 29.60 | 38.19 | 41.13 | 37.94 | 30.56
13  28-03-2023 | 09:51:59 | 19.13 | 842 |  0.88 | 545 | 63.00 | 29.70 | 38.00 | 40.88 | 38.31 | 30.56
14  28-03-2023 | 09:53:59 | 19.33 | 851 |  1.09 | 553 | 63.00 | 29.80 | 37.44 | 39.88 | 38.19 | 30.63
15  28-03-2023 | 09:55:59 | 19.56 | 861 |  1.47 | 567 | 63.00 | 29.90 | 37.06 | 39.94 | 38.19 | 30.56
16  28-03-2023 | 09:57:59 | 19.20 | 845 |  0.96 | 548 | 63.00 | 29.90 | 38.38 | 41.63 | 39.25 | 30.56
17  28-03-2023 | 09:59:59 | 19.63 | 864 |  1.68 | 575 | 63.00 | 29.90 | 39.06 | 42.50 | 39.69 | 30.56
18  28-03-2023 | 10:01:59 | 19.83 | 873 |  2.35 | 600 | 63.00 | 29.80 | 40.25 | 43.88 | 40.56 | 30.63
19  28-03-2023 | 10:03:59 | 19.65 | 865 |  2.11 | 591 | 63.00 | 29.80 | 41.38 | 44.94 | 41.19 | 30.56
20  28-03-2023 | 10:05:59 | 19.72 | 868 |  2.48 | 605 | 63.00 | 29.80 | 42.69 | 46.56 | 42.63 | 30.56
21  28-03-2023 | 10:07:59 | 19.95 | 878 |  2.61 | 610 | 62.00 | 29.90 | 44.13 | 48.25 | 44.25 | 30.63
```

Figure 3.67: Pasting data in Excel sheet

The data copied are all contained in a single column. To separate the data into different columns, user needs to go to Data tab and select Text to Columns.



Date | Time | Voltage(V) | Voltage ADC | Current(A) | Current ADC | Humidity(% … Temp 2(C) | Temp 3(C) | Water Temp(C)

Text to Columns

Split a single column of text into multiple columns.

For example, you can separate a column of full names into separate first and last name columns.

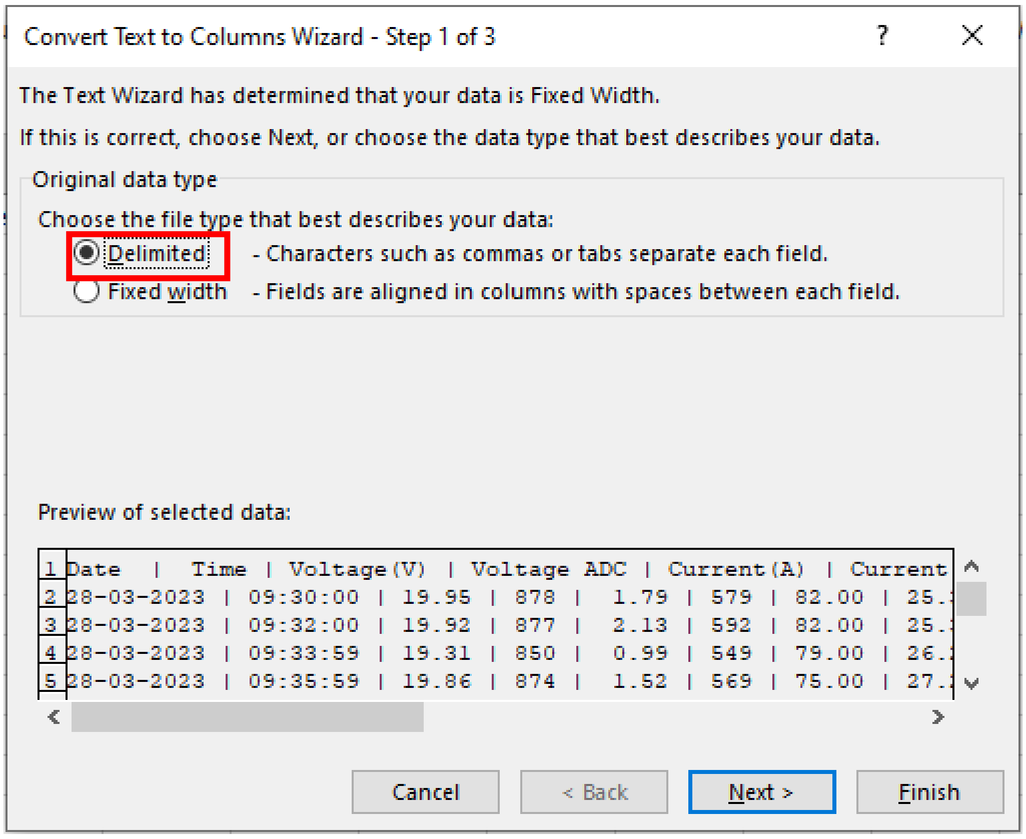
You can choose how to split it up: fixed width or split at each comma, period, or other character.

Tell me more

```
1  Date  | Time | Voltage(V) | Voltage ADC | Current(A) | Current ADC | Humidity(%) | Ambient Temp(C) | Temp 1(C) | Te
2  28-03-2023 | 09:30:00 | 19.95 | 878 | 1.79 | 579 | 82.00 | 25.30 | 35.00 | 37.44 | 35.19 | 30.50
3  28-03-2023 | 09:32:00 | 19.92 | 877 | 2.13 | 592 | 82.00 | 25.30 | 36.19 | 39.25 | 36.38 | 30.50
4  28-03-2023 | 09:33:59 | 19.31 | 850 | 0.99 | 549 | 79.00 | 26.20 | 36.56 | 39.81 | 36.63 | 30.50
5  28-03-2023 | 09:35:59 | 19.86 | 874 | 1.52 | 569 | 75.00 | 27.20 | 36.69 | 39.69 | 36.06 | 30.56
6  28-03-2023 | 09:37:59 | 19.36 | 852 | 1.09 | 553 | 71.00 | 28.20 | 37.25 | 40.44 | 36.63 | 30.56
7  28-03-2023 | 09:39:59 | 19.20 | 845 | 0.91 | 546 | 67.00 | 29.20 | 37.44 | 40.69 | 36.63 | 30.56
8  28-03-2023 | 09:41:59 | 19.49 | 858 | 1.20 | 557 | 65.00 | 29.30 | 37.50 | 40.88 | 36.44 | 30.56
9  28-03-2023 | 09:43:59 | 19.95 | 878 | 2.03 | 588 | 64.00 | 29.40 | 37.69 | 40.94 | 37.00 | 30.56
10 28-03-2023 | 09:45:59 | 19.13 | 842 | 0.88 | 545 | 64.00 | 29.40 | 38.50 | 41.69 | 37.94 | 30.63
11 28-03-2023 | 09:47:59 | 19.52 | 859 | 1.23 | 558 | 64.00 | 29.40 | 38.44 | 41.63 | 37.69 | 30.56
12 28-03-2023 | 09:49:59 | 19.38 | 853 | 1.20 | 557 | 63.00 | 29.60 | 38.19 | 41.13 | 37.94 | 30.56
13 28-03-2023 | 09:51:59 | 19.13 | 842 | 0.88 | 545 | 63.00 | 29.70 | 38.00 | 40.88 | 38.31 | 30.56
14 28-03-2023 | 09:53:59 | 19.33 | 851 | 1.09 | 553 | 63.00 | 29.80 | 37.44 | 39.88 | 38.19 | 30.63
15 28-03-2023 | 09:55:59 | 19.56 | 861 | 1.47 | 567 | 63.00 | 29.90 | 37.06 | 39.94 | 38.19 | 30.56
16 28-03-2023 | 09:57:59 | 19.20 | 845 | 0.96 | 548 | 63.00 | 29.90 | 38.38 | 41.63 | 39.25 | 30.56
17 28-03-2023 | 09:59:59 | 19.63 | 864 | 1.68 | 575 | 63.00 | 29.90 | 39.06 | 42.50 | 39.69 | 30.56
18 28-03-2023 | 10:01:59 | 19.83 | 873 | 2.35 | 600 | 63.00 | 29.80 | 40.25 | 43.88 | 40.56 | 30.63
19 28-03-2023 | 10:03:59 | 19.65 | 865 | 2.11 | 591 | 63.00 | 29.80 | 41.38 | 44.94 | 41.19 | 30.56
20 28-03-2023 | 10:05:59 | 19.72 | 868 | 2.48 | 605 | 63.00 | 29.80 | 42.69 | 46.56 | 42.63 | 30.56
21 28-03-2023 | [illegible] | 19.95 | 878 | 2.61 | 610 | 62.00 | 29.90 | 44.13 | 48.25 | 44.25 | 30.63
```

Figure 3.68: Selecting Text to Columns

Then, select Delimited from the pop-out box. Click Next and select Other for Delimiters. Type “|” symbol beside the box of Other.


Convert Text to Columns Wizard - Step 1 of 3

The Text Wizard has determined that your data is Fixed Width.

If this is correct, choose Next, or choose the data type that best describes your data.

Original data type

Choose the file type that best describes your data:

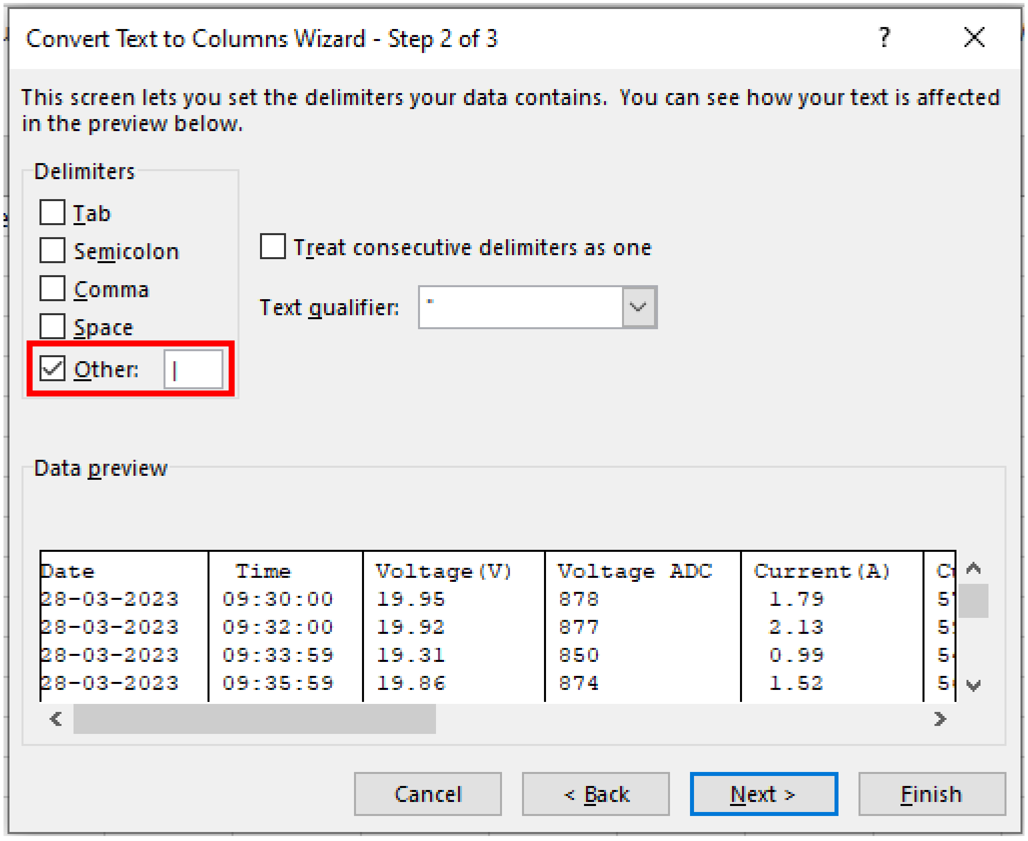
[✓ Delimited] - Characters such as commas or tabs separate each field.

Fixed width - Fields are aligned in columns with spaces between each field.

Preview of selected data:

```
1 Date       |   Time   | Voltage(V)  | Voltage ADC  | Current(A)  | Current
2 28-03-2023 | 09:30:00 | 19.95 | 878 |   1.79 | 579 | 82.00 | 25.
3 28-03-2023 | 09:32:00 | 19.92 | 877 |   2.13 | 592 | 82.00 | 25.
4 28-03-2023 | 09:33:59 | 19.31 | 850 |   0.99 | 549 | 79.00 | 26.
5 28-03-2023 | 09:35:59 | 19.86 | 874 |   1.52 | 569 | 75.00 | 27.
```

Figure 3.69: Selecting Delimited file type


Convert Text to Columns Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

Tab

Semicolon

Comma

Space

[✓ Other:] |

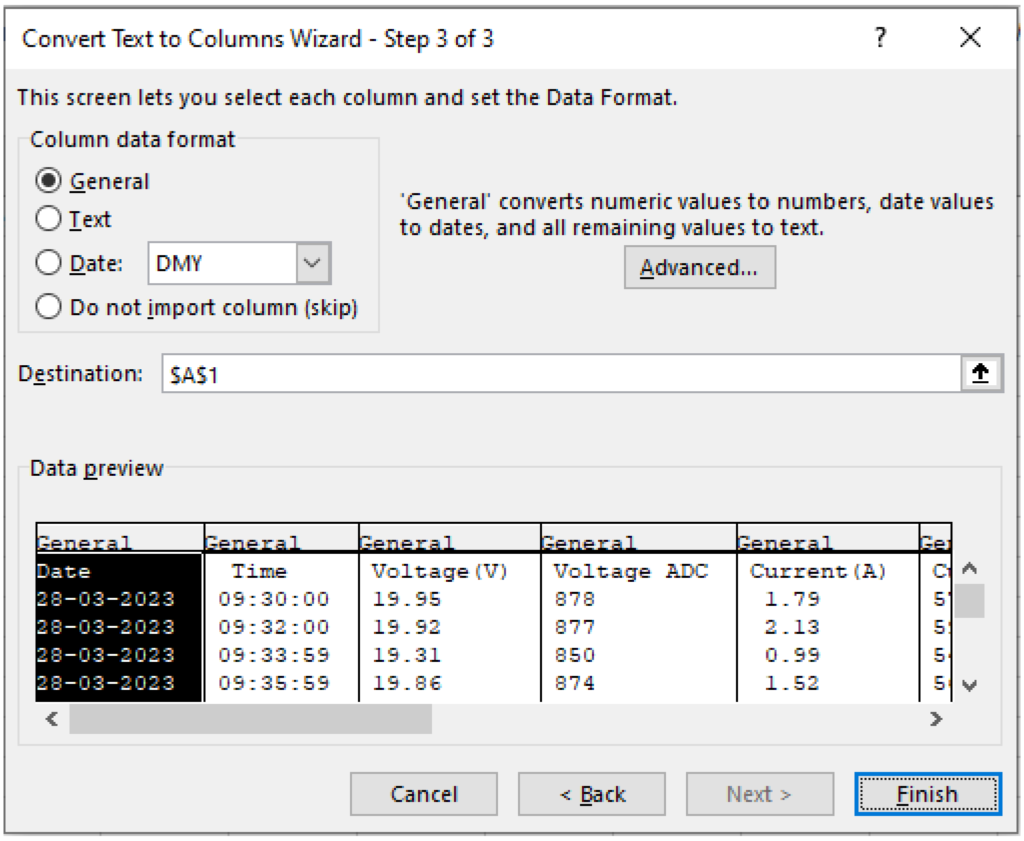
Treat consecutive delimiters as one

Text qualifier: "

Data preview

```
Date         Time      Voltage(V)   Voltage ADC   Current(A)   C
28-03-2023   09:30:00  19.95        878             1.79       5
28-03-2023   09:32:00  19.92        877             2.13       5
28-03-2023   09:33:59  19.31        850             0.99       5
28-03-2023   09:35:59  19.86        874             1.52       5
```

Figure 3.70: Selecting Other and insert “|” symbol



Convert Text to Columns Wizard - Step 3 of 3

This screen lets you select each column and set the Data Format.

Column data format

[✓ General] Text Date: DMY Do not import column (skip)

'General' converts numeric values to numbers, date values to dates, and all remaining values to text.

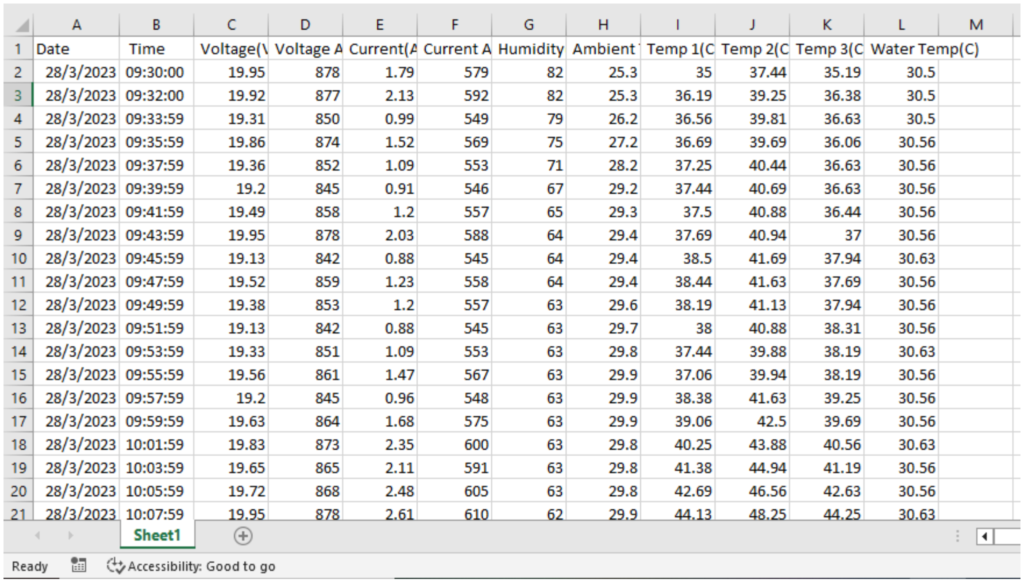
Destination: $A$1

Data preview

```
General     General   General      General       General      Ge
Date        Time      Voltage(V)   Voltage ADC   Current(A)   C
28-03-2023  09:30:00  19.95        878           1.79         5
28-03-2023  09:32:00  19.92        877           2.13         5
28-03-2023  09:33:59  19.31        850           0.99         5
28-03-2023  09:35:59  19.86        874           1.52         5
```

Figure 3.71: Finishing the Text to Columns setting

Click Next and Finished. The data can now be separated into different columns as shown in Figure 3.72.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	Time	Voltage(\	Voltage A	Current(A	Current A	Humidity	Ambient	Temp 1(C	Temp 2(C	Temp 3(C	Water Temp(C)	
2	28/3/2023	09:30:00	19.95	878	1.79	579	82	25.3	35	37.44	35.19	30.5	
3	28/3/2023	09:32:00	19.92	877	2.13	592	82	25.3	36.19	39.25	36.38	30.5	
4	28/3/2023	09:33:59	19.31	850	0.99	549	79	26.2	36.56	39.81	36.63	30.5	
5	28/3/2023	09:35:59	19.86	874	1.52	569	75	27.2	36.69	39.69	36.06	30.56	
6	28/3/2023	09:37:59	19.36	852	1.09	553	71	28.2	37.25	40.44	36.63	30.56	
7	28/3/2023	09:39:59	19.2	845	0.91	546	67	29.2	37.44	40.69	36.63	30.56	
8	28/3/2023	09:41:59	19.49	858	1.2	557	65	29.3	37.5	40.88	36.44	30.56	
9	28/3/2023	09:43:59	19.95	878	2.03	588	64	29.4	37.69	40.94	37	30.56	
10	28/3/2023	09:45:59	19.13	842	0.88	545	64	29.4	38.5	41.69	37.94	30.63	
11	28/3/2023	09:47:59	19.52	859	1.23	558	64	29.4	38.44	41.63	37.69	30.56	
12	28/3/2023	09:49:59	19.38	853	1.2	557	63	29.6	38.19	41.13	37.94	30.56	
13	28/3/2023	09:51:59	19.13	842	0.88	545	63	29.7	38	40.88	38.31	30.56	
14	28/3/2023	09:53:59	19.33	851	1.09	553	63	29.8	37.44	39.88	38.19	30.63	
15	28/3/2023	09:55:59	19.56	861	1.47	567	63	29.9	37.06	39.94	38.19	30.56	
16	28/3/2023	09:57:59	19.2	845	0.96	548	63	29.9	38.38	41.63	39.25	30.56	
17	28/3/2023	09:59:59	19.63	864	1.68	575	63	29.9	39.06	42.5	39.69	30.56	
18	28/3/2023	10:01:59	19.83	873	2.35	600	63	29.8	40.25	43.88	40.56	30.63	
19	28/3/2023	10:03:59	19.65	865	2.11	591	63	29.8	41.38	44.94	41.19	30.56	
20	28/3/2023	10:05:59	19.72	868	2.48	605	63	29.8	42.69	46.56	42.63	30.56	
21	28/3/2023	10:07:59	19.95	878	2.61	610	62	29.9	44.13	48.25	44.25	30.63	

Figure 3.72: Data in text file converted into columns

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the data obtained from the data monitoring system were analysed. The problems encountered and possible root causes leading to the problems were also discussed in the following section.

4.2 Result Analysis

4.2.1 Voltage and Current Calibration Results

The formula for calculating Average Percentage Error is

$$Average\ Percentage\ Error = \frac{\sum_{i=1}^{N}\left\{\frac{|Arduino\ reading - Actual\ Reading|}{Actual\ Reading} \times 100\%\right\}}{N}$$

Whereas the formula for calculating Root Mean Square value is

$$Root\ Mean\ Square\ Error = \sqrt{\frac{\sum_{i=1}^{N}(Arduino\ reading - Actual\ Reading)^2}{N}}$$

where Actual Readings were obtained from IV plotter device.
The analysis result is as shown in Table 4.1 and 4.2 and Figure 4.1 to 4.16 below.

Table 4.1: Voltage and current percentage errors

Arduino system	Voltage measurement		Current measurement	
	Average percentage error before calibration (%)	Average percentage error after calibration (%)	Average percentage error before calibration (%)	Average percentage error after calibration (%)
For Floating PV	8.98	1.06	2.21	1.46
For Ground PV	5.84	0.70	6.18	1.26

Table 4.2: Voltage and current root mean square errors

Arduino system	Voltage measurement		Current measurement	
	Root Mean Square Error before calibration (V)	Root Mean Square Error after calibration (V)	Root Mean Square Error before calibration (A)	Root Mean Square Error after calibration (A)
For Floating PV	1.70	0.21	0.08	0.06
For Ground PV	1.08	0.13	0.21	0.05

The formulae for calibrating the voltage and current values are as shown below. These formulas are added into the coding attached in Appendix G and Appendix H.

Table 4.3: Corrective formulae for calibrating the voltage and current values

Arduino	Coefficient for calibrating voltage value	Coefficient for calibrating current value
For Floating PV	vIN = 0.9176*vIN;	currentValue = 1.0107*currentValue;
For Ground PV	vIN = 0.9448*vIN;	currentValue = 0.9771*currentValue - 0.0946;

IV plotter voltage measurement vs Arduino voltage measurement before calibration (FPV):

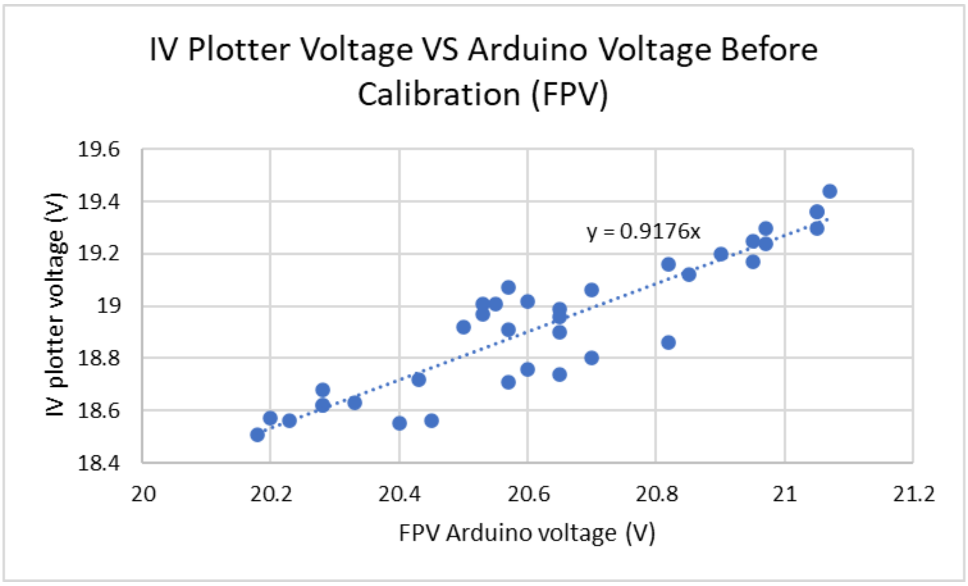


Figure 4.1: IV plotter voltage vs Arduino voltage before calibration (FPV)

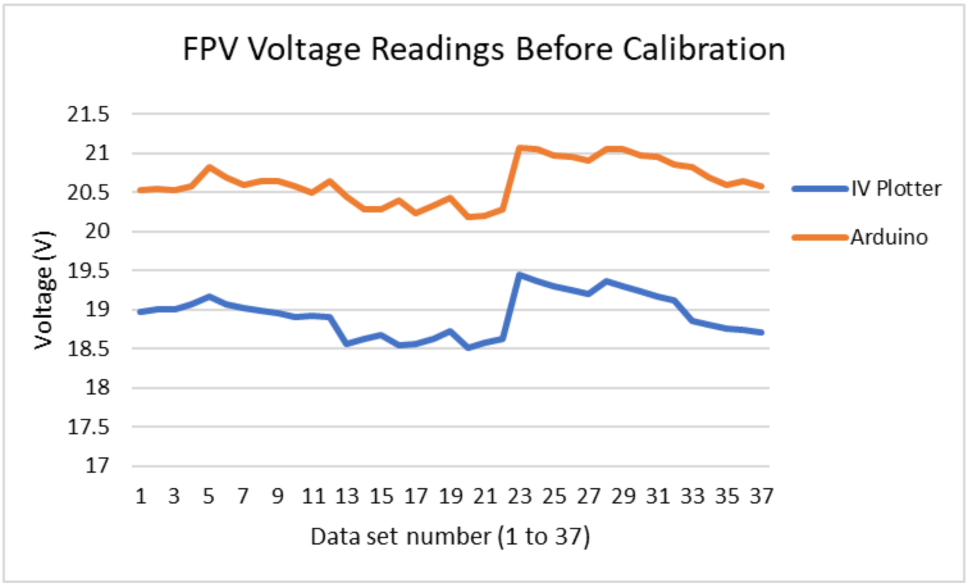


Figure 4.2: 37 sets of IV plotter voltage and Arduino voltage readings comparison before calibration (FPV)

IV plotter voltage measurement vs Arduino voltage measurement after calibration (FPV):

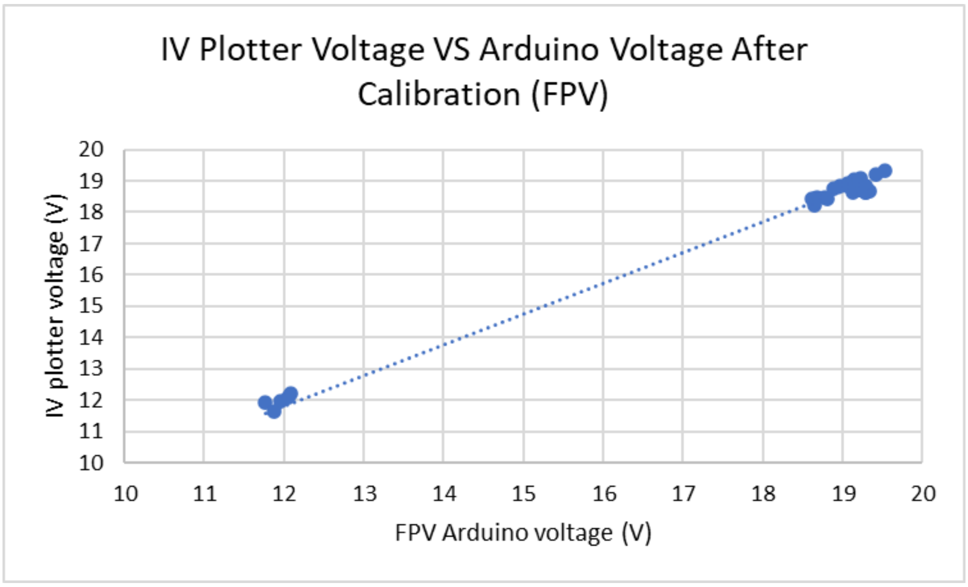


Figure 4.3: IV plotter voltage vs Arduino voltage after calibration (FPV)

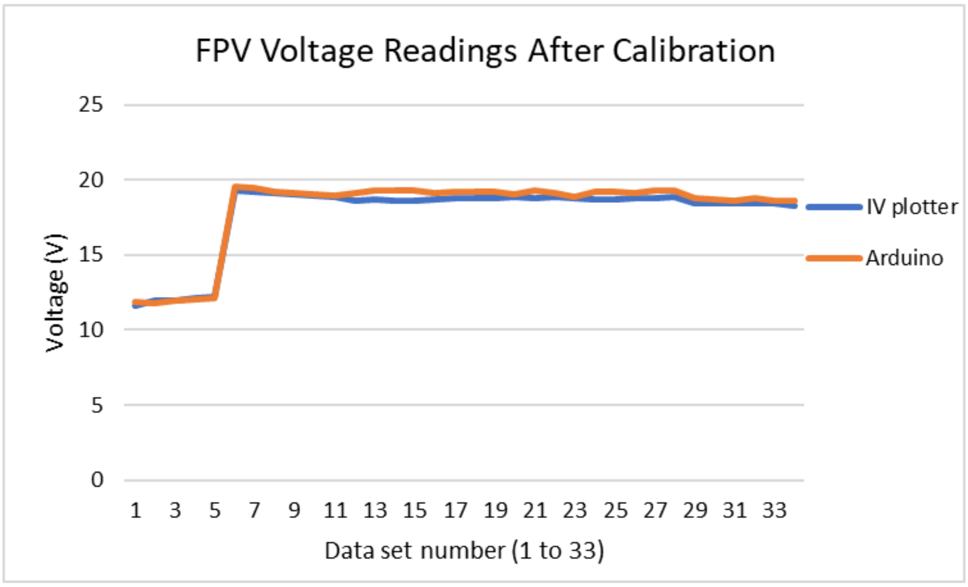


Figure 4.4: 33 sets of IV plotter voltage and Arduino voltage readings comparison (FPV)

IV plotter current measurement vs FPV current measurement before calibration (FPV):

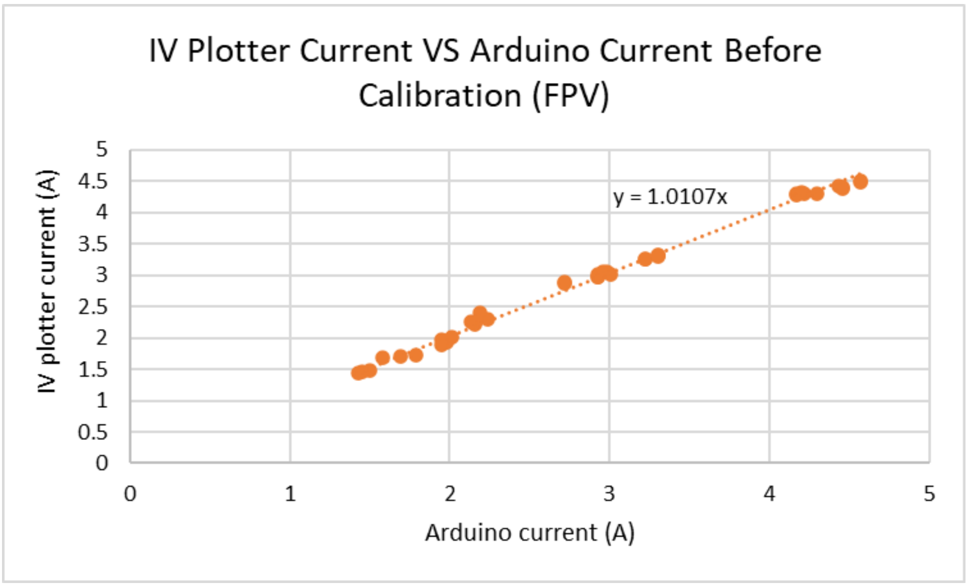


Figure 4.5: IV plotter current vs Arduino current before calibration (FPV)

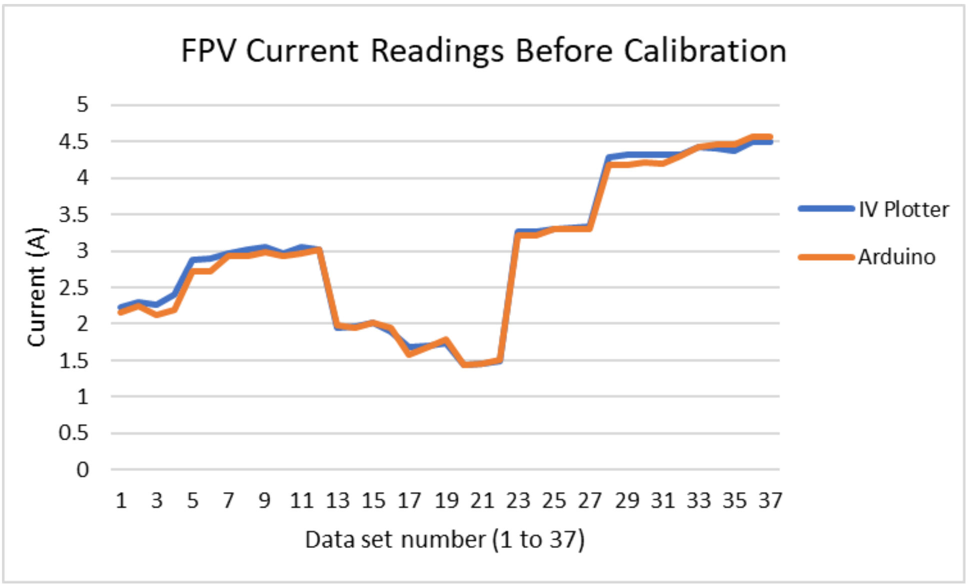


Figure 4.6: 37 sets of IV plotter current and Arduino current readings comparison before calibration (FPV)

IV plotter current measurement vs FPV current measurement after calibration (FPV):

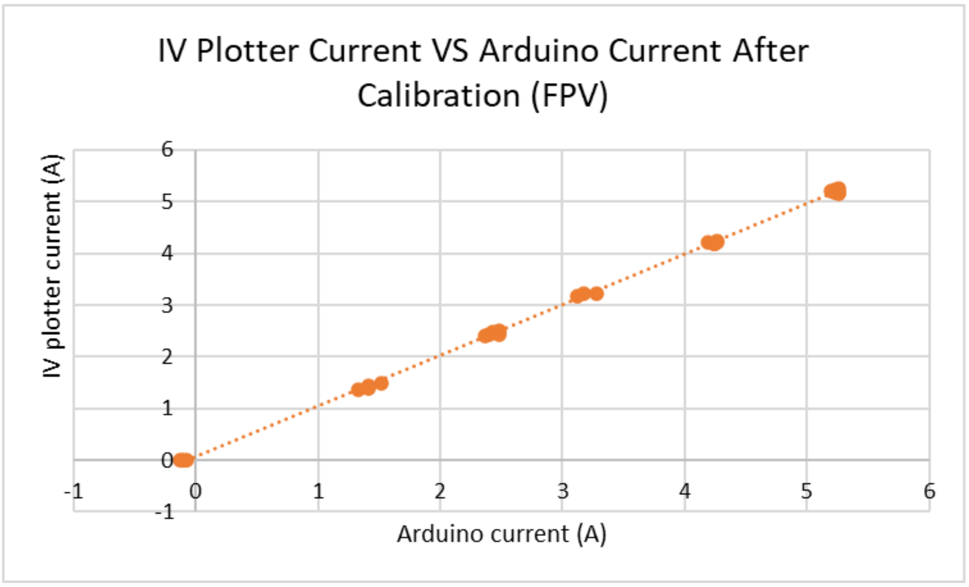


Figure 4.7: IV plotter current vs Arduino current after calibration (FPV)

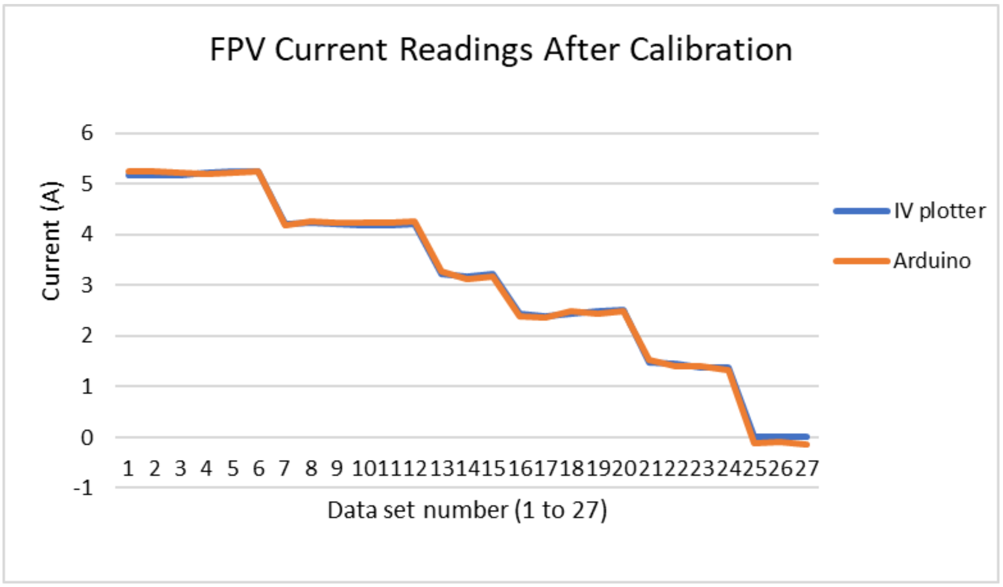


Figure 4.8: 27 sets of IV plotter current and Arduino current readings comparison after calibration (FPV)

IV plotter voltage measurement vs Arduino voltage measurement before calibration (GPV):

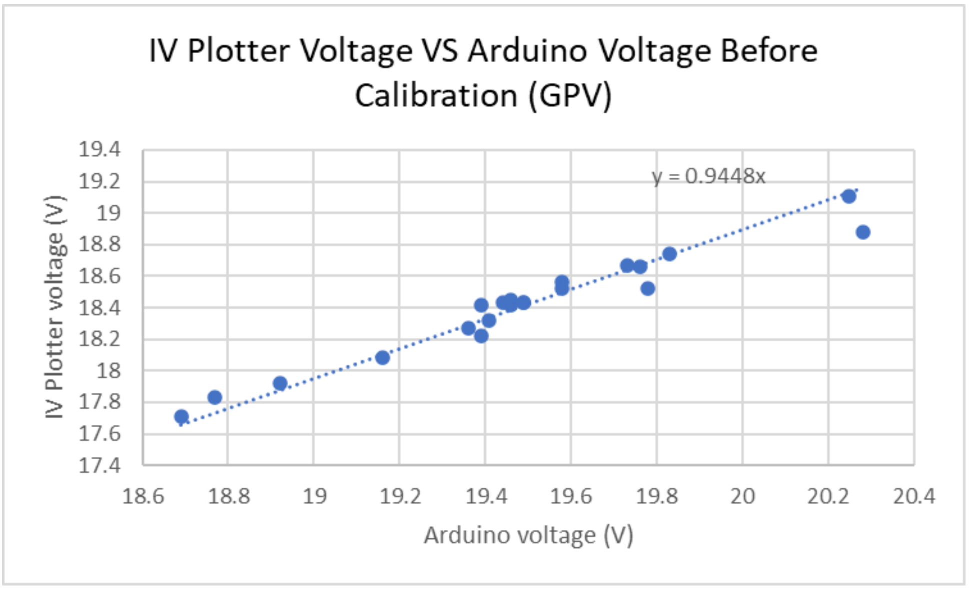


Figure 4.9: IV plotter voltage vs Arduino voltage before calibration (GPV)

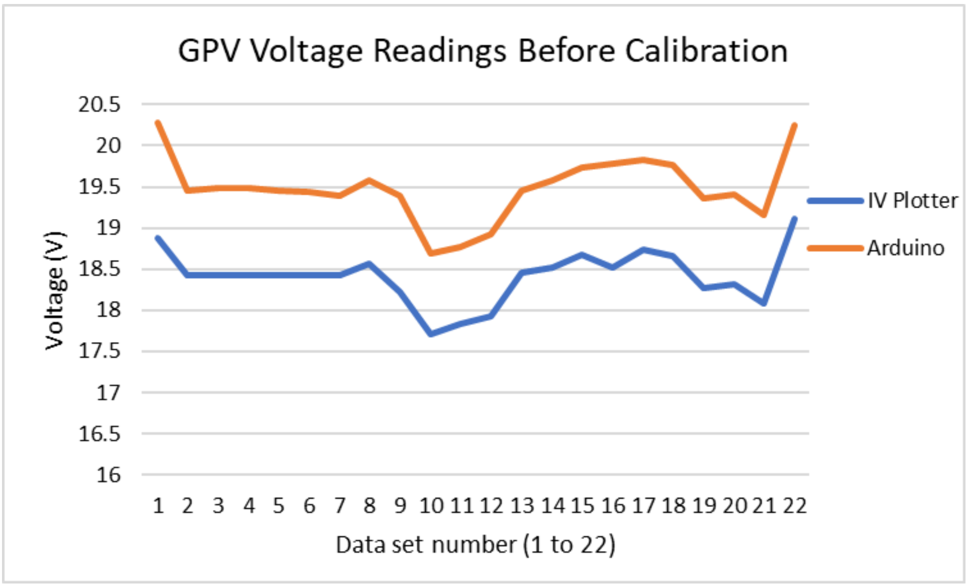


Figure 4.10: 22 sets of IV plotter voltage and Arduino voltage readings comparison before calibration (GPV)

IV plotter voltage measurement vs Arduino voltage measurement after calibration (GPV):

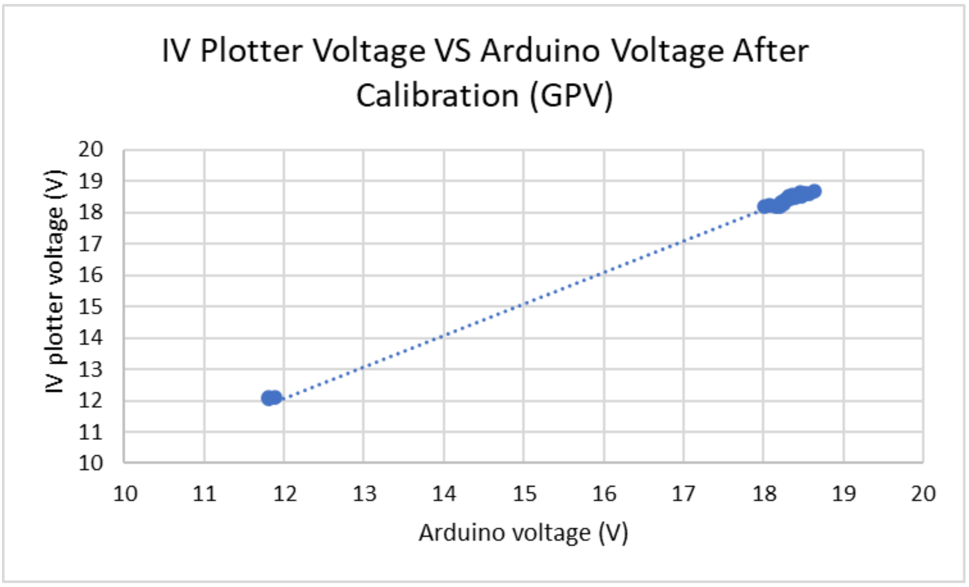


Figure 4.11: IV plotter voltage vs Arduino voltage after calibration (GPV)

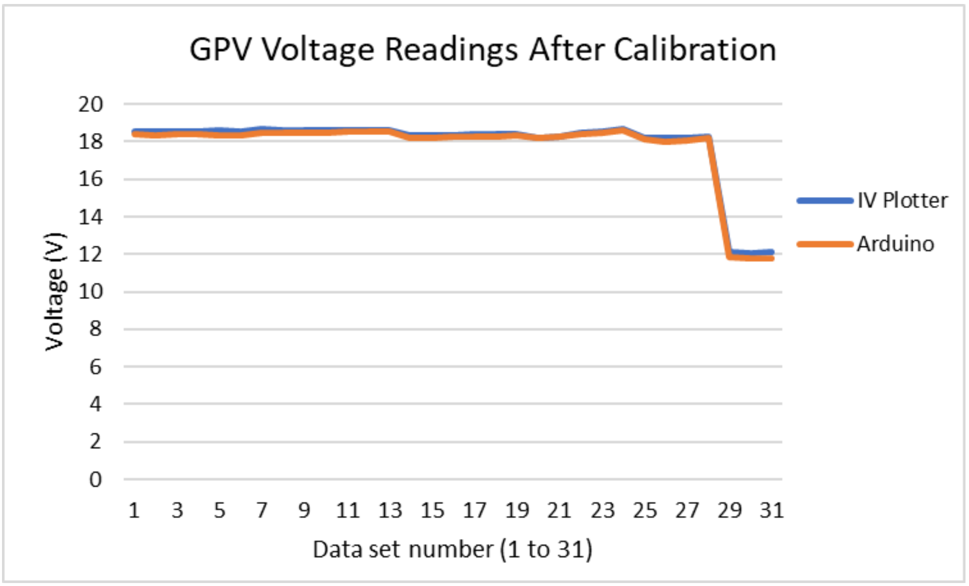


Figure 4.12: 31 sets of IV plotter voltage and Arduino voltage readings comparison after calibration (GPV)

IV plotter current measurement vs Arduino current measurement before calibration (GPV):

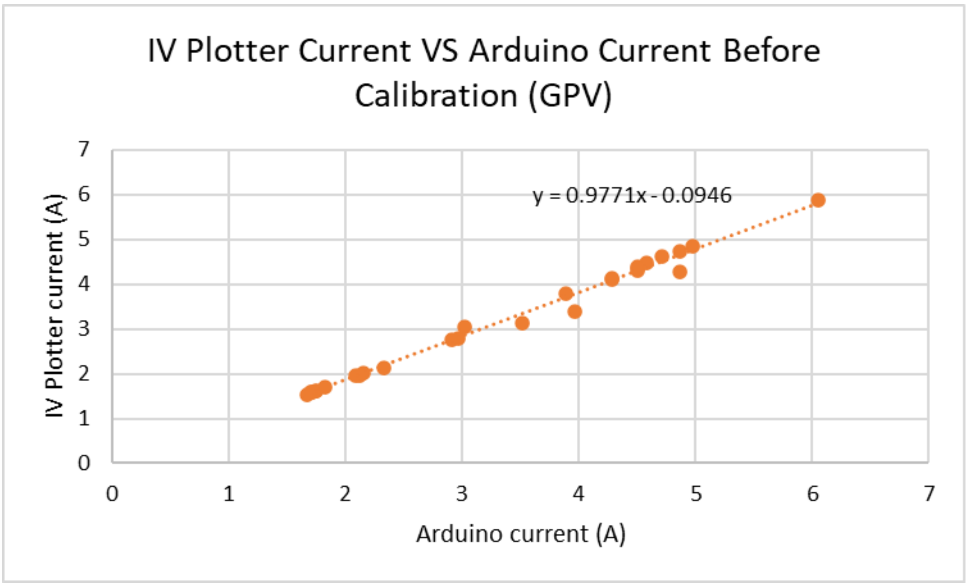


Figure 4.13: IV plotter current vs Arduino current before calibration (GPV)

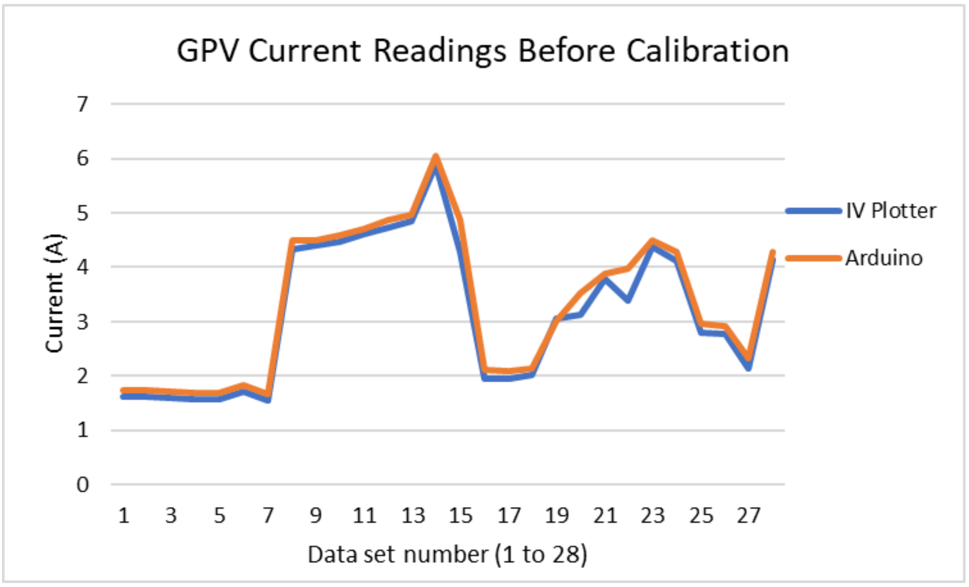


Figure 4.14: 27 sets of IV plotter current and Arduino current readings comparison before calibration (GPV)

IV plotter current measurement vs Arduino current measurement after calibration (GPV):

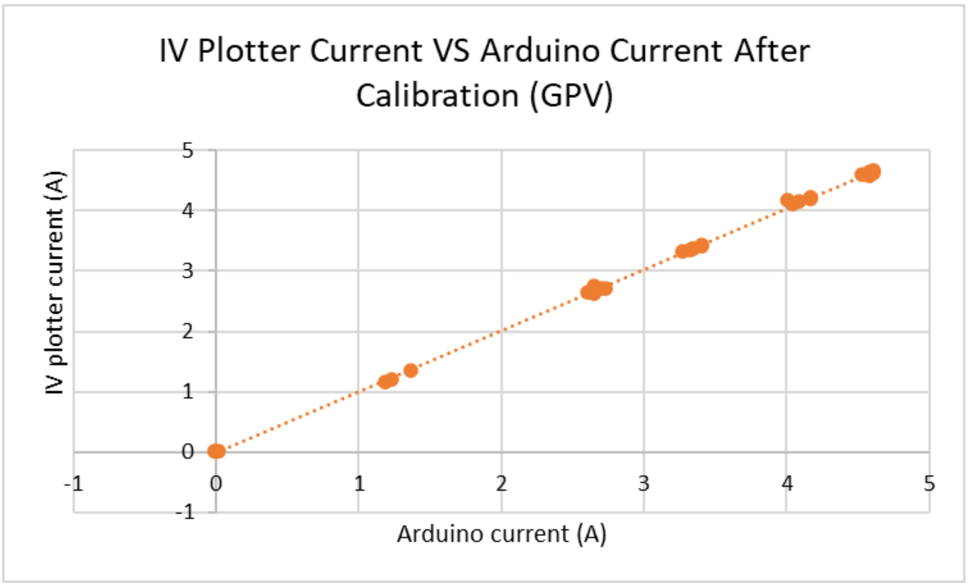


Figure 4.15: IV plotter current vs Arduino current after calibration (GPV)

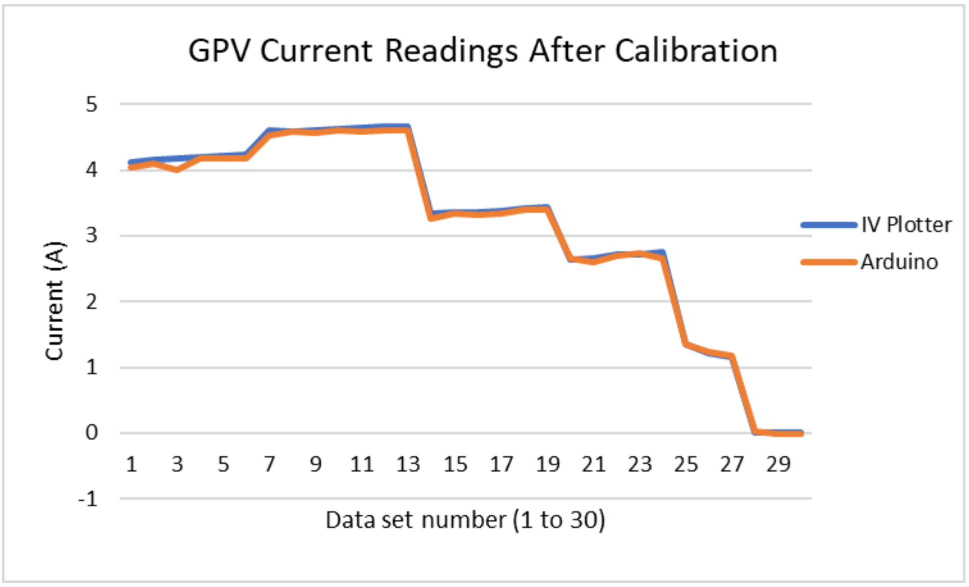


Figure 4.16: 29 sets of IV plotter current and Arduino current readings comparison after calibration (GPV)

4.2.2 DHT11 and DS18B20 Sensors Calibration Results

Table 4.4 below shows the error in degree Celsius of the DS18B20 sensors, with the readings from Fluke's thermometer as reference readings.

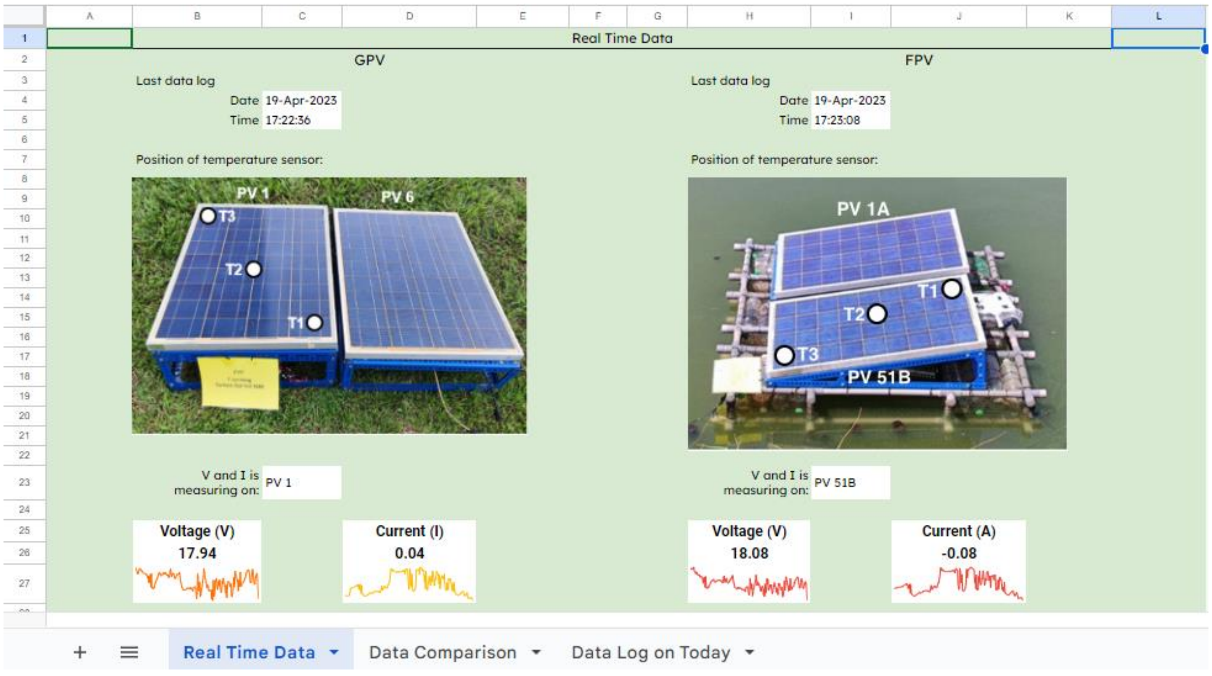
Table 4.4: Temperature sensor accuracy analysis result

Sensor	**From FPV**		**From Ground PV**	
	Average error (℃)	**Percentage error (%)**	**Average error (℃)**	**Percentage error (%)**
DHT11 sensor	0.95	3.08	1.36	4.15
DS18B20 sensor 1	0.07	0.22	0.04	0.13
DS18B20 sensor 2	0.11	0.32	0.24	0.69
DS18B20 sensor 3	0.16	0.46	0.09	0.29
DS18B20 water-proof sensor	0.02	0.07	Not applicable	Not applicable

In general, the DS18B20 sensors are having average error less than 1 ℃ and percentage error of less than 1 %. Therefore, no calibration was done on these sensors. However, both DHT11 sensors were having larger percentage error. The DHT11 sensors were then replaced with new one and the average errors were lowered to less than 1 ℃.

4.2.3 Data Monitoring and Data Collection System

Figure 4.17 to 4.23 below show the data collected from FPV system on Google Spreadsheet on 19 April 2023, from 9.30 am to 5.22 pm.



Real Time Data

GPV

Last data log
Date 19-Apr-2023
Time 17:22:36

Position of temperature sensor:

V and I is measuring on: PV 1

Voltage (V) 17.94

Current (I) 0.04

FPV

Last data log
Date 19-Apr-2023
Time 17:23:08

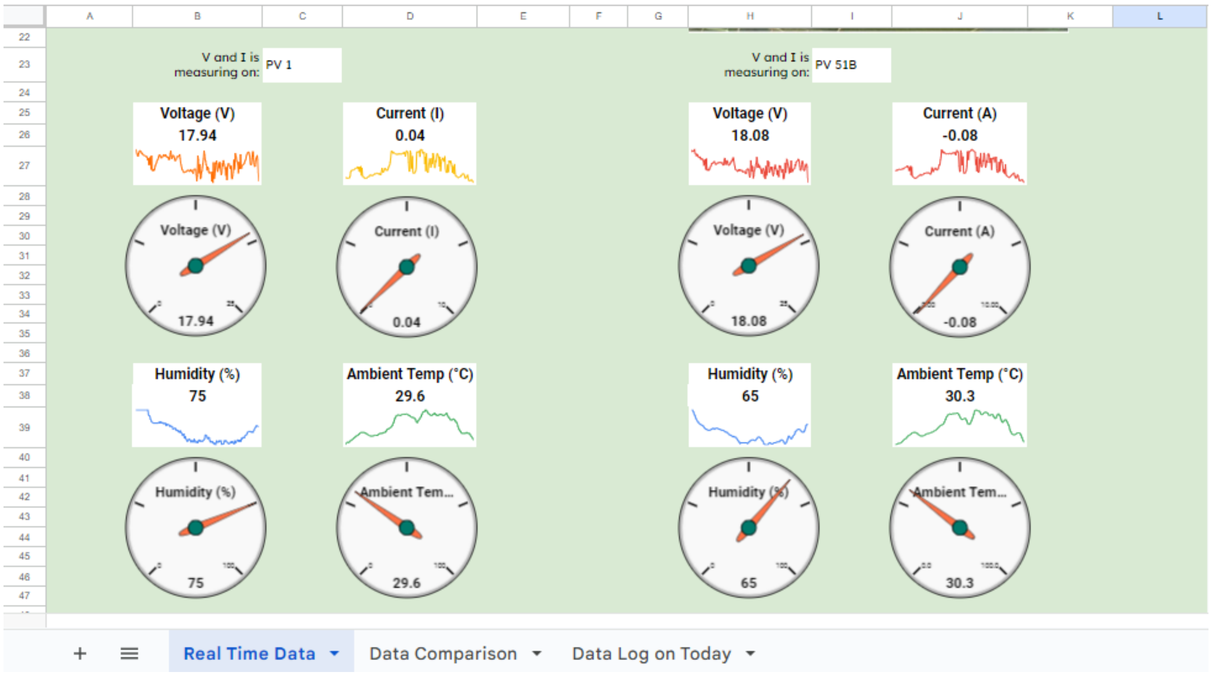
Position of temperature sensor:

V and I is measuring on: PV 51B

Voltage (V) 18.08

Current (A) -0.08

Figure 4.17: Screenshot of FPV data and GPV data on Google Spreadsheet (1 of 3)



V and I is measuring on: PV 1

Voltage (V) 17.94

Current (I) 0.04

Humidity (%) 75

Ambient Temp (°C) 29.6

V and I is measuring on: PV 51B

Voltage (V) 18.08

Current (A) -0.08

Humidity (%) 65

Ambient Temp (°C) 30.3

Figure 4.18: Screenshot of FPV data and GPV data on Google Spreadsheet (2 of 3)

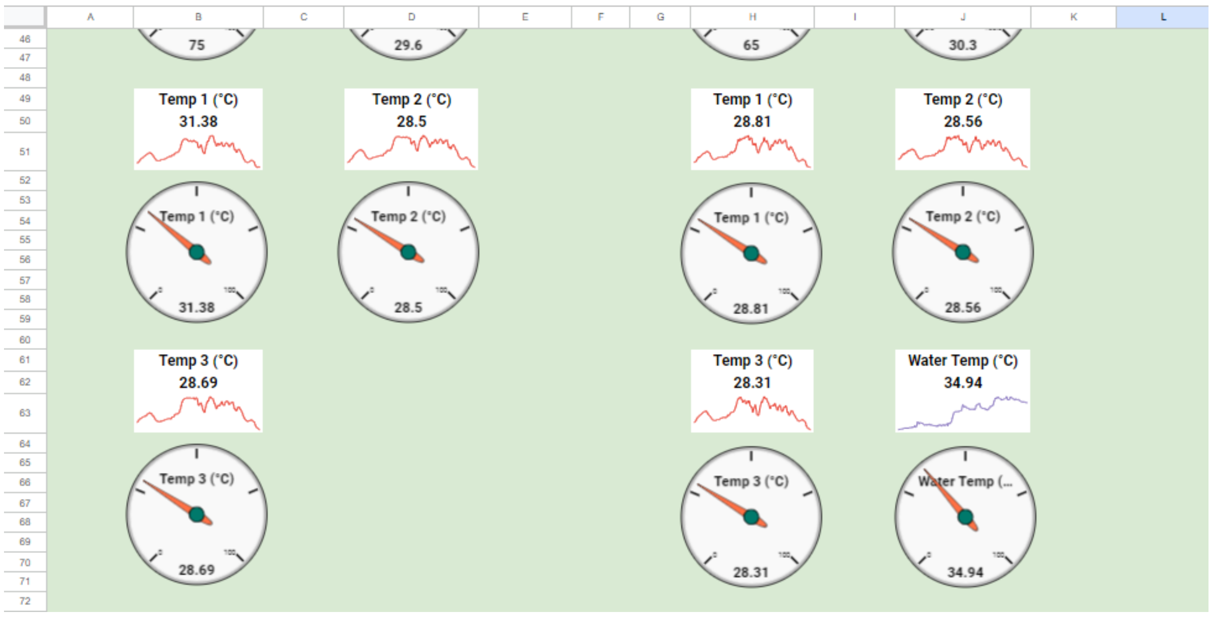


Figure 4.19: Screenshot of FPV data and GPV data on Google Spreadsheet (3 of 3)

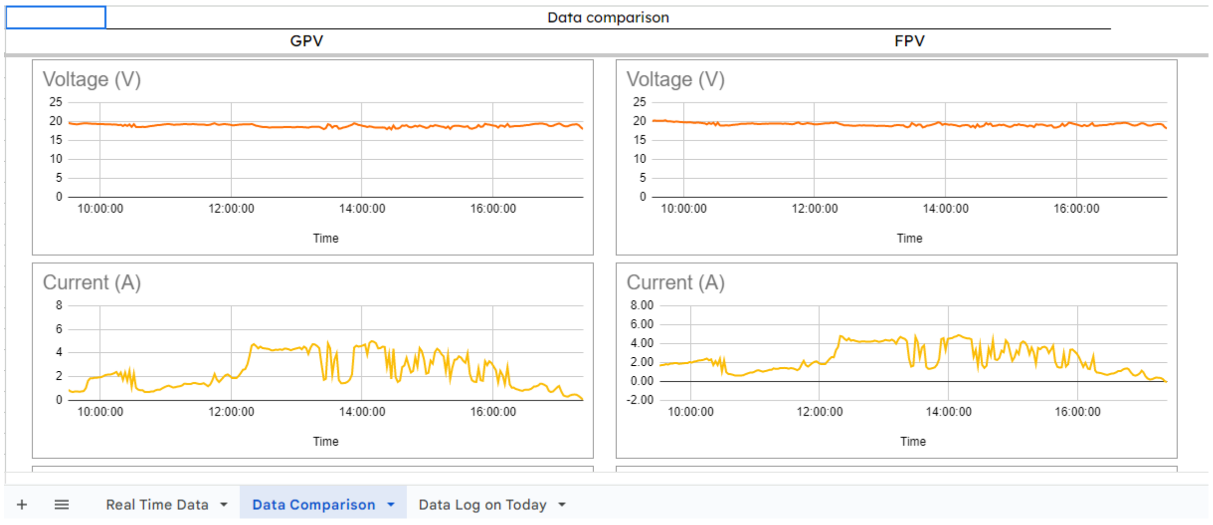


Figure 4.20: Graph of voltage and graph of current against time

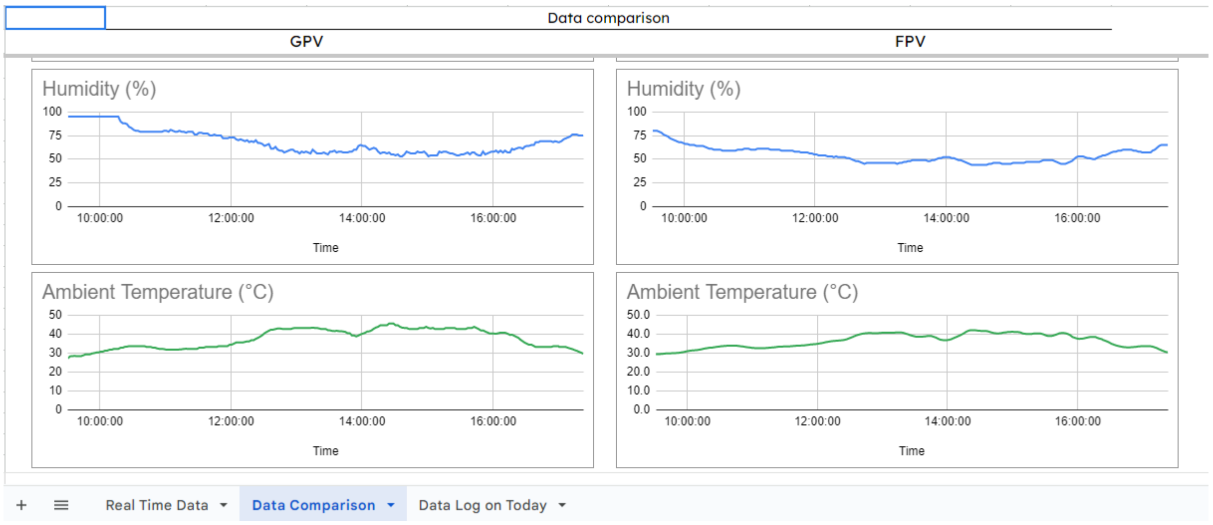


Figure 4.21: Graph of humidity and ambient temperature against time

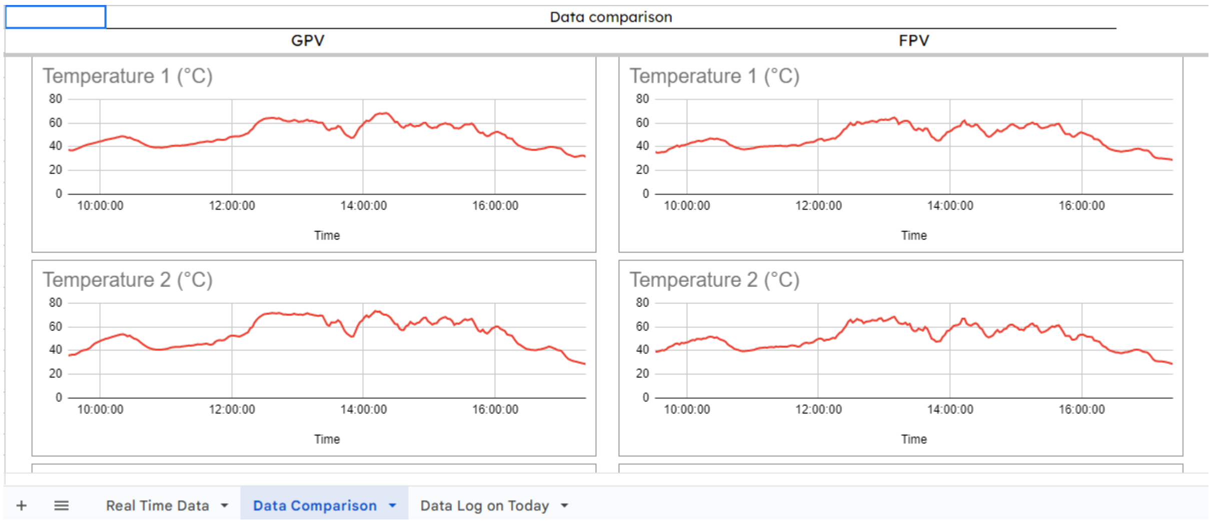


Figure 4.22: Graph of temperature 1 and graph temperature 2 against Time

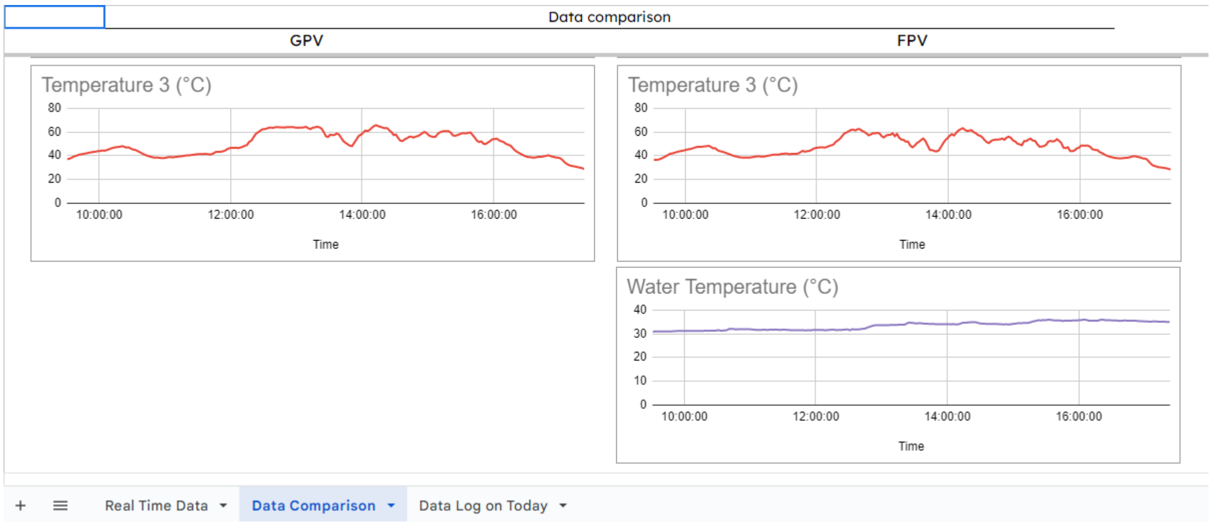


Figure 4.23: Graph of temperature 3 and graph of water temperature against Time

Figure 4.20 to 4.23 above proved that the data monitoring systems for FPV and GPV were able to transmit data to the Internet and then to the Google Spreadsheet. The data were received consistently between interval of 2 minutes for around 8 hours. There were no stoppages shown on the graphs, which proved that the sensors were working fine and do not have malfunction issue.

Throughout the days of the operation of the data monitoring systems, it was estimated that at fully charged condition, the 10 000 mAh power bank can last for more than 8 hours, while the phone used for sharing hotspot can last for around 8 hours. At the same time, network data usage was averagely 0.37 GB per day.

4.2.4 Data Analysis and Comparison Between FPV and GPV

To analyse the performance of the FPV and GPV system, maximum obtainable power, P_{mp} is calculated. However, the information of Fill Factor is required. Fill factor is the ratio of Actual Maximum Obtainable Power, P_{mp} to the product of Open Circuit Voltage, V_{oc} and Short Circuit Current, I_{sc}. Fill Factor was obtained using IV plotter device. The average fill factor, FF is 0.673. With the information of fill factor obtained, the maximum obtainable power is calculated with the formula below.

$$P_{mp} = FF \times V_{oc} \times I_{sc}$$

Where

$V_{oc} = Open\ Circuit\ Voltage, V$

$I_{sc} = Short\ Circuit\ Current, A$

$FF = Average\ Fill\ Factor$

$P_{mp} = Maximum\ Obtainable\ Power, W$

The daily energy yield is then calculated using the formula below.

$$Daily\ energy\ yield = Average\ P_{mp} \times Duration\ of\ data\ collection$$

Figure 4.27 below shows Open Circuit Voltage for FPV and GPV system on 17 March 2023 tabulated in graph.

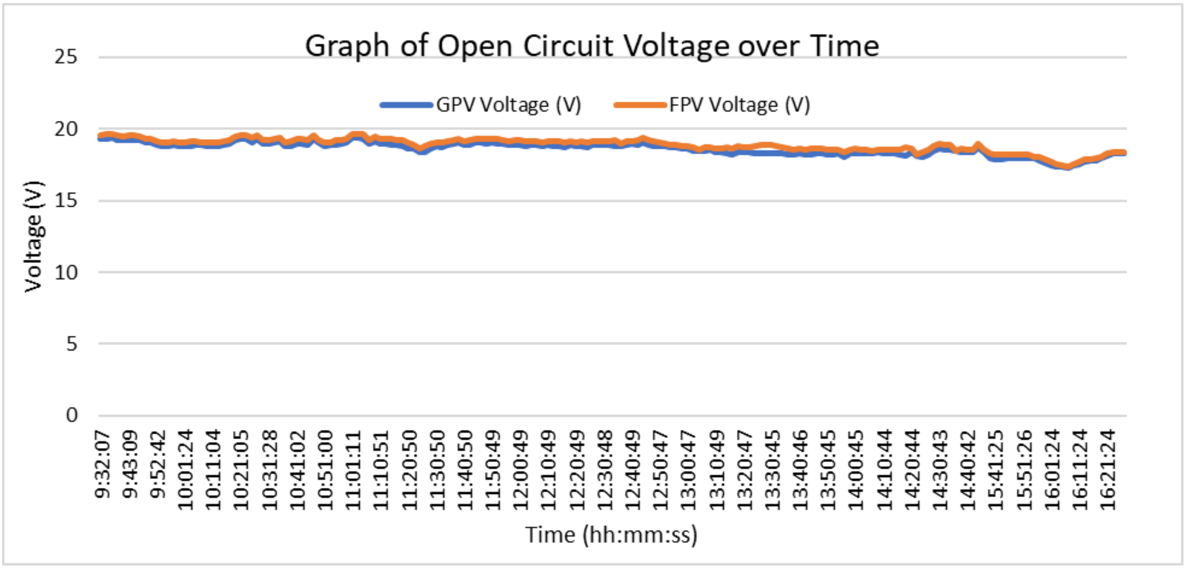


Figure 4.24: Graph of Open Circuit Voltage against Time

Figure 4.25 below shows Short Circuit Current for FPV and GPV system on 17 April 2023 tabulated in graph.

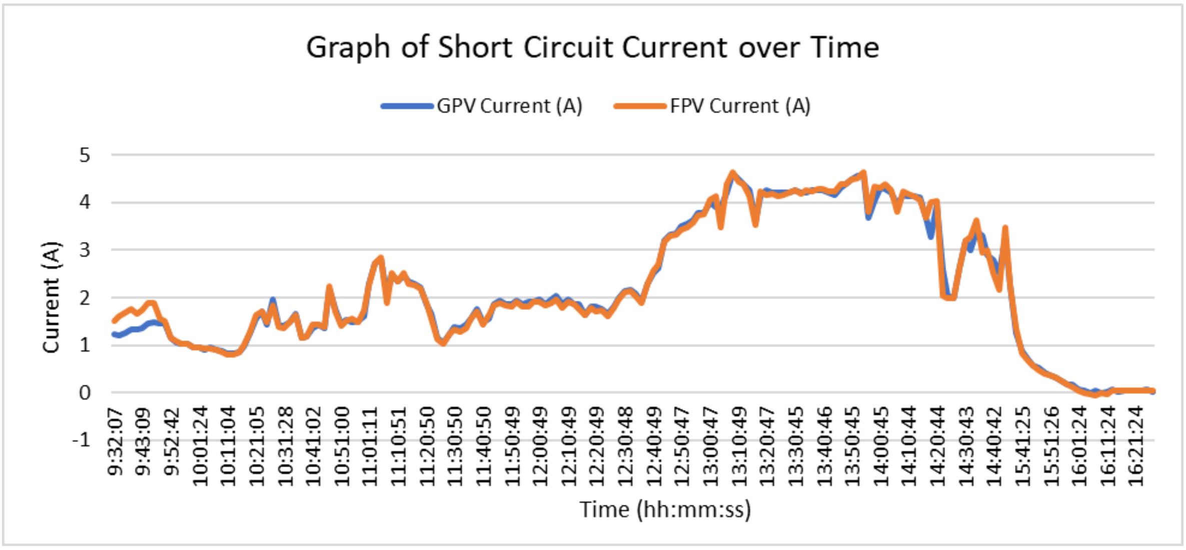


Figure 4.25: Graph of Short Circuit Current against Time

Figure 4.26 below shows Maximum Obtainable Power for FPV and GPV system on 17 March 2023 tabulated in graph.

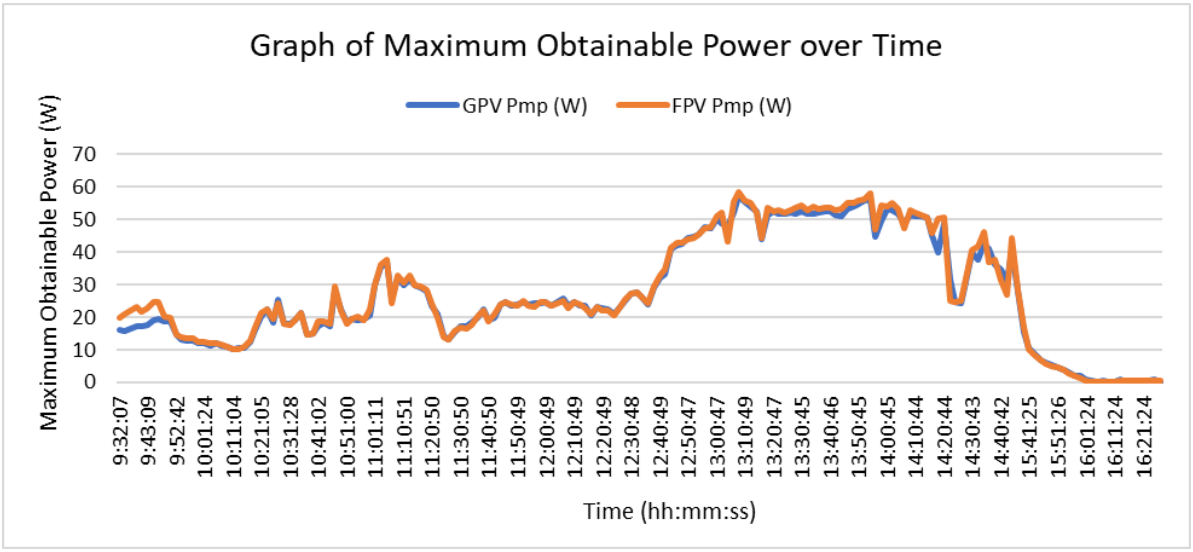


Figure 4.26: Graph of Maximum Obtainable Power against Time

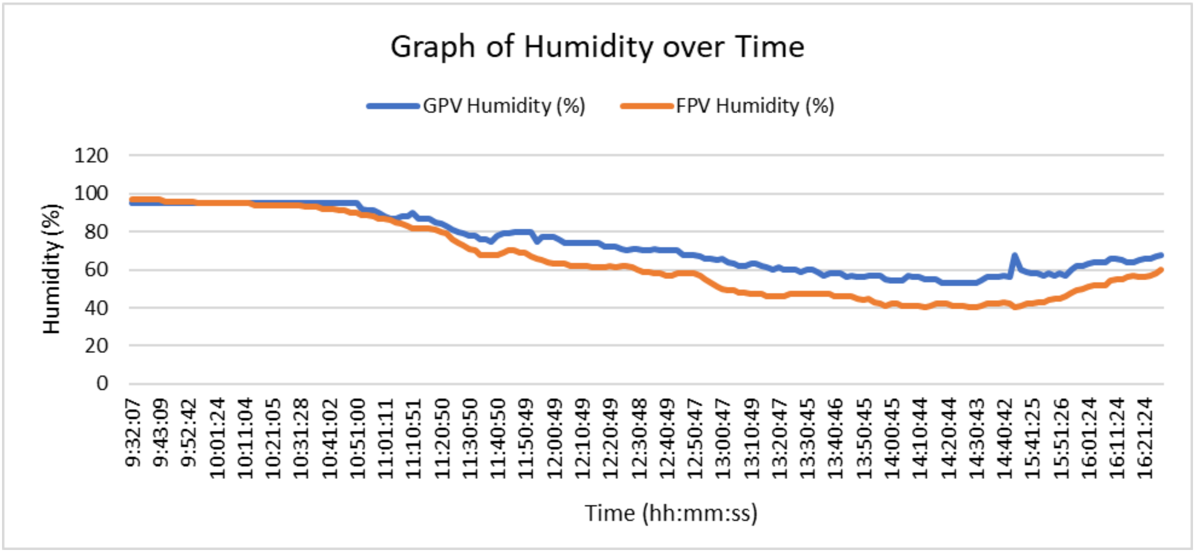


Figure 4.27: Graph of Humidity against Time

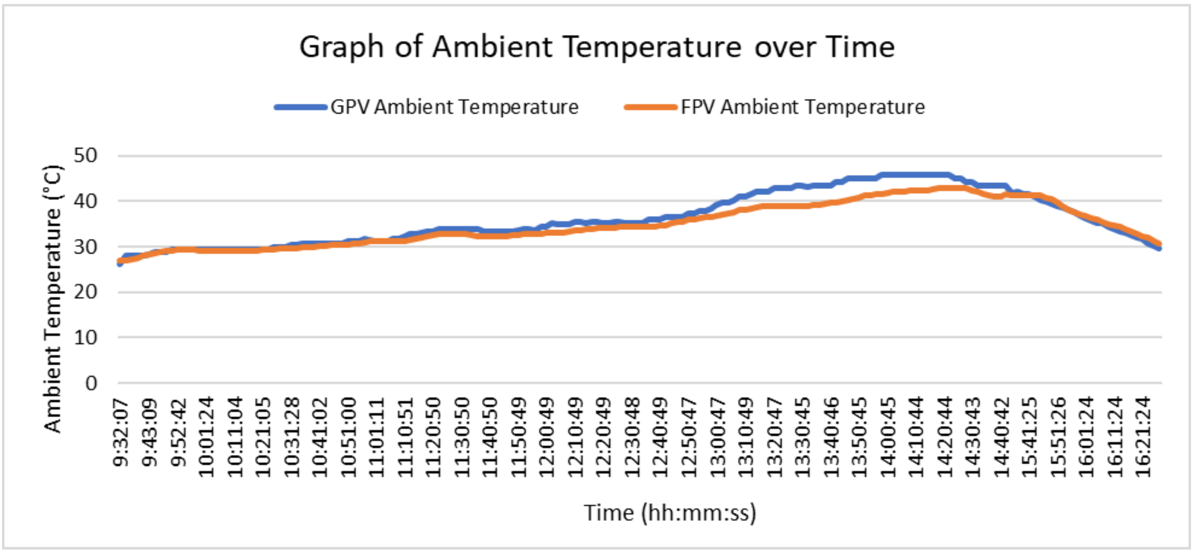


Figure 4.28: Graph of Ambient Temperature against Time

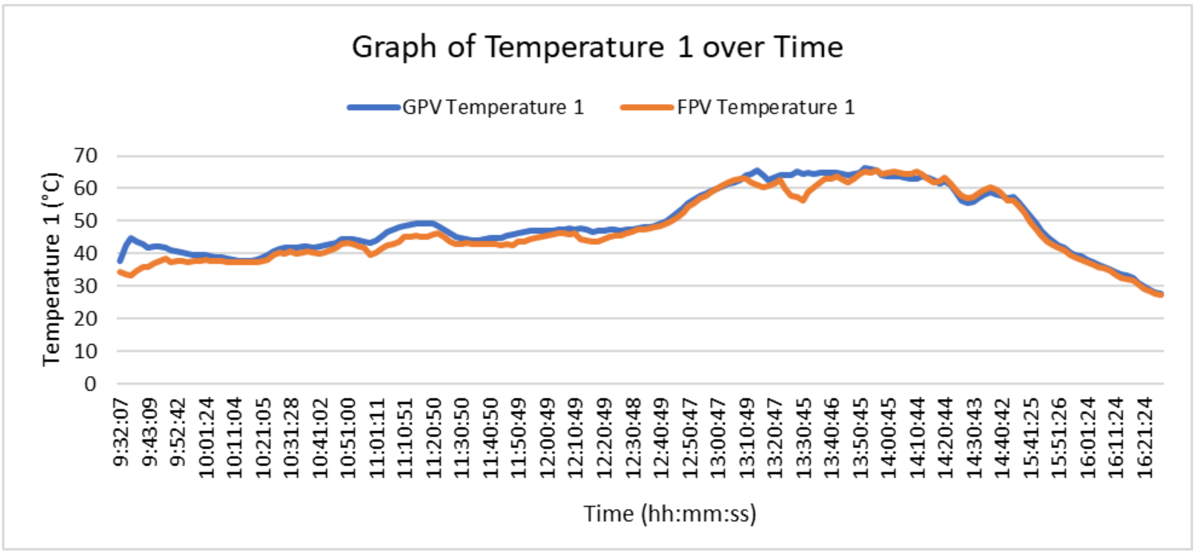


Figure 4.29: Graph of Temperature 1 against Time

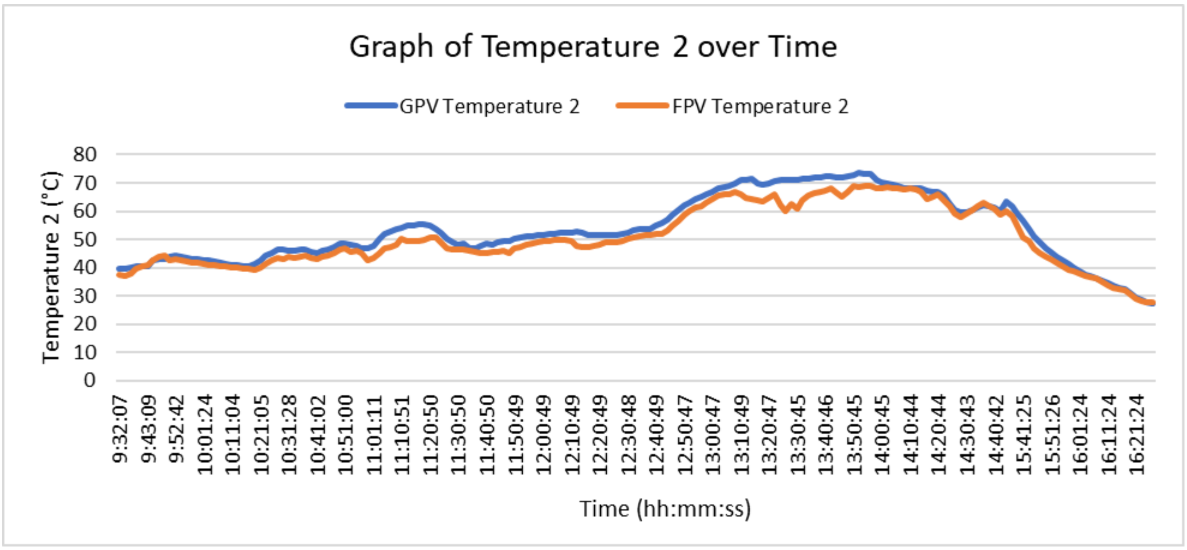


Figure 4.30: Graph of Temperature 2 against Time

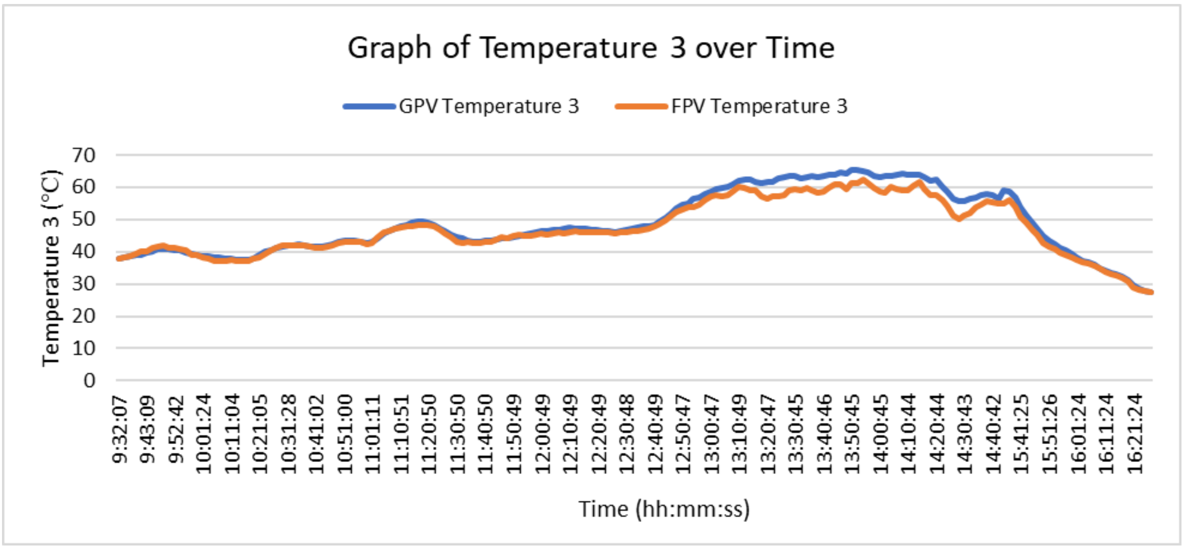


Figure 4.31: Graph of Temperature 3 against Time

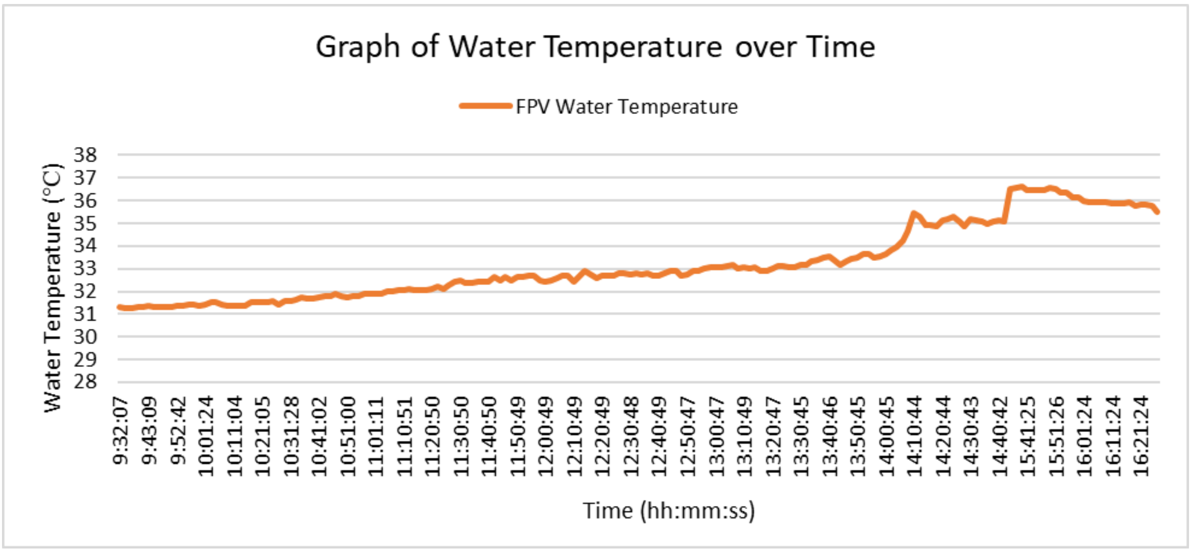


Figure 4.32: Graph of Water Temperature against Time

On 17 April 2023, the Open Circuit Voltage of FPV was slightly higher than that of GPV. The highest V_{oc} value from FPV was 19.67 V, whereas for GPV the value was 19.39 V. Furthermore, the Short Circuit Current of FPV was slightly higher than that of GPV one. The highest I_{sc} value from FPV was 4.64. A, whereas for GPV the value was 4.61 A. For Maximum Obtainable Power, the value from FPV was higher than GPV. The highest P_{mp} value from FPV was 58.46 W, whereas for GPV the value was 57.49 W. Moreover, the daily energy yield for FPV is 199.83 Wh, and 197.38 Wh for GPV. Figure 4.25 shows that before 9.52 am, the GPV Short Circuit Current readings were way lower than that of FPV. This was due to blockage of sunlight by lamp pole at the road side.

Table 4.5 below summarize the highest data reading of Humidity, Ambient Temperature, Temperature 1, Temperature 2, and Temperature 3 on 17 March 2023.

Table 4.5: Highest data reading of FPV system and GPV system

Type of PV system	Highest Humidity (%)	Highest Ambient Temperature (°C)	Highest Temperature 1 (°C)	Highest Temperature 2 (°C)	Highest Temperature 3 (°C)
FPV	97	42.9	65.38	69.06	62.31
GPV	95	45.9	66.13	73.50	65.50

Generally, it can be observed that Temperature 2 on both FPV and GPV was higher than Temperature 1 and Temperature 3. This indicates that the centre of PV panel is hotter than the corners of PV panel. This could be due to heat dissipated by the aluminium frame of PV panel. Meanwhile, the highest reading of Temperature 1 was higher than Temperature 3 on FPV. This could mean that heat dissipates better at the bottom of floating PV since it is closer to water surface. In the other hand, the average reading of Temperature 1 was only slightly higher than Temperature 3 on GPV. This could mean worse heat dissipation at the bottom of ground type PV as there is limited space for air flow as compared to FPV. Although the highest humidity of FPV reached 97 %, the humidity level on GPV throughout the day was still higher than FPV as observed in Figure 4.27. The ambient temperature of GPV was also higher than that of FPV throughout the day.

According to the datasheet of the PV panel in Appendix C, the rated Short Circuit Current is 5.17 A. However, the highest Short Circuit Current value on both FPV and GPV sometimes exceeded 5.17 A. Since the I_{sc} values from both FPV and GPV were during noon time, one possible reason could result this high I_{sc} value. The rated I_{sc} value stated in datasheet may be based on solar irradiance of 1000 W/m^2, but the actual solar irradiance on certain period of the day exceeded this value. Therefore resulting higher I_{sc} value.

4.2.5 Comparison Between Data Collected on IoT System and Micro SD Card

To analyse the amount of variation between the data recorded in SD card and data recorded in the IoT system, which is Google Spreadsheet, percentage of data loss for each day was calculated. When percentage of data loss is greater than zero, there might be some issue in the data recording in the IoT system. The formula is as shown below:

$$Percentage\ of\ data\ loss\ (\%) = \frac{amount\ of\ data\ lost\ in\ IoT}{total\ amount\ of\ data(from\ SD\ card)} \times 100\%$$

Table 4.6 below summarize the percentage of data loss of the IoT system and problems leading to percentage of data lost greater than zero for a total of seven days from 18 March 2023 to 24 March 2023. Table 4.7 summarizes the problems, their possible root causes and solutions.

Table 4.6: IoT system percentage of data loss

Date	**FPV percentage of data lost (%)**	**Problem**	**GPV percentage of data lost (%)**	**Problem**
18 March 2023	0.41	-1 row of data was skipped in IoT system	1.24	-3 rows of data were skipped in IoT system
19 March 2023	6.44	-7 rows of data were skipped in IoT system -Multiple rows of data were shifted;	43.32	-Data suddenly stop updated on IoT system at 2.06 pm
20 March 2023	0.90	-2 rows of data were skipped in IoT system	0.45	-1 row of data was skipped in IoT system
21 March 2023	3.12	- 3 rows of data were skipped in IoT system; -Multiple rows of data were shifted;	0	-
22 March 2023	0	-	0.44	-1 row of data was skipped in IoT system
23 March 2023	3.69	-1 row of data was skipped in IoT system;	0.42	-1 row of data was skipped in IoT system

		-Multiple rows of data were shifted		
24 March 2023	0.49	-1 row of data was skipped in IoT system	0.49	-1 row of data was skipped in IoT system

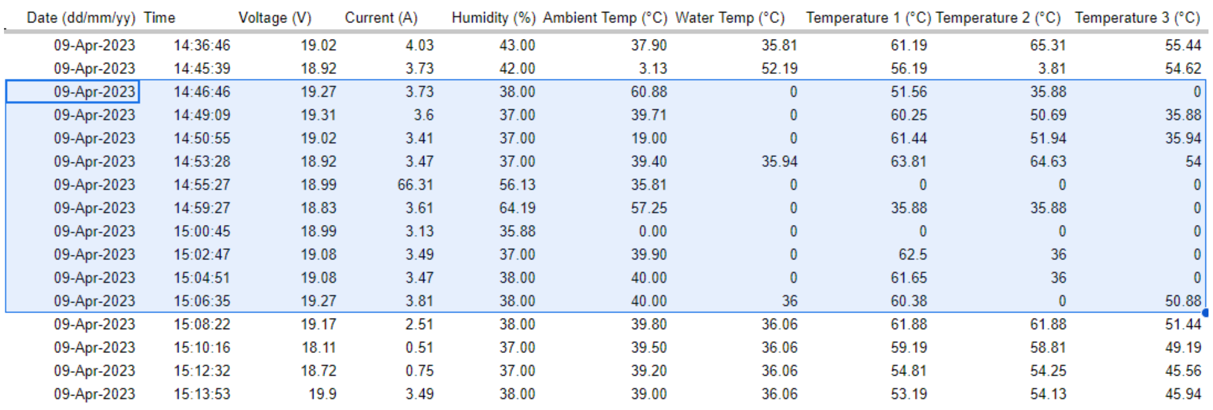
Table 4.7: Problems, possible root causes and solution on data logging in IoT system

Problem	**Possible Root Cause(s)**	**Solution**
Time delay of 1 minute for data to appear in IoT	Slow processing time for data sent from Arduino to NodeMCU then to the IoT platform	Replace Arduino and NodeMCU to a single microcontroller that has decent processing capability and WI-FI chip to save the processing time
Data lost due to data sent from NodeMCU to Google sheet being skipped	Google sheet API service outage or Google server is temporarily unavailable	Not available
	Network connectivity issue	Improve network connectivity of NodeMCU by placing NodeMCU to a spot with better network coverage. Choose better internet service provider.
	Both NodeMCU devices competing for the hotspot signal	Use high speed internet service
Data lost due to multiple rows of data being shifted	Network connectivity issue	Improve network connectivity with solutions mentioned above

	Both NodeMCU devices competing for the hotspot signal	Use high speed internet service
Data lost due to sudden stop of data updating in Google sheet	Goggle sheet API service outage or Google server is temporarily unavailable	Not available
	Network connectivity issue	Improve network connectivity with solutions mentioned above
	Both NodeMCU devices competing for the hotspot signal	Use high speed internet service
	Loose wire connection on VCC, GND, TX and RX pin on NodeMCU board	Replace jumper wire with male pin heads soldered on the PCB, then directly plug-in the entire NodeMCU board's I/O pin onto those pin heads for more secure connection

Several reasons could lead to data sent from NodeMCU to Google sheet being skipped. The first reason is service outages, where either the Goggle sheet API or internet service used by NodeMCU is facing disruptions, data may not be sent or recorded properly. The next reason is network connectivity issue where NodeMCU is not properly connected to the internet. One possible reason to this is PV panel causing blockage of network signal. This is likely to happen because the whole hardware system was placed under the solar PV panel to prevent direct sunlight. However the consequence could be network signal being blocked from transmitting from the phone to the NodeMCU. Furthermore, both NodeMCU devices may be competing for the hotspot signal, resulting slower network connection.

When NodeMCU fails to send data to Google sheet after a long while, the issue elevates to multiple rows of data being shifted as shown in Figure 4.33 below. This is due to large amount of data sent from Arduino accumulated in NodeMCU memory storage and NodeMCU fails to decode the data properly. The issue resolved itself once the network connectivity gets back to normal.



Date (dd/mm/yy)	Time	Voltage (V)	Current (A)	Humidity (%)	Ambient Temp (°C)	Water Temp (°C)	Temperature 1 (°C)	Temperature 2 (°C)	Temperature 3 (°C)
09-Apr-2023	14:36:46	19.02	4.03	43.00	37.90	35.81	61.19	65.31	55.44
09-Apr-2023	14:45:39	18.92	3.73	42.00	3.13	52.19	56.19	3.81	54.62
09-Apr-2023	14:46:46	19.27	3.73	38.00	60.88	0	51.56	35.88	0
09-Apr-2023	14:49:09	19.31	3.6	37.00	39.71	0	60.25	50.69	35.88
09-Apr-2023	14:50:55	19.02	3.41	37.00	19.00	0	61.44	51.94	35.94
09-Apr-2023	14:53:28	18.92	3.47	37.00	39.40	35.94	63.81	64.63	54
09-Apr-2023	14:55:27	18.99	66.31	56.13	35.81	0	0	0	0
09-Apr-2023	14:59:27	18.83	3.61	64.19	57.25	0	35.88	35.88	0
09-Apr-2023	15:00:45	18.99	3.13	35.88	0.00	0	0	0	0
09-Apr-2023	15:02:47	19.08	3.49	37.00	39.90	0	62.5	36	0
09-Apr-2023	15:04:51	19.08	3.47	38.00	40.00	0	61.65	36	0
09-Apr-2023	15:06:35	19.27	3.81	38.00	40.00	36	60.38	0	50.88
09-Apr-2023	15:08:22	19.17	2.51	38.00	39.80	36.06	61.88	61.88	51.44
09-Apr-2023	15:10:16	18.11	0.51	37.00	39.50	36.06	59.19	58.81	49.19
09-Apr-2023	15:12:32	18.72	0.75	37.00	39.20	36.06	54.81	54.25	45.56
09-Apr-2023	15:13:53	19.9	3.49	38.00	39.00	36.06	53.19	54.13	45.94

Figure 4.33: Multiple rows of data being shifted in Google sheet

As for the issue of sudden stop of data updates from NodeMCU to Google sheet, the above-mentioned factors can contribute to this problem. However, it can also be due to loosen wire connection on VCC, GND, TX and RX pin on NodeMCU. Loose connection on VCC or GND results unstable power supply. Insufficient or unstable power supply could NodeMCU to shut down and stop working. While loose connection on TX and RX pin results data sent from Arduino being disrupted. The root cause to loosen wire connection can be due to strong wave movement of the lake during windy weather. Since the connection of the pins on NodeMCU board are not soldered, when the floater wiggles the wire connection can become loose.

4.3 Challenge Encountered

Table 4.8 below summarizes the challenges faced in this project and how they were solved.

Table 4.8: Challenges faced in this project and their solutions

Challenges	Possible Root Cause(s)	Solution/Suggestion
SD card module failed to initialize and record data into SD card	-Too may sensors connected to the circuit. -Sensors were extended with wires that are too long	Finalized hardware configuration to three DS18B20 sensors extended with wire length not more than 1.5 metres.
NodeMCU failed to turn on and function		
Change of accuracy of voltage and current reading	Happens when there is change of hardware configuration	Do calibration on the voltage and current reading after finalizing the hardware configuration
Burn out of crocodile clip cable used for measuring open circuit voltage and short circuit current of PV panel	Bad electrical connection due to rusting on iron crocodile clip	Use better connectors that are water-proof and rust-proof such as MC4 connectors and copper crocodile clips
	High current flow from PV panel to crocodile clip cable	Use copper wire with higher copper diameter or higher American Wire Gauge (AWG) size according to the current rating of PV panel
Inaccurate voltage and current reading	Change of power bank model resulting change of VCC value	Use high quality and same power bank model for both FPV and GPV

Power bank deterioration	Power bank operating under high temperature and humidity environment	Look for power bank placement spot that has good ventilation and optimal operating temperature

The hardware system was initially equipped with six DS18B20 temperature sensors that were extended with wires of 1.5 metres long. When there are many extended temperature sensor in the system, the other components may not be working properly. This can be due to too much current drawn by the long wires, resulting insufficient current flow to the components. For example, SD card module would fail to initialize and record data into SD card. Even if it successfully initialized, the data recording would still fail when relay is switched on. Once relay is switched on it will draw some current from the circuit. The ESP8266 NodeMCU could also be affected by this issue and would fail to turn on. After some trial and error, the hardware configuration was finalized with three DS18B20 temperature sensors with extended wires of around 1.5 metres. This finalized version was able to work without issue for most of the time.

When changes are made on the hardware configuration, the voltage drop across the components on the circuit can be affected. While most of the components are still workable if the voltage supplied is within the rated VCC range, the change of voltage drop can affect the accuracy of voltage and current reading. The calculation of voltage and current value in Arduino requires the input of VCC value. If the actual VCC value changed, it can directly affect the accuracy of these two readings. Therefore, the voltage and current reading may need to be calibrated again if there are any changes made on the hardware configuration that resulted change of voltage drop.

The next issue is crocodile clip cable burn out after around one month of data collection. The crocodile clip cable that burnt out was the one connecting to the positive pole of ground PV panel. Figure 4.34 below shows the burnt out cable that was broken into two pieces. Bad electrical connection due to rusting on iron metal, and high current flow can cause short circuit and burn out the wire. Copper is a type of noble metal and therefore does not get corroded. It was observed that the crocodile clip was corroded as it is made from iron metal and it can absorb moisture from the air.

Furthermore, it was noticed that the positive crocodile clip cable from GPV experience corrosion quicker than FPV one. It could be due to higher humidity environment since the system was installed on grass land instead of concrete land.

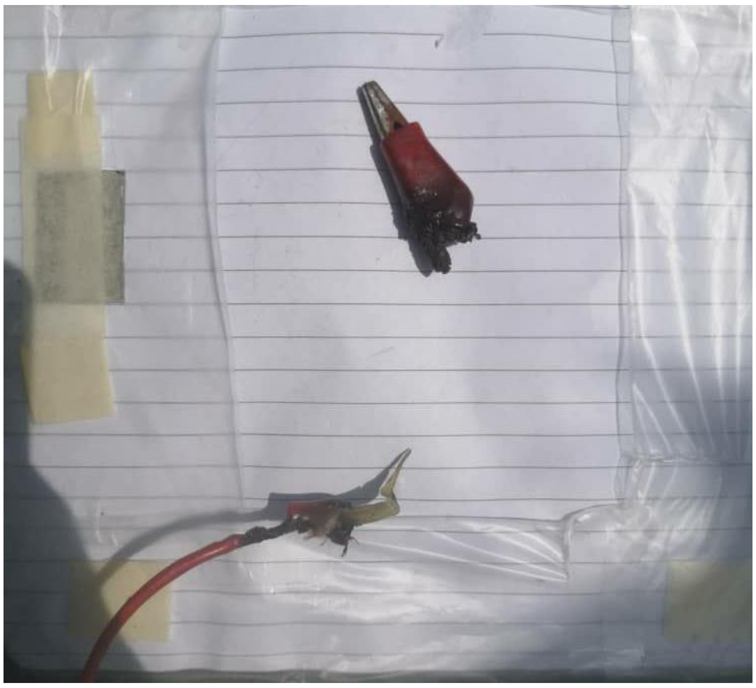


Figure 4.34: Burnt and broken crocodile clip cable from GPV

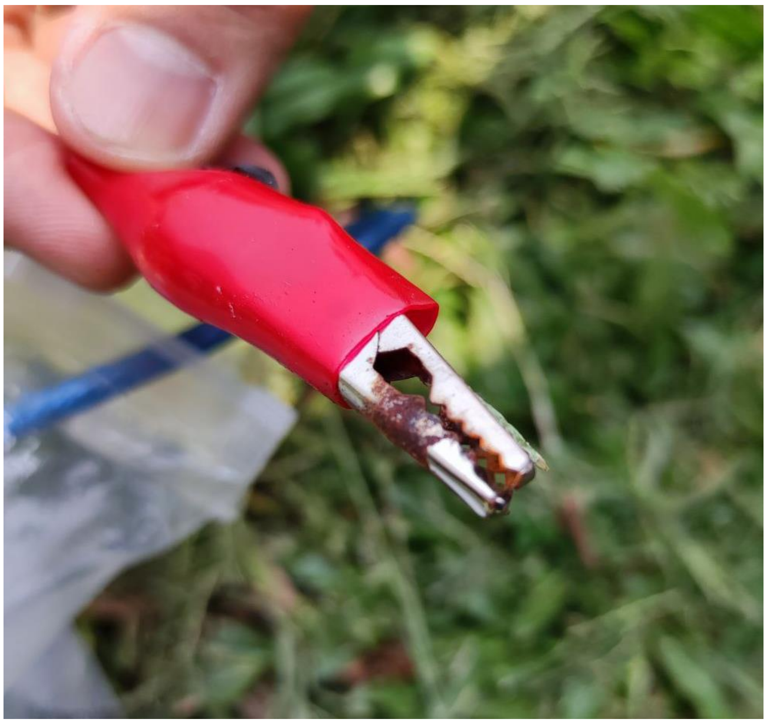


Figure 4.35: Rusted Crocodile Clip on GPV after one week

Moreover, the accuracy of voltage and current reading can be affected by the model or type of power bank. It is suspected due to difference in voltage supply to the Arduino Mega microcontroller when using different power bank model.

The last issue is power bank deterioration. The issue was noticed when the IoT platform stopped updating GPV data. When checked, the power bank's LED battery indicator light shows only one light and was discharged quicker than usual even though it has been fully charged. The power bank was replaced with another one and the same issue was encountered. This could be due to power bank operating under high temperature and humidity environment. According to the datasheet of the power bank, the operating temperature is 0 – 40 ºC for discharge. This indicates that temperature out of this range can damage the power bank, or it may not work according to the specifications. The power bank was placed under the solar panel to avoid expose to direct sunlight. However, since the location has bad ventilation, the temperature of the bottom of PV panel could also be higher than the operating temperature. Furthermore, to avoid water damage, the power bank was put in a zip bag. This may further worsen the issue as heat would trap in the zip bag. Power bank placement spot that has good ventilation and optimal operating temperature.

4.4 Cost Analysis

Table 4.9: Cost of Components and Consumables

Components and consumables	**Quantity**	**Cost per unit (RM)**	**Total Cost (RM)**	**Vendor**
Arduino Mega 2560 R3 with cable	2	69.50	139.00	Techmakers (Shopee)
ESP8266 NodeMCU V3 CH340	2	18.20	36.40	SGROBOT (Shopee)

Micro USB cable	1	5.00	5.00	iConTech Component
ACS712 hall effect current sensor 5A	2	6.50	13.00	Sainapse (Shopee)
DS18B20 temperature sensor	6	3.30	19.80	TGElectronic. (Shopee)
DS18B20 temperature sensor module (waterproof)	1	10.90	10.90	SGROBOT (Shopee)
DHT11 humidity sensor module with LED	2	4.80	9.60	littlecraft (Shopee)
OLED display module (128x64)	2	12.50	25.00	Techmakers (Shopee)
DS3231 AT24C32 RTC module with battery	2	15.90	31.80	SGROBOT (Shopee)
Micro SD Card Adapter Reader Module	2	4.30	8.60	SGROBOT (Shopee)
SanDisk Ultra Micro SD card 16 GB	2	17.90	35.80	PCByte (Shopee)
Resistors 30k ohm (10psc)	1	1.00	1.00	Techmakers (Shopee)
Resistors 7.5k ohm (10psc)	1	1.00	1.00	Techmakers (Shopee)
Resistors 4.7k ohm (10psc)	1	1.00	1.00	Techmakers (Shopee)
Resistors 2.2k ohm (10psc)	1	1.00	1.00	Techmakers (Shopee)
5V SPDT Relay	2	1.20	2.40	Techmakers (Shopee)
1N4142 Diode	2	0.20	0.40	Techmakers (Shopee)
2N2222 NPN Transistor	2	0.30	0.60	SYNACORP @STELECTRONICS (Shopee)

Dupont Jumper wires 20cm male to female (40pcs)	1	2.70	2.70	littlecraft (Shopee)
Dupont Jumper wires 20cm male to male (40pcs)	1	2.80	2.80	Autobotics (Shopee)
Soft silicon wire AWG 20 Black (3 mteres)	1	3.00	3.00	idroNation (Shopee)
Soft silicon wire AWG 20 Red (3 mteres)	1	3.00	3.00	idroNation (Shopee)
Multicore copper wires AWG 20 Black (1 metre)	3	1.70	5.10	HITECTRONS SDN BHD
Multicore copper wires AWG 20 Red (1 metre)	3	1.70	5.10	HITECTRONS SDN BHD
Multicore wires AWG 28 Red (1 metre)	15	0.80	12.00	Techmakers (Shopee)
Multicore wires AWG 28 Blue (1 metre)	15	0.80	12.00	Techmakers (Shopee)
Multicore wires AWG 28 Black (1 metre)	15	0.80	12.00	Techmakers (Shopee)
Single core wire AWG 26 (1 metre)	3	0.50	1.50	Techmakers (Shopee)
40 pin single row female pin header	1	0.50	0.50	Autobotics (Shopee)
2 pin screw terminal block	2	0.20	0.40	Autobotics (Shopee)
Crocodile clip (10psc)	1	8.50	8.50	HITECTRONS SDN BHD
830-hole Breadboard	2	3.75	7.50	iConTech Component (Shopee)
PCB board (FOC)	2	0.00	0.00	UTAR Kampar EE Lab
Cable tie 3x100mm (100pcs)	1	1.50	1.50	Mr. DIY

Heatshrink tube 3mm (5 metres)	1	4.00	4.00	Autobotics (Shopee)
Kapton tape (10mm)	1	5.60	5.60	Techmakers (Shopee)
Electrical insulating tape	1	2.00	2.00	Mr. DIY
Thermal paste GD460 (7 grams)	1	6.9	6.90	Techmakers (Shopee)
Pineng power bank 10000mAh PN936	2	42.70	85.40	138 Phone Accessories (Shopee)
		Total	**523.80**	

Table 4.10: Internet Charges (not inclusive of hardware device)

Description	**Quantity**	**Price per unit (RM)**	**Total price (RM)**	**Vendors**
Celcom 30 GB data plan (30days) + sim card	1	45.00	45.00	D talk (Econsave)
Celcom 30 GB data plan (30days)	1	35.00	35.00	D talk (Econsave)
Celcom 20 GB data plan (30days)	1	20	20.00	Celcom Life App
		Total	**100.00**	

Table 4.11: Total cost spent

Description	**Amount (RM)**
Cost of Components and Consumables	523.80
Internet Charges	100.00
Total	**623.80**

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

To conclude, the project’s aim and objectives were met with success. An IoT data monitoring system with percentage error around 1.5 % was successfully developed and implemented. The system has been operated for more than one month. The Arduino Mega microcontroller is able to collect data from the solar PV panels and record them into SD card. Data transmission from Arduino Mega to the IoT platform is successfully achieved. With Google Spreadsheet as the IoT platform, the received data can be visualized on a customizable dashboard, and recorded for future reference and analysis. Even though challenges were faced, but the project was still conducted successfully.

5.2 Recommendations and Improvements

Throughout the project implementation, he limitations and problems of this project were identified. The section that follows discusses some suggestions for improving the project.

5.2.1 Arduino Mega Data Collection Time Interval

In this project, the time interval for data collection was set using the delay() function in the Arduino coding. The downside of using this function is that the time interval can become inconsistent after operating for several hours. This is because the processing time required for Arduino board to perform the tasks can be increasingly long or short by a few hundreds of milliseconds. The consequence to this is unsynchronized data between GPV and FPV. To overcome this issue, the delay() function should be replaced with another method which is by comparing the program start time and program end time. The former is obtained before Arduino run the tasks in loop() function, whereas the latter is obtained after Arduino finishes running the tasks in loop() function. If the program end time is greater than program start time by a pre-set interval of time, the tasks in loop() function will be executed. Both program start time and program end time are obtained from RTC module. Therefore, RTC module with high accuracy is required to tell the time accurately.

5.2.2 Solar PV Panel as the Power Source

The downside of using power bank to supply power to the data monitoring system is their limited battery capacity. Power bank usually needs to be recharged after 1 to 2 days. To improve this, solar PV panel can be used as the power source for powering the data monitoring system. A solar charge controller and a battery are required. Solar PV panel converts sunlight into DC electrical power and feed in to charge controller. The voltage and current supply from PV panel are regulated at optimal ratings by solar charge controller before being delivered to the battery and the load, which is the data monitoring system. Meanwhile, the battery act as a buffer to receive excess energy from the solar PV panel.

5.2.3 Alternatives of IoT Data Monitoring System

The Google spreadsheet may be an imperfect IoT platform due to the problems mentioned beforehand. Furthermore, setting up the dashboard on Google Spreadsheet require quite a lot of coding and formulae, which may be less user-friendly and difficult for people who are not familiar to them. Therefore, user may try out other IoT platform such as Blynk and Arduino IoT Cloud. Both platforms have mobile app that allows user to build user interface and monitor data from connected devices remotely. Furthermore, user can obtain time via Blynk, which eliminates the use of RTC module provided the network connectivity is stable. Depending on the level of usage, user may need to subscribe to their plan to access more feature. For example, some of the limitations of the free plan on Blynk include limited datastreams (in this case the number of sensor data), limited period of historical data storage etc. Therefore one must allocate their budget wisely.

REFERENCES

Arduino Official Store. n.d. *Arduino Mega 2560 Rev3*. [online] Available at: <https://store.arduino.cc/products/arduino-mega-2560-rev3> [Accessed 22 August 2022].

Arduino Relay Control Tutorial. 2017. [image] Available at: <https://circuitdigest.com/microcontroller-projects/arduino-relay-control> [Accessed 23 August 2022].

Autobotic.com.my. n.d. *Current Sensor Module ACS712 (30A)*. [online] Available at: <https://www.autobotic.com.my/current-sensor-module-acs712-30a> [Accessed 22 August 2022].

Azhari, A., Sopian, K., Zaharim, A. and Al Ghoul, M., 2008. Solar Radiation Maps from Satellite Data for a Tropical Environment – Case Study of Malaysia. In: *3rd IASME/WSEAS Int. Conf. on Energy & Environment*. pp.528-533.

Cazzaniga, R., Cicu, M., Rosa-Clot, M., Rosa-Clot, P., Tina, G. and Ventura, C., 2018. Floating photovoltaic plants: Performance analysis and design solutions. *Renewable and Sustainable Energy Reviews*, 81, pp.1730-1741.

Choi, Y., 2014. A Study on Power Generation Analysis of Floating PV System Considering Environmental Impact. *International Journal of Software Engineering and Its Applications*, 8(1), pp.75-84.

Components101. 2020. *NodeMCU ESP8266*. [online] Available at: <https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet> [Accessed 22 August 2022].

DHT11 sensor module. n.d. [image] Available at: <https://nettigo.eu/products/dht11-sensor-module> [Accessed 23 August 2022].

DroneBot Workshop. 2021. *Measure DC Voltage and Current with an Arduino*. [online] Available at: <https://dronebotworkshop.com/dc-volt-current/> [Accessed 22 August 2022].

DS18B20 Temperature Sensor. 2018. [image] Available at: <https://components101.com/sensors/ds18b20-temperature-sensor> [Accessed 23 August 2022].

DS3231 Real Time Clock (RTC) Module, I2C. n.d. [image] Available at: <https://m.ubitap.com/DS3231-Real-Time-Clock-RTC-Module-I2C-p53018124> [Accessed 24 August 2022].

Dwivedi, L., Yadav, P. and Saket, R. K., 2016. *Partially shaded solar panels*. [image] Available at: <https://www.researchgate.net/figure/Partially-shaded-Solar-panels_fig1_308929021> [Accessed 13 August 2022].

Electronics Tutorials, n.d. *Bypass Diodes in Photovoltaic Arrays*. [image] Available at: <https://www.electronics-tutorials.ws/diode/bypass-diodes.html> [Accessed 13 August 2022].

Engineered Composites, n.d. *Properties & Uses of GRP Products*. [online] Available at: <https://engineered-composites.co.uk/why-grp/> [Accessed 12 August 2022].

FLUKE, n.d. [image] Available at: <https://www.fluke.com/en-my/product/temperature-measurement/ir-thermometers/fluke-54-ii> [Accessed 23 April 2023]

Gudino, M., 2021. *Arduino Uno vs. Mega vs. Micro*. [online] Arrow. Available at: <https://www.arrow.com/en/research-and-events/articles/arduino-uno-vs-mega-vs-micro> [Accessed 22 August 2022].

I2C 0.96" OLED 128x64. n.d. [image] Available at: <https://www.elecrow.com/i2c-096-oled-128x64-blueyellow-p-1086.html> [Accessed 24 August 2022].

Liu, L., Wang, Q., Lin, H., Li, H., Sun, Q. and Wennersten, R., 2017. Power Generation Efficiency and Prospects of Floating Photovoltaic Systems. *Energy Procedia*, 105, pp.1136-1142.

Maghami, M., Hizam, H., Gomes, C., Radzi, M., Rezadad, M. and Hajighorbani, S., 2016. Power Loss Due to Soiling on Solar Panel: A Review. *Renewable and Sustainable Energy Reviews*, 59, pp.1307-1316.

Majid, Z., Ruslan, M., Sopian, K., Othman, M. and Azmi, M., 2014. Study on Performance of 80 Watt Floating Photovoltaic Panel. *JOURNAL OF MECHANICAL ENGINEERING AND SCIENCES*, 7, pp.1150-1156.

Mathur, R., Mehrotra, D., Mittal, S. and Dhariwal, S., 1984. Thermal non-uniformities in concentrator solar cells. *Solar Cells*, 11(2), pp.175-188.

Ranjbaran, P., Yousefi, H., Gharehpetian, G. and Astaraei, F., 2019. A review on floating photovoltaic (FPV) power generation units. *Renewable and Sustainable Energy Reviews*, 110, pp.332-347.

Rosa-Clot, M., Rosa-Clot, P., Tina, G. and Scandura, P., 2010. Submerged photovoltaic solar panel: SP2. *Renewable Energy*, 35(8), pp.1862-1865.

Ost, I., 2020. *Does Solar Panel Temperature Coefficient Matter? | Solar.com*. [online] Solar.com. Available at: <https://www.solar.com/learn/does-solar-panel-temperature-coefficient-matter/#:~:text=A%20solar%20panel's%20temperature%20coefficient%20is%20not%20the%20only%20factor,production%20for%20your%20specific%20setup.> [Accessed 13 August 2022].

Pradhan, A. and Panda, B., 2017. Analysis of Ten External Factors Affecting the Performance of PV System. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*.

Pveducation.org. n.d. *Mismatch Effects | PVEducation*. [online] Available at: <https://www.pveducation.org/pvcdrom/modules-and-arrays/mismatch-effects> [Accessed 13 August 2022].

Sahu, A., Sudhakar, K. and Sarviya, R., 2019. Influence of U.V light on the thermal properties of HDPE/Carbon black composites. *Case Studies in Thermal Engineering*, 15, p.100534.

Sahu, A., Yadav, N. and Sudhakar, K., 2016. Floating photovoltaic power plant: A review. *Renewable and Sustainable Energy Reviews*, 66, pp.815-824.

Svarc, J., 2020. *Solar Panel Construction*. [online] CLEAN ENERGY REVIEWS. Available at: <https://www.cleanenergyreviews.info/blog/solar-panel-components-construction> [Accessed 13 August 2022].

Trapani, K. and Redón Santafé, M., 2014. A review of floating photovoltaic installations: 2007-2013. *Progress in Photovoltaics: Research and Applications*, 23(4), pp.524-532.

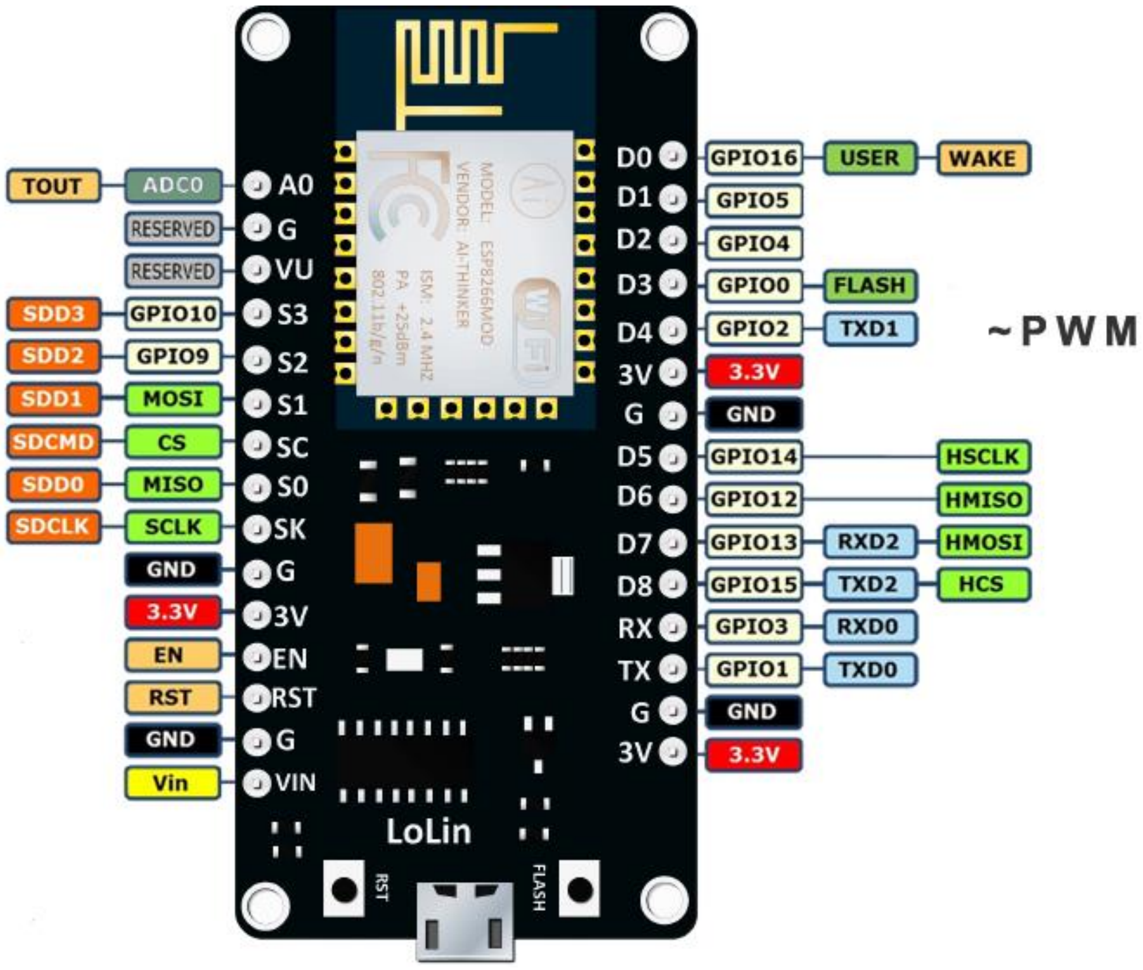
Xia, S., 2021. *19 Defects of Solar Panels and How to Avoid Them*. [image] Available at: <https://www.linkedin.com/pulse/19-defects-solar-panels-how-avoid-them-solar-panel-products> [Accessed 13 August 2022].

5V Compatible Micro SD Card Adapter. n.d. [image] Available at: <https://my.cytron.io/p-5v-compatible-micro-sd-card-adapter?r=1&gclid=CjwKCAjwmJeYBhAwEiwAXlg0AXva-DsJdkUutlE_KlDJ-TBLJpZxVLLrKcNqsyCNdX_ApHj-qV0WlxoC6x0QAvD_BwE> [Accessed 24 August 2022].

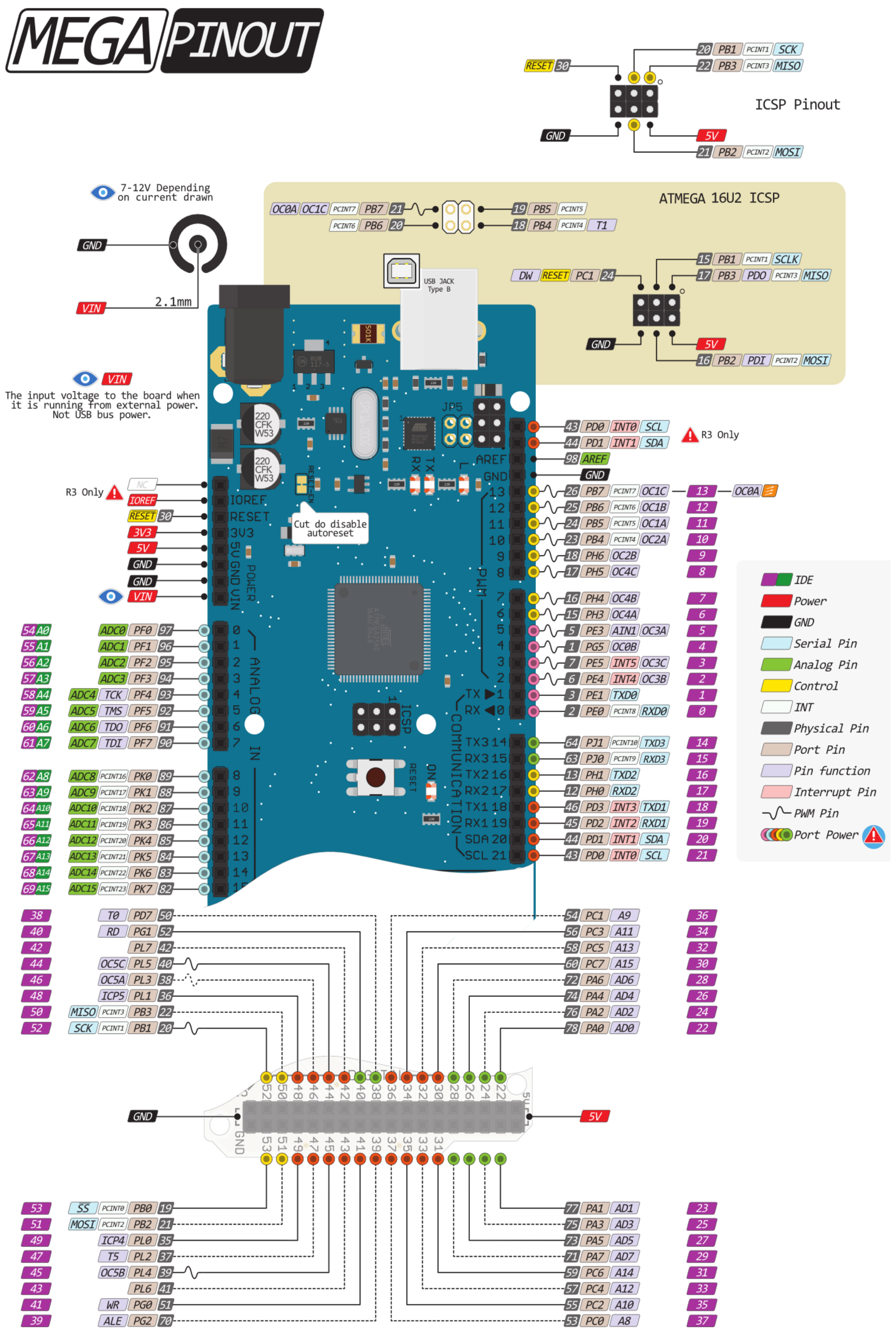
5V Relay. n.d. [image] Available at: <https://www.mgsuperlabs.com/featured-brands/mgsl/5932/5v-relay> [Accessed 23 August 2022].

APPENDIXES

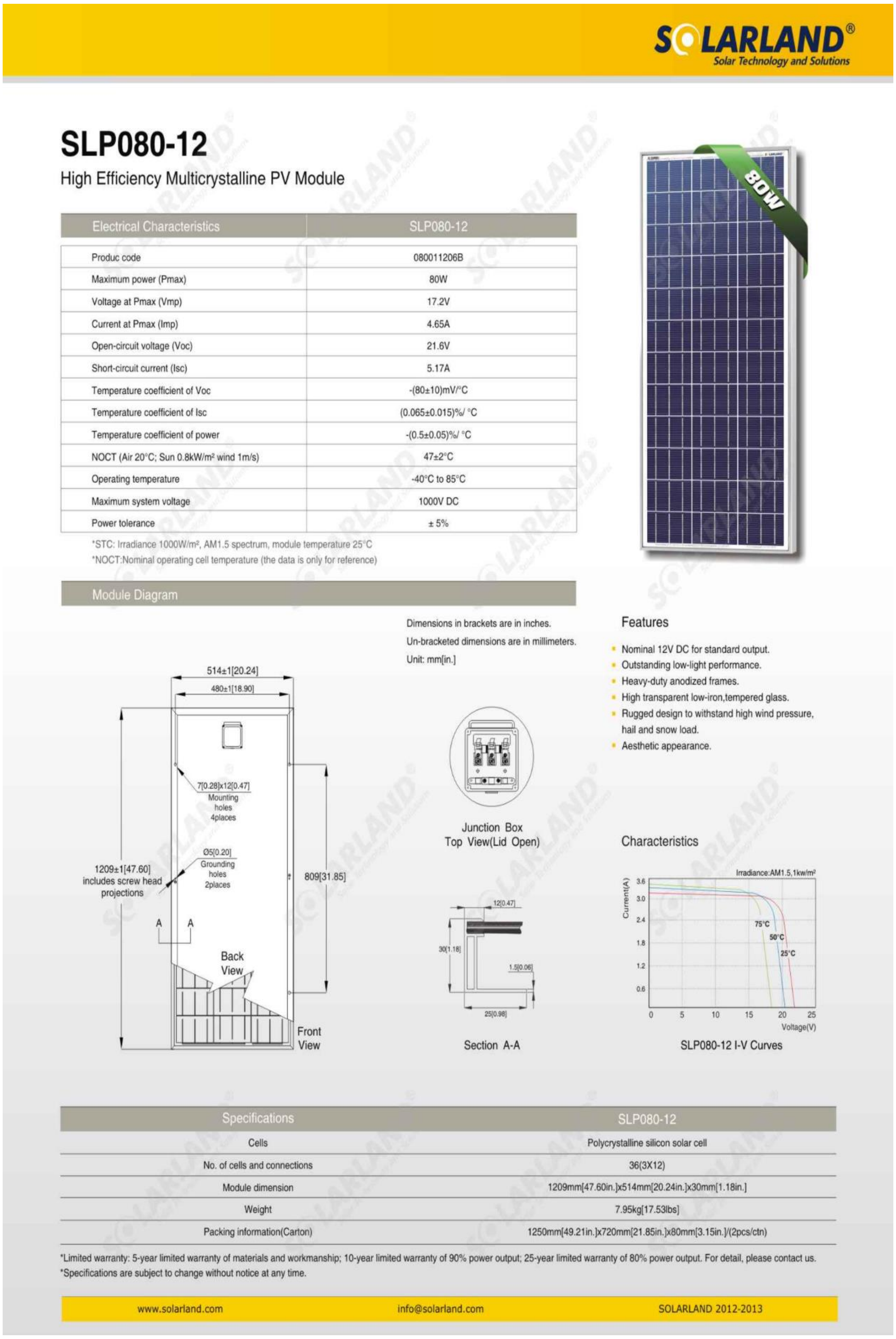
APPENDIX A: ESP8266 NodeMCU V3 Pinout



APPENDIX B: Arduino Mega Pinout



APPENDIX C: SOLARLAND 80W Solar Panel



SOLARLAND®
Solar Technology and Solutions

SLP080-12

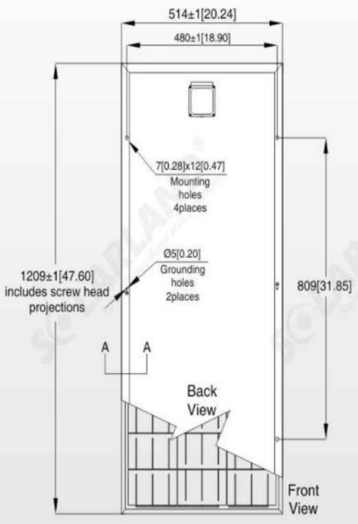
High Efficiency Multicrystalline PV Module

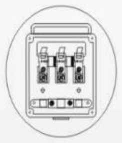
Electrical Characteristics	SLP080-12
Produc code	080011206B
Maximum power (Pmax)	80W
Voltage at Pmax (Vmp)	17.2V
Current at Pmax (Imp)	4.65A
Open-circuit voltage (Voc)	21.6V
Short-circuit current (Isc)	5.17A
Temperature coefficient of Voc	-(80±10)mV/°C
Temperature coefficient of Isc	(0.065±0.015)%/ °C
Temperature coefficient of power	-(0.5±0.05)%/ °C
NOCT (Air 20°C; Sun 0.8kW/m² wind 1m/s)	47±2°C
Operating temperature	-40°C to 85°C
Maximum system voltage	1000V DC
Power tolerance	± 5%

*STC: Irradiance 1000W/m², AM1.5 spectrum, module temperature 25°C
*NOCT:Nominal operating cell temperature (the data is only for reference)

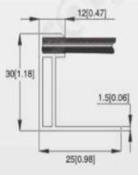
Module Diagram

Dimensions in brackets are in inches.
Un-bracketed dimensions are in millimeters.
Unit: mm[in.]


514±1[20.24]
480±1[18.90]
7[0.28]x12[0.47] Mounting holes 4places
Ø5[0.20] Grounding holes 2places
1209±1[47.60] includes screw head projections
809[31.85]
A
A
Back View
Front View



Junction Box
Top View(Lid Open)

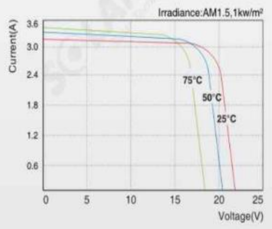


Section A-A

Features

- Nominal 12V DC for standard output.
- Outstanding low-light performance.
- Heavy-duty anodized frames.
- High transparent low-iron,tempered glass.
- Rugged design to withstand high wind pressure, hail and snow load.
- Aesthetic appearance.

Characteristics



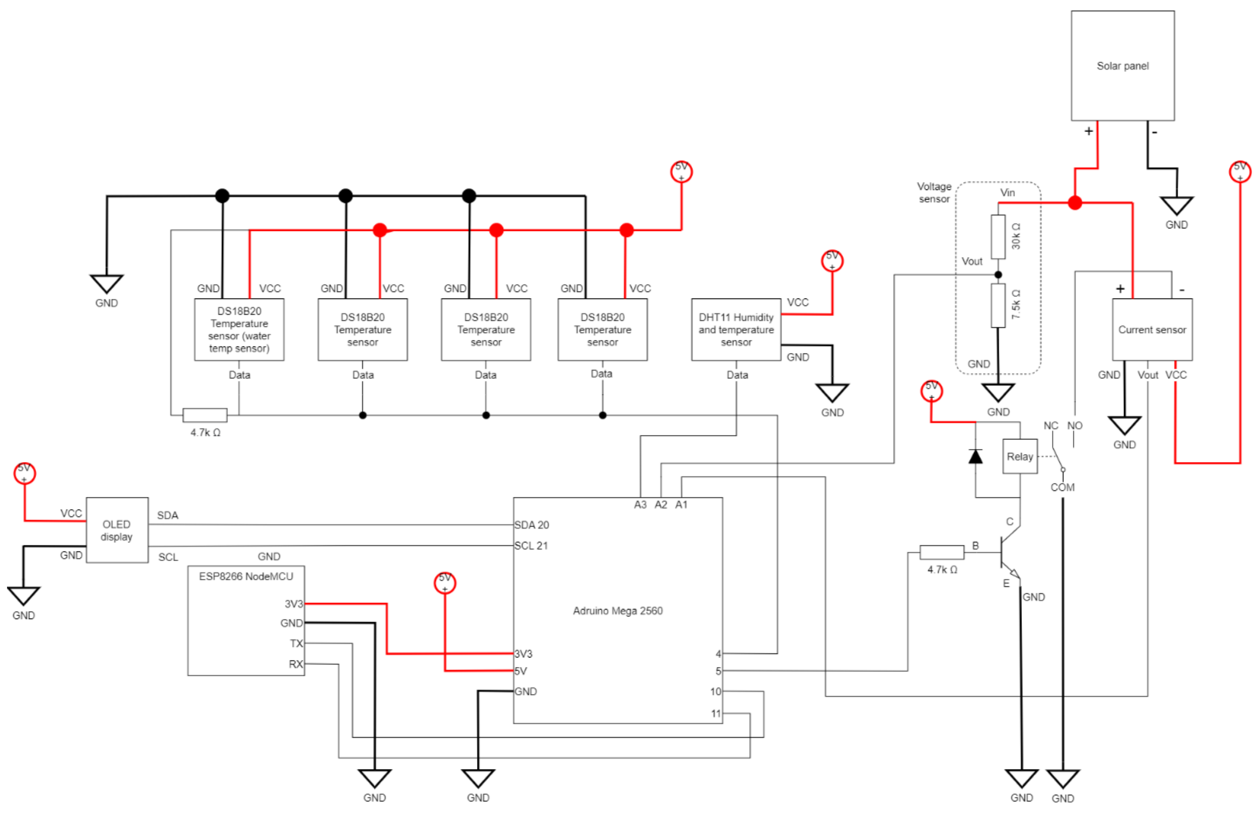
SLP080-12 I-V Curves

Specifications	SLP080-12
Cells	Polycrystalline silicon solar cell
No. of cells and connections	36(3X12)
Module dimension	1209mm[47.60in.]x514mm[20.24in.]x30mm[1.18in.]
Weight	7.95kg[17.53lbs]
Packing information(Carton)	1250mm[49.21in.]x720mm[21.85in.]x80mm[3.15in.]/(2pcs/ctn)

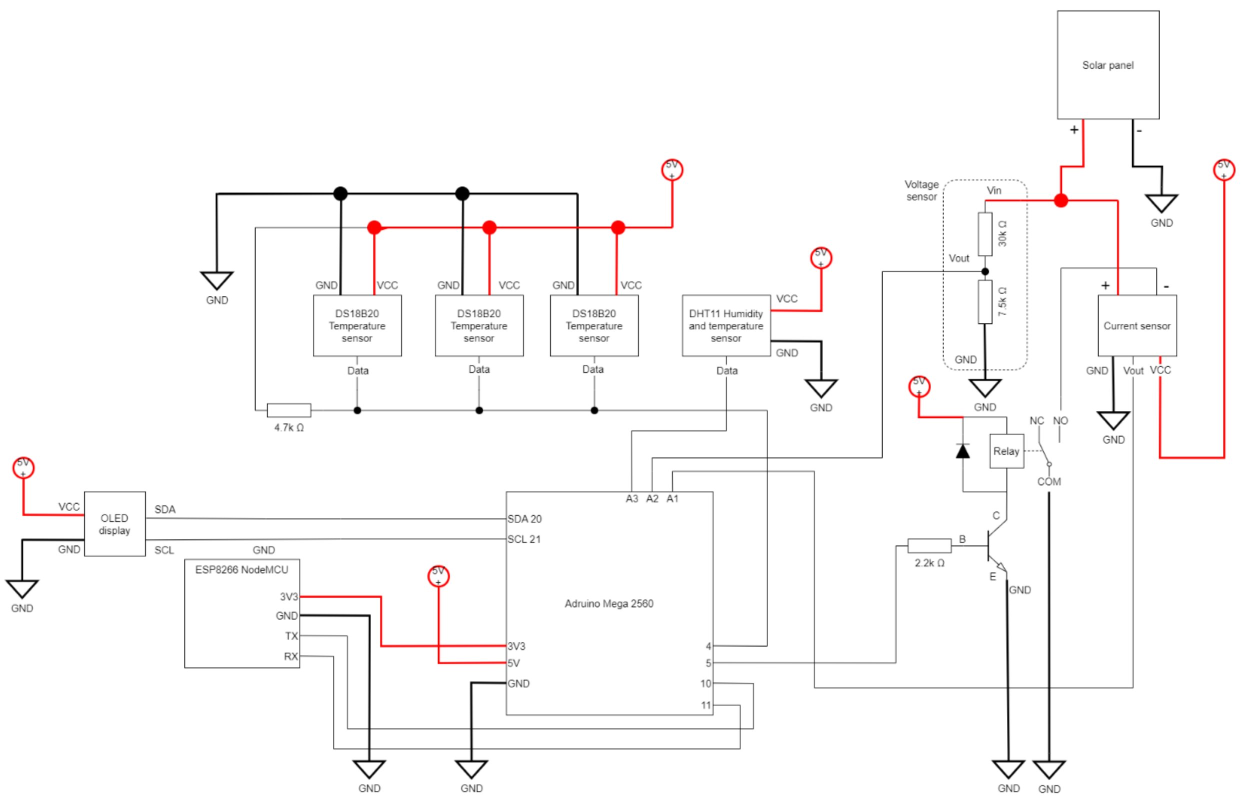
*Limited warranty: 5-year limited warranty of materials and workmanship; 10-year limited warranty of 90% power output; 25-year limited warranty of 80% power output. For detail, please contact us.
*Specifications are subject to change without notice at any time.

www.solarland.com info@solarland.com SOLARLAND 2012-2013

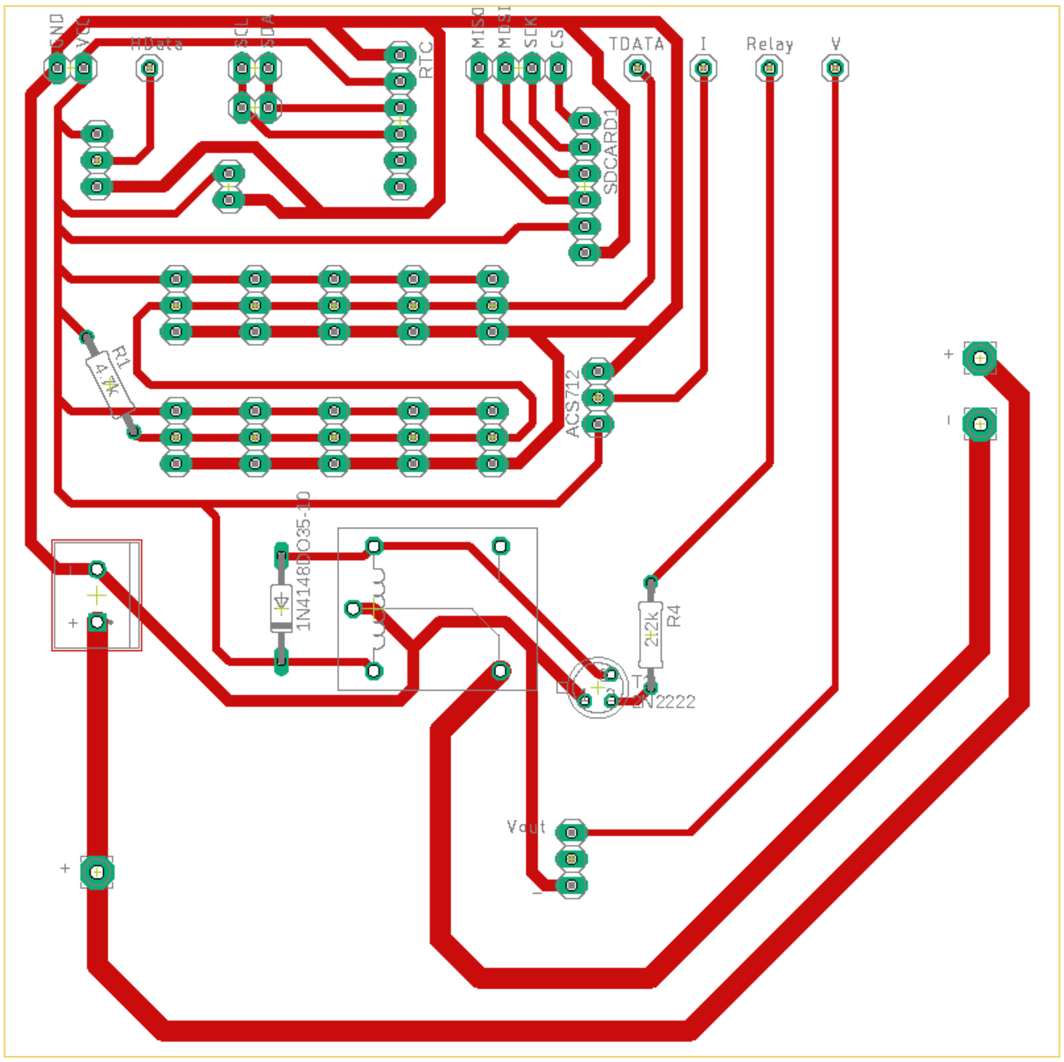
APPENDIX D: FPV Schematic Circuit Diagram



APPENDIX E: GPV Schematic Circuit Diagram



APPENDIX F: FPV and GPV PCB layout diagram



APPENDIX G: Coding for Arduino Mega for GPV

```
#include <SoftwareSerial.h> //Software library (required for serial
communication)
#include <DHT.h> //DHT library (required for DHT11 sensor)
#include <OneWire.h> //OneWire library (required for accessing Dallas 1
wired device)
#include <DallasTemperature.h> //TEMP library (required for DS18B20
sensor)
#include <Adafruit_GFX.h> //TEXT library (required for fonts styles in
OLED)
#include <Adafruit_SSD1306.h> //OLED library (required for the OLED)
#include <SPI.h> //SPI library (required for the SD card)
#include <SD.h> //SD library (required for the SD card)
#include <RTClib.h> //RTC library (required for DS3231 RTC module)

#define DHTPIN A3 //Define the analog pin A3 as the data pin of DHT11
#define DHTTYPE DHT11 //Define the version of DHT sensor used
#define ONE_WIRE_BUS 4 //Define the digital pin 4 as the data pin of
DS18B20
#define SCREEN_WIDTH 128 //Define the OLED display width in pixels
#define SCREEN_HEIGHT 32 //Define the OLED display height in pixels
#define OLED_RESET 4 //Adafruit library requires this element to be
defined but it is not involved in I2C

DeviceAddress thermometerAddress; //Custom array type to hold 64 bit
device address
OneWire oneWire(ONE_WIRE_BUS); //Setup a oneWire instance to communicate
with temperature IC
DallasTemperature tempsensor(&oneWire);//Point oneWire as reference to
DallasTemperature library
DHT dht(DHTPIN, DHTTYPE);
File myFile; // Create a file in the SD card
RTC_DS3231 rtc; // RTC
Adafruit_SSD1306 display(SCREEN_WIDTH,SCREEN_HEIGHT,&Wire,OLED_RESET);
SoftwareSerial espSerial(10, 11);

const int chipSelect = 53; // Define pin 53 as the data pin of SD card
const int relay1Pin = 5; //Define pin 5 as the data pin of relay

//Variables for function of current sensor
int adcValue_I = 0;
float adcVoltage = 0.0;
float currentValue = 0.0;
float sensitivity = 0.185; //Sensitivity of current sensor
float vRef = 5.00;
float offsetVoltage = 2.469; //Arduino mega offset voltage

//Variables for function of voltage sensor
int adcValue_V = 0;
float vOUT = 0.0;
float vIN = 0.0;
float R1 = 30550.0; //First resistor value of the voltage divider
float R2 = 7505.0; //Second resistor value of the voltage divider
```

```
// Addresses of 3 DS18B20s
// Every DS18B20 sensor has its unique address
// Replace the following addresses with those of your DS18B20 sensors
uint8_t sensor1[8] = { 0x28, 0x3E, 0x2F, 0x26, 0x01, 0x00, 0x00, 0xA5 };
uint8_t sensor2[8] = { 0x28, 0x01, 0x9A, 0x25, 0x01, 0x00, 0x00, 0x79 };
uint8_t sensor3[8] = { 0x28, 0xC6, 0x07, 0x26, 0x01, 0x00, 0x00, 0xB0 };

// Store sensor data into str
char str1[6];
char str2[6];
char str3[6];
char str4[6];
char str5[6];
char str6[6];
char str7[6];
String data;                        //contains all sensor data to be
sent to nodemcu


void setup() {
  Serial.begin(115200); //Initialize Serial monitor with 115200 baud
rate
  espSerial.begin(115200);
  dht.begin();                      //Initialize the DHT humidity
sensor
  tempsensor.begin();               //Initialize the DS18B20
temperature sensor
  rtc.begin();                      //Initialize the RTC module
  pinMode(relay1Pin, OUTPUT);       //Initialize the relay

  // setup for the SD card-------------------------------
  Serial.print("Initializing SD card...");

  if(!SD.begin(chipSelect)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
  //open file--------------------------------------------
  myFile=SD.open("GPV_DATA.txt", FILE_WRITE);

  // if the file opened ok, write to it:
  if (myFile) {
    Serial.println("File opened ok");
    // print the headings for our data
    myFile.println();
    myFile.println("Date  |  Time | Voltage(V) | Voltage ADC |
Current(A) | Current ADC | Humidity(%)| Ambient Temp(C)| Temp 1(C)| Temp
2(C)| Temp 3(C)");
  }
  myFile.close();

  //Display these titles on the serial monitor
  Serial.println("Date  |  Time | Voltage(V) | Voltage ADC | Current(A)
| Current ADC | Humidity(%)| Ambient Temp(C)| Temp 1(C)| Temp 2(C)| Temp
3(C)");
```

```
  //Initialize the I2C with address 0x3C (for the 128x32)
  display.begin(SSD1306_SWITCHCAPVCC,0x3C);
  display.clearDisplay();
  display.setTextColor(WHITE, BLACK);
  display.display();                   //Used to display all the data 
which is in buffer

  delay(2000);                         //To allow some time for nodemcu 
to be ready, ie:connect to wifi
}

void loop() {
  DateTime now = rtc.now(); //get time now
  int hh = now.hour();
  int mm = now.minute();
  int time_in_minutes = hh*60 + mm; //convert time now to minutes

  if (time_in_minutes >= 570) //if time now is after/equals to 9:30am = 
9*60+30 = 570 min
  {
    loggingTime();
    loggingVoltage();
    loggingCurrent();
    loggingHumidity();
    loggingTemperature();
    recordData();
    delay(113000); //data collection time interval
  }
}

void loggingTime(){
  DateTime timeNow = rtc.now();

  char dateBuffer[]="DD-MM-YYYY";
  char timeBuffer[]="hh:mm:ss";

  Serial.print(timeNow.toString(dateBuffer));
  Serial.print(" | ");
  Serial.print(timeNow.toString(timeBuffer));
  Serial.print(" | ");

  //record date and time in SD card
  myFile=SD.open("GPV_DATA.txt", FILE_WRITE);
  // if the file opened ok, write to it:
  if (myFile) {
    myFile.print(dateBuffer);
    myFile.print(" | ");
    myFile.print(timeBuffer);
    myFile.print(" | ");
  }
  myFile.close();
}

void loggingVoltage(){
  digitalWrite(relay1Pin,LOW);  //turn relay 1 off
  delay(2000);                  //Set the relay off for 2 second
  //adcValue_V = ((float)sum1/(float)samples);
  adcValue_V = analogRead(A2);
  vOUT = (adcValue_V*vRef)/1024.0;   //formula for arduino to interpret
the data
  vIN = (vOUT/(R2/(R1+R2)));      //formula for voltage calculation
  vIN = 0.9448*vIN;
  dtostrf(vIN,5,2,str1);
  Serial.print(vIN);
  Serial.print("   |   ");
  Serial.print(adcValue_V);
  Serial.print("   |   ");

  //Display the voltage on the OLED display
  display.setTextSize(2);
  display.setCursor(0,0);
  display.print("V= ");
  display.print(vIN);
  display.print(" V");
  display.display();
}

void loggingCurrent(){
  digitalWrite(relay1Pin,HIGH); //Turn relay on
  delay(2000);                  //Set relay on for 2 second

  //--------------------take a number of analog samples and sum them up
  #define samples 10            //Define number of analog samples to be
taken per reading
  int sum1 = 0;
  unsigned char count = 0;
  while (count<samples)
  {
   sum1 += analogRead(A1);
   count++;
   delay(10);
  }
  //-------------------------------------------------------------------

  //Formula to calculate current
  adcValue_I = ((float)sum1/(float)samples);
  adcVoltage = (adcValue_I*vRef)/1024;
  currentValue = ((adcVoltage - offsetVoltage)/sensitivity);
  currentValue = 0.9771*currentValue - 0.0946;
  dtostrf(currentValue,5,2,str2);
  Serial.print(currentValue);
  Serial.print("   |   ");
  Serial.print(adcValue_I);
  Serial.print("   |   ");

  // Display the current on OLED display
  display.setCursor(0,17);
  display.print("I= ");
  display.print(currentValue);
  display.print(" A");
  display.display();
}
```

```
void loggingHumidity(){ //function for humidity sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  // Read temperature as Celsius (the default)
  delay(2000);
  dtostrf(h,4,2,str3);
  dtostrf(t,4,2,str4);
  Serial.print(h);
  Serial.print(" | ");
  Serial.print(t);
}

void loggingTemperature(){ //function for temperature sensor
  //temperature comes in as a float with 1 decimal place
  tempsensor.requestTemperatures(); //request temperature sample from
the sensors through the one wire bus

  //Getting temperatures--------------
  //The circuit have ten DS18B20 on the same bus thus the data are
requested by their addresses
  //example: sensor1 holds the address for the first DS18B20 sensor on
the wire
  //getting temp reading with sensor1 address
  float temp1=tempsensor.getTempC(sensor1);
  //result is string with 5 position + \0 at the end
  //convert float to fprintf type string
  //format 4 positions with 2 decimal places
  //str contains the result
  dtostrf(temp1,4,2,str5);

  float temp2=tempsensor.getTempC(sensor2);
  dtostrf(temp2,4,2,str6);

  float temp3=tempsensor.getTempC(sensor3);
  dtostrf(temp3,4,2,str7);

  //-----------------------------------------------------------------
----

  //Show temperature values on serial monitor
  Serial.print(" | ");
  Serial.print(temp1);
  Serial.print(" | ");
  Serial.print(temp2);
  Serial.print(" | ");
  Serial.println(temp3);
}

void recordData(){
  //record sensor data in SD card
  myFile=SD.open("GPV_DATA.txt", FILE_WRITE);
  //if file open ok, write to it:
  if (myFile){
    myFile.print(str1);
    myFile.print(" | ");
    myFile.print(adcValue_V);   //Voltage adc value
    myFile.print(" | ");
    myFile.print(str2);
    myFile.print(" | ");
    myFile.print(adcValue_I);  //Current adc value
    myFile.print(" | ");
    myFile.print(str3);
    myFile.print(" | ");
    myFile.print(str4);
    myFile.print(" | ");
    myFile.print(str5);
    myFile.print(" | ");
    myFile.print(str6);
    myFile.print(" | ");
    myFile.println(str7);
    myFile.close();
  } else
  {
    Serial.println("File cannot be opened!");
  }


  //record sensor data into "data"
  //send "data" to nodemcu
  data=data+str1+","+str2+","+str3+","+str4+","+str5+","+str6+","+str7;
  espSerial.println(data);
  data=""; //clear data
}
```

APPENDIX H: Coding for Arduino Mega for FPV

```
#include <SoftwareSerial.h> //Software library (required for serial
communication)
#include <DHT.h> //DHT library (required for DHT11 sensor)
#include <OneWire.h> //OneWire library (required for accessing Dallas 1
wired device)
#include <DallasTemperature.h> //TEMP library (required for DS18B20
sensor)
#include <Adafruit_GFX.h> //TEXT library (required for fonts styles in
OLED)
#include <Adafruit_SSD1306.h> //OLED library (required for the OLED)
#include <SPI.h> //SPI library (required for the SD card)
#include <SD.h> //SD library (required for the SD card)
#include <RTClib.h> //RTC library (required for DS3231 RTC module)

#define DHTPIN A3 //Define the analog pin A3 as the data pin of DHT11
#define DHTTYPE DHT11 //Define the version of DHT sensor used
#define ONE_WIRE_BUS 4 //Define the digital pin 4 as the data pin of
DS18B20
#define SCREEN_WIDTH 128 //Define the OLED display width in pixels
#define SCREEN_HEIGHT 32 //Define the OLED display height in pixels
#define OLED_RESET 4 //Adafruit library requires this element to be
defined but it is not involved in I2C

DeviceAddress thermometerAddress; //Custom array type to hold 64 bit
device address
OneWire oneWire(ONE_WIRE_BUS); //Setup a oneWire instance to communicate
with temperature IC
DallasTemperature tempsensor(&oneWire);//Point oneWire as reference to
DallasTemperature library
DHT dht(DHTPIN, DHTTYPE);
File myFile; // Create a file in the SD card
RTC_DS3231 rtc; // RTC
Adafruit_SSD1306 display(SCREEN_WIDTH,SCREEN_HEIGHT,&Wire,OLED_RESET);
SoftwareSerial espSerial(10, 11);

const int chipSelect = 53; // Define pin 53 as the data pin of SD card
const int relay1Pin = 5; //Define pin 5 as the data pin of relay

//Variables for function of current sensor
int adcValue_I = 0;
float adcVoltage = 0.0;
float currentValue = 0.0;
float sensitivity = 0.185; //Sensitivity of current sensor
float vRef = 5.00;
float offsetVoltage = vRef/2.0; //ACS712 offset voltage

//Variables for function of voltage sensor
int adcValue_V = 0; //Voltage value read from analog pin
float vOUT = 0.0;
float vIN = 0.0;
float R1 = 30550.0; //First resistor value of the voltage divider
float R2 = 7505.0; //Second resistor value of the voltage divider
```

```
// Addresses of 4 DS18B20s
// Every DS18B20 sensor has its unique address
// Replace the following addresses with those of your DS18B20 sensors
uint8_t sensor1[8] = { 0x28, 0x53, 0x43, 0x26, 0x01, 0x00, 0x00, 0x1C };
uint8_t sensor2[8] = { 0x28, 0x6E, 0x9B, 0x25, 0x01, 0x00, 0x00, 0x53 };
uint8_t sensor3[8] = { 0x28, 0xD7, 0x47, 0x26, 0x01, 0x00, 0x00, 0x35 };
uint8_t sensor4[8] = { 0x28, 0xD4, 0x2F, 0x79, 0xA2, 0x16, 0x03, 0xB7 };

// Store sensor data into str
char str1[6];
char str2[6];
char str3[6];
char str4[6];
char str5[6];
char str6[6];
char str7[6];
char str8[6];
String data;                           //contains all sensor data to be
sent to nodemcu

void setup() {
  Serial.begin(115200); //Initialize Serial monitor with 115200 baud
rate
  espSerial.begin(115200);
  dht.begin();                         //Initialize the DHT humidity
sensor
  tempsensor.begin();                  //Initialize the DS18B20
temperature sensor
  rtc.begin();                         //Initialize the RTC module
  pinMode(relay1Pin, OUTPUT);          //Initialize the relay

  // setup for the SD card-------------------------------
  Serial.print("Initializing SD card...");

  if(!SD.begin(chipSelect)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");
  //open file-----------------------------------------
  myFile=SD.open("FPV_DATA.txt", FILE_WRITE);

  // if the file opened ok, write to it:
  if (myFile) {
    Serial.println("File opened ok");
    // print the headings for our data
    myFile.println();
    myFile.println("Date  |  Time | Voltage(V) | Voltage ADC |
Current(A) | Current ADC | Humidity(%)| Ambient Temp(C)| Temp 1(C)| Temp
2(C)| Temp 3(C)| Water Temp(C)");
  }
  myFile.close();

  //Display these titles on the serial monitor
  Serial.println("Date  |  Time | Voltage(V) | Voltage ADC | Current(A)
| Current ADC | Humidity(%)| Ambient Temp(C)| Temp 1(C)| Temp 2(C)| Temp
3(C)| Water Temp(C)");
```

```
  //Initialize the I2C with address 0x3C (for the 128x32)
  display.begin(SSD1306_SWITCHCAPVCC,0x3C);
  display.clearDisplay();
  display.setTextColor(WHITE, BLACK);
  display.display();                     //Used to display all the data
which is in buffer

  delay(2000);
}

void loop() {
  DateTime now = rtc.now(); //get time now
  int hh = now.hour();
  int mm = now.minute();
  int time_in_minutes = hh*60 + mm; //convert time now to minutes

  if (time_in_minutes >= 570) //start taking data if time now is
after/equals to 9:30am = 9*60+30 = 570 min
  {
    loggingTime();
    loggingVoltage();
    loggingCurrent();
    loggingHumidity();
    loggingTemperature();
    recordData();
    delay(113000); //data collection time interval
   }
}

void loggingTime(){
  DateTime timeNow = rtc.now();

  char dateBuffer[]="DD-MM-YYYY";
  char timeBuffer[]="hh:mm:ss";

  Serial.print(timeNow.toString(dateBuffer));
  Serial.print(" | ");
  Serial.print(timeNow.toString(timeBuffer));
  Serial.print(" | ");

  //record date and time in SD card
  myFile=SD.open("FPV_DATA.txt", FILE_WRITE);
  // if the file opened ok, write to it:
  if (myFile) {
    myFile.print(dateBuffer);
    myFile.print(" | ");
    myFile.print(timeBuffer);
    myFile.print(" | ");
  }
  myFile.close();
}

void loggingVoltage(){
  digitalWrite(relay1Pin,LOW);  //turn relay 1 off
  delay(2000);                  //Set the relay off for 2 second
  adcValue_V = analogRead(A2);
  vOUT = (adcValue_V*vRef)/1024.0;   //formula for arduino to interpret
the data
  vIN = (vOUT/(R2/(R1+R2)));      //formula for voltage calculation
  vIN = 0.9176*vIN;  //-----------------------------------------Error
adjusting formula
  dtostrf(vIN,5,2,str1);
  Serial.print(vIN);
  Serial.print("   |   ");
  Serial.print(adcValue_V);
  Serial.print("   |   ");

  //Display the voltage on the OLED display
  display.setTextSize(2);
  display.setCursor(0,0);
  display.print("V= ");
  display.print(vIN);
  display.print(" V");
  display.display();
}

void loggingCurrent(){
  digitalWrite(relay1Pin,HIGH); //Turn relay on
  delay(2000);                  //Set relay on for 2 second

  //--------------------take a number of analog samples and sum them up
  #define samples 10            //Define number of analog samples to be
taken per reading
  int sum1 = 0;
  unsigned char count = 0;
  while (count<samples)
  {
   sum1 += analogRead(A1);
   count++;
   delay(10);
  }
  //--------------------------------------------------------------------

  //Formula to calculate current
  adcValue_I = ((float)sum1/(float)samples);
  adcVoltage = (adcValue_I*vRef)/1024;
  currentValue = ((adcVoltage - offsetVoltage)/sensitivity);
  currentValue = 1.0107*currentValue;  //-------------------------------
-Error adjusting formula
  dtostrf(currentValue,5,2,str2);
  Serial.print(currentValue);
  Serial.print("   |   ");
  Serial.print(adcValue_I);
  Serial.print("   |   ");

  // Display the current on OLED display
  display.setCursor(0,17);
  display.print("I= ");
  display.print(currentValue);
  display.print(" A");
  display.display();
}
```

```
void loggingHumidity(){ //function for humidity sensor
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  // Read temperature as Celsius (the default)
  delay(2000);
  dtostrf(h,4,2,str3);
  dtostrf(t,4,2,str4);
  Serial.print(h);
  Serial.print(" | ");
  Serial.print(t);
}

void loggingTemperature(){ //function for temperature sensor
  //temperature comes in as a float with 1 decimal place
  tempsensor.requestTemperatures(); //request temperature sample from
the sensors through the one wire bus

  //Getting temperatures--------------
  //The circuit have ten DS18B20 on the same bus thus the data are
requested by their addresses
  //example: sensor1 holds the address for the first DS18B20 sensor on
the wire
  //getting temp reading with sensor1 address
  float temp1=tempsensor.getTempC(sensor1);
  //result is string with 5 position + \0 at the end
  //convert float to fprintf type string
  //format 4 positions with 2 decimal places
  //str contains the result
  dtostrf(temp1,4,2,str5);

  float temp2=tempsensor.getTempC(sensor2);
  dtostrf(temp2,4,2,str6);

  float temp3=tempsensor.getTempC(sensor3);
  dtostrf(temp3,4,2,str7);

  float waterTemp=tempsensor.getTempC(sensor4);
  dtostrf(waterTemp,4,2,str8);
  //-----------------------------------------------------------------------
----

  //Show temperature values on serial monitor
  Serial.print(" | ");
  Serial.print(temp1);
  Serial.print(" | ");
  Serial.print(temp2);
  Serial.print(" | ");
  Serial.print(temp3);
  Serial.print(" | ");
  Serial.println(waterTemp);
}

void recordData(){
  //record sensor data into SD card
  myFile=SD.open("FPV_DATA.txt", FILE_WRITE);
  //if file open ok, write to it:
```

```
  if (myFile){
    myFile.print(str1);
    myFile.print(" | ");
    myFile.print(adcValue_V);  //Voltage adc value
    myFile.print(" | ");
    myFile.print(str2);
    myFile.print(" | ");
    myFile.print(adcValue_I);  //Current adc value
    myFile.print(" | ");
    myFile.print(str3);
    myFile.print(" | ");
    myFile.print(str4);
    myFile.print(" | ");
    myFile.print(str5);
    myFile.print(" | ");
    myFile.print(str6);
    myFile.print(" | ");
    myFile.print(str7);
    myFile.print(" | ");
    myFile.println(str8);
    myFile.close();
  } else
  {
    Serial.println("File cannot be opened!");
  }


  //record sensor data into "data"
  //send "data" to nodemcu

data=data+str1+","+str2+","+str3+","+str4+","+str5+","+str6+","+str7+","
+str8;
  espSerial.println(data);
  data=""; //clear data
}
```

APPENDIX I: Coding for ESP8266 NodeMCU for GPV

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

#define ON_Board_LED 2

//-------- Customise these values -----------
const char* ssid = "myInternet"; // ID of your internet or wifi name
const char* password = "kaizin1234"; // Password of your internet


//---------Host & httpsPort------------------
const char* host = "script.google.com";
const int httpsPort = 443;

//--> Create a WiFiClientSecure object.
WiFiClientSecure client;
//--> spreadsheet script ID
String GAS_ID =
"AKfycbw8L5qCsUDKqJdKBRHeaPTQjxnmHq0zjnuS0f_xy8ISYePtQQ8G0R0BCPK5hfiIjTd
2";
//ID is required to be changed if changes have made in the App Script
//Obtain new deployment ID from the App script

//define variables for the received data,total variable is 13
float Val_1, Val_2, Val_3, Val_4, Val_5, Val_6, Val_7;
String myString;// complete message from arduino, which consists of
sensors data
char rdata; // received characters byte per byte from arduino

// Space to store values to send
char str_val_1[6];
char str_val_2[6];
char str_val_3[6];
char str_val_4[6];
char str_val_5[6];
char str_val_6[6];
char str_val_7[6];

unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 500;  //the value is a number of
milliseconds


void setup() {
  // put your setup code here, to run once:
  //open serial communications and wait for port to open:
  Serial.begin(115200);
  delay(500);
  while (!Serial){
    ;//wait for serial port to connect. Needed for native USB port only
  }
  wifiConnect();
  startMillis = millis(); //record the program start time
}

void loop() {
  // put your main code here, to run repeatedly:
  // get the current "time" (actually the number of milliseconds since
the program started)
  currentMillis = millis();

  if(WiFi.status() == WL_CONNECTED)
  {
    if (currentMillis - startMillis >= period)
    {
      sendData();  //send the data to Google spreadsheet
      startMillis = currentMillis;
    }
  } else
  {
    WiFi.disconnect();
    wifiConnect();  //reconnect to wifi if disconnected
  }

}

void wifiConnect(){
  WiFi.begin(ssid, password); //--> Connect to your WiFi router
  Serial.println("");
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(ON_Board_LED, HIGH); //--> Turn off Led On Board

  //Wait for connection
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  //Make the On Board Flashing LED on the process of connecting to the
wifi router.
  digitalWrite(ON_Board_LED, LOW);
  delay(250);
  digitalWrite(ON_Board_LED, HIGH);
  delay(250);

  }

  //--> Turn off the On Board LED when it is connected to the wifi
router.
  digitalWrite(ON_Board_LED, HIGH);
  //If successfully connected to the wifi router,
  //the IP Address that will be visited is displayed in the serial
monitor
  Serial.println("");
  Serial.print("Successfully connected to : ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  //----------------------------------------
  client.setInsecure();
}

void sendData() {
// Subroutine for sending data to Google Sheets
  Serial.println("==========");
  Serial.print("connecting to ");
  Serial.println(host);

  //Connect to Google host
  if (!client.connect(host, httpsPort)) {
    Serial.println("connection failed");
    return;
  }

  if (Serial.available()>0) {
    rdata=Serial.read(); //read data from arduino
    myString = myString + rdata; //change data from char to string
    if ( rdata == '\n') {
      //---------------------------------------------Processing data
      String l = getValue(myString, ',', 0);
      String m = getValue(myString, ',', 1);
      String n = getValue(myString, ',', 2);
      String o = getValue(myString, ',', 3);
      String p = getValue(myString, ',', 4);
      String q = getValue(myString, ',', 5);
      String r = getValue(myString, ',', 6);

      Val_1 = l.toFloat();
      Val_2 = m.toFloat();
      Val_3 = n.toFloat();
      Val_4 = o.toFloat();
      Val_5 = p.toFloat();
      Val_6 = q.toFloat();
      Val_7 = r.toFloat();

      // float value is copied onto str_val
      // 4 is mininum width, 2 is precision
      dtostrf(Val_1, 4, 2, str_val_1);
      dtostrf(Val_2, 4, 2, str_val_2);
      dtostrf(Val_3, 4, 2, str_val_3);
      dtostrf(Val_4, 4, 2, str_val_4);
      dtostrf(Val_5, 4, 2, str_val_5);
      dtostrf(Val_6, 4, 2, str_val_6);
      dtostrf(Val_7, 4, 2, str_val_7);

      myString=""; //refresh to clear the old data
      //-----------------------------------------------end of data
processing

      //Sending data
      String url = "/macros/s/" + GAS_ID + "/exec?vIN=" + str_val_1+
"&currentVal=" + str_val_2
      + "&humidity=" + str_val_3 + "&ambTemp=" + str_val_4 + "&temp1=" +
str_val_5 + "&temp2=" + str_val_6
      + "&temp3=" + str_val_7 ;
      Serial.print("requesting URL: ");
      Serial.println(url);

      client.print(String("GET ") + url + " HTTP/1.1\r\n" +
         "Host: " + host + "\r\n" +
         "User-Agent: BuildFailureDetectorESP8266\r\n" +
         "Connection: close\r\n\r\n");

      Serial.println("request sent");
      //----------------------------------------

      //----------------------------------------Checking whether the
data was sent successfully or not
      while (client.connected()) {
        String line = client.readStringUntil('\n');
        if (line == "\r") {
          Serial.println("headers received");
          break;
        }
      }

      String line = client.readStringUntil('\n');
      if (line.startsWith("{\"state\":\"success\"")) {
        Serial.println("esp8266/Arduino CI successfull!");
      } else {
        Serial.println("esp8266/Arduino CI has failed");
      }
      Serial.print("reply was : ");
      Serial.println(line);
      Serial.println("closing connection");
      Serial.println("==========");
      Serial.println();
    }
  }
}

//Subroutine to separate sensor values received from arduino
String getValue(String data, char separator, int index)
{
 int found = 0;
 int strIndex[] = { 0, -1 };
 int maxIndex = data.length() - 1;
 for (int i = 0; i <= maxIndex && found <= index; i++) {
 if (data.charAt(i) == separator || i == maxIndex) {
 found++;
 strIndex[0] = strIndex[1] + 1;
 strIndex[1] = (i == maxIndex) ? i + 1 : i;
 }
 }
 return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

APPENDIX J: Coding for ESP8266 NodeMCU for FPV

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

#define ON_Board_LED 2

//-------- Customise these values -----------
const char* ssid = "myInternet"; // ID of your internet or wifi name
const char* password = "kaizin1234"; // Password of your internet


//---------Host & httpsPort------------------
const char* host = "script.google.com";
const int httpsPort = 443;

//--> Create a WiFiClientSecure object.
WiFiClientSecure client;
//--> spreadsheet script ID
String GAS_ID =
"AKfycbybNFWEMtR8_ZZW0qoeSOqvpPci9vLRod_6z_eK_5V04cS6WntQ0zaC0Jdu2CkROkM
QSw";
//ID is required to be changed if changes have made in the App Script
//Obtain new deployment ID from the App script

//define variables for the received data,total variable is 13
float Val_1, Val_2, Val_3, Val_4, Val_5, Val_6, Val_7, Val_8;
String myString;// complete message from arduino, which consists of
sensors data
char rdata; // received characters byte per byte from arduino

// Space to store values to send
char str_val_1[6];
char str_val_2[6];
char str_val_3[6];
char str_val_4[6];
char str_val_5[6];
char str_val_6[6];
char str_val_7[6];
char str_val_8[6];

unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 500;  //the value is a number of
milliseconds


void setup() {
  // put your setup code here, to run once:
  //open serial communications and wait for port to open:
  Serial.begin(115200);
  delay(500);
  while (!Serial){
    ;//wait for serial port to connect. Needed for native USB port only
  }
  wifiConnect();
  startMillis = millis(); //record the program start time
}

void loop() {
  // put your main code here, to run repeatedly:
  // get the current "time" (actually the number of milliseconds since
the program started)
  currentMillis = millis();

  if(WiFi.status() == WL_CONNECTED)
  {
    if (currentMillis - startMillis >= period)
    {
      sendData();  //send the data to Google spreadsheet
      startMillis = currentMillis;
    }
  } else
  {
    WiFi.disconnect();
    wifiConnect();  //reconnect to wifi if disconnected
  }

}

void wifiConnect(){
  WiFi.begin(ssid, password); //--> Connect to your WiFi router
  Serial.println("");
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(ON_Board_LED, HIGH); //--> Turn off Led On Board

  //Wait for connection
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  //Make the On Board Flashing LED on the process of connecting to the
wifi router.
  digitalWrite(ON_Board_LED, LOW);
  delay(250);
  digitalWrite(ON_Board_LED, HIGH);
  delay(250);
  }

  //--> Turn off the On Board LED when it is connected to the wifi
router.
  digitalWrite(ON_Board_LED, HIGH);
  //If successfully connected to the wifi router,
  //the IP Address that will be visited is displayed in the serial
monitor
  Serial.println("");
  Serial.print("Successfully connected to : ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  //----------------------------------------
  client.setInsecure();
}

void sendData() {
// Subroutine for sending data to Google Sheets
  Serial.println("==========");
  Serial.print("connecting to ");
  Serial.println(host);

  //Connect to Google host
  if (!client.connect(host, httpsPort)) {
    Serial.println("connection failed");
    return;
  }

  if (Serial.available()>0) {
    rdata=Serial.read(); //read data from arduino
    myString = myString + rdata; //change data from char to string
    if ( rdata == '\n') {
      //---------------------------------------------Processing data
      String l = getValue(myString, ',', 0);
      String m = getValue(myString, ',', 1);
      String n = getValue(myString, ',', 2);
      String o = getValue(myString, ',', 3);
      String p = getValue(myString, ',', 4);
      String q = getValue(myString, ',', 5);
      String r = getValue(myString, ',', 6);
      String s = getValue(myString, ',', 7);

      Val_1 = l.toFloat();
      Val_2 = m.toFloat();
      Val_3 = n.toFloat();
      Val_4 = o.toFloat();
      Val_5 = p.toFloat();
      Val_6 = q.toFloat();
      Val_7 = r.toFloat();
      Val_8 = s.toFloat();

      // float value is copied onto str_val
      // 4 is mininum width, 2 is precision
      dtostrf(Val_1, 4, 2, str_val_1);
      dtostrf(Val_2, 4, 2, str_val_2);
      dtostrf(Val_3, 4, 2, str_val_3);
      dtostrf(Val_4, 4, 2, str_val_4);
      dtostrf(Val_5, 4, 2, str_val_5);
      dtostrf(Val_6, 4, 2, str_val_6);
      dtostrf(Val_7, 4, 2, str_val_7);
      dtostrf(Val_8, 4, 2, str_val_8);

      myString=""; //refresh to clear the old data
      //-----------------------------------------------end of data
processing

      //Sending data
      String url = "/macros/s/" + GAS_ID + "/exec?vIN=" + str_val_1+
"&currentVal=" + str_val_2
```

```
      + "&humidity=" + str_val_3 + "&ambTemp=" + str_val_4 + "&temp1=" +
str_val_5
      + "&temp2=" + str_val_6 + "&temp3=" + str_val_7 + "&waterTemp=" +
str_val_8;
      Serial.print("requesting URL: ");
      Serial.println(url);

      client.print(String("GET ") + url + " HTTP/1.1\r\n" +
         "Host: " + host + "\r\n" +
         "User-Agent: BuildFailureDetectorESP8266\r\n" +
         "Connection: close\r\n\r\n");

      Serial.println("request sent");
      //----------------------------------------

      //----------------------------------------Checking whether the
data was sent successfully or not
      while (client.connected()) {
        String line = client.readStringUntil('\n');
        if (line == "\r") {
          Serial.println("headers received");
          break;
        }
      }

      String line = client.readStringUntil('\n');
      if (line.startsWith("{\"state\":\"success\"")) {
        Serial.println("esp8266/Arduino CI successfull!");
      } else {
        Serial.println("esp8266/Arduino CI has failed");
      }
      Serial.print("reply was : ");
      Serial.println(line);
      Serial.println("closing connection");
      Serial.println("==========");
      Serial.println();
    }
  }
}

//Subroutine to separate sensor values received from arduino
String getValue(String data, char separator, int index)
{
 int found = 0;
 int strIndex[] = { 0, -1 };
 int maxIndex = data.length() - 1;
 for (int i = 0; i <= maxIndex && found <= index; i++) {
 if (data.charAt(i) == separator || i == maxIndex) {
 found++;
 strIndex[0] = strIndex[1] + 1;
 strIndex[1] = (i == maxIndex) ? i + 1 : i;
 }
 }
 return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

APPENDIX K: Coding for Google App Script for GPV

```
function doGet(e) {
  Logger.log( JSON.stringify(e) );
  var result = 'Ok';
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
  }
  else {
    var sheet_id = '1qKn2gYADXMVUAtuShwHSQmFZ3oVL6GGmEco3ea4-2qI';
        // Spreadsheet ID
    var sheet = SpreadsheetApp.openById(sheet_id).getSheetByName("All
Data Record");
    var newRow = sheet.getLastRow() + 1;

    var rowData = [];
    var date = new Date();
    var Curr_Date = Utilities.formatDate(date, "Asia/Kuala_Lumpur", 'MM-
dd-yyyy');

    rowData[0] = Curr_Date; // Date in column A
    sheet.getRange(sheet.getLastRow(),1).setNumberFormat("dd-MMM-yyyy");
    var Curr_Time = Utilities.formatDate(date, "Asia/Kuala_Lumpur",
'HH:mm:ss');
    rowData[1] = Curr_Time; // Time in column B

    for (var param in e.parameter) {
      Logger.log('In for loop, param=' + param);
      var value = stripQuotes(e.parameter[param]);
      Logger.log(param + ':' + e.parameter[param]);
      switch (param) {
        case 'vIN' :
          rowData[2] = value; // Vin in column C
          result += 'Vin written on column C. ';
          break;
        case 'currentVal' :
          rowData[3] = value; // Current in column D;
          result += 'Current written on column D. ';
          break;
        case 'humidity':
          rowData[4] = value; // Humidity in column E
          result = 'Humidity written on column E. ';
          break;
        case 'ambTemp':
          rowData[5] = value; // Ambient temp in column F
          result = 'Ambient temperature written on column F. ';
          break;
        case 'temp1':
          rowData[6] = value; // Temperature 1 in column G
          result += 'Temperature 1 written on column G. ';
          break;
        case 'temp2':
          rowData[7] = value; // Temperature 2 in column H
          result += 'Temperature 2 written on column H. ';
          break;
        case 'temp3':
          rowData[8] = value; // Temperature 3 in column I
          result += 'Temperature 3 written on column I. ';
          break;
        default:
          result = "unsupported parameter";
      }
    }
    Logger.log(JSON.stringify(rowData));
    var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
    newRange.setValues([rowData]);
  }
  return ContentService.createTextOutput(result);
}
function stripQuotes( value ) {
  return value.replace(/^["']|['"]$/g, "");
}
```

APPENDIX L: Coding for Google App Script for FPV

```
function doGet(e) {
  Logger.log( JSON.stringify(e) );
  var result = 'Ok';
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
  }
  else {
    var sheet_id = '1P2aIakEACvnCPuFFa0lgN8z-CcwsqbvxoWirtJgYu0w';
        // Spreadsheet ID
    var sheet = SpreadsheetApp.openById(sheet_id).getSheetByName("All
Data Record");
    var newRow = sheet.getLastRow() + 1;

    var rowData = [];
    var date = new Date();
    var Curr_Date = Utilities.formatDate(date, "Asia/Kuala_Lumpur",'MM-
dd-yyyy');

    rowData[0] = Curr_Date; // Date in column A
    sheet.getRange(sheet.getLastRow(),1).setNumberFormat("dd-MMM-yyyy");
    var Curr_Time = Utilities.formatDate(date, "Asia/Kuala_Lumpur",
'HH:mm:ss');
    rowData[1] = Curr_Time; // Time in column B

    for (var param in e.parameter) {
      Logger.log('In for loop, param=' + param);
      var value = stripQuotes(e.parameter[param]);
      Logger.log(param + ':' + e.parameter[param]);
      switch (param) {
        case 'vIN' :
          rowData[2] = value; // Vin in column C
          result += 'Vin written on column C. ';
          break;
        case 'currentVal' :
          rowData[3] = value; // Current in column D;
          result += 'Current written on column D. ';
          break;
        case 'humidity':
          rowData[4] = value; // Humidity in column E
          result = 'Humidity written on column E. ';
          break;
        case 'ambTemp':
          rowData[5] = value; // Ambient Temp in column F
          result = 'Ambient temperature written on column F. ';
          break;
        case 'waterTemp':
          rowData[6] = value; // Water temperature in column G
          result = 'Water temperature written on column G. ';
          break;
        case 'temp1':
          rowData[7] = value; // Temperature 1 in column H
          result += 'Temperature 1 written on column H. ';
          break;
        case 'temp2':
          rowData[8] = value; // Temperature 2 in column I
          result += 'Temperature 2 written on column I. ';
          break;
        case 'temp3':
          rowData[9] = value; // Temperature 3 in column J
          result += 'Temperature 3 written on column J. ';
          break;
        default:
          result = "unsupported parameter";
      }
    }
    Logger.log(JSON.stringify(rowData));
    var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
    newRange.setValues([rowData]);
  }
  return ContentService.createTextOutput(result);
}
function stripQuotes( value ) {
  return value.replace(/^["']|['"]$/g, "");
}
```