

**AN IMPROVED AUTOMATED ELECTRONIC SENSOR SYSTEM WITH  
IOT**

**CREMENT ONG WEN YAO**


**A project report submitted in partial fulfilment of the  
requirements for the award of the degree of  
Bachelor of Technology (Honours) in Electronic Systems**

**Faculty of Engineering and Green Technology  
Universiti Tunku Abdul Rahman**

**May 2022**

## DECLARATION

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

Signature :  \_\_\_\_\_

Name : CREMENT ONG WEN YAO

ID No. : 19AGB03721

Date : 30 September 2022

### APPROVAL FOR SUBMISSION

I certify that this project report entitled “**AN IMPROVED AUTOMATED ELECTRONIC SENSOR SYSTEM WITH IOT**” was prepared by **CREMENT ONG WEN YAO** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) in Electronic Systems at Universiti Tunku Abdul Rahman.

Approved by,



Signature : \_\_\_\_\_

Supervisor : Dr. Lee Yu Jen

Date : 30 September 2022

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

© 2022, Crement Ong Wen Yao. All right reserved.

Specially dedicated to  
my mother and father

## **ACKNOWLEDGEMENTS**

Firstly, I would like to thank Universiti Tunku Abdul Rahman for providing me with the opportunity to complete this research and the necessary equipment and facilities.

In addition, I would like to thank my supervisor, Dr. Lee Yu Jen, for giving me a lot of helpful guidance and advice throughout the development of this project. In addition, I'd like to thank my moderator, Ts. Dr. Toh Pek Lan, for providing me with helpful feedback to improve the completion of my Final Year Project Report.

Furthermore, I am grateful to the Department of Electronic Engineering in the Faculty of Engineering and Green Technology laboratory managers, Mr. Choon Chee Ming, Miss Norazuani Binti Zaharudin, and Mr. Thong Marn Foo, who offered me a great deal of help and assistance and assistance on the laboratory equipment and facilities.

The study was completed with the assistance and support of friends who lent me a hand when I encountered difficulties with this assignment. Lastly, I would like to express my gratitude to my parents. They gave me both financial and emotional help. I wish to express my sincere appreciation to each of you. This Final Year Project has been completed thanks to your encouragement.

## **AN IMPROVED AUTOMATED ELECTRONIC SENSOR SYSTEM WITH IOT**

### **ABSTRACT**

An existing automated electronic sensor system for surface roughness profiling is improved and renamed “ProSight Surface Profiler”. It is aimed to integrate a newly developed automated surface roughness data analysis programme that connects with the prototype surface profiler. The application is managed able to automate the output of surface roughness parameters, including RMS and autocorrelation length. The software supports real-time data analysis. The software can merge effortlessly with the hardware. The resulting graph is flexible, allowing the user to examine it from any angle desired. Moreover, the user can convert the graph into an image with a single click. The software installation is straightforward, with an executable link. The Blynk app is used to operate the Raspberry Pi's GPIO remotely. The data is uploaded automatically to the cloud storage without the need to extract the data using a cable. With these two functionalities, an Internet of Things-integrated contactless machine is developed. The ProSight Surface Profiler application is user-friendly, fast, and effortlessly controlled for surface roughness analysis.

## TABLE OF CONTENTS

<b>DECLARATION</b>	<b>ii</b>
<b>APPROVAL FOR SUBMISSION</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>vi</b>
<b>ABSTRACT</b>	<b>vii</b>
<b>TABLE OF CONTENTS</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>xii</b>
<b>LIST OF FIGURES</b>	<b>xiv</b>
<b>LIST OF SYMBOLS / ABBREVIATIONS</b>	<b>xviii</b>
<b>LIST OF APPENDICES</b>	<b>xxii</b>

## CHAPTER

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background	1
	1.2 Problem Statements	3
	1.3 Aims and Objectives	4
	1.4 Organisation of Thesis	5
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>7</b>
	2.1 Overview	7
	2.2 Automated Electronic Sensor System	8
	2.2.1 Physical Planning and Design	8
	2.2.2 Hardware Specifications	9
	2.2.3 Software Specifications	15
	2.2.4 System Flow Chart	17
	2.2.5 Case Study: Accuracy Test	18



2.2.6 Summary	20
2.3 Graph Optimisation	20
2.3.1 Moving Average Filter	21
2.3.2 Median Filter	22
2.3.3 Savitzky-Golay Filter	24
2.4 Surface Roughness Parameters	26
2.4.1 Standard Deviation of Surface Heights / RMS Height	27
2.4.2 Surface Autocorrelation Length	28
2.5 Programming Languages	29
2.5.1 Python	30
2.5.2 Java	31
2.5.3 C++	32
2.6 GUI Frameworks	32
2.6.1 Tkinter	33
2.6.2 Kivy	34
2.7 Existing Surface Profiler Analysis Software	34
2.7.1 MicroEpsilon scanCONTROL	35
2.7.2 In-Sight Laser Profile Software	36
2.7.3 Comparison of features	37
2.8 IoT Platform	38
2.8.1 RaspController	39
2.8.2 Blynk	40
2.9 Cloud Storage	41
2.9.1 Google Drive	41
2.9.2 Dropbox	42
2.9.3 Comparison of features	42
<b>3 METHODOLOGY</b>	<b>43</b>
3.1 Overview	43
3.2 Selection of Programming language	43
3.3 GUI Framework Selection	44
3.4 System Overview	45
3.5 System Performance Definition	47
3.6 Technologies and Tools involved	48

3.7	Graph Optimisation	48
3.7.1	Moving Average Filter	49
3.7.2	Median Filter	51
3.7.3	Savitzky-Golay Filter	52
3.7.4	Conclusion (Graph Optimisation)	54
3.8	RMS Height	54
3.9	Surface Roughness Autocorrelation Length	55
3.10	Selection of the IoT platform	57
3.11	Selection of the cloud storage platform	60
3.12	Implementation and Testing	64
3.12.1	Import of library	64
3.12.2	Debugging errors	65
3.12.3	Global definition	66
3.12.4	Design for Homepage (Class I)	67
3.12.5	Widgets for Homepage (Class II)	68
3.12.6	PageOne (Class III)	69
3.12.7	Design of PageTwo (Class IV)	72
3.12.8	Design of PageThree (Class V)	76
3.12.9	Final Call	79
3.12.10	Setup for an executable extension	79
3.13	Project Management	80
<b>4</b>	<b>RESULTS AND DISCUSSION</b>	<b>81</b>
4.1	Introduction	81
4.2	Surface Roughness Parameter	81
4.2.1	Surface Roughness Parameters (Calculated with Excel)	82
4.2.2	Surface Roughness Parameters (Software Algorithm)	84
4.2.3	Discussion of the results	87
4.3	Graph optimisation	88
4.3.1	Case Study 1	89
4.3.2	Case Study 2	90
4.3.3	Case Study 3	91
4.3.4	Case Study 4	93
4.3.5	Case Study 5	94

4.3.6 Case Study 6	96
4.3.7 Case Study 7	97
4.3.8 Case Study 8	99
4.3.9 Summary of the case study of graph optimisation	100
4.4 Extra	101
<b>5 CONCLUSIONS AND RECOMMENDATIONS</b>	<b>103</b>
5.1 Conclusion	103
5.2 Limitations	104
5.3 Recommendations for Future Improvement	104
<b>REFERENCES</b>	<b>105</b>
<b>APPENDICES</b>	<b>109</b>

## LIST OF TABLES

<b>TABLE</b>	<b>TITLE</b>	<b>PAGE</b>
1.1	Table of organisation of the thesis	5
2.1	Table summarising the pros and cons of the automated surface profiler	20
2.2	Table summarising the pros and cons of Python	30
2.3	Table summarising the pros and cons of Java	31
2.3	Table summarising the pros and cons of C++	32
2.4	Table summarising the pros and cons of Tkinter	33
2.5	Table summarising the pros and cons of Kivy	34
2.6	Table of feature comparison between the existing surface profiler analysis software	37
2.7	Table summarising the pros and cons of RaspController	39
2.8	Table summarising the pros and cons of Blynk	40
2.9	Table of feature comparisons between Google Drive and Dropbox	42
3.1	Table of laptop specifications	48
3.2	Optimised graph (Dark Blue) of Moving Average Filter with different values of $n$	50
3.3	Optimised graph (Red) of Median Filter with different values of $n$	51
3.4	Optimised graph (Green) of Median Filter with different values of $n$	53

4.1	Autocorrelation Function graph and autocorrelation length of 8 sites (Calculated with Excel)	82
4.2	RMS height of 9 sites (Calculated with Excel)	84
4.3	Autocorrelation Function graph and autocorrelation length of 8 sites (Software Algorithm)	84
4.4	RMS height of 9 sites (Software algorithm)	86
4.5	Percentage error of the results obtained for autocorrelation length	87
4.6	Percentage error of the results obtained for RMS height	88
4.7	Raw data graph compared with the optimised graph to the following object in case study 1.	89
4.8	Raw data graph compared with the optimised graph to the following object in case study 2.	90
4.9	Raw data graph compared with the optimised graph to the following object in case study 3.	92
4.10	Raw data graph compared with the optimised graph to the following object in case study 4.	93
4.11	Raw data graph compared with the optimised graph to the following object in case study 5.	95
4.12	Raw data graph compared with the optimised graph to the following object in case study 6	96
4.13	Raw data graph compared with the optimised graph to the following object in case study 7.	98
4.14	Raw data graph compared with the optimised graph to the following object in case study 8.	99

## LIST OF FIGURES

<b>FIGURE</b>	<b>TITLE</b>	<b>PAGE</b>
1.1	Cognex Laser Profiler (Cognex, 2018)	1
2.1	Sketch of surface profiler (Kang, 2016)	8
2.2	Graph of distance vs voltage (Kang, 2016)	9
2.3	Experiment setup to measure the distance to voltage (Kang, 2016)	10
2.4	Graph of distance vs voltage (average) (Kang, 2016)	10
2.5	ADC0804 IC connection (Kang, 2016)	11
2.6	Connection of SN74166 (Kang, 2016)	12
2.7	Dissembled slider from HP inkjet printer (Kang, 2016)	12
2.8	DC motor attached to the slider (Kang, 2016)	13
2.9	Line follower infrared sensor (Kang, 2016)	13
2.10	Sketch of the encoder system (Kang, 2016)	14
2.11	DC-DC buck converter module (Kang, 2016)	14
2.12	Connection of battery, converter to the supply other devices (Kang, 2016)	15
2.13	Duty cycle of PWM	15
2.14	Flow chart of surface profiler system (Kang, 2016)	17
2.15	Setup of the case study (Kang, 2016)	18

2.16	The best result of the case study (Kang, 2016)	19
2.17	The worst result of the case study (Kang, 2016)	19
2.18	Illustration of least-squares smoothing by locally fitting a second-degree polynomial (solid line) to five input samples (Schafer, 2011)	25
2.19	Random height variations superimposed on a periodic surface (Ulaby, Moore and Fung, 1981)	26
2.20	Random height variations superimposed on a flat surface (Ulaby, Moore and Fung, 1981)	26
2.21	Autocorrelation Function (Ulaby, Moore and Fung, 1981)	29
2.22	MicroEpsilon scanCONTROL software interface (Micro-Epsilon Messtechnik, 2022)	35
2.23	In-Sight Laser Profile software interface (Cognex, 2018)	36
3.1	Flowchart of system overview	45
3.2	Flowchart of software process	46
3.3	The box is used as the reference to the graph optimising filter selection	49
3.4	Function of moving average filter in Python	49
3.5	Command of median filter in Python	51
3.6	Command for Savitzky-Golay Filter	52
3.7	Command to calculate RMS height of surface roughness	54
3.8	Flowchart of finding the autocorrelation length of surface roughness	55
3.9	Finding the intersection point between $y=0.36788$ and autocorrelation function graph	56
3.10	Command for the generation of autocorrelation function graph	56

3.11	Code to find the intersection point of 2 lines	57
3.12	Command line to sync Raspberry Pi with the Blynk Cloud	58
3.13	Function of controlling the GPIO pin virtually	58
3.14	Flowchart of the Blynk controlling the GPIO	59
3.15	Blynk Cloud dashboard showing the device status and virtual pin	59
3.16	Creating OAuth client ID in Google Developer Console	61
3.17	A copy of information after the remote is added	61
3.18	Rclone@.service file content	62
3.19	Mounted Google Drive in Linux system	63
3.20	Google Drive sync with the mounted drive in Linux	63
3.21	Command for importing the libraries of the program	65
3.22	Command to increase the recursion limit	65
3.23	Command to suppress the error of ShapelyDeprecationWarning	66
3.24	Global definition in the program	66
3.25	Class for Homepage design interface	67
3.26	Frame structure of Homepage	68
3.27	Class of StartPage Widgets	68
3.28	Widgets and Fonts on Homepage	69
3.29	PageOne initialisation function	69
3.30	PageOne widgets and interface command	70
3.31	PageOne search file function	70
3.32	PageOne read excel file function	71



3.33	PageOne clear data in the list function	71
3.34	PageOne interface (without selected file)	72
3.35	PageOne interface (with the selected file)	72
3.36	PageTwo initialisation function	73
3.37	PageTwo graph plotting function	74
3.38	PageTwo new analysis function	74
3.39	PageTwo interface (without selected file)	75
3.40	PageTwo interface (with the selected file)	75
3.41	PageThree initialisation function	76
3.42	PageThree Surface Roughness Parameters Calculation and Graph Plotting function	77
3.43	PageThree new analysis function	77
3.44	PageTwo interface (without selected file)	78
3.45	PageTwo interface (with selected file)	78
3.46	Coding part for the initialising stage	79
3.47	Coding part for the setup of executable extension	79
3.48	Gantt Chart of FYP1	80
3.49	Gantt Chart of FYP2	80
4.1	Replace the old Raspberry Pi 2 with Raspberry Pi 3	101
4.2	Replace a new step-down converter (12V to 5V)	102
4.3	Replace a new encoder strip	102

## LIST OF SYMBOLS / ABBREVIATIONS

$\bar{z}$	mean
$\sigma$	standard deviation
$\lambda$	wavelength
$e$	Euler's number [2.71828]
$\infty$	infinity
$l$	surface correlation length
RMS	Root Mean Square
GPIO	General-Purpose Input/Output
GUI	Graphical User Interface
IoT	Internet of Things
HDMI	High-Definition Multimedia Interface
LAN	Local Area Network
PCB	Printed Circuit Board
USB	Universal Serial Bus
OS	Operating System
IR	Infrared
ADC	Analog-to-Digital Converter
IC	Integrated Circuit
PISO	Parallel-In/Serial-Out
I/O	Input/Output
DC	Direct Current
PWM	Pulse-Width Modulation
IP	Internet Protocol
DSP	Digital Signal Processor
API	Application Programming Interface
RAM	Random-Access Memory

UI	User Interface
PC	Personal Computer
HMI	Human-Machine Interface
PDF	Portable Document Format
CAD	Computer-Aided Design
WIFI	Wireless Fidelity

**LIST OF APPENDICES**

<b>APPENDIX</b>	<b>TITLE</b>	<b>PAGE</b>
A	Coding of Main Program	109
B	Coding of Setup of Executional Extension	115
C	Profiler System Coding	115

## CHAPTER 1

### INTRODUCTION

#### 1.1 Background

In geomorphology, the earth's surface roughness, also known as landscape, terrain, and topographic roughness, is directly related to the unevenness of surface elevation values. (Day and Chenoweth, 2013). The surface roughness measurements are then fed into surface dynamics modelling. The application examples are soil erosion, runoff prediction, and microwave remote sensing scattering modelling and calibration. (Gharechelou, Tateishi and A.Johnson, 2018). One of the laser profilers from Cognex is shown in Figure 1.1 below.

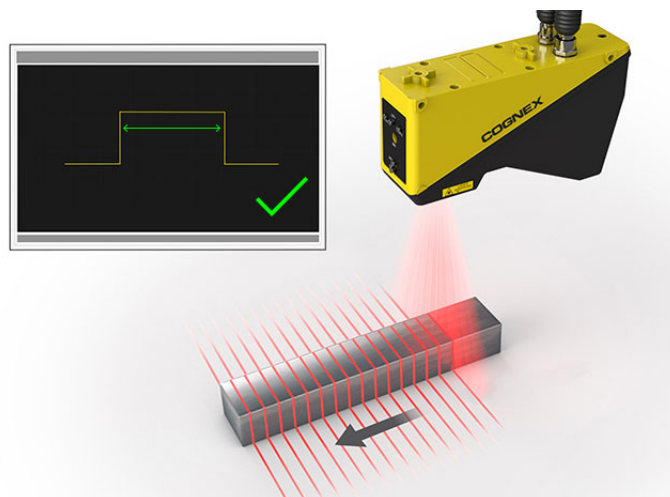


Figure 1.1: Cognex Laser Profiler (Cognex, 2018)

In the past, researchers employed a variety of approaches to estimate roughness parameters based on ground data. Surface measurement is classified into two types: contact and non-contact. Non-contact measurement is faster than contact because contact-type instruments must touch and transverse the object. Non-contact devices can measure multiple points without placing pressure on the item. Although non-contact measurements have advantages, contact-based measurements are preferable in environments with poor cleanliness levels. (MTI Instruments, 2021). As the dust is not affecting the measurement as the noise anymore. An example of a non-contact instrument is a pin profilometer. It requires physical contact with the land and can be time-consuming to conduct (Gharechelou, Tateishi and A.Johnson, 2018). In contrast, non-contact instruments include laser profilers and ultrasonic techniques.

Surface roughness affects natural surfaces' scattering and emission characteristics in microwave remote sensing. The study of height differences recorded along transects characterises land surface roughness. The degree of random surface roughness is defined by statistical characteristics assessed by the wavelength units of the observing sensor (Campbell and Wynne, 2011). Surface roughness measurement is critical in microwave remote sensing. However, the concept of roughness has yet to be fully resolved. Methods of evaluating roughness have developed independently with distinct yet parallel approaches emerging in different disciplines, often originating from outside the Earth Sciences in engineering sciences (Smith, 2014).

The research is inspired by the application of understanding the surface roughness parameters by Dr. Lee Yu Jen and other researchers in Ross Island, Antarctica. The researchers applied remote sensing on the snow and ice surface with the surface profiler to deduce its physical properties and parameters such as autocorrelation length and RMS surface height. Sea ice is a complex, polycrystalline combination of pure ice with random brine and air inclusions, whose volume fraction and geometry drastically depend on temperature, age, and growth conditions, making research difficult. In addition, the surface of the sea ice has varying degrees of roughness and may be coated with a layer of snow, which is still another complex random composite with variable microstructure (Koay et al., 2017). Excel calculations must be performed manually in order to extract surface roughness parameters.

## 1.2 Problem Statements

### i. High instrument cost for a laser profiler

Developing a non-contact measurement is costly; the sensor's accuracy can be affected by many factors. In order to maintain the precision of the measured data, the instrument's specifications must decide cautiously. The specifications include:

- Building material
- Type of sensor
- Safety measure

The considerations above caused the manufacturers to raise their costs. Also, it is difficult to find spare parts replacement. This is a burden for most users.

### ii. Lack of portable non-contact surface measurement instruments in the market

Finding a portable laser profiler measurement instrument in the existing market is difficult. It is inconvenient for users to move to the desired place to collect the data. The available instruments in the market are fixed at a location and non-movable. If the specimen is bulky and fragile, it is difficult for the user to move it on the platform to measure. The machines found in the market are heavy and cannot be easily reassembled. It requires professionals that acquire technical skills to do the work. Therefore, the maintenance and repair work needs a very high cost to sustain the machine.

### iii. Non-user-friendly in software GUI interface

Integrating data collection and obtaining the desired parameters is costly software development. Now, highly accurate, smaller, intuitive packages that archive data points and allow users to parse the data and explore more insights are coming at a price. The main issue faced by the users is that the measured data is not easily accessed. Users cannot get insights into the data and read it easily. The users need to follow the manual to get their desired specific parameters.

**iv. Lack of automated contactless surface roughness parameters software in the market**

No programme on the current market can automatically determine surface roughness metrics such as autocorrelation length and RMS height from raw data input. The researcher can only perform calculations manually, which is inefficient and time-consuming.

### **1.3 Aims and Objectives**

The project is aimed to develop software that integrates with the surface profiler prototype in extracting the measured data. The surface profiler is run in a lateral direction over a distance. Then, the measured data is stored in the system as an excel file. Users can conveniently choose the file they want to analyse from the system. The hardware and software of the instrument are connected seamlessly. Users can install the software online and use it with their devices. The following are the precise project objectives:

- i. To develop an application that can optimise the graph by removing unwanted noises with spikes.
- ii. To design a user-friendly interface that increases the productivity of the user, which can decrease the search time in finding the options and increase the satisfaction of using the application.
- iii. To develop an algorithm that can generate specific parameters such as autocorrelation length and RMS height and automate the process to satisfy the users' needs within seconds.
- iv. To develop a contactless integration with the Internet of Things (IoT) which highly increase user productivity in data analysis.



## 1.4 Organisation of Thesis

The content of each chapter is listed in Table 1.1 below.

**Table 1.1 Table of organisation of the thesis**

Chapter 1: Introduction	<ul style="list-style-type: none"> <li>• Introduce the surface profiler and surface roughness with remote sensing.</li> <li>• Give a brief introduction to the previous research by Dr. Lee Yu Jen and other researchers on the sea and snow surface physical properties.</li> <li>• Subchapters of this chapter include background, problem statements, aims and objectives.</li> </ul>
Chapter 2: Literature Review	<ul style="list-style-type: none"> <li>• Reviews of working principles on the previous project.</li> <li>• Research on the filters optimising the measured data.</li> <li>• Research on the surface roughness parameters (Autocorrelation Length &amp; RMS Surface Height).</li> <li>• Compare the programming language and the GUI framework to use.</li> <li>• Compare the IoT and cloud storage platforms to use.</li> <li>• Reviews of existing surface data analysis software.</li> </ul>

Chapter 3: Methodology	<ul style="list-style-type: none"> <li>• Provides an overview of the proposed method and approach to developing the project.</li> <li>• Shows and visualises system design in the form of system flow charts.</li> <li>• The hardware used is listed.</li> <li>• Explain how the code is executed and also highlight the features.</li> <li>• Shows the timeline of deliverables.</li> </ul>
Chapter 4: Result and Discussion	<ul style="list-style-type: none"> <li>• Measure the reliability and accuracy of the software's calculation on the surface roughness parameters.</li> <li>• Measure the reliability and accuracy of the filter by optimising the raw data and generating a graph.</li> </ul>
Chapter 5: Conclusion and Recommendations	<ul style="list-style-type: none"> <li>• Conclude the project.</li> <li>• Reviews on the project by providing the limitations and areas of improvement.</li> </ul>

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Overview**

A review of the previous project – Automated Electronic Sensor System, is done to understand the working principle and find out the limitations and areas of improvement. A programming language is the first thing to research for developing software with a GUI interface. This is to make sure that the GUI framework can be more focused and ease the further decision next. Then, a few GUI frameworks are compared to ensure the functionality and sustainability of the application can be achieved. Furthermore, graph optimisation by comparing different filters and research on the surface roughness parameters are done to achieve software automation on data analysis. Then, IoT and cloud storage platforms are compared to achieve the contactless machine objective. Afterwards, reviews of existing surface roughness measurement software must be made to highlight the advantage and disadvantages of each application. This is to trigger more innovations and functions to be added to the project.

## 2.2 Automated Electronic Sensor System

This system is utilised to design and construct an automated electronic earth surface. This system should be able to automatically measure and store data at predetermined intervals for later extraction. This gadget is limited to profiling a 30 cm long, 1-dimensional surface at 0.6 cm intervals. A Raspberry Pi in this project controls the entire system. The Raspberry Pi is powered by a Raspbian OS, a Linux-based operating system. A screen is attached to Raspberry Pi's HDMI connector to display the OS's graphical user interface. The entire circuit is constructed with the PCB. Some circuits are manufactured independently from the main PCB to accomplish specific functions. The acquired data is stored in the memory card of the Raspberry Pi. Two methods are available for data extraction: Connect a pendrive or use a LAN cable (Kang, 2016).

### 2.2.1 Physical Planning and Design

A holder or stand is required for measurements to be taken. The design drawing is depicted in Figure 2.1. Aluminium was chosen to construct the profiler's stand. Aluminium is a lightweight material that resists corrosion. Two PCBs: the main board and the encoder board, are required in the design.

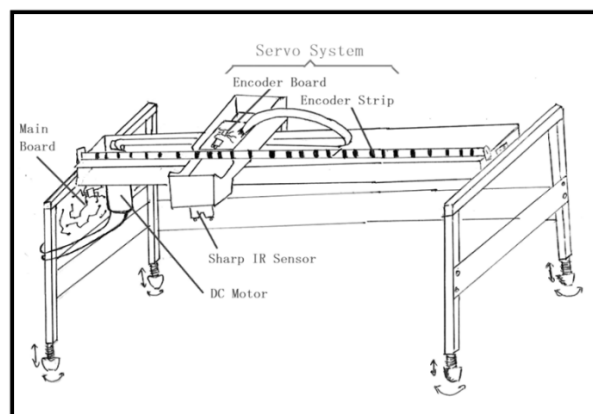


Figure 2.1: Sketch of surface profiler (Kang, 2016)

### 2.2.2 Hardware Specifications

For this project, a Sharp IR Analogue Distance Sensor was chosen. This sensor's beam width is relatively limited. Additionally, it provides accurate measurements. The sensor's measuring range is between 10 cm and 80 cm. Although inexpensive, this sensor performs poorly on black surfaces and in direct sunshine. Due to the inexpensive cost, it is appropriate for study. The Sharp IR sensor operates on the triangulation technique. This sensor features an integrated circuit. The Sharp IR sensor generates an analogue signal based on the detected distance. The distance versus voltage graph from the datasheet is depicted in Figure 2.2 below. According to the graph, the distance between approximately 7 cm and 20 cm has a steeper curve. This study indicates that this sensor is more sensitive at this range and delivers more precise detection at that distance. The maximum height of the sensor design must, therefore, not exceed 20 cm above the ground (Kang, 2016).

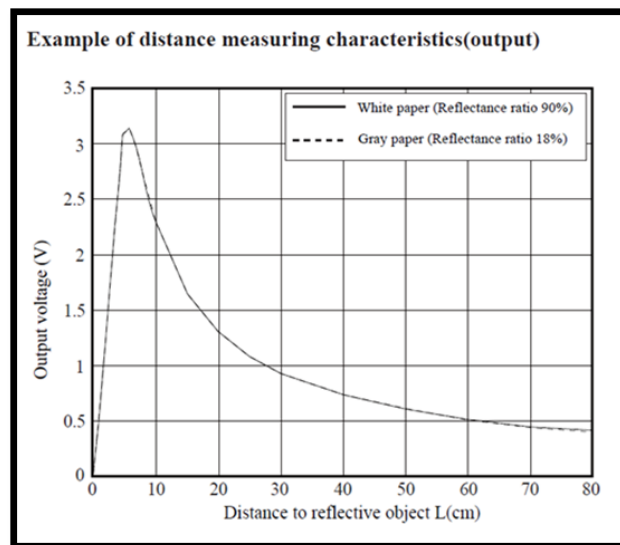


Figure 2.2: Graph of distance vs voltage (Kang, 2016)

Since the Sharp IR sensor datasheet does not include an equation for measuring the distance to voltage, an experimental measurement is conducted. The actual experimental setup is depicted in Figure 2.3 below. The graph of average distance versus voltage is depicted in Figure 2.4. After plotting the average graph, a trendline based on the power function is added to the graph.

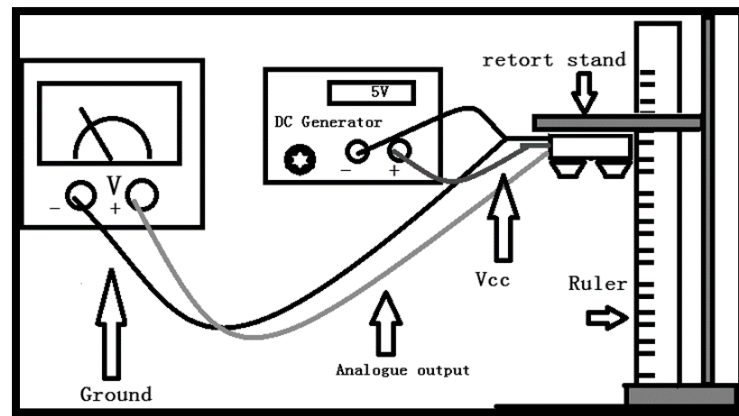


Figure 2.3: Experiment setup to measure the distance to voltage (Kang, 2016)

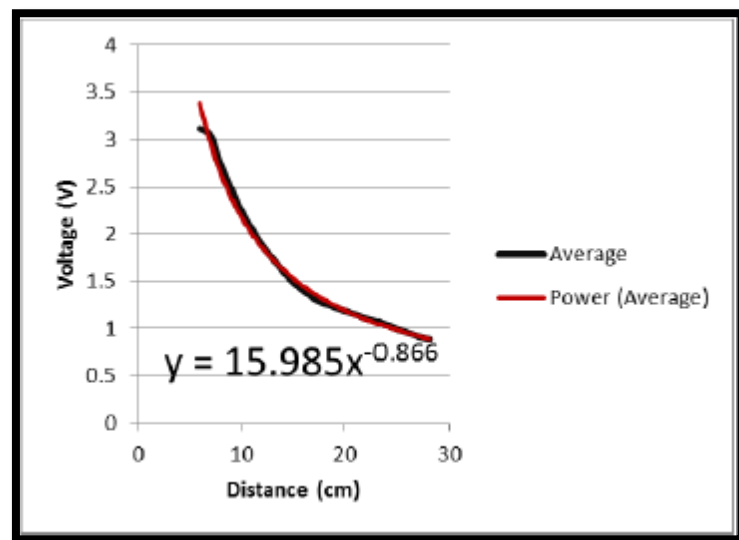


Figure 2.4: Graph of distance vs voltage (average) (Kang, 2016)

Then the equation of distance to voltage is generated using Microsoft Excel. The equation is obtained as shown below:

$$y = 15.985x^{-0.866} \quad (2.1)$$

where

y = voltage, V

x = distances, cm

To derive the voltage-to-distance equation,  $x$  is moved to the left side of the equation, and  $y$  is moved to the right side through a series of mathematical calculations. The derived equation is given by

$$x = (y / 15.985)^{(1 / -0.866)} \quad (2.2)$$

where

$x$  = distance, cm

$y$  = voltage, V

To convert the analogue signal from the Sharp IR distance measuring sensor, an ADC is required. The ADC is necessary because the Raspberry Pi lacks an inbuilt ADC. ADC0804 IC was selected as the eight-bit ADC. The ADC 0804IC contains an oscillator. Figure 2.5 depicts pin 19 as the oscillator's output and pin 4 as the ADC's clock input. ADC0804's datasheet indicates that the ADC can accept a 640 kHz clock. Connected to the output of the Sharp IR sensor is Pin 6. Pin 9 is the voltage reference pin. The voltage provided to this terminal must be divided in half. The SN74166 PISO 8-bit shift register was chosen. This 8-bit shift register is responsible for moving the data from the 8-bit ADC to the controller bit by bit to compensate for the lack of I/O pins. Figure 2.6 below illustrates the IC's pin assignment.

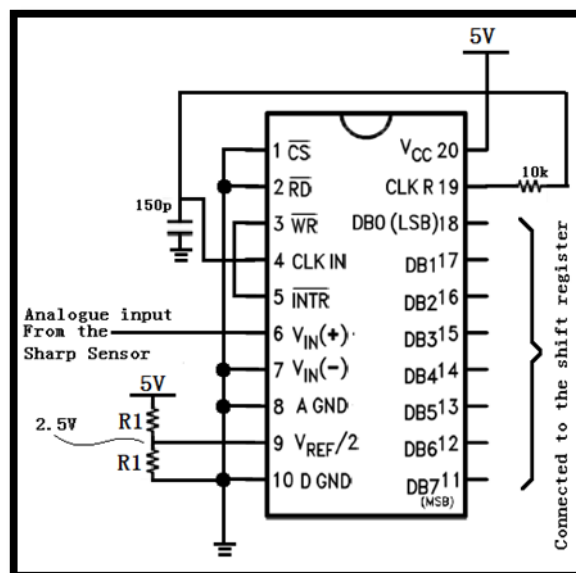


Figure 2.5: ADC0804 IC connection (Kang, 2016)

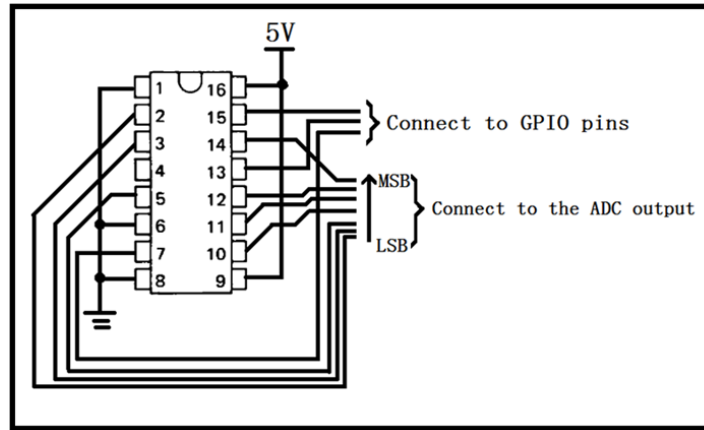


Figure 2.6: Connection of SN74166 (Kang, 2016)

A slider purchased from the market is expensive, and its length is difficult to find. Therefore, a slider pulled from a malfunctioning inkjet printer was selected. It has a length of 35 cm, making it appropriate for carrying in a bag. The disassembled slider from the HP inkjet printer is depicted in Figure 2.7 below. A DC motor was chosen to mount on the slider. The DC motor is shown mounted on the slider in Figure 2.8. The DC motor is the only motor that can fit onto the slider, although it is not the ideal option. The stepper motors could be positioned incorrectly on the slider.



Figure 2.7: Disassembled slider from HP inkjet printer (Kang, 2016)



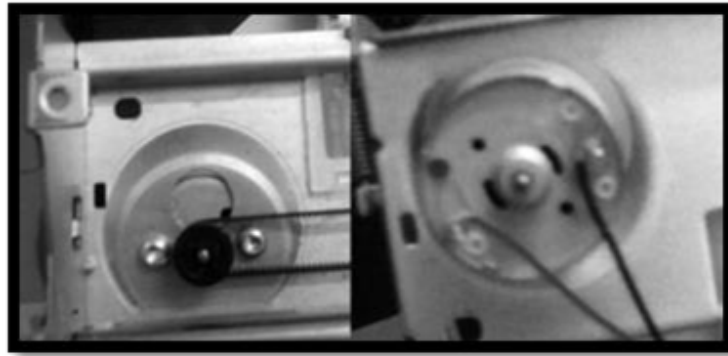


Figure 2.8: DC motor attached to the slider (Kang, 2016)

A servo system has an encoder, motor, and motor driver. The encoder system provides the controller with feedback. Components of the encoder system include an encoder and an encoder strip. As the encoder, a line follower infrared sensor was chosen. This circuit's primary components are an IR transmitter, receiver, IR resistors, and an op-amp comparator. The selected operational amplifier, LM358N, is accessible in the faculty laboratory. The line follower infrared sensor of choice is depicted in Figure 2.9 below. The black and white strip of the encoder is sensed with the black surface as logic "0" and the white surface as logic "1". The position of the slider is fed back by the stripes. The encoder system is depicted in Figure 2.10.

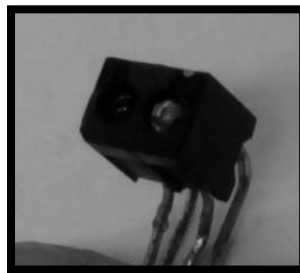


Figure 2.9: Line follower infrared sensor (Kang, 2016)

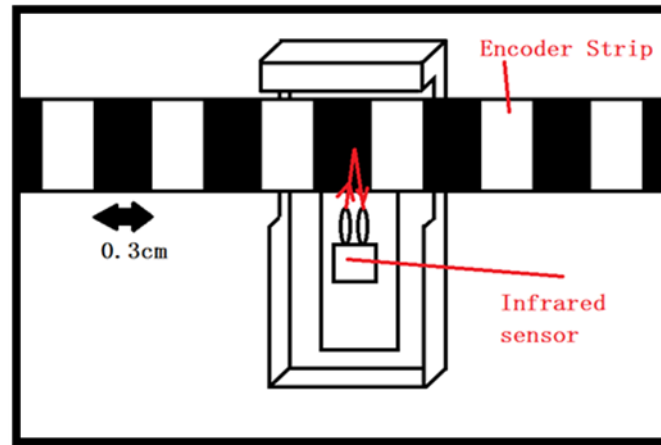


Figure 2.10: Sketch of the encoder system (Kang, 2016)

A tiny lead acid battery with a capacity of 1.2 AH and an operating voltage of 12 V was chosen. This battery is appropriate for powering the DC motor. The 12 V battery is used to power the DC motor, but it cannot power the Raspberry Pi, the Sharp IR sensor, the encoder, the ADC, or the shift register. Consequently, a buck step-down DC-DC converter was used. The selected module for the buck converter is depicted in Figure 2.11 below. This bulk converter can convert voltages between 4.5 V and 35 V; therefore, the fire is not caught when a 12 V source is supplied. The input of this bulk converter module is supplied with 12 V, and 5 V is obtained by adjusting the rheostat using a voltmeter attached to the module's output. The block diagram is depicted in Figure 2.12 below.

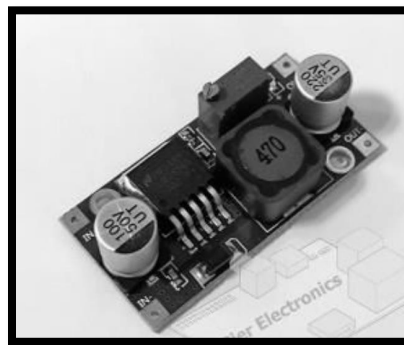


Figure 2.11: DC-DC buck converter module (Kang, 2016)

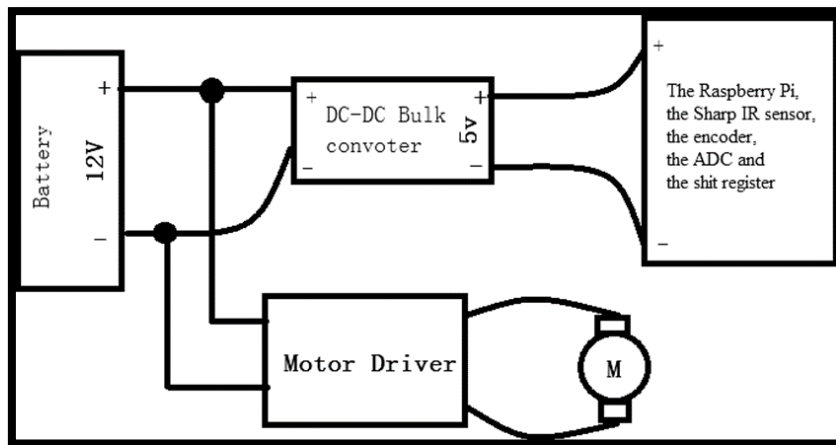


Figure 2.12: Connection of battery, converter to the supply other devices (Kang, 2016)

### 2.2.3 Software Specifications

Python was selected to program the Raspberry Pi to control the hardware. Python is already available in the Raspbian OS, and the libraries required to run the Python script are also available in the OS. PWM is required to control the speed of the motor. The explanation of the duty cycle is shown in Figure 2.13. After running many tests with different duty cycles and frequencies, the maximum duty cycle obtained is 17% at 175 Hz.

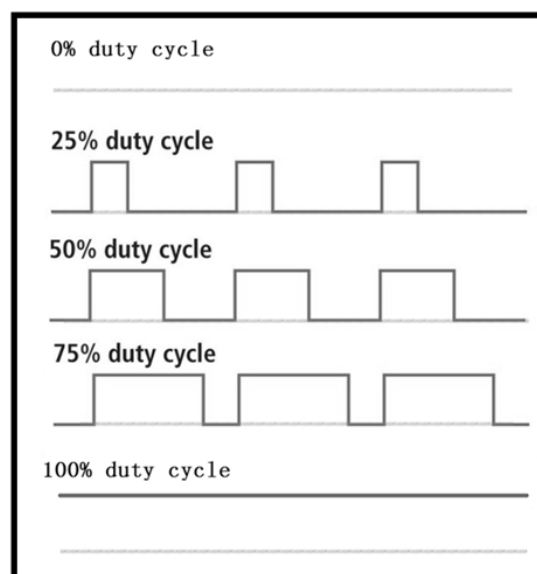


Figure 2.13: Duty cycle of PWM

In order to compute the height of the surface, it is necessary to retrieve data from the Sharp IR sensor using the ADC and shift register. The shift register's binary code is translated to decimal. Using the formula  $v = 5 * (\text{the retrieved decimal value}) / 255$ , the voltage value is calculated. By using the voltage to distance equation, the distance between the surface and the Sharp IR sensor is calculated as  $x = (y / 15,985) (1 / 0.866)$ . In the code, the first value is designated as the reference *distance*. Then, store the initial and subsequent values in reference *distance1*. The height of the surface is then calculated using the formula *height = distance - distance1*.

Python requires an additional library to save data in an Excel file. It was the XlsxWriter library that was installed. The terminal command "`sudo pip install XlsxWriter`" is used to install the library. Following installation, the library is immediately usable. The code "`import xlsxwriter`" is placed at the beginning of the script to import the library. Two lines of code are used to create an Excel file: "`workbook = xlsxwriter.Workbook (filename.xls>)`" and "`worksheet = workbook.add worksheet()`." After the data is transferred to the file, the file is closed. Excel is closed using the "`workbook.close()`" function. The retrieved data files are kept at `/home/pi/Desktop`. There are two methods of extraction. The first technique involves transferring the Excel file to a flash drive. A flash drive is required to connect to the Raspberry Pi's USB port. The excel files created in the `/home/pi/Desktop` directory are then copied to a flash drive.

The second option is to use a LAN cable to transport data to the personal computer. Software is necessary for data transfer over a LAN. The software is named Filezilla. For this software to function, the IP address of the Raspberry Pi must be obtained. Figure 3.34 depicts a screenshot of the Filezilla application. At the very top of the software are four text boxes. The Raspberry Pi's IP address is entered in the Host text box. The username is "pi", and the password is "raspberrypi", while the port number is "22".

### 2.2.4 System Flow Chart

Figure 2.14 depicts the system flow for the surface profiler. The surface profiler is initiated by pressing the button on the PCB. When the button is pressed, an Excel file is created, and the motor is controlled via PWM. Along the encoder strip, the infrared sensor collects data. The program's algorithm processes and saves the data in the Excel file. The motor is finally returned to its initial position, and the Excel file is prepared for analysis.

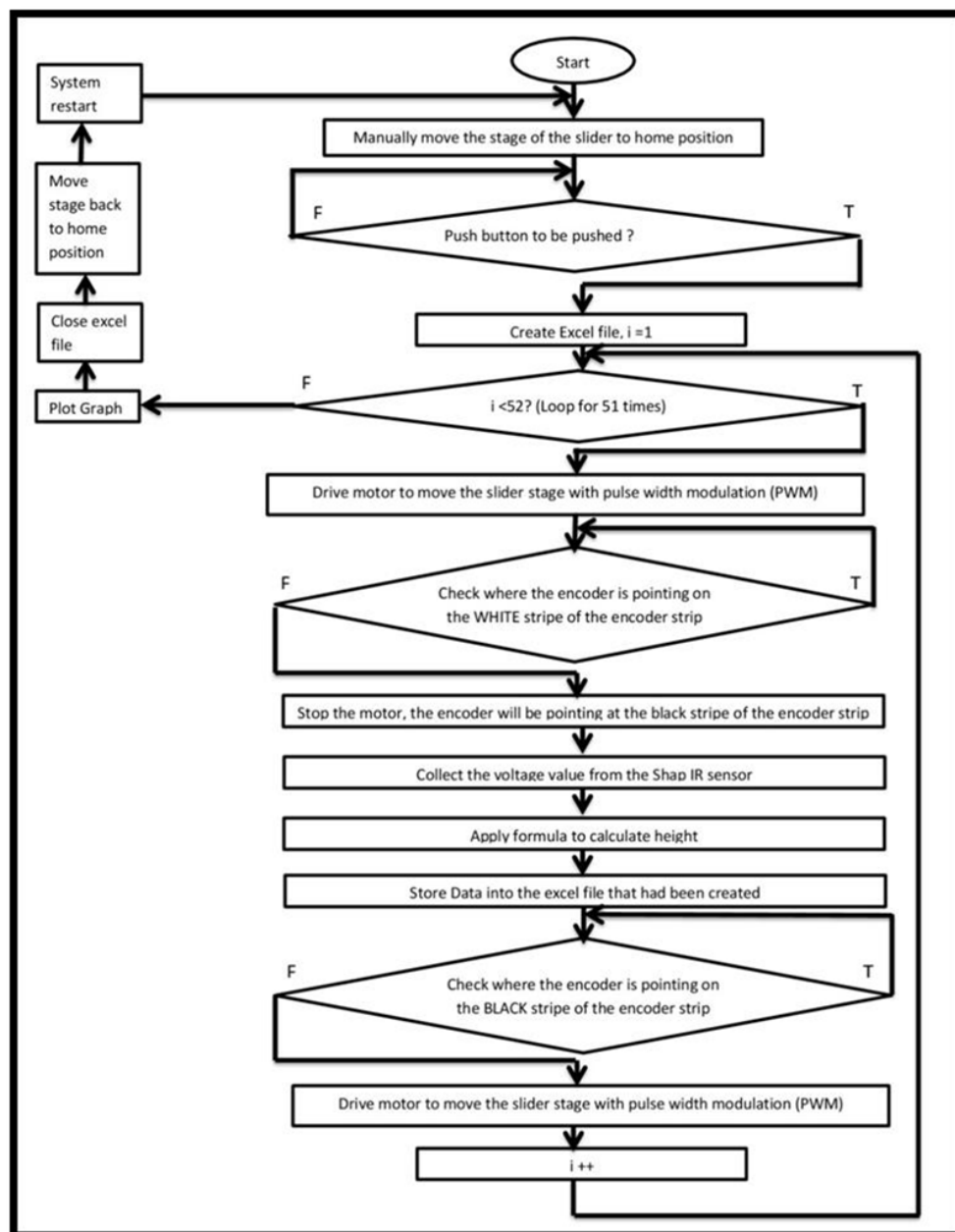


Figure 2.14: Flow chart of surface profiler system (Kang, 2016)

### 2.2.5 Case Study: Accuracy Test

The profiler is examined as it traversed a white, flat surface 19 cm beneath the Sharp IR sensor. The physical arrangement is shown in Figure 2.15. The optimal outcome is depicted in Figure 2.16, whereas the worst outcome is depicted in Figure 2.17. Due to the highest fluctuations of the graphs being less than 0.5 cm, the result obtained is deemed satisfactory. The optimal outcome yielded a maximum variation of 0.26 cm. On the other side, the obtained value showed a maximum variation of greater than 1.3 cm. The highest fluctuation is 1.88 cm at its worst. The outcomes demonstrated that the graphs had sharp edges and fluctuations along a flat surface. The accuracy is improved when the surface is closer to the Sharp IR sensor.



Figure 2.15: Setup of the case study (Kang, 2016)

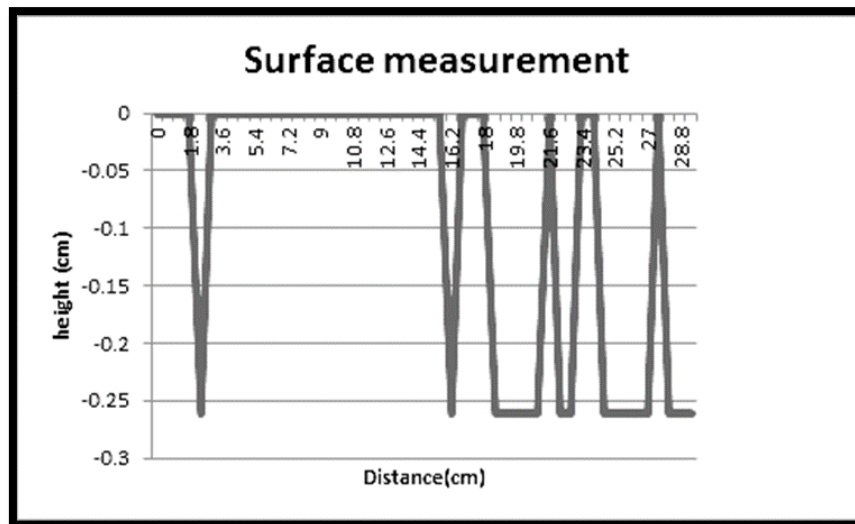


Figure 2.16: The best result of the case study (Kang, 2016)

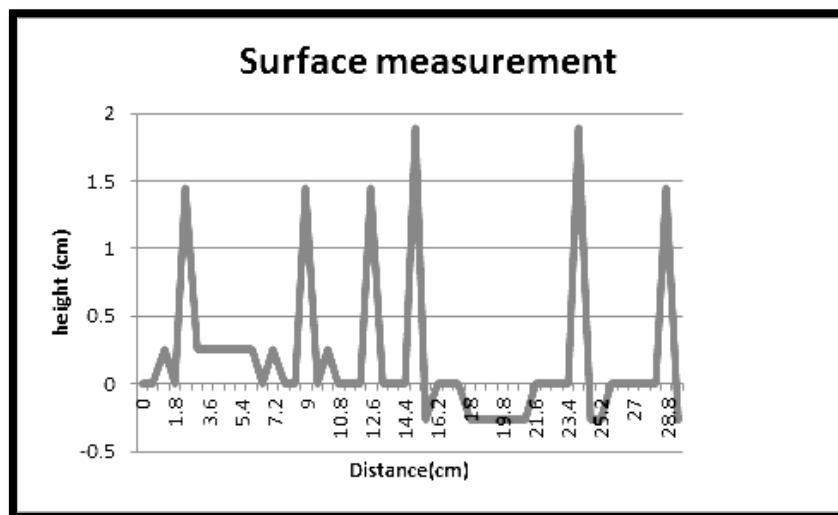


Figure 2.17: The worst result of the case study (Kang, 2016)

### 2.2.6 Summary

The pros and cons of the automated surface profiler are summarised in Table 2.1 below.

**Table 2.1: Table summarising the pros and cons of the automated surface profiler**

Pros	Cons
<b>Portable –</b> Easy to move around	<b>Battery Consumption –</b> Short usage time
<b>Data Automation –</b> Graph of surface measurement is generated in Excel file every run	<b>Unoptimized Data –</b> Data collected contain noise
<b>Accessibility –</b> Easy to control and use	<b>Surface Roughness Parameter –</b> Lack of data analysis on the surface roughness

### 2.3 Graph Optimisation

Graph optimisation is a statistical technique for eliminating datasets' noise to identify patterns. This is achieved by removing statistical noise from datasets using algorithms. The data may be adjusted during compilation to decrease or remove significant disparities. It focuses on establishing a fundamental direction for the key data points by avoiding any volatile data points and simplifying the formation of a smoother curve across all data points (Bhalerao, 2021). As data is compiled, it can be efficiently altered to eliminate or reduce any volatility or other forms of disturbances. This is referred to as the data smoothing procedure (Fincash, 2022).

Digital Signal Processing (DSP) technology is being applied in data smoothing. DSP is an integral component of several measurement systems and is used to post-process data before real analysis. Signal processing aims to enhance the quality of a



signal by eliminating noise. It can also eliminate redundant information for more effective data storage and transmission. DSP focuses on digital signal processing on computers. Resistors, capacitors, and inductors are employed to execute analogue signal processing. Digital filters are more precise, less expensive, and age-resistant than analogue ones (Pastell, 2016). It is feasible to optimise a filter for either time domain or frequency domain performance, but not both. There are several functions to designing and analysing filters. For instance, the Moving Average Filter, Median Filter, and Savitzky-Golay Filter (Pastell, 2016).

### 2.3.1 Moving Average Filter

The moving average is a simple lowpass filter. It substitutes the average of  $n$  samples for each sample. The most common variation is the central moving average; here's how to compute it for 5 points:

$$y_t = \frac{x_{t-2} + x_{t-1} + x_t + x_{t+1} + x_{t+2}}{5} \quad (2.3)$$

where  $y_t$  is  $t$  th of the filtered signal and  $x$  is the original signal. The general formula of such an averaging operation is  $y(k)$ . The value of  $y(k)$  is defined to be the arithmetic average of the preceding  $N - 1$  points of  $x(k)$  itself:

$$y(k) = \frac{1}{N} \sum_{j=k-N+1}^k x(j) \quad (2.4)$$

Moving average is the ideal filter for white noise-containing time domain signals (Pastell, 2016). The signal-to-noise ratio can be improved by widening the filter or smoothing the data several times. After each filter run, the first  $n$  and last  $n$  points are lost. This technique's findings are deceptively impressive due to excessive filtering. Information is lost or corrupted due to giving too much statistical weight to points that are distant from the focal point. The moving average method is highly detrimental when the filter traverses narrow peaks compared to the filter width (Constantinos, 2018).

### 2.3.2 Median Filter

The median filter is a discrete-time operation in which a window with a width of  $2N+1$  points is moved across an input signal. At each phase, the points within the window are sorted according to their respective values, with the median value of the sorted set serving as the output value for each window position. Consider a real, discrete-time sequence  $\{a(n)\}$ , where  $a$  is an  $M$ -level signal. The output of the median filter  $y(n)$  is given by  $y(n) = \text{median} [a(n - N), \dots, a(n), \dots, a(n + N)]$ . The median operation has the property of commuting with the thresholding operation. Consequently, a multilayer running median can be achieved by thresholding the input signal at all potential levels, filtering each binary thresholded signal with a median filter, and summing the filtered binary outputs. Define  $\{a_i(n)\}, i = 1, 2, \dots, M - 1$  as the  $i$ th level binary signal sequence of  $\{a_i(n)\}$ , where  $a_i(n) = 0$  if  $a(n) < i$ , and  $a_i(n) = 1$  if  $a(n) > i$ . The output of the median filter at the  $i$ th level is given by  $y_i(n) = \text{median} [a_i(n - N), \dots, a_i(n), \dots, a_i(n + N)]$ . It has been shown in (2.4) that median filtering of an arbitrary level signal is equivalent to decomposing the signal into binary signals, filtering each binary signal with a binary median filter, and then reversing the decomposition (Qiu, 1994).

$$y(n) = \sum_{i=1}^{M-1} y_i(n) \quad (2.5)$$

For analysis purposes, the  $\{0,1\}$  binary of  $\{a_i(n)\}$  is transferred into  $\{-1,1\}$  binary of  $b_i(n)$  by the operation:  $b_i(n) = 2a_i(n) - 1$ . Define  $V_i(n)$  as the output of the median filter of  $\{b_i(n)\}$ , which is given by  $V_i(n) = \text{median} [b_i(n - N), \dots, b_i(n), \dots, b_i(n + N)]$  and  $V_i(n) = 2y_i(n) - 1$ .

For median filtering of the binary sequence  $\{b_i(n)\}$ , the output of the median filter is as follows:

$$V_i(n) = \begin{cases} +1 & \text{if } S(n) \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (2.6)$$

where

$$S(n) = \sum_{\substack{j=-N \\ j \neq 0}}^{j=N} b_i(n + j) + b_i(n) \quad (2.7)$$

The cost function of the output state of the median filter is defined as below:

$$E_i(V_i(n)) = \sum_{\substack{j=-N \\ j \neq 0}}^{j=N} V_i(n)b_i(n + j) - V_i(n)b_i(n) \quad (2.8)$$

It is straightforward to show that the median filter operation of (2.6) always forces (2.8) into its minimum:  $E_i(+1) = -S(n)$  and  $E_i(-1) = S(n)$ . If  $V_i(n) = +1, S(n) \geq 0, E_i(+1) < E_i(-1)$ . If  $V_i(n) = -1, S(n) < 0, E_i(-1) < E_i(+1)$ . This property of median filtering is known as the functional optimization property. This functional optimization characteristic holds at every level of the thresholded space. Thus, median filtering is an optimization process in which the output of the filter is always set to the least of a cost function of the filter's output state. The cost function of (2.8) is called the median cost function (Qiu, 1994). It may appear odd at first look because the equation (2.8) is written with two terms rather than a single sum. As will become evident in the analysis that follows, these two concepts have distinct meanings. The first term quantifies the smoothness between the filter output and its neighbouring points inside the filtering window, while the second quantifies the difference between the filter output and the original signal (Qiu, 1994).

If the output of the filter is the same as its neighbouring input,  $V_i(n) = b_i(n + j), j \neq 0$ , the median cost function is reduced. If the output of the filter differs from its neighbour,  $V_i(n) \neq b_i(n + j), j \neq 0$ , the median cost function is increased. The median filtering operation (2.6) encourages the output to have the same value as its neighbours as it always minimises the cost function. Consequently, median filtering has a tendency to smooth the signal. According to the above analysis, the median filtering function consists of two components: The median filtering favours a smooth signal that is identical to the original signal. This may explain why median filters have the fundamental property of removing noise without considerable blurring or edge loss (Qiu, 1994).

### 2.3.3 Savitzky-Golay Filter

Savitzky and Golay present a method for data smoothing based on approximating local least-squares polynomials. It corresponds to discrete convolution with a fixed impulse response. The lowpass filters resulting from this approach are commonly referred to as Savitzky-Golay filters. Savitzky and Golay demonstrated that least squares smoothing decreases noise while preserving a waveform's peak shape and height (Schafer, 2011).

The idea of least-squares polynomial smoothing is depicted in Figure 2.18, which shows a sequence of samples  $x[n]$  of a signal as solid dots. Considering, for the moment, the group of  $2M + 1$  samples centred at  $n = 0$ , the coefficients of a polynomial

$$p(n) = \sum_{k=0}^N a_k n^k \quad (2.9)$$

that minimises the mean-squared approximation error for the group of input samples centred on  $n = 0$ ,

$$\begin{aligned} \varepsilon_N &= \sum_{n=-M}^M (p(n) - x[n])^2 \\ &= \sum_{n=-M}^M (a_k n^k - x[n])^2 \end{aligned} \quad (2.10)$$

are obtained (Schafer, 2011).

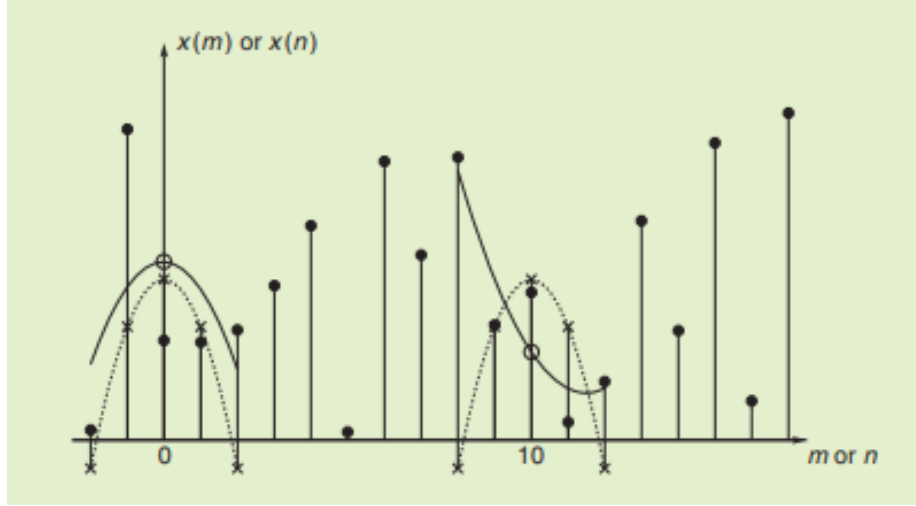


Figure 2.18: Illustration of least-squares smoothing by locally fitting a second-degree polynomial (solid line) to five input samples (Schafer, 2011)

The analysis is the same for any other group of  $2M + 1$  input samples.  $M$  is the “half width” of the approximation interval. In Figure 2.17, where  $N=2$  and  $M=2$ , the solid curve on the left in Figure 2.17 is the polynomial  $p(n)$  evaluated on a fine grid between  $-2$  and  $+2$ , and the smoothed output value is obtained by evaluating  $p(n)$  at the central point  $n=0$ . That is,  $y[0]$ , the output at  $n=0$ , is

$$y[0] = p(0) = a_0 \quad (2.11)$$

The interval of approximation need not be symmetric about the evaluation point. This results in nonlinear phase filters, which are useful for smoothing the endpoints of sequences of finite length. Using these weighting coefficients, also known as convolution integers, is exactly equal to fitting the data to a polynomial; nevertheless, it is computationally more efficient and considerably faster. Therefore, the smoothed data point  $(y_k)_s$  is given by the following equation:

$$(y_k)_s = \frac{\sum_{i=-n}^n A_i y_{k+i}}{\sum_{i=-n}^n A_i} \quad (2.12)$$

## 2.4 Surface Roughness Parameters

A surface that "appears" very rough to an optical wave may seem very smooth to a microwave. This is because the degree of roughness, or simply the roughness, of a random surface is described by statistical characteristics measured in wavelengths. The standard deviation of the surface height fluctuation (or RMS height) and the surface correlation length are the two most frequently used metrics to assess surface roughness (Ulaby, Moore and Fung, 1981).

The standard deviation of surface height ( $\sigma$ ) and the surface correlation length ( $\ell$ ) reflect the statistical variance of the random surface height component compared to a reference surface. The reference surface may be the undisturbed surface of a periodic pattern (as in the case of row-tilled soil surfaces) or the mean surface if only random changes exist, as depicted in Figures 2.19 and 2.20, respectively (Ulaby, Moore and Fung, 1981).

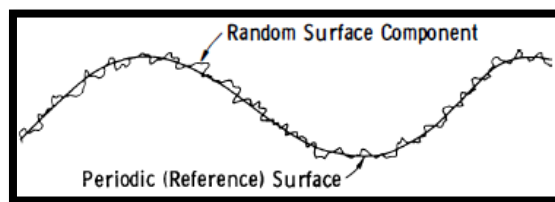


Figure 2.19: Random height variations superimposed on a periodic surface (Ulaby, Moore and Fung, 1981)

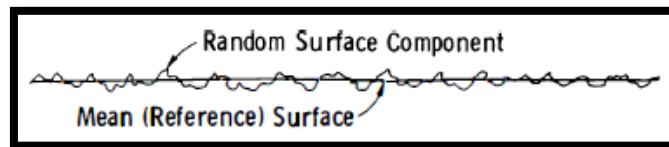


Figure 2.20: Random height variations superimposed on a flat surface (Ulaby, Moore and Fung, 1981)

### 2.4.1 Standard Deviation of Surface Heights / RMS Height

Consider a surface in the x-y plane whose height at a point  $(x, y)$  is  $z(x, y)$  above the x-y plane. For a statistically representative surface section of dimensions  $L_x$  and  $L_y$ , centred at the origin, the mean height of the surface is

$$\bar{z} = \frac{1}{L_x L_y} \int_{-\frac{L_x}{2}}^{\frac{L_x}{2}} \int_{-\frac{L_y}{2}}^{\frac{L_y}{2}} z(x, y) dx dy \quad (2.13)$$

And the second moment is

$$\overline{z^2} = \frac{1}{L_x L_y} \int_{-\frac{L_x}{2}}^{\frac{L_x}{2}} \int_{-\frac{L_y}{2}}^{\frac{L_y}{2}} z^2(x, y) dx dy \quad (2.14)$$

The standard deviation of the surface height,  $\sigma$ , is then given by

$$\sigma = \left( \overline{z^2} - \bar{z}^2 \right)^{\frac{1}{2}} \quad (2.15)$$

If  $z(x, y)$  is statistically independent of the x-y plane's azimuth angle, the above formulation can be simplified to a single dimension (Ulaby, Moore and Fung, 1981). For the one-dimensional surface profile,  $\sigma$  is computed, in practice, by digitising the profile into discrete values  $z_i(x_i)$  at an appropriate spacing  $\Delta x$ . Assuming that the height variation  $\Delta z$  corresponding to a horizontal segment  $\Delta x$  is much smaller than the wavelength  $\lambda$  of the incident wave, this variation  $\Delta x$  will have no appreciable effect on the reflection by the surface of the segment  $\Delta x$ . As a rule of thumb, the spacing  $\Delta x$  should be chosen such that  $\Delta x \leq 0.1\lambda$  (Ulaby, Moore and Fung, 1981).

The standard deviation for the discrete one-dimensional case is given by

$$\sigma = \left[ \frac{1}{N-1} (\sum_{i=1}^N (z_i)^2 - N(\bar{z})^2) \right] \quad (2.16)$$

where

$$\bar{z} = \frac{1}{N} \sum_{i=1}^N z_i \quad (2.17)$$

and  $N$  is the number of samples.

### 2.4.2 Surface Autocorrelation Length

The normalised autocorrelation function for a one-dimensional surface profile  $z(x)$  is defined as

$$\rho(x') = \frac{\int_{-\frac{L_x}{2}}^{\frac{L_x}{2}} z(x)z(x+x')dx}{\int_{-\frac{L_x}{2}}^{\frac{L_x}{2}} z^2(x)dx} \quad (2.18)$$

and is a comparison of similarities between the height  $z$  at a point  $x$  and a point  $x'$  distant from  $x$ . For the discrete case, the normalised autocorrelation function for a spatial displacement  $x' = (j - 1)\Delta x$ , where  $j$  is an integer  $\geq 1$ , is given by

$$\rho(x') = \frac{\sum_{i=1}^{N+1-j} z_i z_{j+i-1}}{\sum_{i=1}^N z_i^2} \quad (2.19)$$

A plot of  $\rho(x')$  as a function of the displacement  $x'$  for the surface profile is shown in Figure 2.21. The surface correlation length usually is defined as the displacement  $x'$  for which  $\rho(x')$  is equal to  $\frac{1}{e}$ :

$$\rho(l) = 1/e, \quad (2.20)$$

as indicated in Figure 2.21.



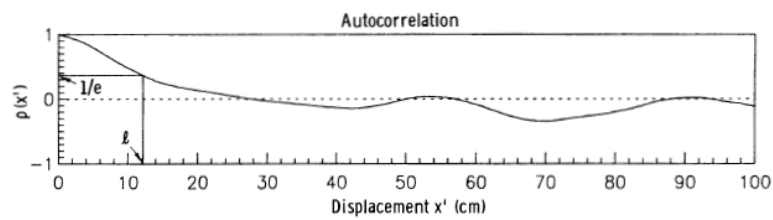


Figure 2.21: Autocorrelation Function (Ulaby, Moore and Fung, 1981)

If two points on a surface are separated by a horizontal distance greater than  $l$ , their heights may be statistically independent. In the extreme case of a perfectly smooth surface, every point on the surface is correlated with every other point with a correlation coefficient of unity. Hence,  $l = \infty$  in this case (Ulaby, Moore and Fung, 1981).

## 2.5 Programming Languages

A programming language is used to communicate with computers. The computers can perform a specific task with a set of instructions that the developer specifies. The familiarisation of a developer directly affects the communication effectiveness between the developer and the computer. Ineffective communication finds the software to have many bugs left to be resolved by the developer. It is a significant problem that all programmers are worried about. Therefore, a familiar and well-handling programming language for software development is essential. Research is done on different mainstream programming languages for GUI interfaces.

### 2.5.1 Python

Python is a modern language and widely used language since 2004 and was developed in the 1980s. Python has the most popular because it has a solution for every field. It can be used in Artificial Intelligence, Web, Data Analysis, and Application Development. The efficiency of Python in multiple technical domains advocates the reason for such a large and active Python community (upGrad, 2021).

Python is an easy-to-read language; it is interpretable even for amateur programmers. It belongs to a dynamically typed language. It also has automatic memory management. For GUI development, Python has an extensive library. Python has a library that lets the developer detect the mouse and keyboard input and show it on the screen (Shepherd, 2020). Furthermore, the pros and cons of the language are summarised in Table 2.2 below.

**Table 2.2: Table summarising the pros and cons of Python**

Pros	Cons
<b>Faster build –</b> Easy to understand and write	<b>Speed Limitations –</b> Slower reading speed than other languages
<b>Large libraries –</b> More options for GUI frameworks	<b>Limited environment –</b> Weak language for the mobile environment
<b>Cross Platforms –</b> Coding run on any platforms	<b>Memory Consumption –</b> High memory consumption with active RAM

### 2.5.2 Java

Java is object-oriented and helps in creating programs on any platform. Java has ruled over all languages for over 20 years (TechVidvan, 2020). Java was initially designed for embedded network applications on multiple platforms (Austerlitz, 2003). The most recognisable Android application is also written in the Java programming language. Nowadays, Java is used by developers to build software in many forms. These products serve in many fields, for instance, video games and websites. Due to its security, maintainability, platform independence, and many other benefits, Java is one of the most used languages in the software industry (BairesDev, 2022). Furthermore, the pros and cons of the language are summarised in Table 2.3 below.

**Table 2.3: Table summarising the pros and cons of Java**

Pros	Cons
<b>Stable and Secure–</b> Avoiding the use of explicit pointers to store a memory address	<b>Verbose –</b> Results in a long and complex coding
<b>Various APIs –</b> More options for methods of communication	<b>Unattractive –</b> Less mature in good-looking UI development
<b>Cross Platforms –</b> Coding run on any platforms	<b>Memory Consumption –</b> High memory consumption with active RAM

### 2.5.3 C++

C++ is a C programming language framework introduced with object-oriented programming principles. It was created as a cross-platform enhancement to C to give developers more control over memory and system resources (Buttice, 2021). C++ is a compiler-based programming language. The coding is pre-interpreted and makes the code run faster. The syntax of C++ is similar to C#, C, and Java. It is easier for the developer to familiarise the language well and switch quickly. C++ is widely used as it can also be run as the C program without changing the code. The flexibility of the language is the reason for its popularity. Furthermore, the pros and cons of the language are summarised in Table 2.3 below.

**Table 2.3: Table summarising the pros and cons of C++**

Pros	Cons
<b>Fast and Powerful –</b> The code is pre-interpreted	<b>Verbose –</b> Results in a long and complex coding
<b>Multi-paradigm –</b> Generic, imperative, and object-oriented style of program	<b>Unsafe –</b> The presence of pointers and global variables made the system easily corrupt
<b>Cross Platforms –</b> Coding run on any platforms	<b>Memory Consumption –</b> High memory consumption with active RAM

## 2.6 GUI Frameworks

For application developers, choosing a suitable GUI framework is important to simplify the work and increase the efficiency in developing an application with the expected outcome. There are several criteria that developers have to consider, such as overall complexity, ability to handle and adaptability to emerging technologies. In the

progress of developing a UI interface application, it alleviates a lot of grunt work. Therefore, simplifying tasks such as error handling is important to reduce the debugging job after the structure of the classes has been completed. Before choosing a framework to learn, the developer can handle it. With the same function, different types of frameworks address issues uniquely. Therefore, the developer must understand the framework structure before working on it. The framework should be able to update constantly to keep up with the latest technology. This is to ensure that the platform can support the latest framework (Mckenzie, 2016).

### 2.6.1 Tkinter

Tkinter is the standard built-in GUI library for Python. It is the most used Python GUI framework as it is a mature framework that provides many widgets such as canvas, graph generation and pop-up messages. The 'Tk' extension actively involves a big community on the web. Any extension is immediately accessible from Tkinter.

The logo of the framework is shown in Figure 2.24 below. Furthermore, the pros and cons of the framework are summarised in Table 2.4 below.

**Table 2.4: Table summarising the pros and cons of Tkinter**

Pros	Cons
<b>User-friendly –</b> The syntax is simple, and no need to download	<b>Unattractive –</b> Lack of native look and feel
<b>Free for all –</b> Free licensed for all usage	<b>Platform Limitation –</b> Unable to develop a mobile application
<b>More accessible –</b> Many libraries can be accessed online	

### 2.6.2 Kivy

Kivy is an open-source multi-platform GUI development framework for Python. It can run on Windows, OS X, and mobile OS such as iOS and Android. It assists in developing apps that take advantage of creative, multi-touch user interfaces. Kivy's core concept allows developers to create an app once and utilise it across all devices, making the code reusable and deployable and enabling quick and easy interaction design and rapid prototyping (GeeksforGeeks, 2020). Furthermore, the framework's pros and cons are summarised in Table 2.5 below.

**Table 2.5: Table summarising the pros and cons of Kivy**

Pros	Cons
<b>User-friendly –</b> Easy-to-use widgets built with multi-touch support	<b>Unmature –</b> Lack of good examples and community support
<b>Free for all –</b> Free licensed for all usage	<b>Storage –</b> Large package size
<b>Multi-platforms –</b> Use across all devices	

### 2.7 Existing Surface Profiler Analysis Software

In the current market, there is no software capable of receiving raw input data from any equipment for surface roughness analysis. The majority of synchronisation software is connected with the specific instrument. Therefore, MicroEpsilon scanCONTROL and Insight Laser Profile software are compared in terms of their capabilities.

### 2.7.1 MicroEpsilon scanCONTROL

The Smart series sensors contain an intelligent controller that enables simple profile analysis without using a PC. The scanCONTROL Configuration Tools software is used to configure the profile analysis parameters. As well as the configuration of the sensor, this also enables the parameters of the measurement task to be set up and of the outputs, resulting in a compact, industrial, inline measurement solution. The software has the features of:

- Plug & Play solution for complex measurement tasks
- Real-time profile analysis inside the controller
- Real-time saving of profile data
- Load and save parameters
- Protocol feature for measured values
- Easy online and offline analysis

The interface of the software is shown in Figure 2.22 below.

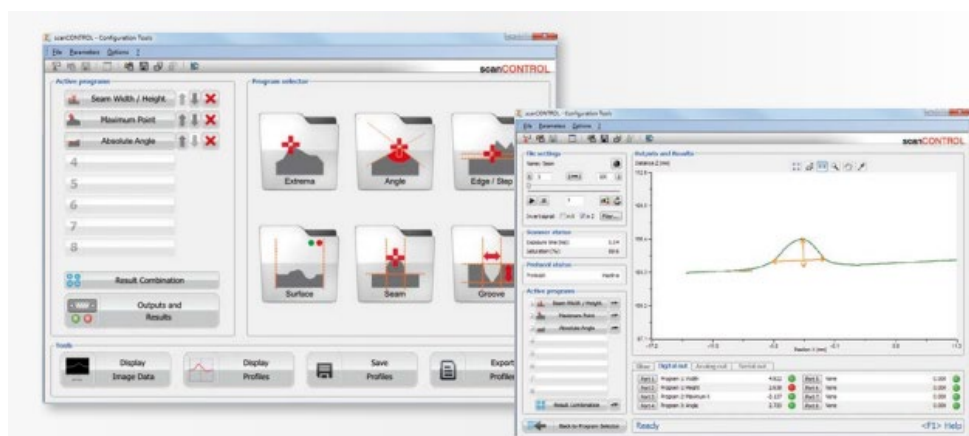


Figure 2.22: MicroEpsilon scanCONTROL software interface (Micro-Epsilon Messtechnik, 2022)

## 2.7.2 In-Sight Laser Profile Software

The In-Sight laser profiler uses In-Sight VC Explorer with EasyBuilder to set up and monitor a variety of measurements. The user-friendly interface guides operators through a step-by-step setup process, allowing rookie and expert users to deploy measuring applications rapidly and easily. The In-Sight laser profiler toolset is specifically designed for laser profiling and includes extraction, construction, and measurement toolsets. The software has the features of:

- Real-time monitoring
- Load and save parameters
- Monitor across all devices

The interface of the software is shown in Figure 2.23 below.

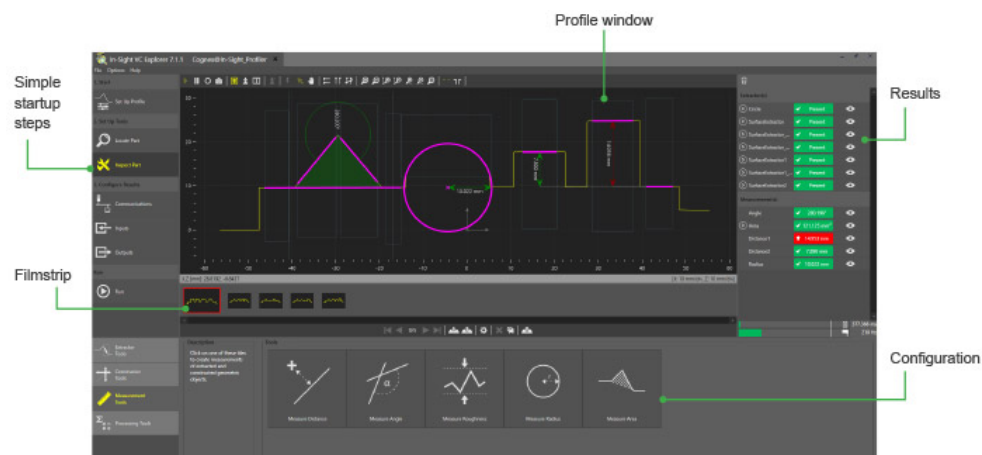


Figure 2.23: In-Sight Laser Profile software interface (Cognex, 2018)



### 2.7.3 Comparison of features

The software's features are compared and shown in Table 2.6 below.

**Table 2.6: Table of feature comparison between the existing surface profiler analysis software**

Features	MicroEpsilon scanCONTROL	In-Sight Laser Profiler
Plug-play	/	
Real-time analysis	/	/
Real-time saving data	/	/
Save analysis as an image		
Use across all devices		/
Compare data		

## 2.8 IoT Platform

The Internet of Things (IoT) aims to connect objects remotely for streamlined operation and convenience. A platform for the Internet of Things connects device sensors to data networks. It provides visibility into the data utilised by the application's backend. An IoT platform comprises components that enable developers to distribute programmes, collect data remotely, secure connectivity, and manage sensors. An IoT platform oversees the connectivity of the devices and allows developers to construct new mobile software apps. It facilitates device data collecting and promotes business change. It ensures a continuous flow of communication between devices by connecting disparate components (Hcltech, 2022).

In light of the foregoing, an IoT platform can perform in various use-case situations, depending on the requirements. It is also referred to as middleware, especially when discussing its ability to connect remote devices to applications and other devices and how it regulates interactions between the hardware and application layers (Upswift, 2022).

### 2.8.1 RaspController

The RaspController application facilitates remote Raspberry Pi control. Now files may control the GPIO ports, issue commands directly through the terminal, examine photos from a linked camera, and acquire data from various sensors. Finally, wiring diagrams, pins, and other relevant information are accessible to correctly use the Raspberry Pi (Egal Net, 2022.). Furthermore, the pros and cons of the RaspController are shown in Table 2.7 below. The application has the features:

- GPIO management (On/Off or impulsive function)
- File manager (Explore the content of Raspberry Pi, copy, paste, delete, download and visualise properties of files, text editor)
- Shell SSH (Send custom commands to your Raspberry Pi)
- Cpu, Ram, Storage, Network Monitoring

**Table 2.7: Table summarising the pros and cons of RaspController**

Pros	Cons
<b>User-friendly –</b> Easy to connect and control	<b>Unmature –</b> Lack of good examples and community support
<b>Free for all –</b> Free licensed for all usage	<b>Uncustomizable –</b> Fewer settings to work on for specific requirements
<b>Multi-platforms –</b> Use across all mobile operating systems (Android and iOS)	

### 2.8.2 Blynk

Blynk is an IoT platform for iOS and Android devices that enables Internet-based control of Arduino, Raspberry Pi, and NodeMCU. This application is used to generate a graphical user interface (GUI) or human-machine interface (HMI) by compiling and providing the correct widget address (Media's, Syufrijal and Rifan, 2019). Blynk was developed for the Internet of Things. It can remotely manage hardware, display sensor data, save data, analyse it, and do many other fascinating things. Furthermore, the pros and cons of the Blynk are shown in Table 2.8. Three primary components comprise the platform:

- Blynk App - It allows users to design stunning interfaces for projects utilising the given widgets.
- Blynk Server - It is in charge of all smartphone-to-hardware communications.
- Blynk Cloud - It is open-source, can easily manage thousands of devices, and can even be run on a Raspberry Pi.

**Table 2.8: Table summarising the pros and cons of Blynk**

Pros	Cons
<b>User-friendly –</b> Easy to connect and control	<b>Payment Needed –</b> Need to pay for more features
<b>Mature –</b> Large community support with good examples	<b>Data Storage –</b> Readings from hardware are auto-deleted after a specific time
<b>Multi-platforms –</b> Use across all mobile operating systems (Android and iOS)	

## **2.9 Cloud Storage**

Cloud storage is a service model in which data is transferred and kept on remote storage devices, where it is managed, backed up, and made accessible to users over a network — usually the internet. Users typically pay a monthly consumption-based fee for cloud data storage. Cloud service providers manage and maintain cloud-stored information. In the cloud, storage services are offered on demand, expanding capacity and contracting as needed. Cloud service providers operate expansive data centres in numerous global locations. Customers who acquire cloud storage from a provider delegate the majority of data storage responsibilities to the provider, including security, capacity, storage servers and computing resources, data availability, and network delivery. The cloud data can be accessed via traditional storage protocols or application programming interfaces (APIs), or customer applications can be migrated to the cloud (Chai, Castagna and Lelii, 2021).

### **2.9.1 Google Drive**

Google Drive is a cloud-based file storage and synchronisation service. It lets users store and share their files and personal information. It offers 15 GB of free space for storage. Google started the service in 2012. It is utilised by nearly everyone connected to the internet in some way. Everyone uses this to store personal and professional information (Mixon and Wigmore, 2016).

Drive connects with Docs, Sheets, and Slides, cloud-native collaboration apps that enable the team to more effectively produce content and collaborate in real-time. Drive enhances and connects with the team's existing technology. Collaborate in Microsoft Office files without converting file formats, and edit and save over 100 different file kinds, including PDFs, CAD files, and pictures (Google, 2019).

### 2.9.2 Dropbox

Dropbox is a personal cloud storage service (sometimes known as an online backup service) that is commonly used for file sharing and collaboration. Dropbox is compatible with Windows, Mac OS X, and Linux desktop operating systems. Apps are also available for the iPhone, iPad, Android, and Blackberry. The service offers 2 GB of free storage and up to 100 GB via various paid plans (Wigmore, 2011).

### 2.9.3 Comparison of features

Both cloud storage features are compared and shown in Table 2.9 below.

**Table 2.9: Table of feature comparisons between Google Drive and Dropbox**

<b>Google Drive</b>	<b>Dropbox</b>
Charges <b>\$9.99</b> per month for additional 2TB storage	Charges <b>\$11.99</b> per month for additional 2TB storage
Provides <b>full</b> security of data	Provides full security of data but comparatively <b>less</b>
The file upload limit is <b>5TB</b>	The file upload limit is <b>50GB</b>
Integrate with <b>Google's software suite</b>	Integrate with <b>Dropbox's software suite</b>

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 Overview**

This chapter discusses software configuration via programming language choices and GUI framework. The system overview is then presented, followed by a discussion of the filter used to optimise the graph and the software's implementation of surface roughness parameter calculation. The deployment of remote control and cloud storage on the prototype is examined to automate the system further. Hardware debugging and project management are discussed at the end of this chapter.

#### **3.2 Selection of Programming language**

After comparing the languages, Python is chosen as the primary language to code. There are several reasons and concerns for choosing that. Python is less challenging to learn than C++. C++ features additional syntax rules and programming practices. C++'s code is more extended, resulting in more challenging debugging duties. This is not advantageous for the software developer. In addition, the Python library enables customers to utilise whatever function they choose, particularly in data analysis and machine learning. This characteristic suggests that Python is an excellent option for the surface roughness parameters' analysis task. Python is well-known for its ease of use, less complex syntax, good readability, and active community support (SoftwareTestingHelp, 2022).

Since its debut in 1995, Java has been the most popular and sought-after programming language. Python has reached the top three most popular programming languages within a few years, as its popularity increases annually. The apparent disadvantage of Java compared to Python is that Java's syntax is far more complex than Python's. In Java, a simple task such as reading the contents of a file requires the import of numerous classes, but in Python, only two lines. Therefore, creating an application with Python saves much time and enhances productivity. Python has a built-in standard library known as TK (Tkinter for GUI interface), another feature. They have fewer GUI features than C++ and Java (SoftwareTestingHelp, 2022).

### **3.3 GUI Framework Selection**

Tkinter and Kivy are the Python language's GUI frameworks. Tkinter is selected as the primary framework for several reasons. This framework requires no installation as Tkinter is the standard Python interface to the TK GUI toolkit. By importing Tkinter, developers may immediately use the framework; this saves time and makes learning easier. However, Kivy requires installation, which is complicated for novice application developers. Second, Tkinter is an established and robust framework with an active community. It is well-documented on the Python website and in numerous tutorials written by bloggers or developers. This allows a newbie developer to utilise the community's excellent resources. Kivy is not popular enough to attract a large community as Tkinter did. This is because the majority of Android developers prefer to use Java. Java has additional capabilities and configuration flexibility. Therefore, it is advisable to use Tkinter as the primary programme development framework for simple stage development. Unquestionably, it is conceivable to use Kivy as the framework after the fundamental structure and functionality have reached maturity and are prepared for a platform move, such as a mobile application. It must be entertaining and an added software function (Kravchenko, 2022).



### 3.4 System Overview

As an overview, the surface profiler is used to extract the surface measurement data. Then, the extracted data is imported into the software to analyse the surface roughness. The flowchart of the system between the hardware and software is shown in Figure 3.1 below.

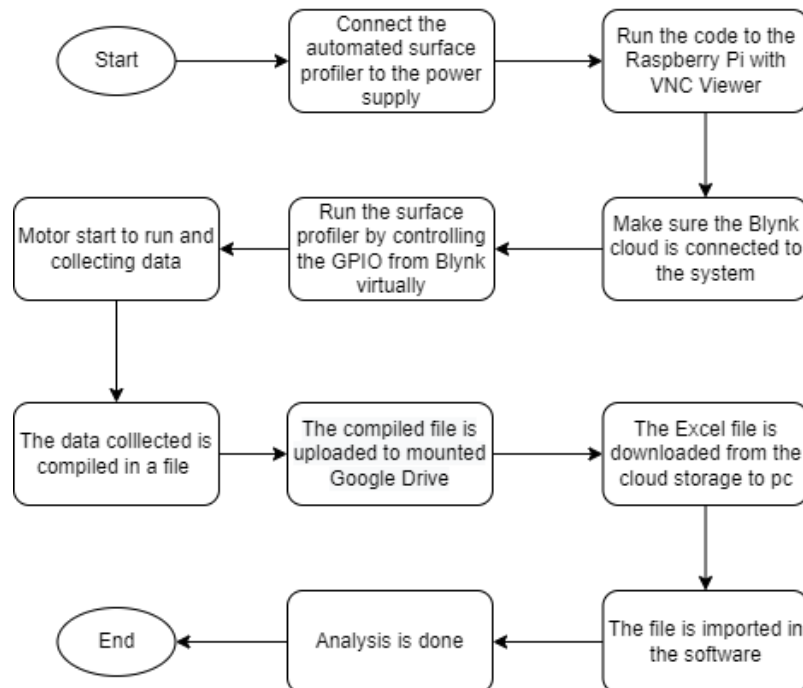


Figure 3.1: Flowchart of system overview

The ProSight Surface Profiler software is an automated system that reads data from the surface profiler's data file. A graph is constructed for analysis. Users can easily relocate the graph to increase the efficiency and effectiveness of the analysis task. In addition, users can save the graph as an image for record-keeping purposes and distribute it to whomever they choose. With the software's.exe extension, the developer can send the download link directly to the engineer. Figure 3.2 is a software flowchart illustrating automated data analysis.

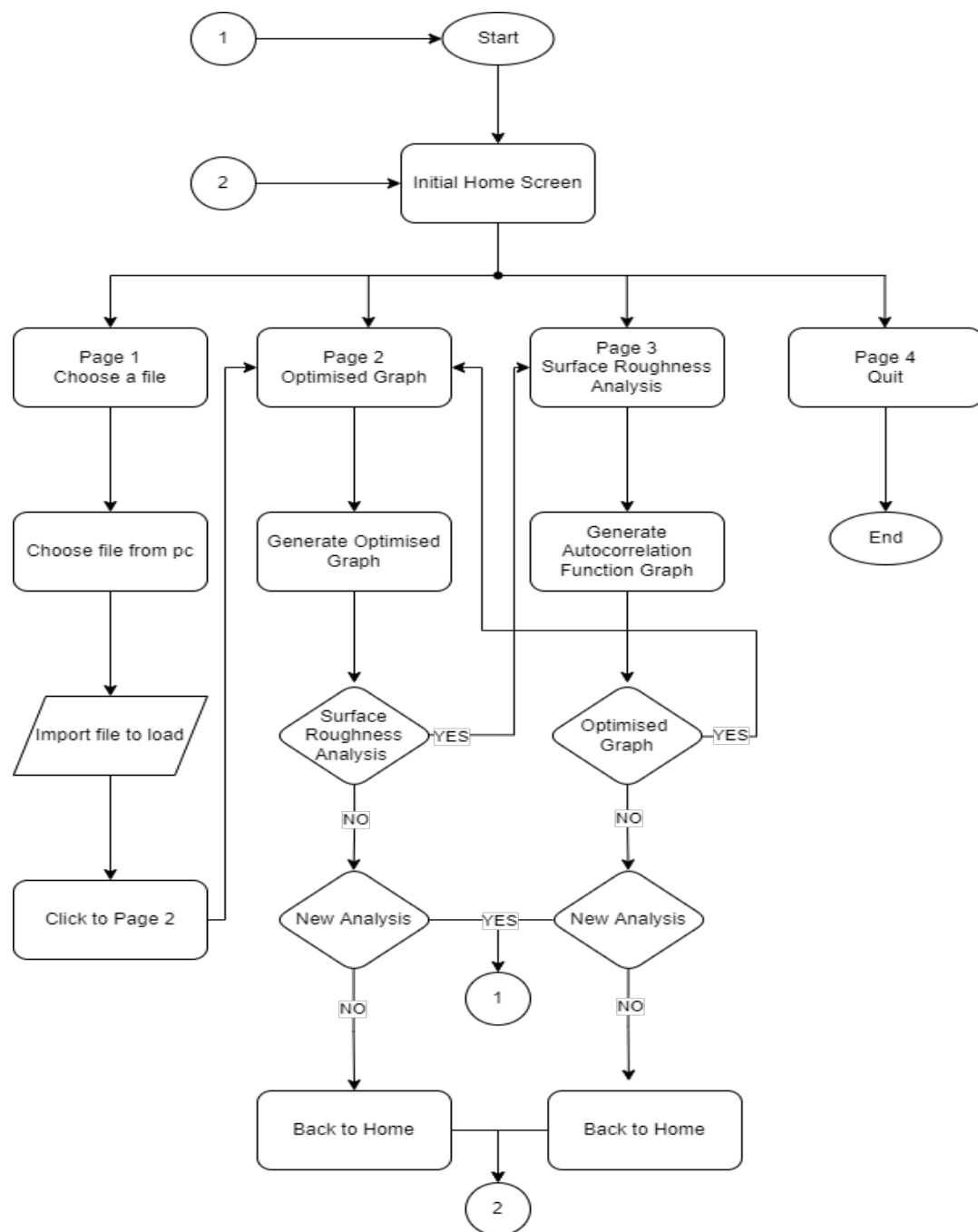


Figure 3.2: Flowchart of software process

### **3.5 System Performance Definition**

The system performance can be defined in detail from multiple perspectives:

#### **1. Real-time analysis**

- i. The file generated from the surface profiler can be imported in real-time from Google Drive and run in the software.
- ii. The software can calculate the autocorrelation length and RMS height of the surface.
- iii. Both .xlsx and .csv file are readable.

#### **2. Data View**

- i. The imported data can be viewed in a table form with each heading.
- ii. The data is used to extract for the generating graph purpose.

#### **3. Graph generated**

- i. The data from the imported file is generated with the x-axis of distance and the y-axis of height.
- ii. The graph can compare with the following imported data.
- iii. An optimised graph and autocorrelation function graph are generated seamlessly.

#### **4. Saved data**

- i. The graph can be saved as an image permanently on the pc.
- ii. Users can save the image in any view they want.

### 3.6 Technologies and Tools involved

#### 1. Hardware

As shown in Table 3.1 below, a laptop with specifications is used.

**Table 3.1: Table of laptop specifications**

Hardware	Specification
Operating System	Windows 11
Processor	Intel® Core™ i5-8250H CPU @ 1.60GHz
Installed memory (RAM)	8GB
System type	64-bit operating system

#### 2. Software

VNC Viewer is used to configure the Raspberry Pi 3 remotely. No cable is needed when uploading the Python code to the Raspberry Pi 3. However, the IP address of both devices has to be the same.

### 3.7 Graph Optimisation

As shown in Figure 3.3 below, the box is positioned and runs the scanning with the Automated Electronic Sensor System. The box is measured with phone AI measurement. The dimension calculated is  $10\text{ cm} \times 3\text{ cm}$ . The raw data is being compared with 3 different filters to optimise the graph generated.

The filters used are moving average, median, and Savitzky-Golay Filter. These filters are tested on different parameters to find the most suitable and accurate outcome.



Figure 3.3: The box is used as the reference to the graph optimising filter selection

### 3.7.1 Moving Average Filter

A library called SciPy must be imported into the system. The library is called to import 'signal'. SciPy does not have a built-in implementation of a moving average filter, but it is easy to implement it. A moving average of order  $n$  has an impulse response with  $n$  elements that all have the value of  $1/n$ . The  $n$  value can only be an odd number. So,  $n=5, 7$  &  $9$  are used to optimise the graph. The function and outcomes are shown in Figure 3.4 and Table 3.2 below.

```
def ma(x, n = 5):
    b = np.repeat(1.0/n, n) #Create impulse response
    xf = scipy.signal.lfilter(b, 1, x) #Filter the signal
    return(xf)
```

Figure 3.4: Function of moving average filter in Python

**Table 3.2: Optimised graph (Dark Blue) of Moving Average Filter with different values of  $n$**

$n$	Optimised graph (Dark Blue)
5	
7	
9	

The optimised graph removes spikes as  $n$  increases. However, the object's height is not optimally optimised. The upward gradient of the optimised graph is tilted, which is a negative result. All optimised graphs for  $n = 5, 7$  &  $9$  show a sloping line as it approaches 34 cm. The box's surface is not flat in the optimal graph for all  $n$ . Therefore, the software does not employ the moving average filter.

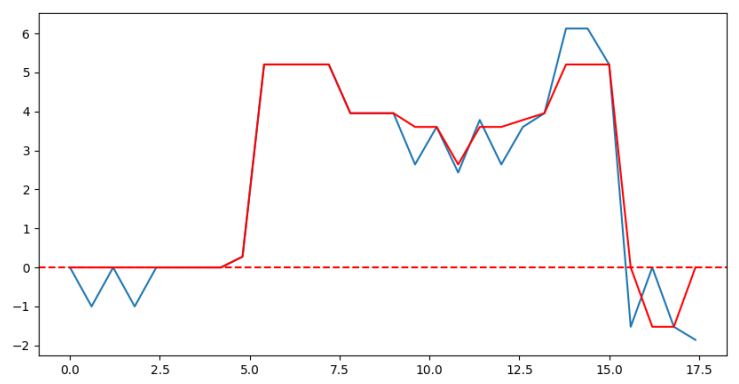
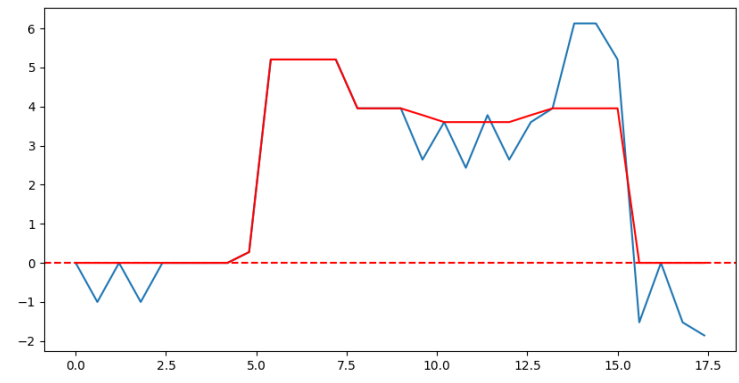
### 3.7.2 Median Filter

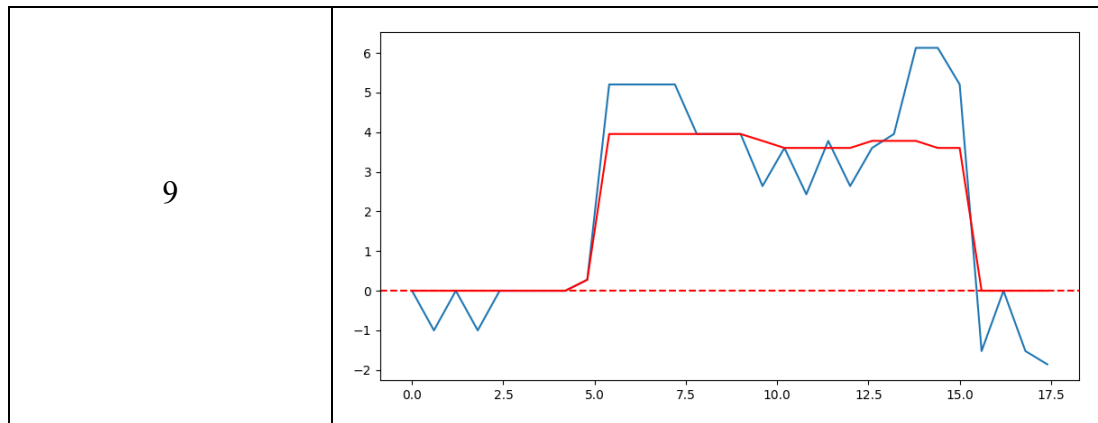
A library called SciPy must be imported into the system. The library is called to import 'signal'. SciPy has a command for the median filter, as illustrated in Figure 3.7. Thus, its implementation is straightforward. The filter has a parameter of *kernel\_size*. It is a scalar or an N-length list giving the size of the median filter window in each dimension. Elements of *kernel\_size* should be odd. If *kernel\_size* is a scalar, then this scalar is used as the size in each dimension (Virtanen et al., 2020). The default size is 3 for each dimension. So,  $n = 5, 7$  &  $9$  are used to optimise the graph. The function and outcomes are shown in Figure 3.5 and Table 3.3 below.

```
y_med11 = scipy.signal.medfilt(y, kernel_size=9)
```

Figure 3.5: Command of median filter in Python

**Table 3.3: Optimised graph (Red) of Median Filter with different values of  $n$**

$n$	Optimised graph (Red)
5	
7	



When the value of  $n$  increases, the optimised graph eliminates the spikes and calculates the average height of the newly created spikes to retrieve the actual height of the item. The box in Figure 3.6 has a dimension of  $10\text{ cm}$  in length and  $3\text{ cm}$  in height. The optimised graph of  $n = 9$  shows that the height is between  $3 - 4\text{ cm}$ . However, the optimised graphs of  $n = 5$  &  $7$  show that the spikes are not eliminated evenly. The state of the box's height is unfavourable. The box's surface is not flat in the optimised graph of  $n = 5$  and  $7$ . Therefore,  $n = 9$  of the median filter is the most accurate among other values of  $n$ .

### 3.7.3 Savitzky-Golay Filter

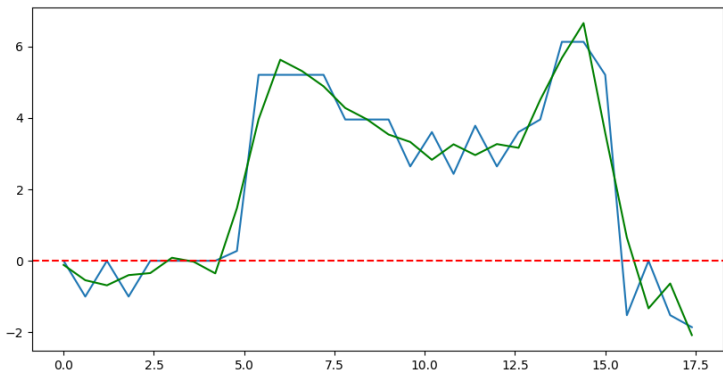
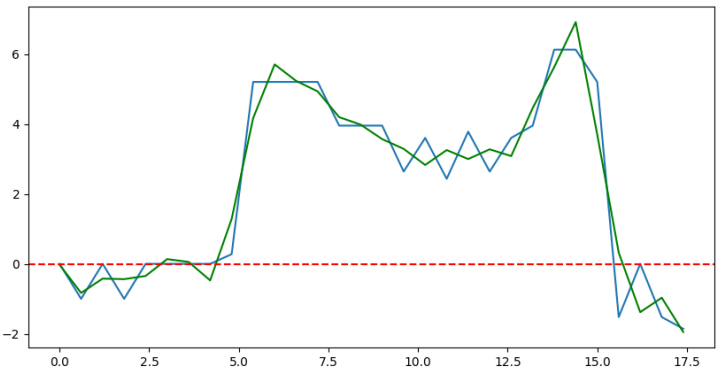
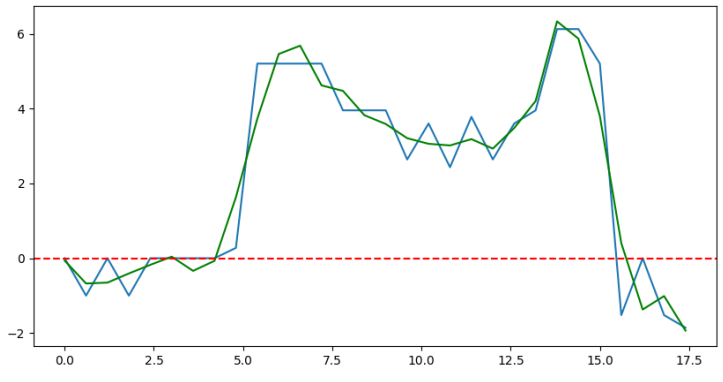
A library called SciPy must be imported into the system. The library is called to import 'signal'. SciPy has a command for the median filter, so it is easy to implement. A Savitzky-Golay filter calculates a polynomial fit of each window based on polynomial degree and window size. Scipy `savgol_filter()` function requires one-dimensional array data, window length, polynomial order, and other optional parameters. The parameters are shown in the command in Figure 3.8 as (y, window size = 20, polynomial order = 5). So, polynomial order = 5, 7 & 9 are used to optimise the graph. The function and outcomes are shown in Figure 3.6 and Table 3.4 below.

```
yhat = scipy.signal.savgol_filter(y, 20, 5)
```

Figure 3.6: Command for Savitzky-Golay Filter



**Table 3.4: Optimised graph (Green) of Median Filter with different values of  $n$** 

Polynomial order	Optimised graph (Green)
5	
7	
9	

When the value of polynomial order increases, the optimised graph does not eliminate the spikes. Also, the height of the object is not accurately optimised. When moving up, the gradient of the optimised graph is slanted, which is a negative outcome. All the optimised graph of polynomial order = 5, 7 & 9 shows the slanting line when going up to 4 – 6 cm. The box's surface is not flat in the optimised graph of all polynomial order. Therefore, the Savitzky-Golay Filter is not being used in the software.

### 3.7.4 Conclusion (Graph Optimisation)

After comparing 3 filters, the most accurate and suitable filter is the **Median Filter with the n value of 9**. For the Moving Average Filter and Savitzky-Golay Filter, both filters resulted in a negative result as they are not removing the spikes evenly, and the height surface is non-flat as the actual object.

## 3.8 RMS Height

RMS height is also known as the standard deviation of surface height. Therefore, a library called NumPy is imported. NumPy provides extensive mathematical functions, random number generators, linear algebra operations, and Fourier transform, among other things. The standard deviation measures the dispersion of an array of element distributions. By default, the standard deviation is computed for the flattened array; otherwise, it is computed for the chosen axis (Harris et al., 2020).

The existing data from the previous research on the sea and snow surface roughness measurement are used to validate the accuracy of the calculation. The calculated data is given a tolerance of  $\pm 5\%$ . There are 8 existing data to be used to measure the precision of the calculation. The results are shown and discussed in Chapter 4.

The command below, shown in Figure 3.7, is used to calculate the RMS height of input data. The 'y' is the height of the surface as input data. While 'np.float64' is to allow the calculated answer to be more accurate with more decimal numbers.

```
std=np.std(y,dtype=np.float64)
```

Figure 3.7: Command to calculate RMS height of surface roughness

### 3.9 Surface Roughness Autocorrelation Length

Calculating the autocorrelation length of the surface roughness is a bit tricky. This is because there is no existing library to calculate the length automatically. Therefore, the. Next, a line of  $\frac{1}{e} = 0.36788$  is drawn horizontally in the autocorrelation function graph. Then, the x value, which intersects between the  $y = 0.36788$  and the autocorrelation function graph, is listed. The x-axis value is the correlation length. The flowchart of the steps finding the autocorrelation length is shown in Figure 3.8 below.

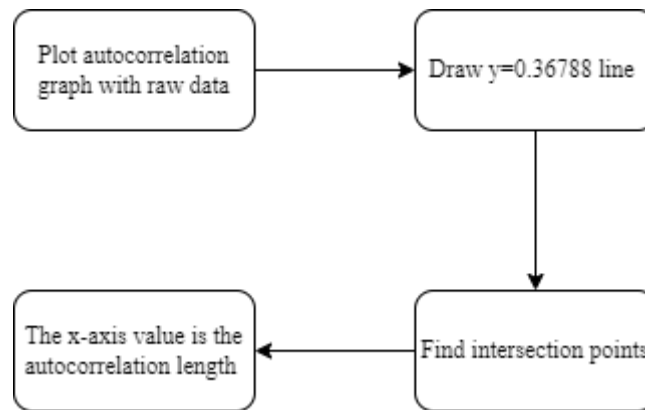


Figure 3.8: Flowchart of finding the autocorrelation length of surface roughness

The graph shown in Figure 3.9 below is the autocorrelation function graph (blue line) with the raw data from the snow site. The orange dot is the intersection point between the horizontal line of  $y=0.36788$  and the autocorrelation function graph. The intersection point's x-axis value is the surface roughness's autocorrelation value.

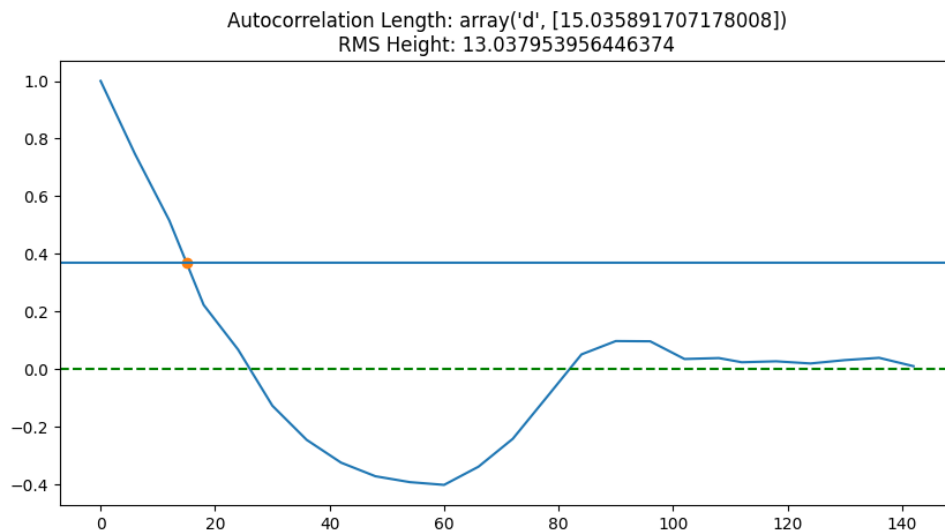


Figure 3.9: Finding the intersection point between  $y=0.36788$  and the autocorrelation function graph

The autocorrelation function graph is plotted by using `tsaplots.plot_acf( )` function from the Statsmodels library. The command is run as shown in Figure 3.10 below. The x-axis displays the number of lags, and the y-axis displays the autocorrelation at that number of lags. By default, the plot starts at lag = 0, and the autocorrelation is always 1 at lag = 0.

```
lags=range(30)
acorr = sm.tsa.acf(y, nlags = len(lags)-1)
```

Figure 3.10: Command for the generation of the autocorrelation function graph

The intersection point is found using a library called “shapely.geometry” that imports a function called `LineString`. The function separated the autocorrelation function graph and the horizontal line of  $y=0.36788$  into 2 lines, and then the intersection of both is located. Finally, the autocorrelation length of the surface roughness is found. The code to find the intersection point is stated as shown in Figure 3.11 below.

```

first_line = LineString(np.column_stack((xc, np.array(acorr))))
second_line = LineString(np.column_stack((xc, np.array(y0))))
intersection = first_line.intersection(second_line)

if intersection.geom_type == 'MultiPoint':
    a.plot(*LineString(intersection).xy, 'o')
elif intersection.geom_type == 'Point':
    a.plot(*intersection.xy, 'o')

x,y=intersection.xy
print(x,y)

```

Figure 3.11: Code to find the intersection point of 2 lines

### 3.10 Selection of the IoT platform

In Chapter 2, RaspController and Blynk are compared. Blynk is picked for numerous reasons. Although RaspController is the most suitable platform with Raspberry Pi for monitoring the system and managing the GPIO, it is not the only option. Nonetheless, the RaspController community is still immature. Compared to Blynk, there are fewer resources and decent examples to refer. Blynk provides multiple platforms for developers to simply configure the system. Blynk makes it simple for developers to add additional sensors for data collection. Blynk requires the developer to pay for additional functionality, but the free version is sufficient for enhancing the system. In addition, Blynk can operate the Raspberry Pi GPIO from various devices, including smartphones, tablets, and computers. It offers the developer a great deal of convenience.

It is simple to configure Blynk and connect hardware to the Blynk app. Create a Blynk account via the Blynk.Console. Then, Blynk will send an email containing the Blynk.Cloud authorization code for the Raspberry Pi. On the Raspberry Pi running Linux, the Blynk library must be downloaded. In order to synchronise with Blynk.Cloud, the Blynk authorization code is inserted during the coding phase. Figure 3.12 depicts the format of the command. Following Raspberry Pi synchronisation, a function is written to setup the GPIO.

```
BLYNK_AUTH = '-7a6qkCMCVZ9P1k0oAhJlSzNdFj9esRE'
blynk = BlynkLib.Blynk(BLYNK_AUTH, server="blynk.cloud")
```

Figure 3.12: Command line to sync Raspberry Pi with the Blynk Cloud

During this pandemic, contactless machines are already a notable subject of discussion. Therefore, a contactless function is implemented in the system by remotely activating the start button compared to physically pressing it. By integrating the virtual pin GPIO, the automated surface profiler researcher does not have to enter a hazardous area to click the start button. As seen in Figure 3.13, a function is written to manage the GPIO pin of the Raspberry Pi.

```
# Push button to start running
@blynk.VIRTUAL_WRITE(0)
def write_virtual_pin_handler(value):
    if value == ["1"]:
        GPIO.output(startrun, GPIO.HIGH)
        print("START")
        main()
        sys.stdout.flush()
        restart_program()
    elif value == ["0"]:
        GPIO.output(startrun, GPIO.LOW)
        print("PENDING")
try:
    while True:
        blynk.run()
```

Figure 3.13: Function of controlling the GPIO pin virtually

A condition is applied to the function. The signal is sent if the GPIO value is determined to be one, allowing the motor to keep running. After the application has been entirely executed, the system is restarted and continues to detect the GPIO's value. If the detected value is zero, the motor is not driven, and the Python IDLE shell displays "Pending." Figure 3.14 is the flowchart describing how the Blynk controls the GPIO. The Blynk Cloud dashboard displays the online status when the device is connected. Then, the device is prepared to operate the hardware by clicking the virtual button illustrated in Figure 3.15 on the smartphone/display.

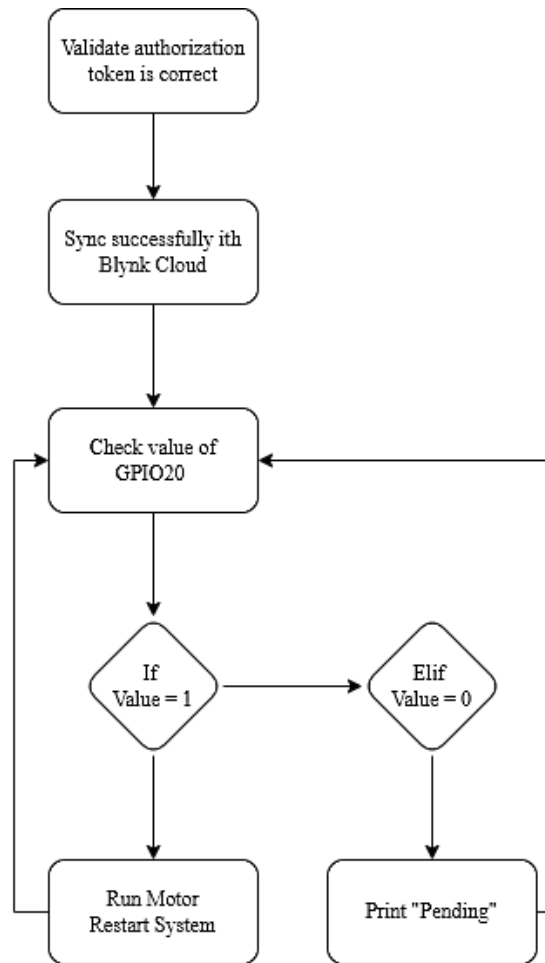


Figure 3.14: Flowchart of the Blynk controlling the GPIO

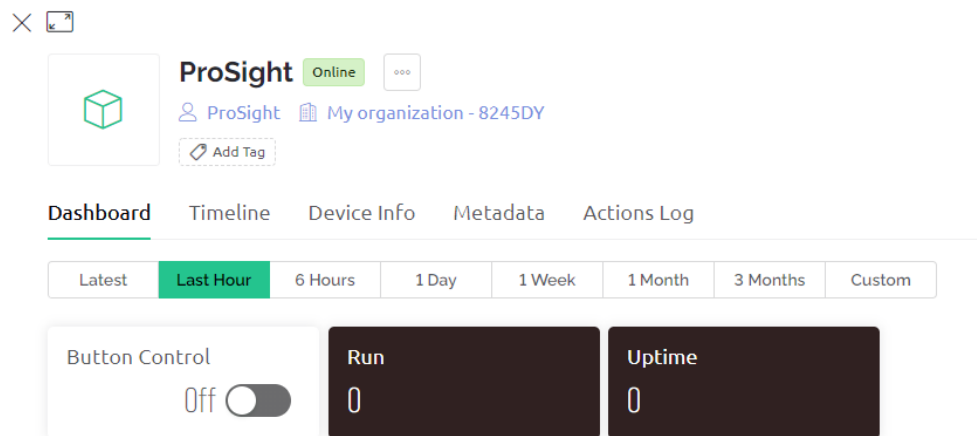


Figure 3.15: Blynk Cloud dashboard showing the device status and virtual pin

### 3.11 Selection of the cloud storage platform

In Chapter 2, Google Drive and Dropbox are evaluated to determine the best cloud storage solution for the system. Google Drive is eventually selected as the cloud storage for all measured data. This is because Google Drive is significantly less expensive than Dropbox. Google Drive offers a bigger maximum storage capacity even with the free version. In addition, Google Drive offers complete protection for files that interface with Google's software suite, including Excel, Word, and Gmail. In addition to the file format of our raw data, Google's software is highly well-known and recognisable on the market.

The implementation of cloud storage is intended to automate data extraction. In the previous project, data extraction was performed using a USB cable. This makes it difficult for researchers to analyse data in real-time and slows job progress. The automation of data extraction facilitates research and increases the efficiency of data analysis tasks. Mounting Google Drive in a Linux environment on a Raspberry Pi requires importing a library called 'rclone', which is actively maintained and offers a great deal of functionality. With the assistance of the FUSE userspace filesystem layer, it is possible to mount Google Drive as part of your Linux file system, dramatically simplifying cloud storage from Raspberry Pi.

Since remote files are exclusive to a given user, Rclone maintains distinct configuration files for each user. All secret access tokens required to communicate with cloud storage are kept in a user's configuration file, preventing access to cloud storage by other users. By default, the config file is in *\$HOME/.config/rclone/rclone.conf* and should have permissions of 0600 (read-write only for file owner) to ensure user access tokens remain private. For Google Drive, there is an additional aspect to consider. With the default settings, the Rclone application's client id is used. Google imposes per-client rate limits on Google Drive interactions. This implies that by utilising the Rclone default client id, the user is competing with all other Rclone users worldwide for operation bandwidth to Google Drive. Even though the Rclone writers attempt to minimise this by requesting greater quotas from Google, utilising the default is not ideal. This necessitates establishing a client id with Google and avoiding the stampede.



First, go to the Google Developer Console in a web browser and sign in with the Google account. A new project is established, and the Google Drive API and services are enabled. The consent screen configuration is then selected on the Credentials tab to generate an OAuth client ID. Finally, the client ID and client secret are generated for the user's project. The tab appears as illustrated in Figure 3.16.

Name *	ProSight Surface Profiler
The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.	
Client ID	648599136713-7a0fg32o0us34ba4f2kndm2ovcr8p7c6.apps.googleusercontent.com
Client secret	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
Creation date	August 11, 2022 at 7:29:15 PM GMT+8

Note: It may take 5 minutes to a few hours for settings to take effect

Figure 3.16: Creating OAuth client ID in Google Developer Console

After obtaining the client ID and secret, the Rclone configuration must be performed to mount Google Drive on the Linux system. The configuration is performed by entering the command "Rclone config" in the Linux system's terminal. By following the given instructions, a new remote is generated. An authorisation code is issued after verification with the link provided at the end of the steps. The area depicted in Figure 3.17 is a replica of the information displayed when the remote is successfully inserted.

```
[gdrive]
type = drive
client_id = 648599136713-7a0fg32o0us34ba4f2kndm2ovcr8p7c6.apps.googleusercontent.com
client_secret = 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
scope = drive
token = {"access_token": "ya29.a0AVA9y1uAWcCclZ-FkfmxiujQmEZxcftwwV_EGguZi2ICntY3apB5SF2CsmvSJ3AUy3bXeDgZDd6hsIYqKRXZMtYVsErxCly_F5o0X4jk0uSY1Bz5QpaaziRHgsxzbI85YHGeuyejfibGt2t2GTzDgEgJeXAaCgYKATASQAFQE65dr8tKtJq6dpj_iJ7Vd4BqdgpQ0165",
token_type": "Bearer", "refresh_token": "1//0g0dnM0_CZ5sgCgYIARAAGBASNwF-L9IrsC38RHy3NYFz2yBH3TnAn2SNHsgrfazP8yJBbKpaj_t82mxP0nJ2jC9M8Wg_LNGI08", "expiry": "2022-08-04T19:13:41.070773405+01:00"}
-----
y) Yes this is OK (default)
e) Edit this remote
d) Delete this remote
y/e/d> █
```

Figure 3.17: A copy of information after the remote is added

Following Google Drive is added to a system, the Google Drive is unmounted after a system restart. Consequently, a configuration is introduced to enable the storage to be automatically mounted once the respective user logs in. This is possible with `systemd`, a daemon that automatically launches and disables services. `Systemd` operates in one of two modes: a system mode that handles hardware services such as setting up networking, and a user mode that handles per-user services such as launching the desktop environment. However, mounting the Rclone FUSE file system for Google Drive is required in this circumstance.

To tell `systemd` what to do depends on configuration files in some standard locations. One of these locations where `systemd` looks for user config files is `$HOME/.config/systemd/user`. To have the Rclone FUSE file system mounted automatically at login, a `~/.config/systemd/user/rclone@.service` file contains the following contents, as shown in Figure 3.18 below.

```
[Unit]
Description=rclone: Remote FUSE filesystem for cloud storage config %i
Documentation=man:rclone(1)

[Service]
Type=notify
ExecStartPre=/bin/mkdir -p %h/mnt/%i
ExecStart= \
    /usr/bin/rclone mount \
        --fast-list \
        --vfs-cache-mode writes \
        --vfs-cache-max-size 100M \
        %i: %h/mnt/%i

[Install]
WantedBy=default.target
```

Figure 3.18: `rclone@.service` file content

The file is mounted in the system. Every time a user logs in to a Raspberry Pi, Google Drive is automatically added to the file system, providing access to the cloud storage's contents. Figures 3.19 and 3.20 depict that the Google Drive mounted on a Linux system is synchronised with the online drive.

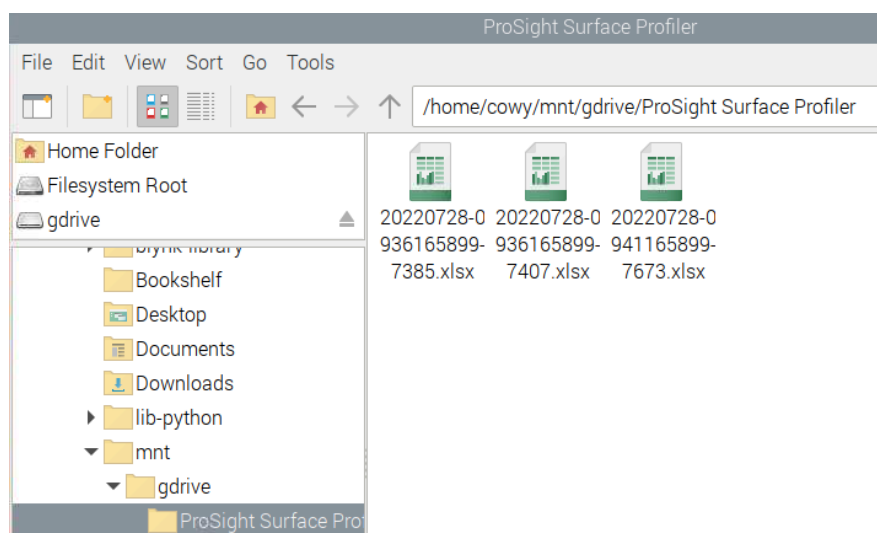


Figure 3.19: Mounted Google Drive in Linux system

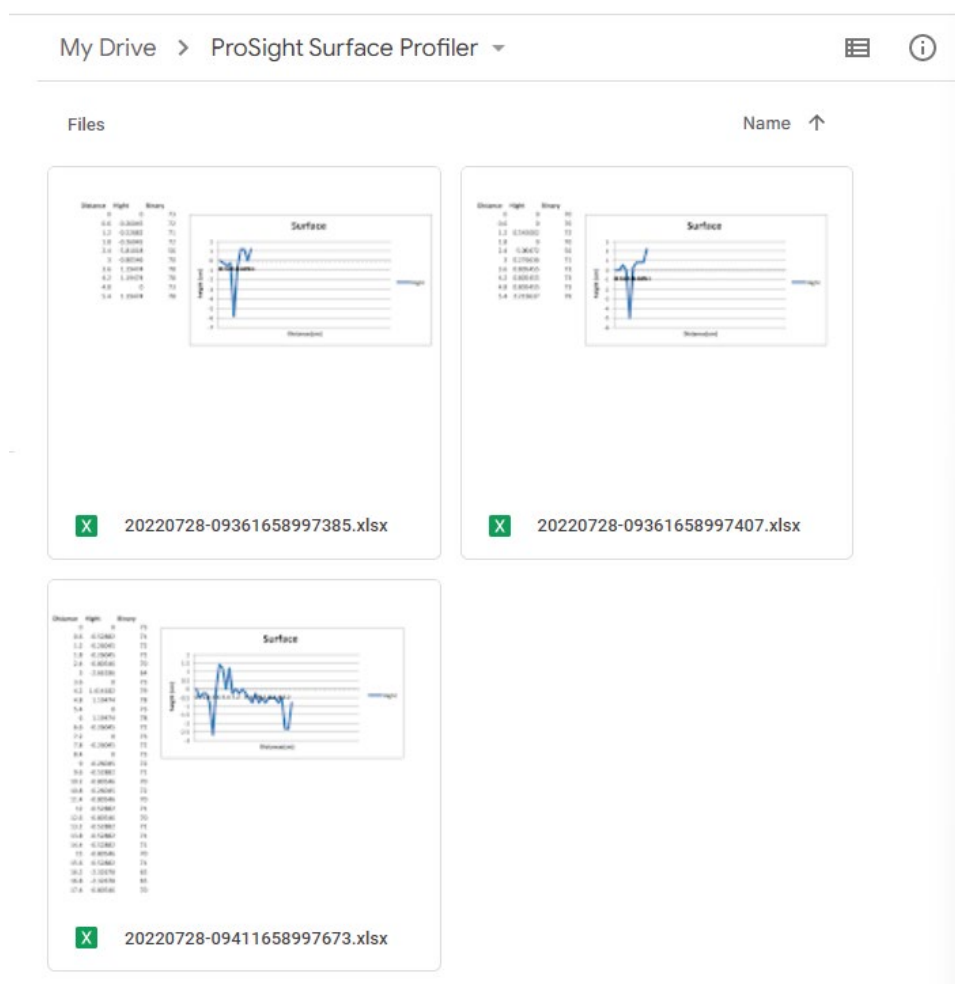


Figure 3.20: Google Drive sync with the mounted drive in Linux

### 3.12 Implementation and Testing

#### 3.12.1 Import of library

Beginning stage of the software development, a few libraries are imported.

- Matplotlib – Used to create a static, interactive and animated canvas.
- Tkinter – Used for the widget and button generation feature to design the interface.
- Pandas – Used for data analysis, such as reading the excel data from pc.
- Shapely – Used for manipulation and analysis of planar geometric objects. Mainly in finding the intersection point of graph analysis in the system.
- SciPy – Used for signal processing in the median filter of graph optimisation.
- NumPy – Used for comprehensive mathematical functions, integrated with Scipy and Shapely to calculate autocorrelation function and RMS value.
- Stasmodels – Used for conducting statistical tests and data exploration. The system is used for calculating the autocorrelation function in the data analysis.
- Pillow – Used for image processing. It is used to import and display an image on the system's interface.
- System, Contextlib & Warnings – Used for managing resources within a program to prevent the specific error message from popping out.

The command to import the libraries is shown in Figure 3.21 below.

```

import matplotlib
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
from matplotlib.backends_bases import key_press_handler
from matplotlib.figure import Figure
matplotlib.use("TkAgg")

import tkinter as tk
from tkinter import ttk, filedialog, messagebox, PhotoImage

import shapely
from shapely.geometry import LineString

import pandas as pd

import scipy.signal
from scipy.signal import medfilt

import numpy as np
from numpy import ones, vstack
from numpy.linalg import lstsq

import statsmodels.api as sm

from PIL import ImageTk, Image

import sys
import contextlib
import warnings
from packaging import version

```

Figure 3.21: Command for importing the libraries of the program

### 3.12.2 Debugging errors

When a recursive function is executed in Python on a large input ( $> 10^4$ ), a "maximum recursion depth exceeded error" might be encountered. This is a common error when executing algorithms such as DFS and factorial on large inputs. This is common in competitive programming on multiple platforms when running a recursive algorithm on various test cases. Therefore, a command line is coded, as shown in Figure 3.22, to modify the recursion limit in Python.

```
sys.setrecursionlimit(2000)
```

Figure 3.22: Command to increase the recursion limit

Shapely provides an array interface to access the coordinates, such as NumPy array, easily. Starting with Shapely 1.8, converting a geometry object to a NumPy array raises a warning: "ShapelyDeprecationWarning: The array interface is deprecated and

will no longer work in Shapely 2.0." NumPy attempts to access the array interface of objects or determine if an object is iterable or has a length, which is now a deprecated action. Even though the result is still correct, the warnings appear. In order to preserve the code that depends on Shapely, the code seen in Figure 3.23 is used to suppress the error and enable the programme to execute in various versions of Shapely.

```
SHAPELY_GE_20 = version.parse(shapely.__version__) >= version.parse("2.0a1")

try:
    from shapely.errors import ShapelyDeprecationWarning as shapely_warning
except ImportError:
    shapely_warning = None

if shapely_warning is not None and not SHAPELY_GE_20:
    @contextlib.contextmanager
    def ignore_shapely2_warnings():
        with warnings.catch_warnings():
            warnings.filterwarnings("ignore", category=shapely_warning)
        yield
else:
    @contextlib.contextmanager
    def ignore_shapely2_warnings():
        yield
```

Figure 3.23: Command to suppress the error of ShapelyDeprecationWarning

### 3.12.3 Global definition

The variables of `df_columns`, `df_columns1`, and `file_path` are used as the list for the usage of different classes. The font elements are defined for the visual interface design. The definitions 'f' and 'a' are used for plotting graphs from Matplotlib. The figure below, shown in Figure 3.24, is the code for global definition to be used in the program.

```
df_columns = []
df_columns1 = []
file_path = []

LARGE_FONT= ("Verdana", 12)
NORM_FONT = ("Helvetica", 10)
SMALL_FONT = ("Helvetica", 8)

f = Figure(figsize=(5,5), dpi=100)
a = f.add_subplot(111)
```

Figure 3.24: Global definition in the program

### 3.12.4 Design for Homepage (Class I)

Class SeaofBTCapp has functioned for the homepage interface design. The design elements include the icon and header of the software, logo interface, widget orientation, menu bar, and most importantly, the frame for the buttons for entering the main features page. The figures below, as shown in Figure 3.25 and Figure 3.26, are the code for class I and the homepage interface, respectively.

```
class SeaofBTCapp(tk.Tk):

    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)

        tk.Tk.iconbitmap(self, default="clienticon.ico")
        tk.Tk.wm_title(self, "Automated Surface Profiler")

        prop = ImageTk.PhotoImage(Image.open("2.png"))
        l4 = ttk.Label(borderwidth=2, image=prop)
        l4.image = prop
        l4.pack(side="top")

        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand = True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        for F in (StartPage, PageOne, PageTwo, PageThree):

            frame = F(container, self)

            self.frames[F] = frame

            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame(StartPage)

    def show_frame(self, cont):

        frame = self.frames[cont]
        frame.tkraise()
```

Figure 3.25: Class for Homepage design interface

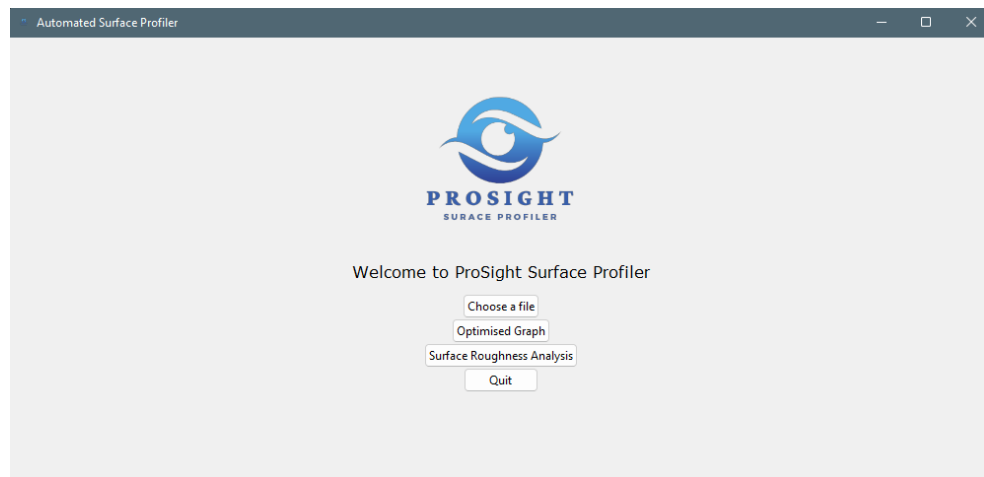


Figure 3.26: Frame structure of Homepage

### 3.12.5 Widgets for Homepage (Class II)

The class StartPage is used to set up font and buttons to press for entering the feature page. The 'controller.show\_frame' is to call the classes for the defined function. The class "Startpage" command and homepage widgets are shown in Figure 3.27 and Figure 3.28 below.

```
class StartPage(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Welcome to ProSight Surface Profiler", font=LARGE_FONT)
        label.pack(pady=10, padx=10)

        button1 = ttk.Button(self, text="Choose a file",
                              command=lambda: controller.show_frame(PageOne))
        button1.pack()

        button2 = ttk.Button(self, text="Optimised Graph",
                              command=lambda: controller.show_frame(PageTwo))
        button2.pack()

        button3 = ttk.Button(self, text="Surface Roughness Analysis",
                              command=lambda: controller.show_frame(PageThree))
        button3.pack()

        button4 = ttk.Button(self, text="Quit",
                              command=quit)
        button4.pack()
```

Figure 3.27: Class of StartPage Widgets



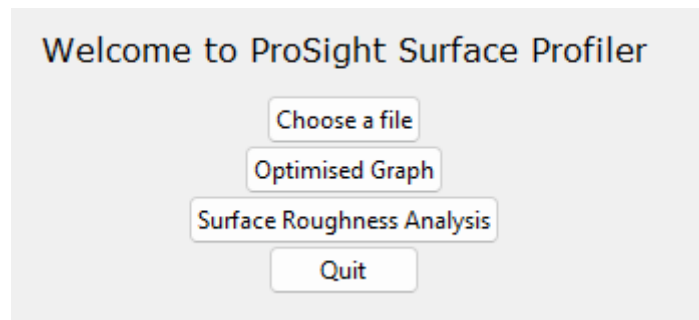


Figure 3.28: Widgets and Fonts on Homepage

### 3.12.6 PageOne (Class III)

The initialisation function (`def __init__`) defines the page's frame and font size. It also initiates the TreeView for data visualisation purposes. The function is shown in Figure 3.29 below.

```
class PageOne(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Excel Data", font=LARGE_FONT)
        label.pack(pady=10, padx=10)

        # Frame for TreeView
        frame1 = tk.LabelFrame(self, text="Excel Data")
        frame1.place(height=350, width=1000)
```

Figure 3.29: PageOne initialisation function

The buttons to decide which action to perform next are implemented with Tkinter and summon the function with "command=lambda". The functions of "File\_dialog" and Load\_excel\_data" are included. When there is no file selected on the page, a label is shown to indicate "No File Selected". The command is shown in Figure 3.30 below. Then a function called (File\_dialog) is in the ' \_\_init\_\_ ' function to open the file explorer and assign the chosen file to 'label\_file'. The function is shown in Figure 3.31 below.

```
# Buttons
button1 = tk.Button(self, text="Browse A File", command=lambda: File_dialog())
button1.place(rely=0.55, relx=0.30)

button2 = tk.Button(self, text="Load File", command=lambda: Load_excel_data())
button2.place(rely=0.55, relx=0.45)

button3 = tk.Button(self, text="Click to proceed", command=lambda: controller.show_frame(PageThree))
button3.place(rely=0.55, relx=0.58)

# The file/file path text
label_file = ttk.Label(self, text="No File Selected")
label_file.place(rely=0, relx=0)
```

Figure 3.30: PageOne widgets and interface command

```
def File_dialog():
    """This Function will open the file explorer and assign the chosen file path to label_file"""
    filename = filedialog.askopenfilename(initialdir="/",
                                          title="Select A File",
                                          filetype=(("xlsx files", "*.xlsx"), ("All Files", "*.*")))
    label_file["text"] = filename
    return None
```

Figure 3.31: PageOne search file function

The 'Load excel data' function reads the previously selected Excel file and puts the list into the treeview frame. The selected file is categorised into different lists to facilitate the analysis process. The Pandas library handles the classification process. The function also imports the data as a list into global variables for use by other classes. The function is depicted in Figure 3.32. A clear\_data is also defined to ensure the list is clear after every data import. The function is shown in Figure 3.33 below. The interface of PageOne without selected file and selected file are shown in Figure 3.34 and Figure 3.35 below, respectively.

```

def Load_excel_data():
    global df_columns, df_columns1, file_path
    """If the file selected is valid this will load the file into the Treeview"""
    file_path = label_file["text"]
    try:
        excel_filename = r"{}".format(file_path)
        if excel_filename[-4:] == ".csv":
            df = pd.read_csv(excel_filename)
        else:
            df = pd.read_excel(excel_filename)

    except ValueError:
        tk.messagebox.showerror("Information", "The file you have chosen is invalid")
        return None
    except FileNotFoundError:
        tk.messagebox.showerror("Information", f"No such file as {file_path}")
        return None

    clear_data()
    tv1["column"] = list(df.columns)
    tv1["show"] = "headings"
    for column in tv1["columns"]:
        tv1.heading(column, text=column) # let the column heading = column name

    df_rows = df.to_numpy().tolist() # turns the dataframe into a list of lists

    df_columns = df["Distance"].to_numpy().tolist()
    df_columns1 = df["Hight"].to_numpy().tolist()

    for row in df_rows:
        tv1.insert("", "end", values=row) # inserts each list into the treeview
    return None

```

Figure 3.32: PageOne read excel file function

```

def clear_data():
    tv1.delete(*tv1.get_children())
    return None

```

Figure 3.33: PageOne clear data in the list function

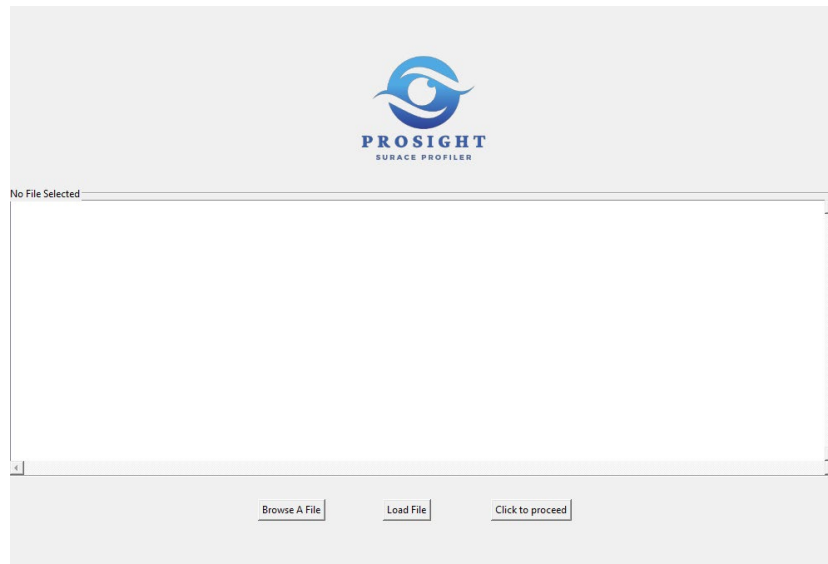


Figure 3.34: PageOne interface (without selected file)

Distance	Hight	Binary	
0.0	0.23439130434782385	79.0	
6.0	0.4517826086956518	73.0	
12.0	1.1039565217391214	86.0	
18.0	1.5387391304347773	86.0	
24.0	1.1039565217391214	80.0	
30.0	0.8865652173912935	79.0	
36.0	0.23439130434782385	87.0	
42.0	-1.0699565217391296	86.0	
48.0	-1.2873478260869575	96.0	
54.0	-0.41778260869566	86.0	
60.0	0.016999999999999907	96.0	
66.0	-1.2873478260869575	79.0	
72.0	0.016999999999999907	87.0	
78.0	0.23439130434782385	78.0	
84.0	0.23439130434782385	78.0	

Figure 3.35: PageOne interface (with the selected file)

### 3.12.7 Design of PageTwo (Class IV)

The class initialise the frame, font, header, define a purpose, and widget button generation. With the buttons widget features, the function of "graph" and "clear" is called. Also, the function calls the canvas drawing and navigation tool function for the graph generation. The function is shown in Figure 3.36 below.

```

class PageTwo(tk.Frame):

    def __init__(self, parent, controller):

        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Optimised Graph", font=LARGE_FONT)
        label.pack(pady=10,padx=10)

        button3 = ttk.Button(self, text="Press to Generate Graph",
                             command=lambda: graph())
        button3.pack()

        button4 = ttk.Button(self, text="Surface Roughness Analysis",
                             command=lambda: controller.show_frame(PageThree))
        button4.pack()

        button5 = ttk.Button(self, text="New Graph",
                             command=lambda: clear())
        button5.pack()

        button1 = ttk.Button(self, text="Back to Home",
                             command=lambda: controller.show_frame(StartPage))
        button1.pack()

        f = Figure(figsize=(10,5), dpi=100)
        a = f.add_subplot(111)

        canvas = FigureCanvasTkAgg(f, self)
        canvas.draw()
        canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

        toolbar = NavigationToolbar2Tk(canvas, self)
        toolbar.update()
        canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

```

Figure 3.36: PageTwo initialisation function

In the inner loop, a graph function is called to extract the data from the lists that are defined as the global variables in the earlier stage. The raw data graph and optimised graph with median filter are generated and displayed in a display. Canvas library is called to plot and implement an interactive graph interface. The function is shown in Figure 3.37 below. The clear function is used to open up a new window for new analysis to be done so that 2 or more analyses can be done simultaneously. The function is shown in Figure 3.38 below. The interface of PageTwo without selected file and selected file are shown in Figure 3.39 and Figure 3.40 below, respectively.

```
def graph():
    x = df_columns
    y = df_columns1
    ymed11 = scipy.signal.medfilt(y, kernel_size=9)

    a.plot(x,y)
    a.plot(x,ymed11, color='red')
    a.axhline(y = 0, color = 'g', linestyle = '--')

    df=pd.DataFrame()
    df['optimized']=ymed11
    print(ymed11)

    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

    toolbar.update()
    canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)
```

Figure 3.37: PageTwo graph plotting function

```
def clear():
    app = SeaofBTCapp()
    app.mainloop()
```

Figure 3.38: PageTwo new analysis function

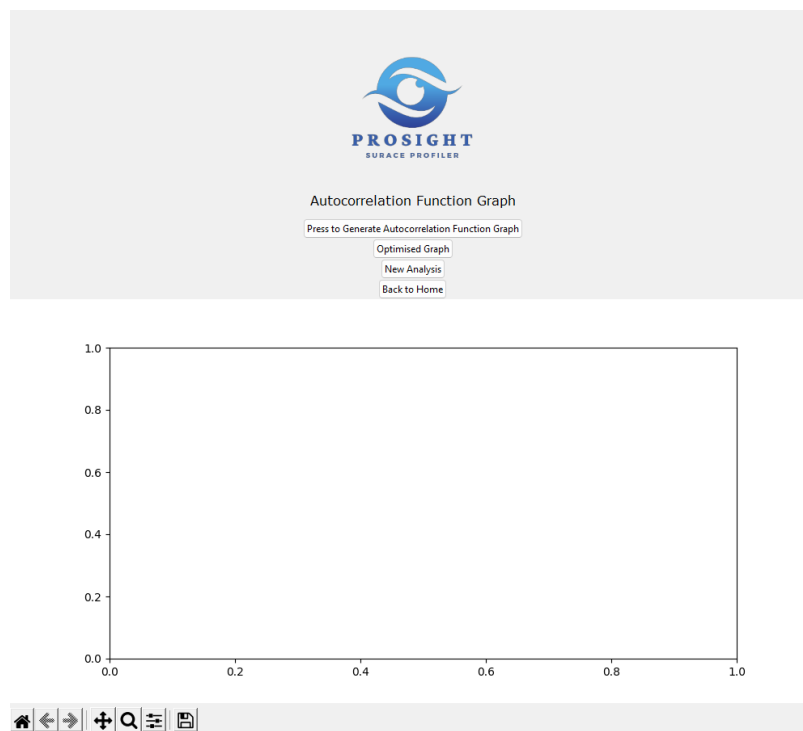


Figure 3.39: PageTwo interface (without selected file)

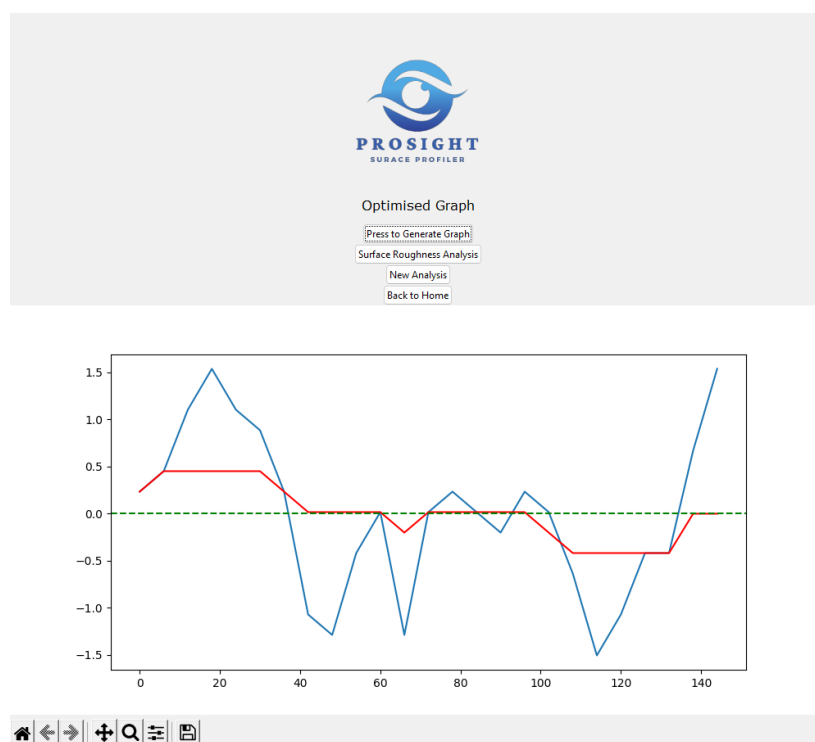


Figure 3.40: PageTwo interface (with the selected file)

### 3.12.8 Design of PageThree (Class V)

The class initialise the frame, font, header, define a purpose, and widget button generation. With the buttons widget features, a function of "graph1" and "clear1" is called. Also, the canvas drawing and navigation tool function for graph generation. The function is shown in Figure 3.41 below.

```
class PageThree(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Autocorrelation Function Graph", font=LARGE_FONT)
        label.pack(pady=10, padx=10)
        labelFrame = ttk.LabelFrame(self, text = "Parameters")

        button3 = ttk.Button(self, text="Press to Generate Autocorrelation Function Graph",
                              command=lambda: graph1())
        button3.pack()

        button1 = ttk.Button(self, text="Optimised Graph",
                              command=lambda: controller.show_frame(PageTwo))
        button1.pack()

        button5 = ttk.Button(self, text="New Analysis",
                              command=lambda: clear1())
        button5.pack()

        button1 = ttk.Button(self, text="Back to Home",
                              command=lambda: controller.show_frame(StartPage))
        button1.pack()

        f = Figure(figsize=(10,5), dpi=100)
        a = f.add_subplot(111)

        canvas = FigureCanvasTkAgg(f, self)
        canvas.draw()
        canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

        toolbar = NavigationToolbar2Tk(canvas, self)
        toolbar.update()
        canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)
```

Figure 3.41: PageThree initialisation function

In the inner loop, a graph function is called to extract the data from the lists that are defined as the global variables in the earlier stage. A series of calculations with Statmodels, NumPy and Shapely libraries are used to do the algorithm for autocorrelation length and RMS surface height calculation. Canvas library is called to plot and implement an interactive graph interface. The function is shown in Figure 3.42 below. The clear function is used to open up a new window for new analysis to be done so that 2 or more analyses can be done simultaneously. The function is shown in Figure 3.43 below. The interface of PageTwo without selected file and selected file are shown in Figure 3.44 and Figure 3.45 below, respectively.



```

def graph1():
    x = df_columns
    y = df_columns1
    ymed11 = scipy.signal.medfilt(y, kernel_size=9)

    lags=range(30)
    acorr = sm.tsa.acf(y, nlags = len(lags)-1)
    print(acorr)
    xc=[0,6,12,18,24,30,36,42,48,54,60,66,72,78,84,90,96,102,108,112,118,124,130,136,142]
    a.plot(xc,acorr)
    y0=[0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,
        0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,
        0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788]

    std=np.std(y,dtype=np.float64)
    print(std)

    first_line = LineString(np.column_stack((xc, np.array(acorr))))
    second_line = LineString(np.column_stack((xc, np.array(y0))))
    intersection = first_line.intersection(second_line)

    if intersection.geom_type == 'MultiPoint':
        a.plot(*LineString(intersection).xy, 'o')
    elif intersection.geom_type == 'Point':
        a.plot(*intersection.xy, 'o')

    x,y=intersection.xy
    print(x,y)

    a.axhline(y=0.36788)
    a.axhline(y = 0, color = 'g', linestyle = '--')

    title = "Autocorrelation Length: "+str(x)+"\n"+"RMS Height: "+str(std)
    a.set_title(title)

    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

    toolbar.update()
    canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

```

Figure 3.42: PageThree Surface Roughness Parameters Calculation and Graph Plotting function

```

def clear():
    app = SeaofBTCapp()
    app.mainloop()

```

Figure 3.43: PageThree new analysis function

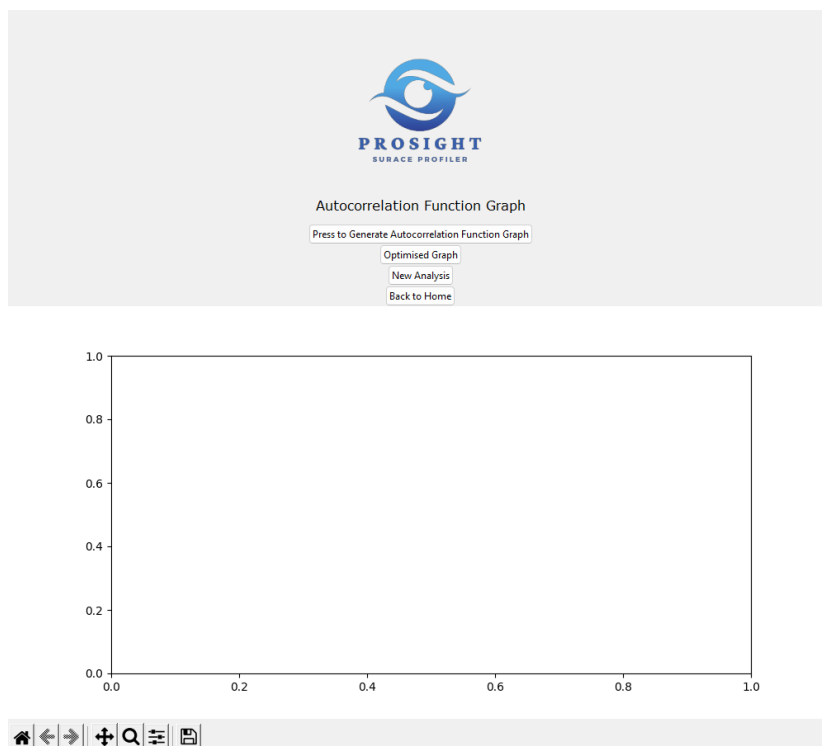


Figure 3.44: PageTwo interface (without selected file)

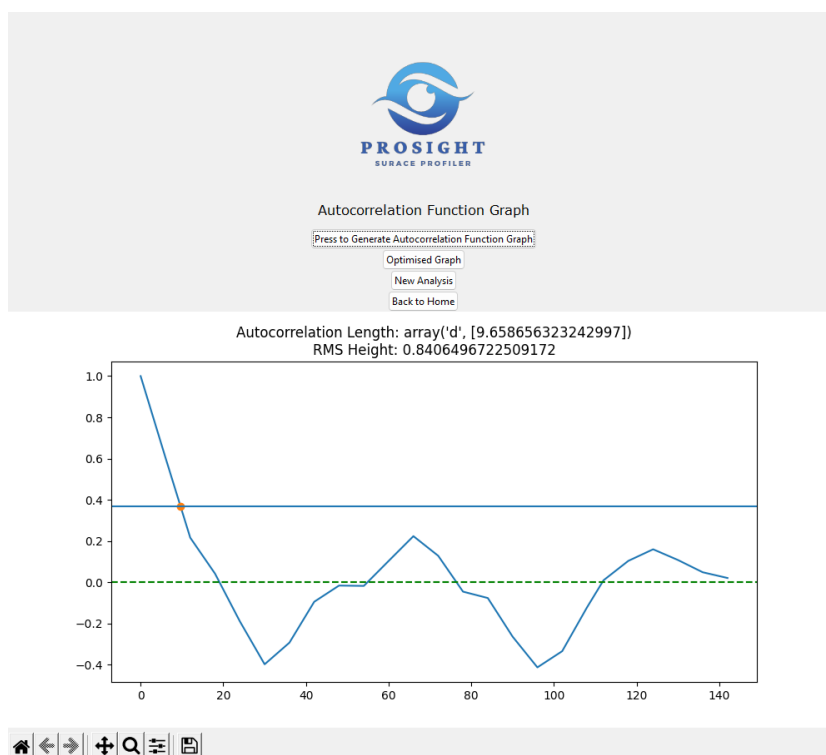


Figure 3.45: PageTwo interface (with selected file)

### 3.12.9 Final Call

Two lines of code are summoned to run the whole system, as shown in Figure 3.46 below.

```
app = SeaofBTCapp()
app.mainloop()
```

Figure 3.46: Coding part for the initialising stage

### 3.12.10 Setup for an executable extension

The code shown in Figure 3.47 below is run to generate an executable extension (.exe) for installation. Users can download the software with the extension on their PC easily.

```
import cx_Freeze
import sys
import matplotlib
import os
import scipy

base = None

if sys.platform == 'win32':
    base = "Win32GUI"

executables = [cx_Freeze.Executable("ProSight Surface Profiler.py", base=base, icon = "clienticon.ico")]

cx_Freeze.setup(
    name = "ProSight Surface Profiler",
    options = {"build_exe": {"packages": ["scipy.optimize", "scipy.integrate", "scipy", "tkinter", "matplotlib"],
                             "include_files": [os.path.join(sys.prefix, "Lib", "site-packages", "Shapely.libs"), "clienticon.ico", "2.png"]}},
    version = "0.01",
    description = "ProSight Data Analysis Application",
    executables = executables
)
```

Figure 3.47: Coding part for the setup of executable extension

### 3.13 Project Management

As shown in Figure 3.48 and Figure 3.49, the Gantt charts for FYP1 and FYP 2 are constructed as follows.

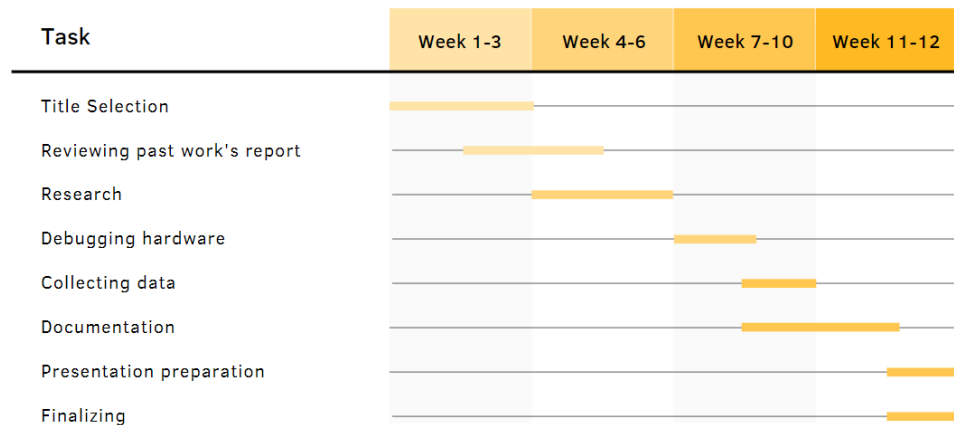


Figure 3.48: Gantt Chart of FYP1

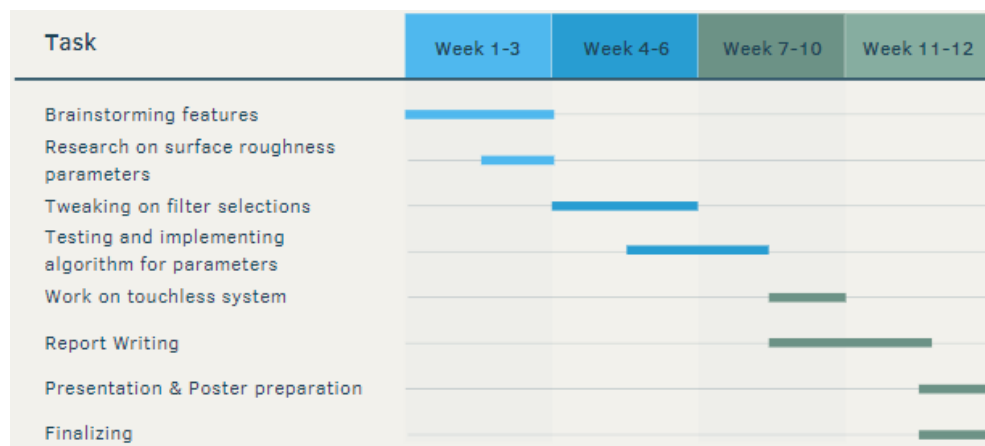


Figure 3.49: Gantt Chart of FYP2

## **CHAPTER 4**

### **RESULTS AND DISCUSSION**

#### **4.1 Introduction**

The reliability and precision of the graph optimisation and surface roughness parameter method are discussed. The software is evaluated using data collected by the researcher regarding electromagnetic wave scattering in dense media at sea and snow surface sites. Furthermore, the graph optimisation for the obtained raw data graph is examined with respect to relevant surfaces. In addition, the hardware debugging is highlighted.

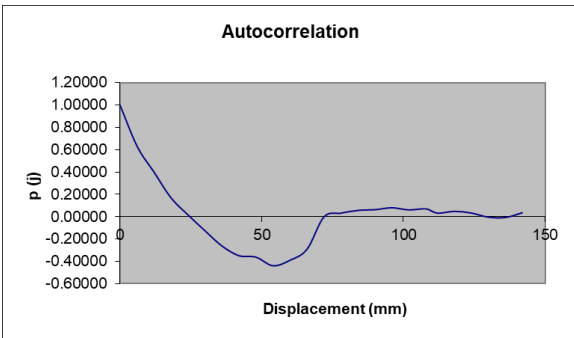
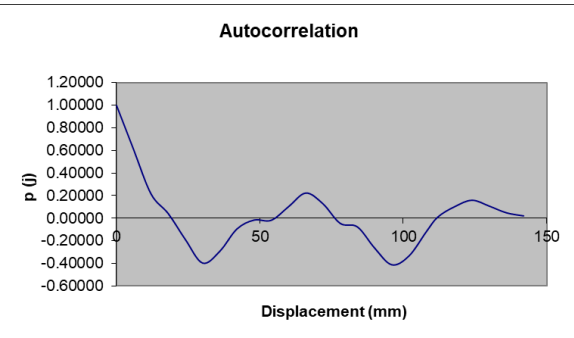
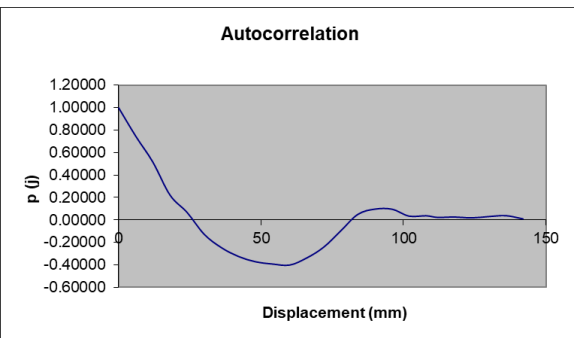
#### **4.2 Surface Roughness Parameter**

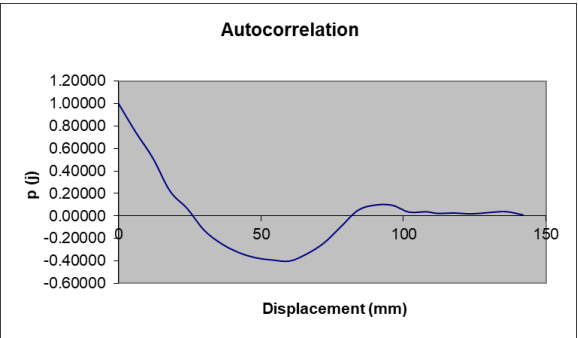
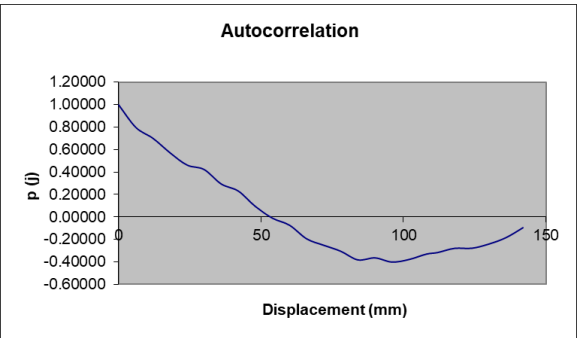
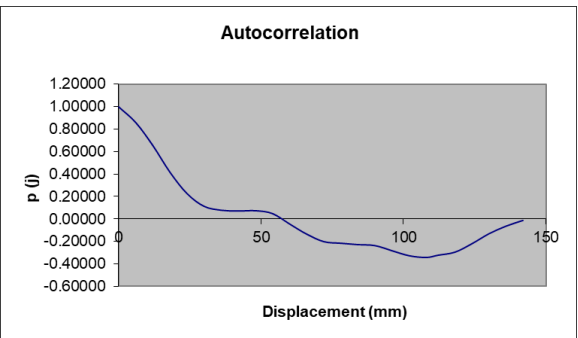
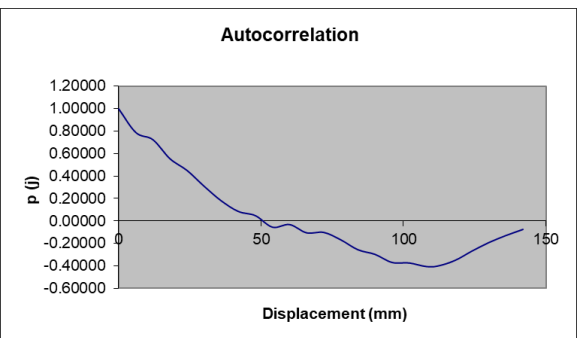
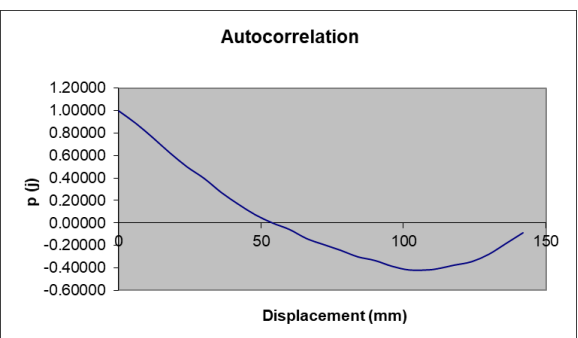
The surface measurement results are from Dr. Lee Yu Jen and other researchers' efforts to understand the surface roughness parameters on Ross Island, Antarctica. The researchers used remote sensing to determine the snow and ice surface's physical features and parameters, such as autocorrelation length and RMS surface height. Therefore, the parameters computed manually by the researchers are compared to those calculated automatically by the software's algorithm.

### 4.2.1 Surface Roughness Parameters (Calculated with Excel)

The calculated parameters and autocorrelation function graph of 8 sites are shown in Table 4.1 and 4.2 below.

**Table 4.1: Autocorrelation Function graph and autocorrelation length of 8 sites (Calculated with Excel)**

Site	Autocorrelation Function Graph	Autocorrelation Length (mm)
1		12.8181
2		9.6609
3		15.036

4	<p><b>Autocorrelation</b></p>  <p>The plot shows the autocorrelation function <math>p(\tau)</math> on the y-axis (ranging from -0.60000 to 1.20000) against Displacement (mm) on the x-axis (ranging from 0 to 150). The curve starts at 1.00000 at <math>\tau=0</math>, decreases to a minimum of approximately -0.45000 at <math>\tau \approx 75</math> mm, and then rises to a local maximum of approximately 0.15000 at <math>\tau \approx 100</math> mm before settling near zero.</p>	12.3864
5	<p><b>Autocorrelation</b></p>  <p>The plot shows the autocorrelation function <math>p(\tau)</math> on the y-axis (ranging from -0.60000 to 1.20000) against Displacement (mm) on the x-axis (ranging from 0 to 150). The curve starts at 1.00000 at <math>\tau=0</math>, decreases steadily to a minimum of approximately -0.40000 at <math>\tau \approx 100</math> mm, and then rises slightly to approximately -0.10000 at <math>\tau = 150</math> mm.</p>	32.6075
6	<p><b>Autocorrelation</b></p>  <p>The plot shows the autocorrelation function <math>p(\tau)</math> on the y-axis (ranging from -0.60000 to 1.20000) against Displacement (mm) on the x-axis (ranging from 0 to 150). The curve starts at 1.00000 at <math>\tau=0</math>, decreases to a minimum of approximately -0.30000 at <math>\tau \approx 110</math> mm, and then rises to approximately 0.00000 at <math>\tau = 150</math> mm.</p>	19.5564
7	<p><b>Autocorrelation</b></p>  <p>The plot shows the autocorrelation function <math>p(\tau)</math> on the y-axis (ranging from -0.60000 to 1.20000) against Displacement (mm) on the x-axis (ranging from 0 to 150). The curve starts at 1.00000 at <math>\tau=0</math>, decreases to a minimum of approximately -0.40000 at <math>\tau \approx 110</math> mm, and then rises to approximately -0.10000 at <math>\tau = 150</math> mm.</p>	27.5201
8	<p><b>Autocorrelation</b></p>  <p>The plot shows the autocorrelation function <math>p(\tau)</math> on the y-axis (ranging from -0.60000 to 1.20000) against Displacement (mm) on the x-axis (ranging from 0 to 150). The curve starts at 1.00000 at <math>\tau=0</math>, decreases to a minimum of approximately -0.45000 at <math>\tau \approx 110</math> mm, and then rises to approximately -0.10000 at <math>\tau = 150</math> mm.</p>	31.4285

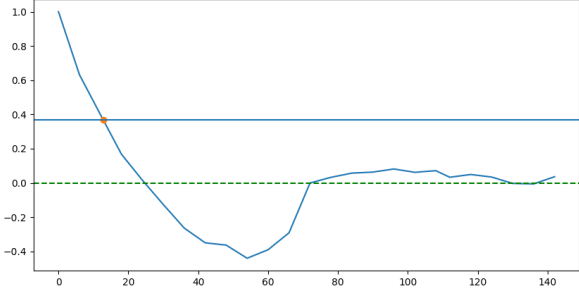
**Table 4.2: RMS height of 9 sites (Calculated with Excel)**

Site	RMS Height (mm)
1	0.290219
2	0.857984
3	13.30681
4	3.625384
5	4.246466
6	2.761772
7	2.038615
8	3.540231

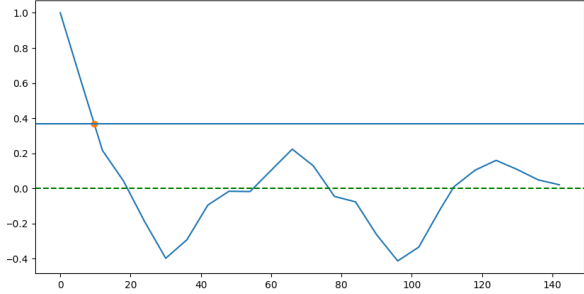
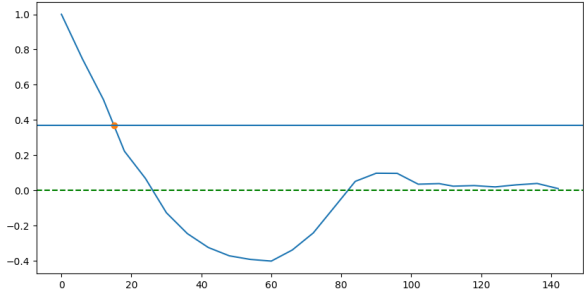
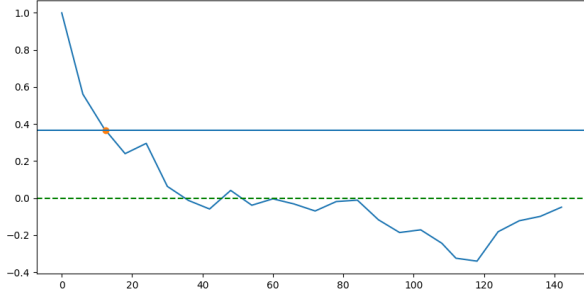
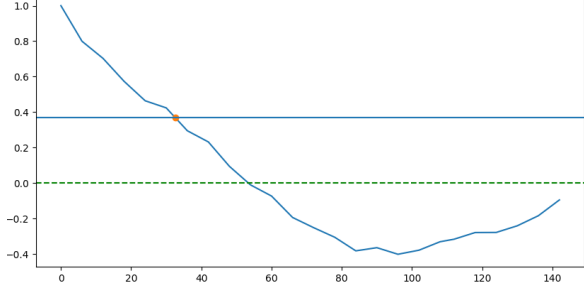
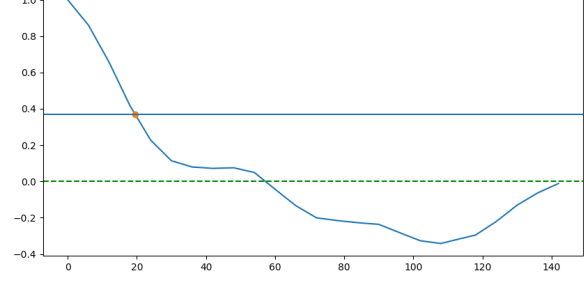
#### 4.2.2 Surface Roughness Parameters (Software Algorithm)

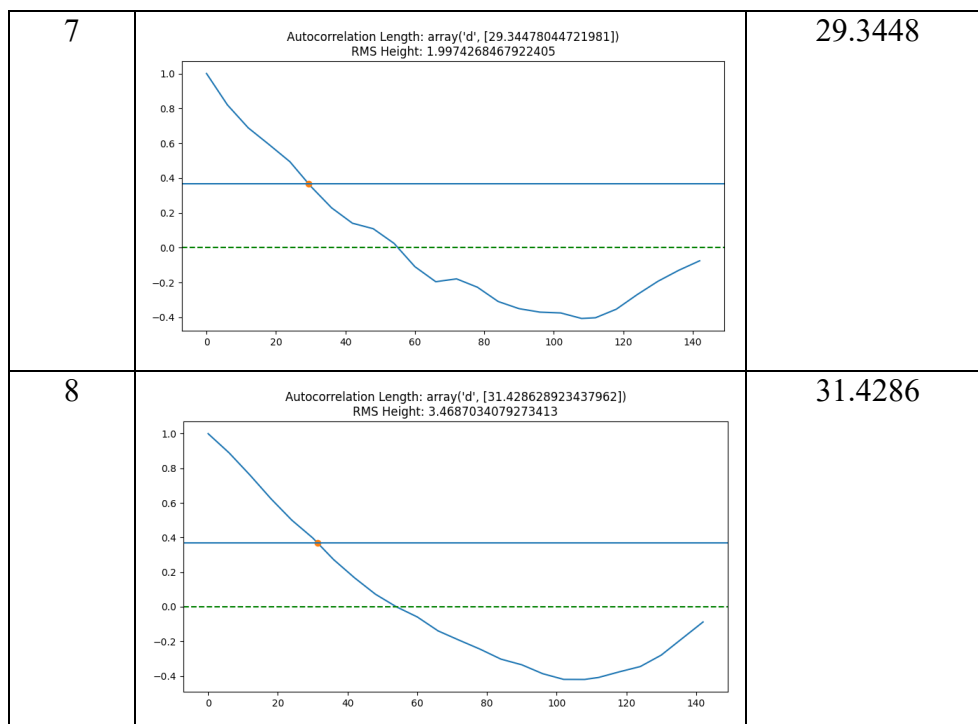
The parameters and autocorrelation function graph of 8 sites from the software algorithm are shown in Table 4.3 & 4.4 below.

**Table 4.3: Autocorrelation Function graph and autocorrelation length of 8 sites (Software Algorithm)**

Site	Autocorrelation Function Graph	Autocorrelation Length (mm)
1	<p>Autocorrelation Length: array('d', [12.816394876398064]) RMS Height: 2.8435567369767676</p> 	12.8164



2	<p>Autocorrelation Length: array('d', [9.658656323242997]) RMS Height: 0.8406496722509172</p>  <p>The plot shows an autocorrelation function starting at 1.0 at lag 0 and decaying towards zero. A solid blue horizontal line is at y=0.4, and a dashed green horizontal line is at y=0.0. An orange dot marks the intersection of the curve and the blue line at approximately lag 10.</p>	9.6587
3	<p>Autocorrelation Length: array('d', [15.035891707178008]) RMS Height: 13.037953956446374</p>  <p>The plot shows an autocorrelation function starting at 1.0 at lag 0 and decaying towards zero. A solid blue horizontal line is at y=0.4, and a dashed green horizontal line is at y=0.0. An orange dot marks the intersection of the curve and the blue line at approximately lag 15.</p>	15.036
4	<p>Autocorrelation Length: array('d', [12.386492418484423]) RMS Height: 3.5521359387224827</p>  <p>The plot shows an autocorrelation function starting at 1.0 at lag 0 and decaying towards zero. A solid blue horizontal line is at y=0.4, and a dashed green horizontal line is at y=0.0. An orange dot marks the intersection of the curve and the blue line at approximately lag 12.</p>	12.3865
5	<p>Autocorrelation Length: array('d', [32.60758883908172]) RMS Height: 4.160669737976493</p>  <p>The plot shows an autocorrelation function starting at 1.0 at lag 0 and decaying towards zero. A solid blue horizontal line is at y=0.4, and a dashed green horizontal line is at y=0.0. An orange dot marks the intersection of the curve and the blue line at approximately lag 33.</p>	32.6076
6	<p>Autocorrelation Length: array('d', [19.556339076762157]) RMS Height: 2.7059729095916976</p>  <p>The plot shows an autocorrelation function starting at 1.0 at lag 0 and decaying towards zero. A solid blue horizontal line is at y=0.4, and a dashed green horizontal line is at y=0.0. An orange dot marks the intersection of the curve and the blue line at approximately lag 19.</p>	19.5563



**Table 4.4: RMS height of 9 sites (Software algorithm)**

Site	RMS Height (mm)
1	2.90219
2	0.857984
3	13.30681
4	3.625384
5	4.246466
6	2.761772
7	2.038615
8	3.540231

### 4.2.3 Discussion of the results

Based on the results obtained from the software algorithm for autocorrelation length and RMS height, the percentage error is calculated as shown in Table 4.5 and 4.6.

**Table 4.5: Percentage error of the results obtained for autocorrelation length**

Site	Accepted Value, $V_A$ (Excel)	Observed Value, $V_O$ (Software Algorithm)	Percentage Error (%) [ $ \frac{V_O - V_A}{V_A}  \times 100\%$ ]
1	12.8181	12.8164	0.0133
2	9.6609	9.6587	0.0228
3	15.036	15.036	0
4	12.3864	12.3865	0
5	32.6075	32.6076	0
6	19.5564	19.5563	0
7	27.5201	29.3448	6.2180
8	31.4285	31.4286	0

$$\text{Average percentage error} = \frac{0.0133+0.0228+6.2180}{8} = 0.7818 \% \quad (4.1)$$

The average percentage error of 8 sites is 0.7818%. The error is accepted and can be neglected. Among 8 sites, 5 have 0% error, and the success rate is 62.5%. Therefore, the software algorithm is accepted and can be used in the study.

**Table 4.6: Percentage error of the results obtained for RMS height**

Site	Accepted Value, $V_A$ (Excel)	Observed Value, $V_O$ (Software Algorithm)	Percentage Error (%) $[ \frac{V_O - V_A}{V_a}  \times 100\%]$
1	2.90219	2.843557	2.062
2	0.857984	0.840650	2.062
3	13.30681	13.037954	2.062
4	3.625384	3.552136	2.062
5	4.246466	4.160670	2.062
6	2.761772	2.705973	2.062
7	2.038615	1.997467	2.062
8	3.540231	3.468703	2.062

$$\text{Average percentage error} = 2.062 \% \quad (4.2)$$

The average percentage error of 8 sites is 2.062 %. The error is accepted as the tolerance level set is 5%. Therefore, the software algorithm is accepted and can be used in the study.

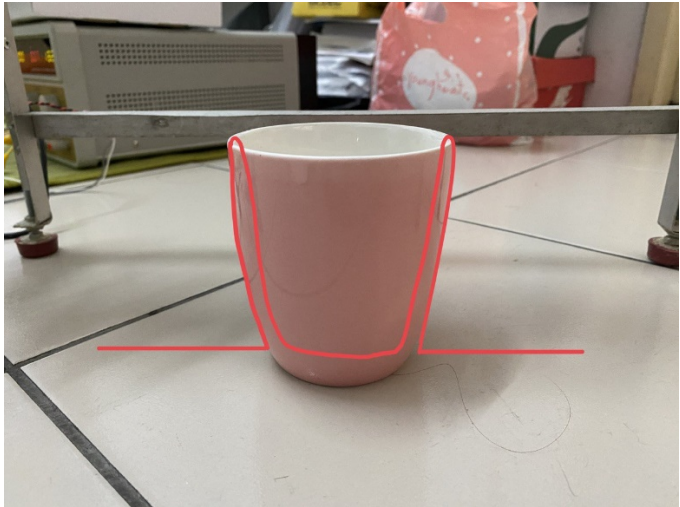
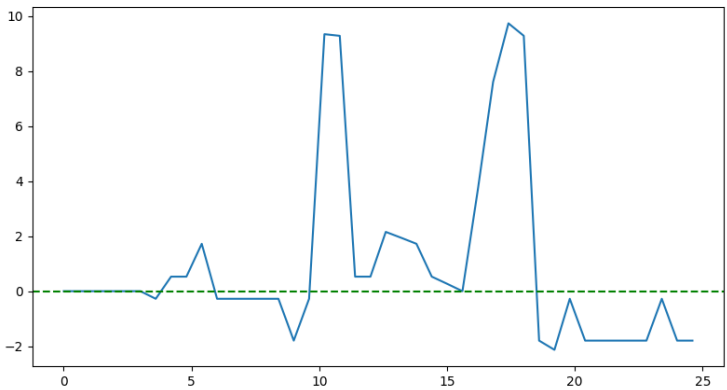
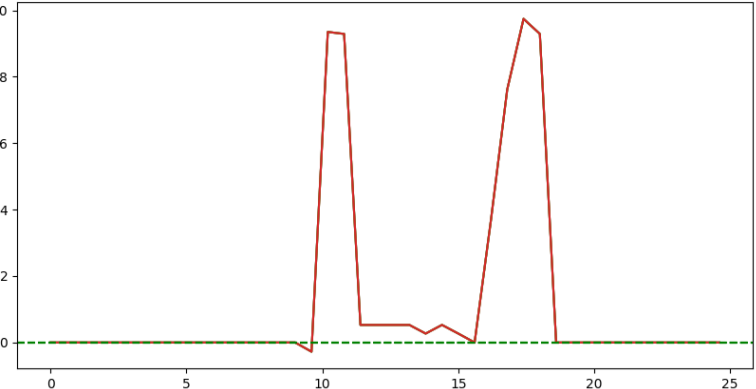
### 4.3 Graph optimisation

A median filter with a kernel size of 9 is applied to optimise the graph. There are a total of 8 case studies to show the results of the optimisation. The object used is shown with the raw data and optimised graph in the subchapters below.

### 4.3.1 Case Study 1

The object shown in Table 4.7 below is drawn with the red line to visualise the expected outcome of the graph. The outcome of the optimised graph is plotted and shown in the table.

**Table 4.7: Raw data graph compared with the optimised graph to the following object in case study 1**

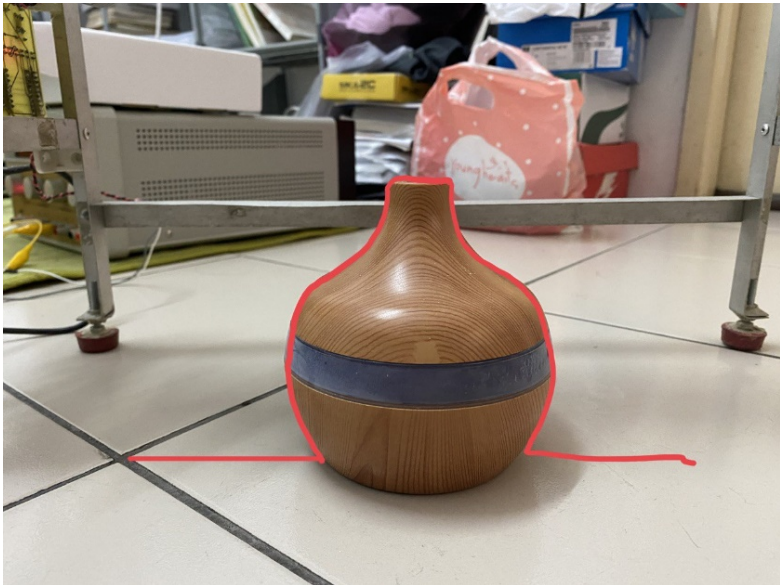
Object	
Raw Data Graph	
Optimised Graph	

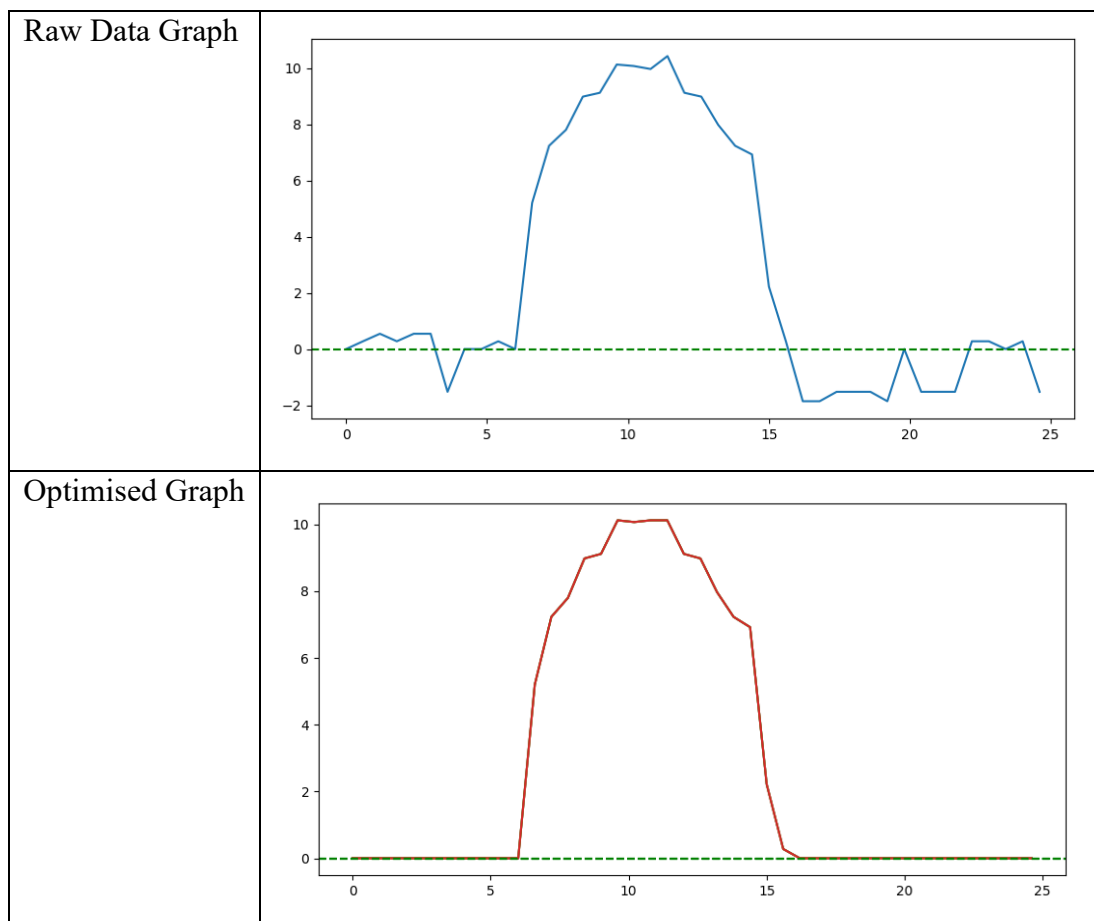
The used object is an empty cup. The raw data graph is shown, but it is full of noise and spikes at the beginning and end. The bottom of the cup is quite uneven, primarily because the interior is black. If the surrounding lighting is inadequate, noise levels increases. As a result of the median filter's optimization, most of the spikes have vanished, and the points have reclaimed their proper locations. The result of case study 1 is therefore accepted.

#### 4.3.2 Case Study 2

The object shown in Table 4.8 below is drawn with a red line to visualise the expected outcome of the graph. The outcome of the optimised graph is plotted and shown in the table.

**Table 4.8: Raw data graph compared with the optimised graph to the following object in case study 2**

Object	
--------	--


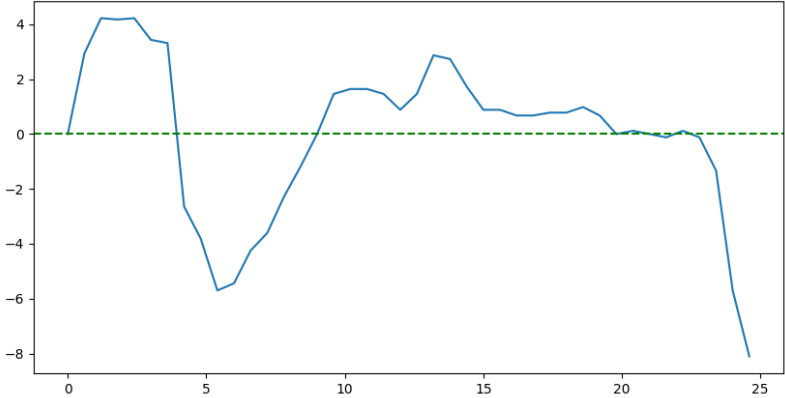
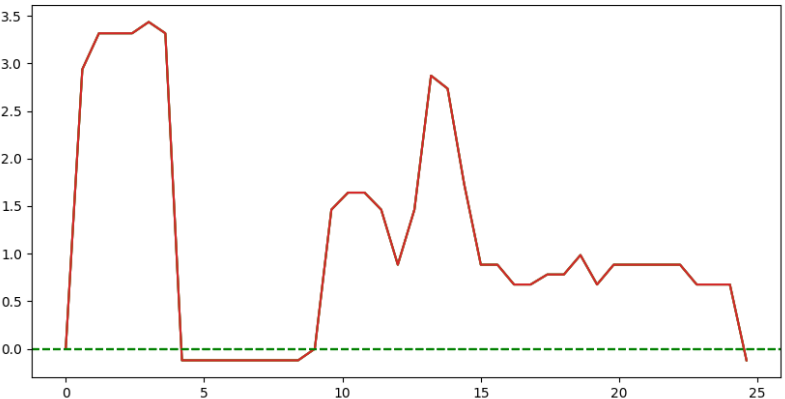


The used object is a humidifier. The raw data graph is shown, but it is full of noise and spikes at the beginning and end. The top of the form is quite irregular. As a result of the median filter's optimisation, most of the spikes have vanished, and the points have reclaimed their proper locations. Additionally, the upper surface of the form became flat. The finding of the second case study is therefore accepted.

### 4.3.3 Case Study 3

The object shown in Table 4.9 below is drawn with a red line to visualise the expected outcome of the graph. The outcome of the optimised graph is plotted and shown in the table.

**Table 4.9: Raw data graph compared with the optimised graph to the following object in case study 3**

Object	
Raw Data Graph	
Optimised Graph	

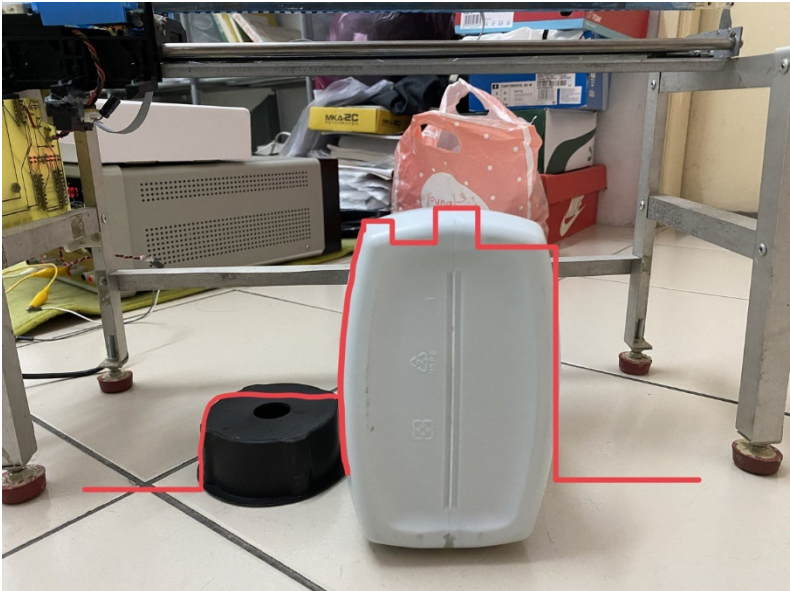


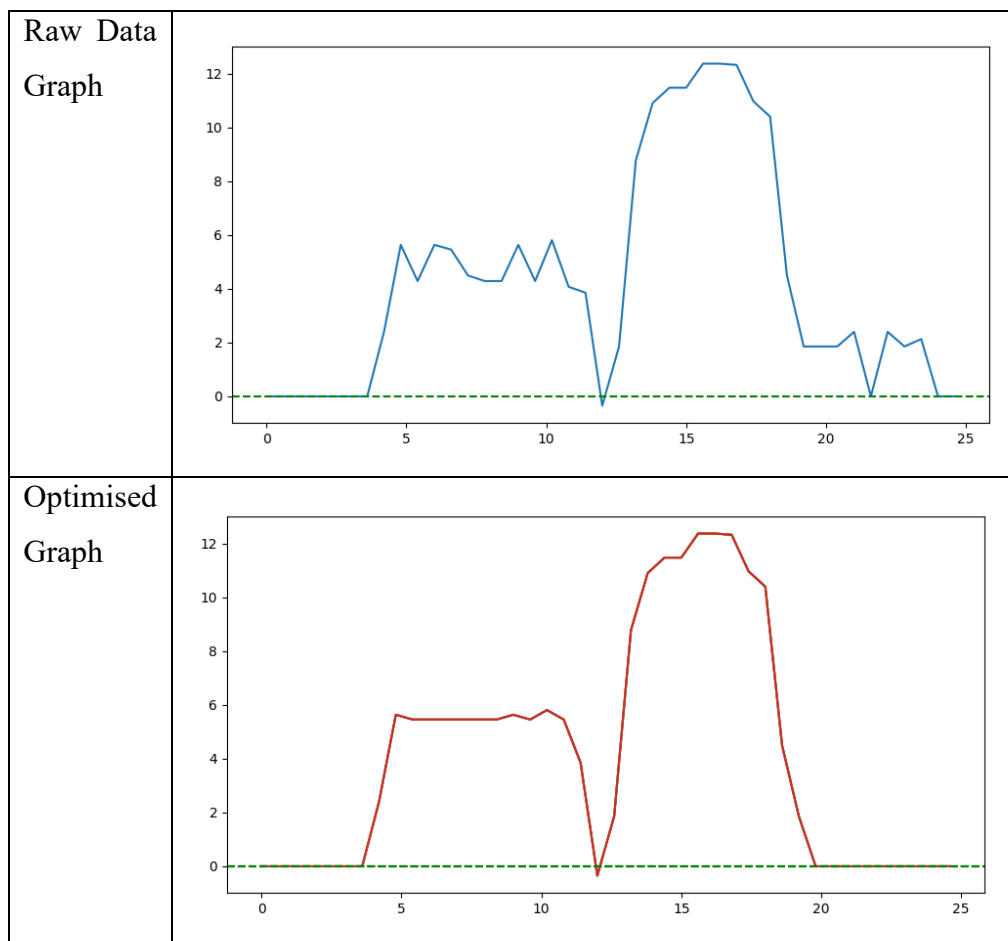
The item used is a pair of hiking shoes. Unexpectedly, the central portion of the plotted graph of raw data is in the negative region. The shoe's insole is on the negative side of the graph due to the inadequate lighting within. If the surrounding lighting is inadequate, noise levels increase. As a result of the median filter's optimisation, most of the spikes have vanished, and the points have reclaimed their proper locations. The graph's negative side is close to the origin. The result of case study 3 is therefore accepted.

#### 4.3.4 Case Study 4

The object shown in Table 4.10 below is drawn with a red line to visualise the expected outcome of the graph. The outcome of the optimised graph is plotted and shown in the table.

**Table 4.10: Raw data graph compared with the optimised graph to the following object in case study 4**

Object	
--------	--


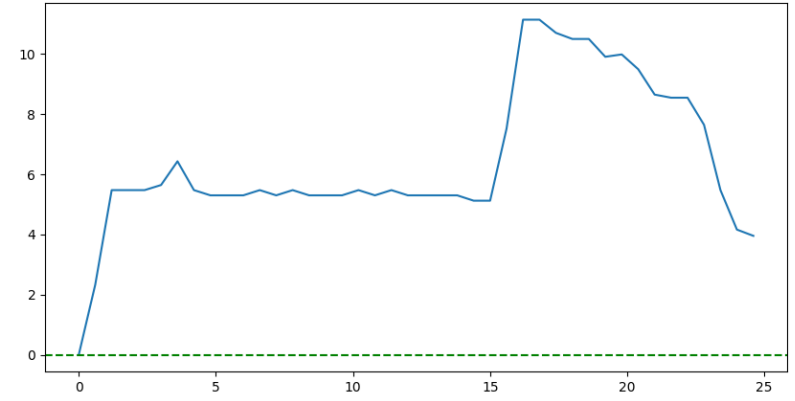
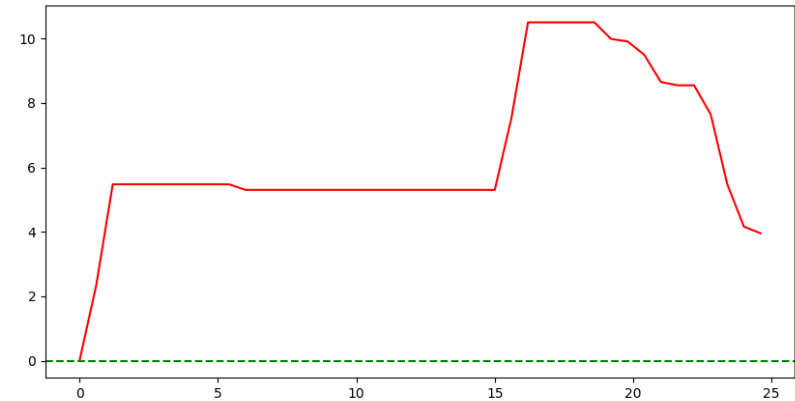


The item used is an inverted cup holder and a bottle of detergent. The raw data graph is shown, but it is full of noise and spikes at the beginning and end. The top of the form is quite irregular. As a result of the median filter's optimisation, most of the spikes have vanished, and the points have reclaimed their proper locations. However, the detergent bottle's cap is inconsistent and not optimised. The finding of case study 4 is therefore rejected.

#### 4.3.5 Case Study 5

The object, as shown in Table 4.11 below, is drawn with a red line to visualise the expected outcome of the graph. The outcome of the optimised graph is plotted and shown in the table.

**Table 4.11: Raw data graph compared with the optimised graph to the following object in case study 5**

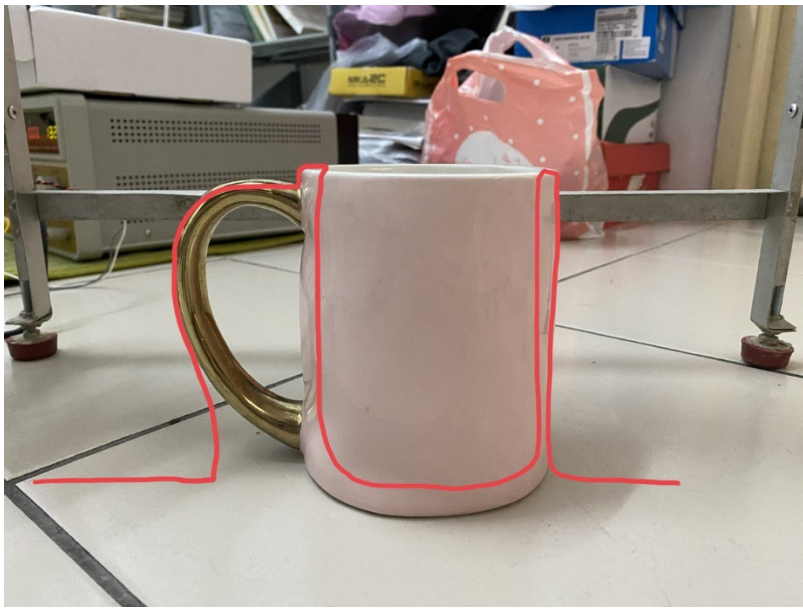
Object	
Raw Data Graph	
Optimised Graph	

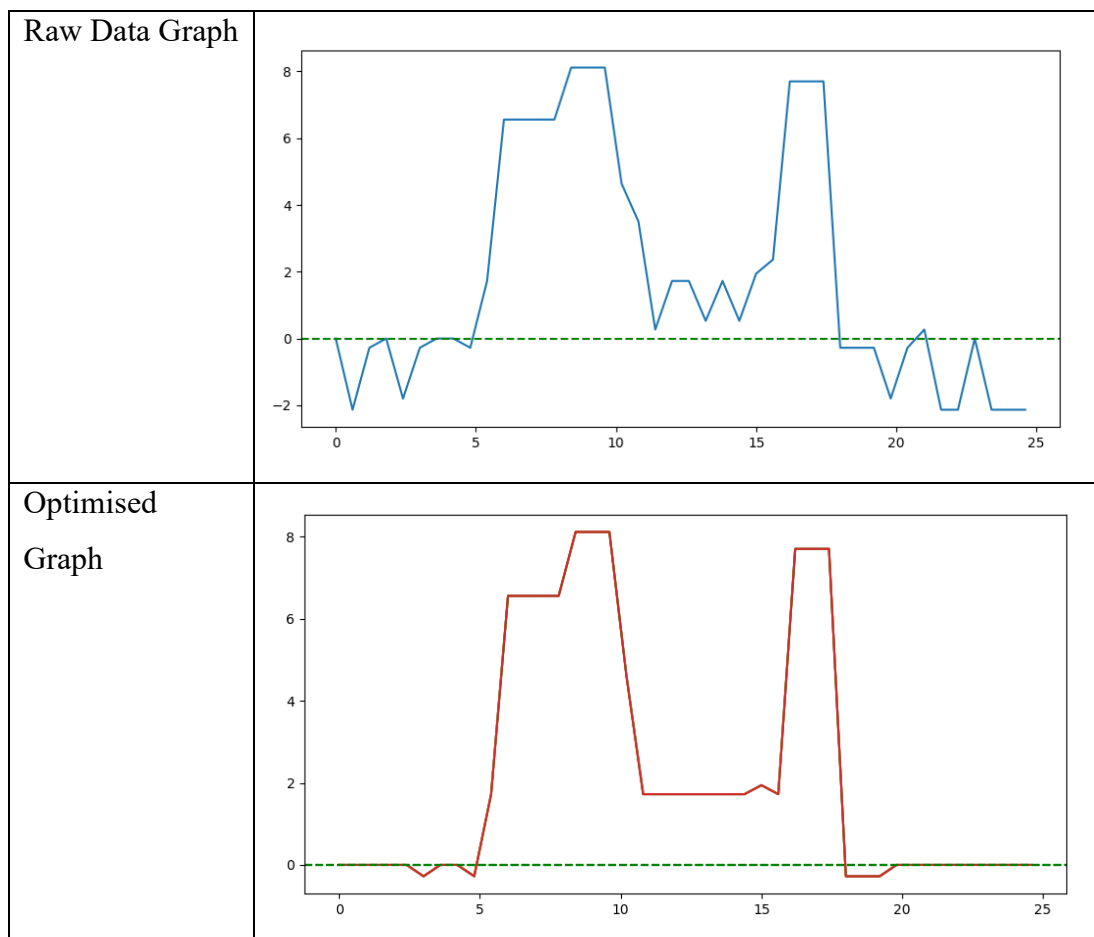
The item used was a slipper. The graph of raw data is depicted with fewer spikes. As a result of the median filter's optimisation, most of the spikes have vanished, and the points have reclaimed their proper locations. The result of case study 5 is therefore accepted.

#### 4.3.6 Case Study 6

The object shown in Table 4.12 below is drawn with a red line to visualise the expected outcome of the graph. The outcome of the optimised graph is plotted and shown in the table.

**Table 4.12: Raw data graph compared with the optimised graph to the following object in case study 6**

Object	
--------	--

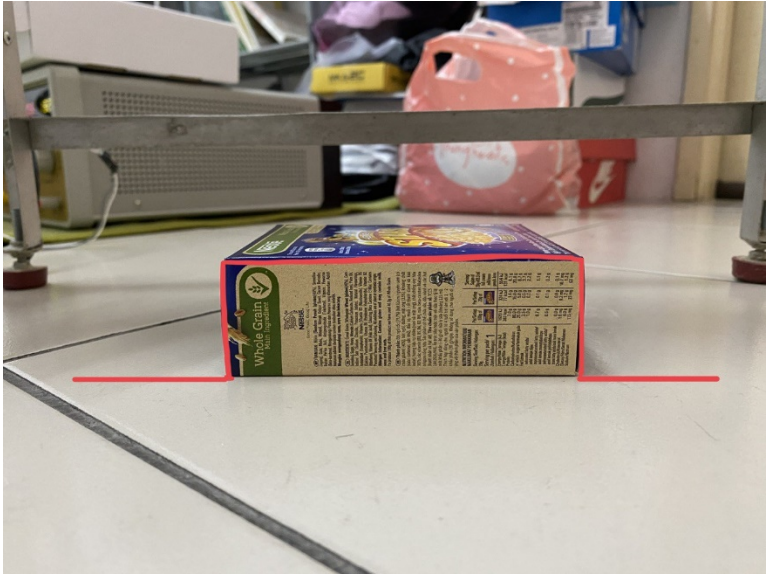
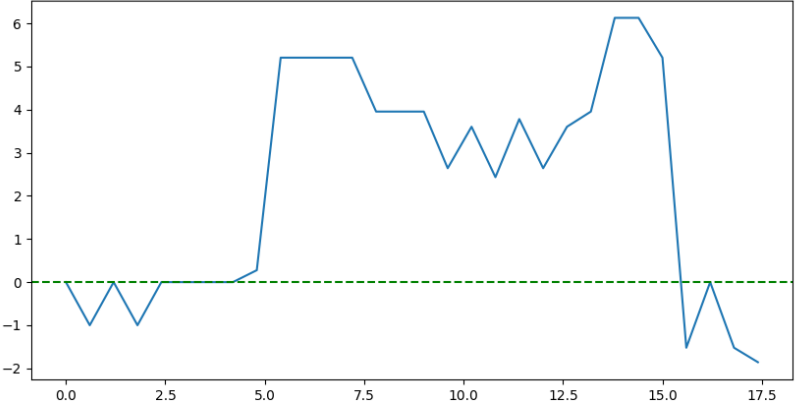
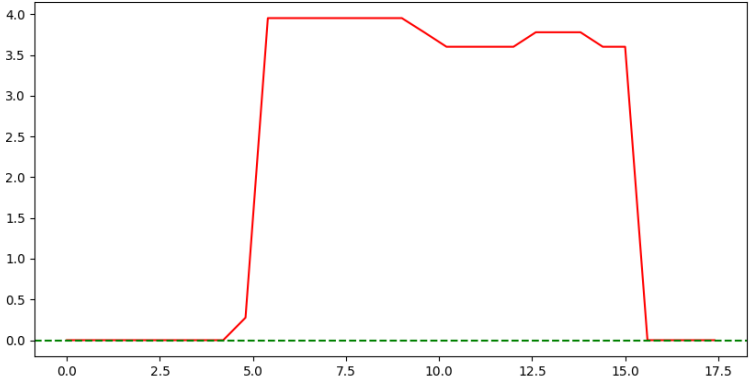


The object used is a cup with a handle. The raw data graph is shown, but it is full of noise and spikes at the beginning and end. The bottom of the cup is quite uneven, primarily because the interior is black. If the surrounding lighting is inadequate, noise levels increase. As a result of the median filter's optimisation, most of the spikes have vanished, and the points have reclaimed their proper locations. The result of case study 6 is therefore accepted.

#### 4.3.7 Case Study 7

The object shown in Table 4.13 below is drawn with a red line to visualise the expected outcome of the graph. The outcome of the optimised graph is plotted and shown in the table.

**Table 4.13: Raw data graph compared with the optimised graph to the following object in case study 7**

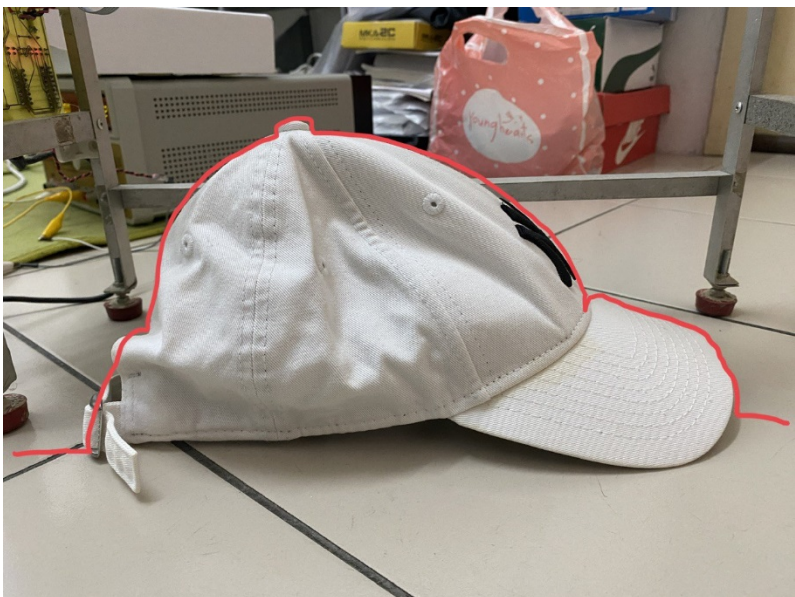
Object	
Raw Data Graph	
Optimised Graph	

The item used is a box. The raw data graph is shown, but it is full of noise and spikes at the beginning and end. The case study is used to determine whether or not a flat surface may be optimised appropriately. As a result of the median filter's optimisation, most of the spikes have vanished, and the points have reclaimed their proper locations. The result of case study 7 is therefore accepted.

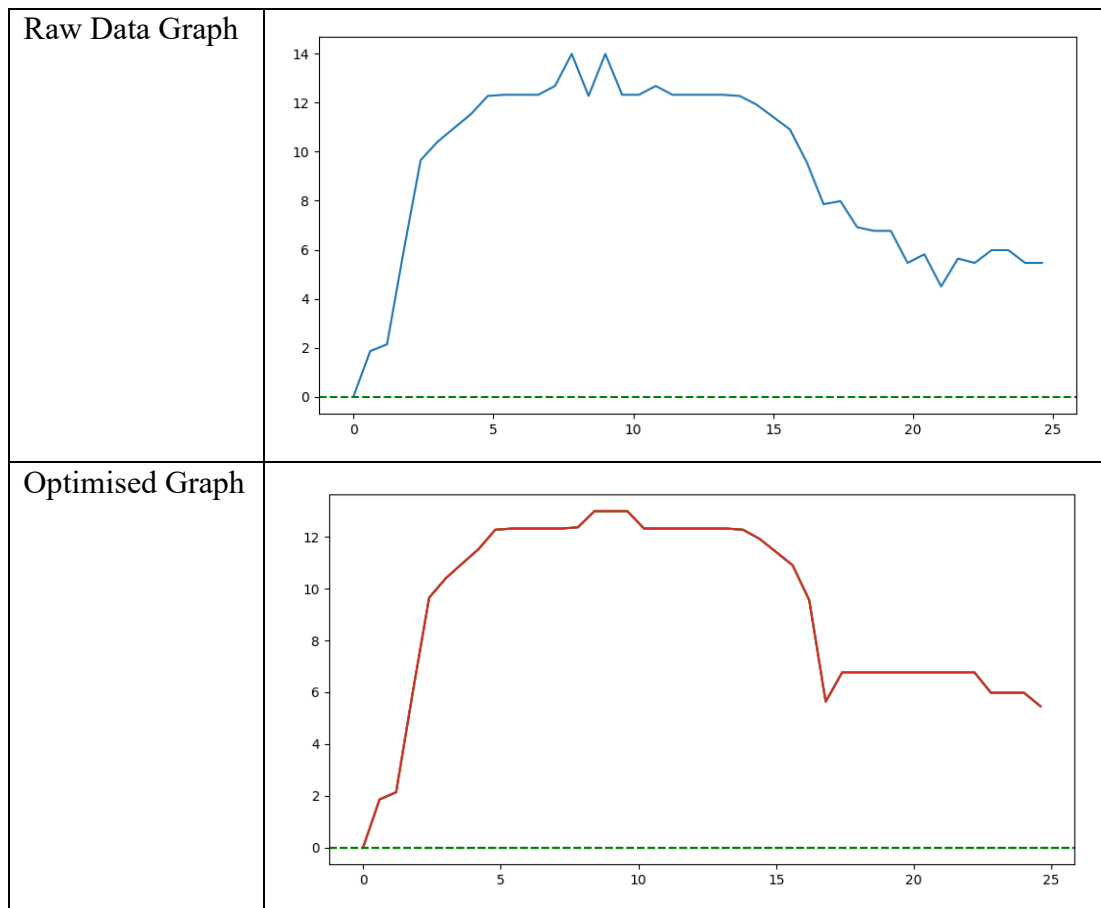
#### 4.3.8 Case Study 8

The object shown in Table 4.14 below is drawn with a red line to visualise the expected outcome of the graph. The outcome of the optimised graph is plotted and shown in the table.

**Table 4.14: Raw data graph compared with the optimised graph to the following object in case study 8**

Object	
--------	--





The object used is a cap. The graph of raw data is drawn. However, it is full of noise and spikes towards the end. As a result of the median filter's optimisation, most of the spikes have vanished, and the points have reclaimed their proper locations. The cap's little tip is recovered, and the outcome is positive overall. The result of case study 8 is therefore accepted.

#### 4.3.9 Summary of the case study of graph optimisation

After the graph optimisation analysis, seven of the eight case study recommendations are accepted. The majority of spikes are gone. However, the results are unreliable for dark surface objects and gloomy environments. The issue can be resolved by modifying the hardware. The graph optimisation returned the raw data graph to its original form.



#### 4.4 Extra

As the prototype ages, the hardware requires an upgrade. For instance, the Raspberry Pi 2 is upgraded to the Raspberry Pi 3 to provide Bluetooth and internet wireless connectivity. The Raspberry Pi 3 is mounted to the PCB board, as depicted in Figure 4.1.

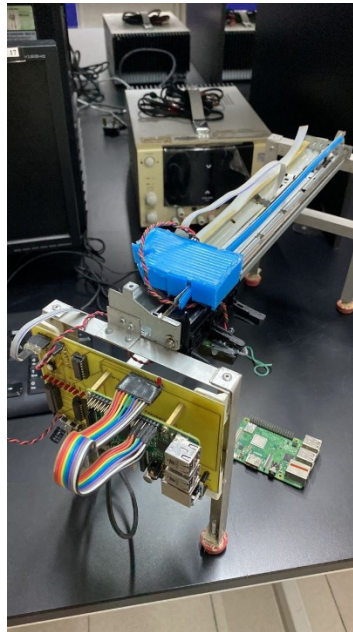


Figure 4.1: Replace the old Raspberry Pi 2 with Raspberry Pi 3

Also, the step-down converter is burnt and malfunctioned. The component has to replace. The replacement is done by configuring the step-down converter, as shown in Figure 4.2 below, to ensure the source's output is 5V.

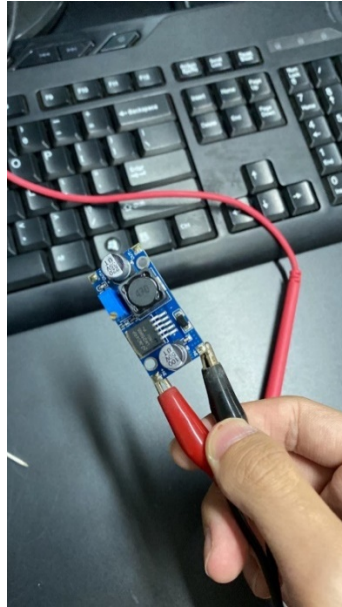


Figure 4.2: Replace a new step-down converter (12V to 5V)

The encoder strip is changed as the old one is decolourised, and the paper is not in good condition for the servo system to function correctly. A new encoder strip is printed and installed on the prototype, as shown in Figure 4.3 below.



Figure 4.3: Replace a new encoder strip

## CHAPTER 5

### CONCLUSIONS AND RECOMMENDATIONS

#### 5.1 Conclusion

All objectives are accomplished. The ProSight Surface Profiler software can connect to all Blynk Cloud devices and extract data in real-time from Google Drive. The clear GUI interface increases the data analysis productivity of the user. In addition, the software can automate the calculation of autocorrelation length and RMS height. The calculation's output can be completed within a second. The output graph is optimised so that spikes and noises are eliminated. With the executable file prepared, the software can be downloaded publicly.

The autocorrelation length and RMS height error percentages are 0.7818 % and 2.062 %, respectively. Given that the tolerance limit is 5 %, the percentage error can be neglected. For graph optimisation, if the surrounding condition has good lighting, the optimisation result is positive. The outcome is still satisfactory, despite not all the spikes being eliminated. The ProSight Surface Profiler programme is fully operational with all its advanced features.

## 5.2 Limitations

The software has some drawbacks. First, the distance IR sensor has a limited sensitivity when operating in low light. The output of the measured data is inconsistent, and the data cannot be optimised accurately. Furthermore, the motor movement is sometimes unstable, which affects the measurement's resolution. Next, the contactless system cannot be implemented without a WIFI connection because the Blynk app and Google Drive require an internet connection.

## 5.3 Recommendations for Future Improvement

- Software

With the existing Python library, image processing technology can be added to the system for future implementation. The user can import an image and analyse the surface's roughness in real-time without requiring an instrument. In addition, since the RMS height has a predetermined error percentage of 2.062 %, the error can be eliminated by adding a formula to the code. Additionally, files can be imported by mounting Google Drive within the software, allowing the user to select the file directly. Finally, the application can be implemented on the website, allowing users worldwide to access it online.

- Hardware

The factors considered as the object's surface colour and the surrounding lightning condition can affect the sensor accuracy. Therefore, the factors should be a topic for further research because the measure of the factors is inaccurate and without statistical support. A countermeasure should be implemented to increase the sensor's precision after the research. Also, the servo system can be improved by changing a newly developed servo system with stable readability so that the motor moves in a smaller step and the resolution of the surface measurement increases.

## REFERENCES

- Arora, S., 2021, *Smooth Data in Python* [Online]. Available at: <https://www.delftstack.com/howto/python/smooth-data-in-python/> [Accessed: 5 September 2022].
- Austerlitz, H., 2003. *Computer Programming Languages. Data Acquisition Techniques Using PCs*, [e-journal] pp.326–360. Available at: <http://dx.doi.org/10.1016/B978-012068377-2/50013-9>.
- BairesDev, 2022, *The Pros and Cons of Java Development - BairesDev* [Online]. Available at: <https://www.bairesdev.com/technologies/java-dev-pros-cons/> [Accessed: 15 April 2022].
- Bhalerao, S., 2021, *What is data smoothing and what are the effects? - MENTOR ME CAREERS* [Online]. Available at: <https://mentormecareers.com/data-smoothing/> [Accessed: 5 September 2022].
- Buttice, C., 2021, *What is C++ Programming Language? - Definition from Techopedia* [Online]. Available at: <https://www.techopedia.com/definition/26184/c-plus-plus-programming-language> [Accessed: 16 August 2022].
- Campbell, J.B. and Wynne, R.H., 2011. *Introduction to remote sensing* 5th ed., Guilford Press, New York.
- Cass, S., 2018, *The 2018 Top Programming Languages* [Online]. Available at: <https://spectrum.ieee.org/the-2018-top-programming-languages> [Accessed: 15 April 2022].
- Chai, W., Castagna, R. and Lelii, S., 2021, *What is Cloud Storage? Cloud Storage Definition | Search Storage* [Online]. Available at: <https://www.techtarget.com/searchstorage/definition/cloud-storage> [Accessed: 16 August 2022].
- Cognex, 2022, *In-Sight Laser Profiler - Software & Tools| Cognex* [Online]. Available at: <https://www.cognex.com/products/machine-vision/3d-machine-vision-systems/in-sight-laser-profiler/software> [Accessed: 15 April 2022].
- Constantinos, E., 2018, *Signal-Smoothing Algorithms* [Online]. Available at: [http://195.134.76.37/applets/AppletSmooth/App1\\_Smooth2.html](http://195.134.76.37/applets/AppletSmooth/App1_Smooth2.html) [Accessed: 15 September 2022].

- Day, M. and Chenoweth, S., 2013. 6.14 *Surface Roughness of Karst Landscapes. Treatise on Geomorphology*, [e-journal] pp.157–163. Available at: <http://dx.doi.org/10.1016/B978-0-12-374739-6.00108-1>.
- Egal Net, 2022, *RaspController Android* [Online]. Available at: [https://www.gallinaettore.com/android\\_apps/raspcontroller/](https://www.gallinaettore.com/android_apps/raspcontroller/) [Accessed: 5 September 2022].
- Fincash, 2022, *Data Smoothing | What is Data Smoothing? - Fincash* [Online]. Available at: <https://www.fincash.com/l/basics/data-smoothing> [Accessed: 5 September 2022].
- GeeksforGeeks, 2020, *What is Kivy?* [Online]. Available at: <https://www.geeksforgeeks.org/what-is-kivy/> [Accessed: 15 April 2022].
- Gharechelou, S., Tateishi, R. and A. Johnson, B., 2018. *A Simple Method for the Parameterization of Surface Roughness from Microwave Remote Sensing. Remote Sensing*, [e-journal] 10(11), p.1711. Available at: <http://dx.doi.org/10.3390/rs10111711>.
- Google, 2019, *Google Drive: Free Cloud Storage for Personal Use* [Online]. Available at: <https://www.google.com/drive/> [Accessed: 16 August 2022].
- Guoping Qiu, 1994. *Functional optimization properties of median filtering. IEEE Signal Processing Letters*, [e-journal] 1(4), pp.64–65. Available at: <http://dx.doi.org/10.1109/97.295334>.
- Harris, C. et al., 2020, *numpy.std — NumPy v1.21 Manual* [Online]. Available at: <https://numpy.org/doc/stable/reference/generated/numpy.std.html> [Accessed: 21 August 2022].
- Hcltech, 2022, *What are IoT platforms? | HCL Technologies* [Online]. Available at: <https://www.hcltech.com/technology-qa/what-are-iot-platforms#main-content> [Accessed: 5 September 2022].
- Kang, E.H., 2016. *Automated Electronic Sensor System*. Final Year Project Universiti Tunku Abdul Rahman, pp.24-53.
- Koay, J.Y., Lee, Y.J., Ewe, HT and Chuah, H.T., 2017. *Electromagnetic Wave scattering In Dense Media: Applications In The Remote Sensing Of Sea Ice And Vegetation*. *Electromagnetic Scattering*, [e-journal] pp.303–339. Available at: [http://dx.doi.org/10.1142/9789813209954\\_0008](http://dx.doi.org/10.1142/9789813209954_0008).
- Kravchenko, I., 2022, *Pros and Cons of Java: Main Advantages and Disadvantages* [Online]. Available at: <https://diceus.com/why-is-java-so-popular/> [Accessed: 1 September 2022].
- Mckenzie, C., 2016, *Five tips for choosing a UI development framework* [Online]. Available at: <https://www.theserverside.com/tip/Five-tips-for-choosing-a-UI-development-framework> [Accessed: 15 April 2022].

- Media's, E., Syufrijal and Rif'an, M., 2019. Internet of Things (IoT): BLYNK Framework for Smart Home. *KnE Social Sciences*, 3(12), p.579.
- Micro-Epsilon Messtechnik, 2022, *Configuration Tools - easy setup for scanCONTROL laser scanners* [Online]. Available at: [https://www.micro-epsilon.com/2D\\_3D/laser-scanner/Software/scanCONTROL-Configuration-Tools/](https://www.micro-epsilon.com/2D_3D/laser-scanner/Software/scanCONTROL-Configuration-Tools/) [Accessed: 15 April 2022].
- Mixon, E. and Wigmore, I., 2016, *What is Google Drive? - Definition from WhatIs.com* [Online]. Available at: <https://www.techtarget.com/searchmobilecomputing/definition/Google-Drive> [Accessed: 4 September 2022].
- Pastell, M., 2016, *Measurements and Data Analysis for Agricultural Engineers using Python* [Online]. Available at: [https://pyageng.mpastell.com/book/dsp.html#figma\\_example](https://pyageng.mpastell.com/book/dsp.html#figma_example) [Accessed: 5 September 2022].
- Schafer, R., 2011. What Is a Savitzky-Golay Filter? [Lecture Notes]. *IEEE Signal Processing Magazine*, 28(4), pp.111–117. Available at: <http://dx.doi.org/10.1109/MSP.2011.941097>.
- Shepherd, A., 2020, *What is the Best Programming Language for GUI?* [Online]. Available at: <https://mockitt.wondershare.com/ui-ux-design/gui-python.html> [Accessed: 15 April 2022].
- Smith, M.W., 2014. *Roughness in the Earth Sciences*. *Earth-Science Reviews*, [e-journal] 136, pp.202–225. Available at: <http://dx.doi.org/10.1016/j.earscirev.2014.05.016>.
- SoftwareTestingHelp, 2022, *Python Vs C++ (Top 16 Differences Between C++ And Python)* [Online]. Available at: <https://www.softwaretestinghelp.com/python-vs-cpp/> [Accessed: 2 May 2022].
- Ulaby, F.T., Moore, R.K. and Fung, A.K., 1981. *Microwave remote sensing : active and passive*, Addison-Wesley Publishing Company, Advanced Book Program/World Science Division, Reading, Massachusetts Etc.
- UpGrad, 2021, *Top 10 Reasons Why Python is So Popular With Developers in 2022* [Online]. Available at: [https://www.upgrad.com/blog/reasons-why-python-popular-with-developers/#Why\\_is\\_Python\\_becoming\\_so\\_popular\\_in\\_this\\_decade](https://www.upgrad.com/blog/reasons-why-python-popular-with-developers/#Why_is_Python_becoming_so_popular_in_this_decade) [Accessed: 15 April 2022].
- Upswift, 2022, *What Is An IoT Platform? - JFrog Connect (formerly Upswift)* [Online]. Available at: <https://jfrog.com/connect/post/what-is-an-iot-platform/> [Accessed: 5 September 2022].
- Virtanen, P. et al., 2020, *scipy.signal.medfilt — SciPy v1.7.1 Manual* [Online]. Available at: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.medfilt.html> [Accessed: 16 August 2022].

Vitrek, 2021, *Contact vs. Non-Contact Measurement: Examples of Each* [Online]. Available at: <https://mtiinstruments.com/contact-vs-non-contact-measurement/> [Accessed: 15 April 2022].

Wigmore, I., 2011, *What is Dropbox? - Definition from WhatIs.com* [Online]. Available at: <https://www.techtarget.com/searchmobilecomputing/definition/Dropbox> [Accessed: 16 August 2022].



## APPENDICES

### APPENDIX A: Coding of Main Program

```
import matplotlib
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
matplotlib.use("TkAgg")

import tkinter as tk
from tkinter import ttk, filedialog, messagebox, PhotoImage

import shapely
from shapely.geometry import LineString

import pandas as pd

import scipy.signal
from scipy.signal import medfilt

import numpy as np
from numpy import ones, vstack
from numpy.linalg import lstsq

import statsmodels.api as sm

from PIL import ImageTk, Image

import sys
import contextlib
import warnings
from packaging import version

sys.setrecursionlimit(2000)

SHAPELY_GE_20 = version.parse(shapely.__version__) >= version.parse("2.0a1")

try:
    from shapely.errors import ShapelyDeprecationWarning as shapely_warning
except ImportError:
    shapely_warning = None

if shapely_warning is not None and not SHAPELY_GE_20:
    @contextlib.contextmanager
    def ignore_shapely2_warnings():
        with warnings.catch_warnings():
            warnings.filterwarnings("ignore", category=shapely_warning)
        yield
else:
    @contextlib.contextmanager
    def ignore_shapely2_warnings():
        yield
```

```

df_columns = []
df_columns1 = []
file_path = []

LARGE_FONT= ("Verdana", 12)
NORM_FONT = ("Helvetica", 10)
SMALL_FONT = ("Helvetica", 8)

f = Figure(figsize=(5,5), dpi=100)
a = f.add_subplot(111)

def quit():
    quit()

class SeaofBTCapp(tk.Tk):

    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)

        tk.Tk.iconbitmap(self, default="clienticon.ico")
        tk.Tk.wm_title(self, "Automated Surface Profiler")

        prop = ImageTk.PhotoImage(Image.open("2.png"))
        l4 = ttk.Label(borderwidth=2, image=prop)
        l4.image = prop
        l4.pack(side="top")

        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand = True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        for F in (StartPage, PageOne, PageTwo, PageThree):

            frame = F(container, self)

            self.frames[F] = frame

            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame(StartPage)

    def show_frame(self, cont):

        frame = self.frames[cont]
        frame.tkraise()

```

```

class StartPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Welcome to ProSight Surface Profiler", font=LARGE_FONT)
        label.pack(pady=10, padx=10)

        button1 = ttk.Button(self, text="Choose a file",
                              command=lambda: controller.show_frame(PageOne))
        button1.pack()

        button2 = ttk.Button(self, text="Optimised Graph",
                              command=lambda: controller.show_frame(PageTwo))
        button2.pack()

        button3 = ttk.Button(self, text="Surface Roughness Analysis",
                              command=lambda: controller.show_frame(PageThree))
        button3.pack()

        button4 = ttk.Button(self, text="Quit",
                              command=quit)
        button4.pack()

class PageThree(tk.Frame):

    def __init__(self, parent, controller):

        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Autocorrelation Function Graph", font=LARGE_FONT)
        label.pack(pady=10, padx=10)
        labelFrame = ttk.LabelFrame(self, text = "Parameters")

        button3 = ttk.Button(self, text="Press to Generate Autocorrelation Function Graph",
                              command=lambda: graph1())
        button3.pack()

        button1 = ttk.Button(self, text="Optimised Graph",
                              command=lambda: controller.show_frame(PageTwo))
        button1.pack()

        button5 = ttk.Button(self, text="New Analysis",
                              command=lambda: clear1())
        button5.pack()

        button1 = ttk.Button(self, text="Back to Home",
                              command=lambda: controller.show_frame(StartPage))
        button1.pack()

```

```

f = Figure(figsize=(10,5), dpi=100)
a = f.add_subplot(111)

canvas = FigureCanvasTkAgg(f, self)
canvas.draw()
canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

toolbar = NavigationToolbar2Tk(canvas, self)
toolbar.update()
canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

def graph1():

    x = df_columns
    y = df_columns1

    lags=range(30)
    acorr = sm.tsa.acf(y, nlags = len(lags)-1)
    print(acorr)
    xc=[0,6,12,18,24,30,36,42,48,54,60,66,72,78,84,90,96,102,108,112,118,124,130,136,142]
    a.plot(xc,acorr)
    y0=[0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,
        0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,
        0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788,0.36788]

    std=np.std(y,dtype=np.float64)
    print(std)

    first_line = LineString(np.column_stack((xc, np.array(acorr))))
    second_line = LineString(np.column_stack((xc, np.array(y0))))
    intersection = first_line.intersection(second_line)

    if intersection.geom_type == 'MultiPoint':
        a.plot(*LineString(intersection).xy, 'o')
    elif intersection.geom_type == 'Point':
        a.plot(*intersection.xy, 'o')

    x,y=intersection.xy
    print(x,y)

    a.axhline(y=0.36788)
    a.axhline(y = 0, color = 'g', linestyle = '--')

    title = "Autocorrelation Length: "+str(x)+"\n"+"RMS Height: "+str(std)
    a.set_title(title)

    canvas.draw()
    canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

    toolbar.update()
    canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

```

```

def clear1():
    app = SeaofBTcapp()
    app.mainloop()

class PageOne(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Excel Data", font=LARGE_FONT)
        label.pack(pady=10, padx=10)

        # Frame for TreeView
        frame1 = tk.LabelFrame(self, text="Excel Data")
        frame1.place(height=350, width=1000)

    def File_dialog():
        """This Function will open the file explorer and assign the chosen file path to label_file"""
        filename = filedialog.askopenfilename(initialdir="/",
                                                title="Select A File",
                                                filetype=(("xlsx files", "*.xlsx"), ("All Files", "*.*")))
        label_file["text"] = filename
        return None

    def Load_excel_data():
        global df_columns, df_columns1, file_path
        """If the file selected is valid this will load the file into the Treeview"""
        file_path = label_file["text"]
        try:
            excel_filename = r"{}".format(file_path)
            if excel_filename[-4:] == ".csv":
                df = pd.read_csv(excel_filename)
            else:
                df = pd.read_excel(excel_filename)

        except ValueError:
            tk.messagebox.showerror("Information", "The file you have chosen is invalid")
            return None
        except FileNotFoundError:
            tk.messagebox.showerror("Information", f"No such file as {file_path}")
            return None

        clear_data()
        tv1["column"] = list(df.columns)
        tv1["show"] = "headings"
        for column in tv1["columns"]:
            tv1.heading(column, text=column) # let the column heading = column name

        df_rows = df.to_numpy().tolist() # turns the dataframe into a list of lists

        df_columns = df["Distance"].to_numpy().tolist()
        df_columns1 = df["Height"].to_numpy().tolist()

        for row in df_rows:
            tv1.insert("", "end", values=row) # inserts each list into the treeview
        return None

    def clear_data():
        tv1.delete(*tv1.get_children())
        return None

    # Buttons
    button1 = tk.Button(self, text="Browse A File", command=lambda: File_dialog())
    button1.place(rely=0.55, relx=0.30)

    button2 = tk.Button(self, text="Load File", command=lambda: Load_excel_data())
    button2.place(rely=0.55, relx=0.45)

    button3 = tk.Button(self, text="Click to proceed", command=lambda: controller.show_frame(PageTwo))
    button3.place(rely=0.55, relx=0.58)

    # The file/file path text
    label_file = ttk.Label(self, text="No File Selected")
    label_file.place(rely=0, relx=0)

    ## Treeview Widget
    tv1 = ttk.Treeview(frame1)
    tv1.place(relheight=1, relwidth=1) # set the height and width of the widget to 100% of its container (frame1).

    treescrollx = tk.Scrollbar(frame1, orient="vertical", command=tv1.yview) # command means update the yaxis view of the widget
    treescrollx = tk.Scrollbar(frame1, orient="horizontal", command=tv1.xview) # command means update the xaxis view of the widget
    tv1.configure(xscrollcommand=treescrollx.set, yscrollcommand=treescrollx.set) # assign the scrollbars to the Treeview Widget
    treescrollx.pack(side="bottom", fill="x") # make the scrollbar fill the x axis of the Treeview widget
    treescrollx.pack(side="right", fill="y") # make the scrollbar fill the y axis of the Treeview widget

```

```

class PageTwo(tk.Frame):

    def __init__(self, parent, controller):

        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Optimised Graph", font=LARGE_FONT)
        label.pack(pady=10,padx=10)

        button3 = ttk.Button(self, text="Press to Generate Graph",
                              command=lambda: graph())
        button3.pack()

        button4 = ttk.Button(self, text="Surface Roughness Analysis",
                              command=lambda: controller.show_frame(PageThree))
        button4.pack()

        button5 = ttk.Button(self, text="New Analysis",
                              command=lambda: clear())
        button5.pack()

        button1 = ttk.Button(self, text="Back to Home",
                              command=lambda: controller.show_frame(StartPage))
        button1.pack()

        f = Figure(figsize=(10,5), dpi=100)
        a = f.add_subplot(111)

        canvas = FigureCanvasTkAgg(f, self)
        canvas.draw()
        canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

        toolbar = NavigationToolbar2Tk(canvas, self)
        toolbar.update()
        canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

    def graph():

        x = df_columns
        y = df_columns1
        ymed11 = scipy.signal.medfilt(y, kernel_size=9)

        a.plot(x,y)
        a.plot(x,ymed11, color='red')
        a.axhline(y = 0, color = 'g', linestyle = '--')

        df=pd.DataFrame()
        df['optimized']=ymed11
        print(ymed11)

        canvas.draw()
        canvas.get_tk_widget().pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

        toolbar.update()
        canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

    def clear():
        app = SeaofBTCapp()
        app.mainloop()

app = SeaofBTCapp()
app.mainloop()

```

## APPENDIX B: Coding of Setup of Executional Extension

```
import cx_Freeze
import sys
import matplotlib
import os
import scipy

base = None

if sys.platform == 'win32':
    base = "Win32GUI"

executables = [cx_Freeze.Executable("ProSight Surface Profiler.py", base=base, icon = "clienticon.ico")]

cx_Freeze.setup(
    name = "ProSight Surface Profiler",
    options = {"build_exe": {"packages": ["scipy.optimize", "scipy.integrate", "scipy", "tkinter", "matplotlib"],
    "include_files": [os.path.join(sys.prefix, "Lib", "site-packages", "Shapely.libs"), "clienticon.ico", "2.png"]}},
    version = "0.01",
    description = "ProSight Data Analysis Application",
    executables = executables
)
```

## APPENDIX C: Profiler System Coding

```
#include libraies
import RPi.GPIO as GPIO
import time
from datetime import datetime
import xlswriter
import time
import os # include this library for the code to acces to the os directory
import sys
import BlynkLib

#systemctl --user start rclone@gdrive

BLYNK_AUTH = '-7a6qkCMCVZ9P1k0oAhJlszNdFj9esRE'
blynk = BlynkLib.Blynk(BLYNK_AUTH, server="blynk.cloud")
GPIO.setwarnings(False)
# setting up GPIO
GPIO.setmode(GPIO.BCM) # set GPIO Mode
CLK=6 #pin for clock of the Flip-flope
MUX=5 #pin for multiplexer of the Flip-flope
ADC=12 #pin for Flip-flope output
M1=13 #pin for Motor to move foward
M2=16 #pin for Motor to move backward
Decoder=19 #pin for Decoder input
startrun=20 #pin for push button to start the system
LED=21 #pin for LED to indicate
```

```

#set input and output pins
GPIO.setup(LED,GPIO.OUT)
GPIO.setup(startrun,GPIO.OUT)
GPIO.setup(ADC,GPIO.IN)
GPIO.setup(Decoder,GPIO.IN)
GPIO.setup(M1,GPIO.OUT)
GPIO.setup(M2,GPIO.OUT)
GPIO.setup(CLK,GPIO.OUT)
GPIO.setup(MUX,GPIO.OUT)

#light up LED to indicate that the system it ready to go
GPIO.output(LED,GPIO.HIGH)

def main():
    #blinking linght to indicat the process start
    for i in range (7):
        GPIO.output(LED,GPIO.HIGH)
        time.sleep((7-i)*0.01)
        GPIO.output(LED,GPIO.LOW)
        time.sleep((7-i)*0.01)

    #put the year,month, day, hour, minutes into a string 'timestr'
    timestr = time.strftime("/home/cowy/mnt/gdrive/%Y%m%d-%H%M%S")+".xlsx"
    # save file in /home/pi/Desktop/ directory and name it using year month time

    print (timestr) #display the string

    #####create Excel File#####
    # Create a workbook and add a worksheet.
    workbook = xlswriter.Workbook(timestr)
    worksheet = workbook.add_worksheet()
    # Add a bold format to use to highlight cells.
    bold = workbook.add_format({'bold': 1})
    # Write some data headers.
    worksheet.write('A1', 'Distance',bold)
    worksheet.write('B1', 'Hight', bold)
    worksheet.write('C1', 'Binary', bold)
    #####create Excel File#####
    Dcycle = 30#dutycycle for motor M1
    delay = 0.8 #delay for the Sharp IR and ADC to get ready for measurment
    Delay1=0.005 #delay for shift register CLOCK
    bit=0 #ADC bit detection
    detect1=0 #Flip-flop detect, detect to prevent value from overloading
    i=0

    p=GPIO.PWM(M1,175) #set PWM for motor to move foward in to p
    q=GPIO.PWM(M2,175) #set PWM for motor to move foward in to q
    p.start(Dcycle) #start pwm of p with Dcycle% duty cycle
    distance=0

```



```

while i<42: #loop for 42 times
    #
    while GPIO.input(Decoder)==True :
        pass
    time.sleep (0.005)#delay
    if GPIO.input(Decoder)==False:
        p.ChangeDutyCycle(0) #stop motor
        i=i+1
        time.sleep(delay)

    #####_ADC_#####
    # initiate the Flip-flop
    b=0
    GPIO.output(CLK,GPIO.LOW)
    GPIO.output(MUX,GPIO.HIGH)
    time.sleep(Delay1)
    #load values form ADC into the Flip-flope
    GPIO.output(MUX,GPIO.LOW)
    time.sleep(Delay1)
    GPIO.output(CLK,GPIO.HIGH)
    time.sleep(Delay1)
    #change MUX to high to for value shifting
    GPIO.output(MUX,GPIO.HIGH)

    #give 8 clock pulses into the Flip-flop to shift the values then
    #convert them from binary into decimal
    for T in range (8):
        GPIO.output(CLK,GPIO.HIGH)
        time.sleep(Delay1)
        if T == 7:
            if GPIO.input(ADC)==True:
                b=b+1
        if T == 6:
            if GPIO.input(ADC)==True:
                b=b+2
        if T == 5:
            if GPIO.input(ADC)==True:
                b=b+4
        if T == 4:
            if GPIO.input(ADC)==True:
                b=b+8
        if T == 3:
            if GPIO.input(ADC)==True:
                b=b+16
        if T == 2:
            if GPIO.input(ADC)==True:
                b=b+32
        if T == 1:
            if GPIO.input(ADC)==True:
                b=b+64

```

```

        if T == 0:
            if GPIO.input(ADC)==True:
                b=b+128
            GPIO.output(CLK,GPIO.LOW)
            time.sleep(Delay1)

v=5*b/255.0 # convert decimal value into voltage value
#set the first value taht is detected as reference
if i ==1:
    #using formula that had obtained to claculate the distance
    distance = (v/15.985)**(1/-0.866)

    distance1= (v/15.985)**(1/-0.866)
    h=distance-distance1 #calculate the hight
    print (h)

    # Some data we want to write to the worksheet.
    j=i+1
    k="A"+str(j)
    l="B"+str(j)
    m="C"+str(j)
    worksheet.write(k,(i-1)*0.6) #lenght
    worksheet.write(l, h) #hight
    worksheet.write(m, b)

#####_ADC_#####
p.ChangeDutyCycle(Dcycle)
while GPIO.input(Decoder)==False:
    pass

# Create a new chart object. In this case an embedded chart.
chart1 = workbook.add_chart({'type': 'line'})

# Configure the first series.
chart1.add_series({
    'name':      '=Sheet1!$B$1',
    'categories': '=Sheet1!$A$2:$A$51',
    'values':     '=Sheet1!$B$2:$B$51',
    })

# Add a chart title and some axis labels.
chart1.set_title ({'name': 'Surface measurement'})
chart1.set_x_axis({'name': 'Distance(cm)'})
chart1.set_y_axis({'name': 'height (cm)'})

# Set an Excel chart style. Colors with white outline and shadow.
chart1.set_style(10)

# Insert the chart into the worksheet (with an offset).
worksheet.insert_chart('D2', chart1, {'x_offset': 25, 'y_offset': 10})

```

```

workbook.close() #close Excel file

p.ChangeDutyCycle(0) #stop motor
q.start(35) #start PWM to drive motor to move backwards
time.sleep(1) #run motor for 1 second
q.ChangeDutyCycle(0) #stop motor

p.stop() #stop PWM
q.stop() #stop PWM
GPIO.cleanup() #clear all the GPIO Pins

def restart_program():
    python=sys.executable
    os.execl(python,python,*sys.argv)
# Push button to start running
@blynk.VIRTUAL_WRITE(0)
def write_virtual_pin_handler(value):
    if value == ["1"]:
        GPIO.output(startrun,GPIO.HIGH)
        print("START")
        main()
        sys.stdout.flush()
        restart_program()
    elif value == ["0"]:
        GPIO.output(startrun,GPIO.LOW)
        print("PENDING")
try:
    while True:
        blynk.run()
except KeyboardInterrupt:
    print("Quit")

```