

**A LOW COST ALL-SKY IMAGER FOR SHORT-
TERM FORECASTING HORIZON OF SOLAR
IRRADIANCE**

TAN JING HONG

UNIVERSITI TUNKU ABDUL RAHMAN

**A LOW-COST ALL-SKY IMAGER FOR SHORT-TERM
FORECASTING HORIZON OF SOLAR IRRADIANCE**

TAN JING HONG


**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Electrical and Electronic
Engineering with Honours**

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

May 2023

DECLARATION

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

Signature :  _____

Name : TAN JING HONG _____

ID No. : 18 UEB 02837 _____

Date : 19/5/2023 _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**A LOW-COST ALL-SKY IMAGER FOR SHORT-TERM FORECASTING HORIZON OF SOLAR IRRADIANCE**” was prepared by **TAN JING HONG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Electrical and Electronic Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : IR.DR.LIM BOON HAN

Date : _____

Signature : _____

Co-Supervisor : _____

Date : _____

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.

© 2023, Tan Jing Hong. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Ir.Dr. Lim Boon Han for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement. Without their unwavering support, this project would not have been possible. Thank you all once again for your valuable contributions.

ABSTRACT

Solar power is an important source of renewable energy, and accurate solar forecasting is essential for its reliability supply. The variability of solar irradiance due to cloud cover makes it difficult to accurately forecast the amount of electricity that can be generated by a solar power plant. An all-sky imager is an innovative technology that can significantly enhance the accuracy of solar forecasting by providing real-time data on cloud cover. The objective of this project is to develop an automated prototype that captures sky images and logs solar irradiance data. The prototype is built with a combination of hardware components, including a Raspberry Pi 3B+ as the microprocessor, a SOZ-03 solar irradiance meter, an ADC (ADS1115), a temperature sensor, and a Raspberry Pi camera module. The parameters of the camera were fine-tuned to capture clear sky images, and an observation was made to identify which parameter affects the camera's exposure. To prevent water droplets from accumulating on the dome, a rainproof radical coat solution was applied. The prototype is enclosed in a waterproof PVC IP66 junction box, and a Raspberry Pi fan module is connected to the microprocessor to prevent overheating. Additionally, a heat sink is attached to the CPU and NIC of the Raspberry Pi 3B+ to help dissipate heat. These measures ensure that the prototype is durable and can function effectively even in harsh weather conditions. The prototype is designed to capture sky images every 5 minutes from 7am to 7pm, and log solar irradiance data every 10 seconds during this period. The use of an automated system will ensure that the data is consistently captured at regular intervals, providing a more accurate representation of the solar irradiance patterns. The captured sky images are uploaded in real-time to Google Drive, providing easy access to the images for analysis and further processing. The solar irradiance data is logged in an Excel file and uploaded to Google Drive at 7.30pm daily, enabling users to quickly access and analyze the data. This automated process eliminates the need for manual data collection, saving time and reducing the risk of errors.

TABLE OF CONTENTS

DECLARATION		i
APPROVAL FOR SUBMISSION		ii
ACKNOWLEDGEMENTS		iv
ABSTRACT		v
TABLE OF CONTENTS		vi
LIST OF TABLES		ix
LIST OF FIGURES		x
LIST OF SYMBOLS / ABBREVIATIONS		xiii
LIST OF APPENDICES		xiv
CHAPTER		
1	INTRODUCTION	15
1.1	General Introduction	15
1.2	Importance of the Study	16
1.3	Problem Statement	17
1.4	Aim and Objectives	17
1.5	Scope and Limitation of the Study	17
1.6	Contribution of the Study	18
1.7	Outline of the Report	18
2	LITERATURE REVIEW	20
2.1	Introduction	20
2.2	Literature Review	20
2.2.1	Types of Solar Irradiance	20
2.2.2	Factors Causing Intermittency of Solar Irradiance	21
2.2.3	Effect of Solar Photovoltaic Power Intermittency on Power Systems	22
2.2.4	Alternative to Overcome Solar Photovoltaic Intermittency on Power Systems	22

	2.2.5 Models Developed to Predict Solar Irradiance Availability	25
	2.2.6 Research on Forecasting Method and Forecasting Horizon for Solar Irradiance in Different Locations	26
	2.2.7 Cloud Detection and Coverage Area using All Sky Imager (all sky camera)	28
	2.2.8 Solar Irradiance Forecasting with All-Sky Imager	29
3	METHODOLOGY AND WORK PLAN	31
	3.1 Introduction	31
	3.2 Circuit construction	31
	3.3 Raspberry Pi 3B+ microprocessor	35
	3.4 DS18B20 temperature sensor	37
	3.5 ADS1115 (Analog to Digital Converter)	38
	3.6 NES Silicon Solar Irradiance Meter	41
	3.7 Raspberry Pi Camera Module	42
	3.8 Waterproof Enclosure for the prototype	43
	3.9 Heat sink and Raspberry Pi Fan Module	44
	3.10 Google Drive API	46
	3.11 Cronjob in Raspberry Pi	49
	3.12 Rain-Proof Radical Coat solution	50
	3.13 Verifying the Functionality of Prototype	51
	3.14 Testing the Prototype Outdoors	52
	3.15 Work plan	54
4	RESULTS AND DISCUSSION	56
	4.1 Introduction	56
	4.2 Automated Process for Data Logging and Image Capturing	56
	4.3 Tuning the Parameter of Raspberry Pi Camera Module	58
	4.4 Adjusting the Position of the Raspberry Pi Camera	

4.5	Waterproofing for the Circuit and Enclosure for the Raspberry Pi Camera	62
4.6	Summary on the Process to Capture a Sky Image with least Overposure and Reflection Observed	64
4.7	Data Logging and Image Uploading with Google Drive	64
4.8	Sky Images Captured under Different Sky Conditions	67
5	CONCLUSIONS AND RECOMMENDATIONS	72
5.1	Conclusions	72
5.2	Recommendations for future work	72
	REFERENCES	74
	APPENDICES	76

LIST OF TABLES

Table 3.1:	The specifications of DS18B20	37
Table 3.2:	Specification of ADS1115	39
Table 3.3:	Specification of SOZ-03 Silicon Solar Irradiance Meter	41

LIST OF FIGURES

Figure 2.1:	Graph of voltage drop before the installation of PV system	23
Figure 2.2:	Graph of voltage drop after the installation of PV system	24
Figure 2.3:	Graph of current before and after the installation of PV system	24
Figure 2.4:	Flowchart of All Sky Camera system	28
Figure 3.1:	Block Diagram of Circuit Construcion	32
Figure 3.2:	Flowchart of Solar Irradiance and Temperature Data Logging Process	33
Figure 3.3:	Flowchart of Image Capturing and Uploading Process	34
Figure 3.4:	GPIO pins of Raspberry Pi 3B+	35
Figure 3.5:	Parts of Raspberry Pi 3B+	36
Figure 3.6:	Connection of DS18B20 temperature sensor	37
Figure 3.7:	Waterproof version of DS18B20	38
Figure 3.8:	ADS1115	40
Figure 3.9:	Pinout of ADS1115	40
Figure 3.10:	SOZ-03 Silicon Solar Radiation Sensor	41
Figure 3.11:	Connection of Raspberry Pi camera module to CSI port of Raspberry Pi	42
Figure 3.12:	Weatherproof PVC IP66 junction box	43
Figure 3.13:	Waterproof cable joints IP68	44
Figure 3.14:	Image Captured by Thermal Camera while Raspberry Pi 3B+ Microprocessor is Working	45
Figure 3.15:	Heat Sink for Raspberry Pi 3B+	45
Figure 3.16:	Raspberry Pi Fan Module	46
Figure 3.17:	Relationship diagram of Google Drive API	46

Figure 3.18:	Access token generated from OAuth 2.0 Playground	47
Figure 3.19:	Client secret and client ID obtained from user account	49
Figure 3.20:	5 Parameters of Cronjob	50
Figure 3.21:	Rain-proof Radical Coat solution from MR.DIY	51
Figure 3.22:	Gantt Chart for FYP 1	54
Figure 3.23:	Gantt Chart for FYP 2	55
Figure 4.1:	List of Cronjob for the All-Sky Imager Prototype	57
Figure 4.2:	Temporary Storage (SD card) when the Wi-Fi Connection is weak	57
Figure 4.3:	Sky image captured which is overexposed	59
Figure 4.4:	Sky image captured with minimal exposure	60
Figure 4.5:	Initial position of the Raspberry Pi Camera (beneath the waterproof casing)	61
Figure 4.6:	Position of Raspberry Pi Camera when moved to the tip of the dome	61
Figure 4.7:	Sky image with lesser reflection observed	62
Figure 4.8:	Prototype of the All-Sky Imager	63
Figure 4.9:	Sky image captured when water droplets accumulate on the surface of the dome	63
Figure 4.10:	Sky image captured after the Rain-proof Radical Coat Solution is applied during rainy day	64
Figure 4.11:	Excel Files containing Solar Irradiance Data that is uploaded to Google Drive everyday at 7.30 pm	66
Figure 4.12:	Images uploaded each 5 minutes from 7am to 7pm to Google Drive	67
Figure 4.13:	Sky Image Captured on 2023-03-09_12:35	67
Figure 4.14:	Sky Image Captured on 2023-04-19_14:20	68
Figure 4.15:	Sky Image Captured on 2023-04-22_14:25	68
Figure 4.16:	Sky Image Captured on 2023-03-11_16:40	69

Figure 4.17: Sky Image Captured on 2023-04-20_17:20	69
Figure 4.18: Sky Image Captured on 2023-04-21_11:00	70
Figure 4.19: Sky Image Captured on 2023-03-09_17:30	70
Figure 4.20: Sky Image Captured on 2023-03-09_17:35	71
Figure 4.21: Sky Image Captured on 2023-04-29_15:00	71

LIST OF SYMBOLS / ABBREVIATIONS

ADC	Analog to Digital Converter
AEC	Automatic Exposure Control
AI	artificial intelligence
ANN	Artificial neural network
ASI	All-Sky Imager
AWB	Automatic White Balance
CNN	Convolution Neural Network
CPU	Central Processing Unit
CSI	Camera Module Interface
DLSR	digital-single-lens-reflective
IMD	Indian Meteorological Department
ISO	International Organization for Standardization
K_T	clear sky index
K_t	clearness index
LSTM	Long Short-Term Memory
MAE	mean absolute error
MAPE	mean absolute percentage error
MBE	mean bias error
MLP	Multi-layer Perceptron
NASA	National Aeronautics and Space Administration
NET	Net Energy Metering
NIC	Network Interface Controller
OS	Operating System
PV	solar photovoltaic
R^2	coefficient of determination
RAM	Random Access Memory
RGB	red-green blue
RMSE	root mean square error
WANN	wavelet artificial neural networks
WRF	Weather Research and Forecasting Model
WVM	Wavelet-base Variability Model

LIST OF APPENDICES

Appendix A: writetoexcel.py	76
Appendix B: uploadexcel.py	79
Appendix C: imagecapturinganduploading.py	81

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Over the past few years, the industrial revolution and the expeditious growth of the population in Malaysia have significantly raised energy demand and electricity consumption. Furthermore, in addition to the environmental pollution from the power generation with coal and natural gas, solar photovoltaic technology has become the desired alternative for the power generation in Malaysia.

Due to its geographical advantage, Malaysia has high solar irradiation throughout the year. Therefore, solar photovoltaic technology is the most promising renewable energy source for Malaysia. The Malaysian government also introduced the Net Energy Metering (NET) scheme in November 2016 to promote the uptake of renewable energy. The NEM concept is that the electricity produced from the photovoltaic (PV) system are consumed by the customer first, and the excess energy will be exported to TNB at the prevailing displaced cost (SEDA MALAYSIA, 2021). As the installation of PV systems in Malaysia has significantly increased, the efficiency of the PV system has become the most concerning matter for the end user. Moreover, Malaysia is a tropical country and is geographically surrounded by the sea. The vaporization of seawater, together with the seasonal winds, creates a large area of clouds. Hence, sunlight is highly scattered and can cause fluctuation to the PV system output. Solar irradiance intermittency can be caused by several factors, for an example shading caused by moving clouds (Fondriest Environmental, 2014).

Other than that, the study of the cloud movement and cloud thickness is also important for forecasting solar irradiance intermittency. The short time horizon of cloud movement and cloud coverage are studied by using All-Sky Imager (ASI). First, the image of the cloud and sky clearness will be collected. After that, to eliminate colour casts white balancing process is carried out. Cloud segmentation is then carried out to differentiate sky, cloud and other non-related things. Next, the collected images will be fed to the correlation model and machine learning techniques will be implemented to compute cloud coverage,

3D volumetric of clouds, changing patterns and the moving speed of the cloud. Finally, with the information from the correlation model, the solar irradiance value is then computed.

The measurement of solar irradiance has become crucial in the operation of a PV power plant. By monitoring key factors affecting the performance of the PV system, such as solar irradiation, a solar irradiance meter can be used. However, the coverage of high-frequency solar variability is still limited, and low temporal resolution measurements of solar irradiance for a month or an hour can be inaccurate by up to 27% (Lave et al., n.d.). To collect a comprehensive data set, it is necessary to develop a solar irradiance measurement technique with better resolution, which can gather a larger amount of solar irradiance data in a shorter time and transfer the acquired data to cloud storage, allowing users to view the data even when they are not at the solar site.

1.2 Importance of the Study

After fossil fuel and hydropower generation, solar photovoltaic (PV) energy has become the dominant modern renewable energy in Malaysia's electricity generation mix. However, the main concern of solar photovoltaic energy is the intermittency of solar irradiance mainly caused by the cloud movement and cloud coverage which directly affect the grid reliability. Intra-hour forecasting which is forecasting below one hour is normally utilised in the PV grid to ensure the reliability and economy of the power system. However, the frequent happening of passing clouds and sudden thunderstorms in the tropical region makes this intra-hour forecasting limited to low spatial resolution, low accuracy and shortening of the forecast horizon. To improve the short-term solar irradiance forecasting horizon or accuracy, a network of multiple low-cost digital-single-lens-reflective (DLSR) cameras surrounding a primary standard all-sky imager is proposed. However, before setting up the network of multiple DSLRs, a single low-cost all-sky imager has to be built first. This is to ensure that the functioning and the quality of the images logged by this single low-cost all-sky imager are suitable for the further use of short-term solar irradiance forecasting.

1.3 Problem Statement

Although solar energy is now both economical and sustainable, the main challenge with PV generation is the inconsistency and unpredictability of sunshine. Solar irradiance's highly intermittent nature directly threatens voltage regulation and demand management issues, such as grid reliability. To ensure grid reliability, an all-sky imager is normally used for solar power forecasting in PV power plants. However, in the tropical region, due to the frequent cloud passing and sudden thunderstorms, the forecast capability of this technique is limited to low spatial resolution, which is less than 2km, a short forecast horizon which is less than 20 minutes and accuracy which is less than 85%. By implementing a network of all-sky imagers surrounding a primary standard all-sky imager, the temporal forecast horizon is expected to be improved to more than 30 minutes or the accuracy is expected to be improved. How do we build a low-cost all-sky imager?

1.4 Aim and Objectives

This project aims to build a low-cost all-sky imager capture sky images and a high-resolution data logger which logs the data of solar irradiance for solar power forecasting purposes.

- To build a data logger to log solar irradiance data using Raspberry Pi 3B+ as a microprocessor.
- To log the images of the sky clearness and cloud by using a Raspberry Pi camera module with the combination of a fisheye lens to mimic the all-sky camera technology.
- To upload the collected solar irradiance data which stored in excel files and sky images to Google Drive effectively.

1.5 Scope and Limitation of the Study

This project focuses on building a low-cost all-sky imager for the collection of sky images and a high-resolution data logger to log solar irradiance data.

There are a few limitations that can be found in this project. First, the camera module used might not be as good as the camera used by large scale all-sky imager as it is only cheap in price. Next, the field of view of a single all-sky imager is limited.

1.6 Contribution of the Study

This project contributes to the advancement of solar forecasting technology by developing an automated prototype for capturing sky images and logging solar irradiance data. The use of an all-sky imager greatly enhances the accuracy of solar forecasting by providing real-time data on cloud cover, which is a major factor affecting solar irradiance. The automated system developed in this project eliminates the need for manual data collection, ensuring consistent and accurate capture of data at regular intervals.

The prototype developed in this project has the potential to benefit the solar power industry by improving the efficiency and cost-effectiveness of solar power plants. The accurate forecasting of solar irradiance enables better planning and management of power generation, reducing the risk of power shortages and increasing the reliability of solar power as a source of renewable energy.

1.7 Outline of the Report

Chapter 1 highlights solar photovoltaic technology's significance in Malaysia and the obstacles it faces due to solar irradiance intermittency caused by cloud movements. It emphasizes the importance of all-sky imagers in solar irradiance forecasting and discusses the study's scope and limitations.

Chapter 2 summarizes the key findings and limitations of previous research studies related to solar irradiance forecasting. The chapter also examines the various theoretical frameworks that support solar irradiance forecasting, including physical models, statistical models, and artificial intelligence (AI) models. It highlights the strengths and weaknesses of each framework and provides a rationale for their use in solar irradiance forecasting.

Chapter 3 outlines the detailed methodology and work plan for the study, including the construction of the prototype circuit and the selection of components for its assembly. It also explains the coding process and verification steps taken to ensure the proper functionality of the prototype.

Chapter 4 presents the results and discussion of the study, including the verification process of the prototype's functionality. The chapter also details the process of adjusting the parameters of the Raspberry Pi camera and its position

to capture sky images with the least reflection and overexposure observed. In addition, it provides a brief estimation of how long the 20 GB free storage of Google Drive can be used to upload the sky images and Excel files.

Chapter 5 summarizes the study's findings and provides recommendations for future research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The measuring of solar irradiance is essential for determining the reliability of a solar system. Numerous research on solar irradiance has been reviewed, and different approaches are being developed to acquire irradiance data. Other than that, cloud detection and coverage area by using an all-sky imager plays an important role in studying the cloud cover which helps in climate modelling and weather forecasting.

2.2 Literature Review

2.2.1 Types of Solar Irradiance

Numerous sectors benefit from solar power, including agriculture, construction, and the life sciences; photovoltaic cells and solar thermal energy are the two most common methods of harnessing this resource. While solar photovoltaic cells convert light into electricity, solar thermal energy converts the heat and cold from the sun's rays into usable heat and cold. Solar-powered unmanned aerial vehicles are just one of many cutting-edge technologies that have been unveiled recently. The development of highly efficient solar energy systems relies, in part, on an accurate prediction of available solar irradiation.

There are three basic types of solar irradiance, and they are called "global," "direct," and "diffused," respectively. Next, the amount of solar energy that reaches Earth's surface perpendicular to a given beam of sunlight is known as direct solar irradiance. Cloud surfaces, particulate matter, pollutants, water vapour, aerosols, and other substances in the atmosphere can scatter and reflect some of the sun's rays, resulting in a phenomenon known as diffuse solar irradiance. As a result, the amount of solar energy that may be collected may decrease if the sun beams are not able to effectively reach surface of the earth (Fondriest Environmental, 2014).

2.2.2 Factors Causing Intermittency of Solar Irradiance

The variability of solar irradiance over time at a particular location is referred to as the intermittency of solar irradiance. The intermittency of solar irradiance is driven by a few factors, the solar altitude angle, scattering components like clouds and the site's elevation above sea level.

The distance solar irradiance from the atmosphere is shortened with increasing elevation. This may suggest a rise in irradiation, but not heat. Ozone acts as a shield against the sun, reducing its intensity when the sun is at a lower inclination. The path the sun's rays have to travel is shortest when it's directly overhead. This is why annual net sun irradiance is greatest in the equator compared to the poles. The amount of solar irradiation that reaches Earth's surface can also be decreased by factors such as cloud cover and air pollution. Clouds and aerosols in the sky When there are more clouds in the sky, the amount of sunlight reaching the Earth is less (this phenomenon is known as "irradiance reduction"). Thus, when taking solar irradiance readings, the angle of the sun becomes less of a factor as cloud cover grows.

The spatial characteristics of intermittency can be reduced if solar energy can be collected over a larger area. Intermittency can be avoided if solar farms are built all over the world to supply electricity needs. The temporal characteristic is the length of time over which measurements of irradiance were taken. Variability in solar irradiance has been observed on the time scale of seconds. The degree of variation in irradiance is determined by two indices, namely the clear sky index (K_t) and the clearness index (K_f). The clearness index measures the amount of solar radiation that can penetrate the Earth's atmosphere and reach the planet's surface, while the clear sky index is the ratio of the irradiance received at the ground to the irradiance received at the ground during clear-sky conditions (Perez et al., 2016).

2.2.3 Effect of Solar Photovoltaic Power Intermittency on Power Systems

Solar photovoltaic (PV) energy's intermittent nature can have several unintended consequences for power grid efficiency and reliability. Depending on factors like PV capacity, load profile, dispatchable generation unit flexibility, and the system network, these effects could be localised or system-wide (Lannoye et al., 2011).

The fluctuation of sun irradiation at particular locations causes varying of solar PV power output. This fluctuation as result will cause power quality distortion at that site or feeder, like flicker and overvoltage. Variations in voltage induced by fluctuating PV power can be slow (steady) on sunny days, but fast (transient) on partially cloudy days due to passing clouds. The tap changers and voltage regulators may be impacted by voltage fluctuations. Cloudy weather can also enhance harmonic distortion from PV systems. Increased net load variability may result from high variability of PV power output. This can cause thermal-based generation units to cycle more frequently (turn on/off). Thus, increased cycling leads to higher wear-and-tear costs as well as higher emissions (Albadi, 2019).

2.2.4 Alternative to Overcome Solar Photovoltaic Intermittency on Power Systems

To balance short-term demand changes, an operating reserve is required. Operators would require higher operating reserves to maintain the supply-demand balance as the proportion of intermittent PV electricity in a power system increases. However, adding additional spinning reserve capacity will result in more units working at sub-optimal operating points with higher marginal costs (Lannoye et al., 2011).

Other than that, there is another alternative to compensate for the intermittency of PV electricity which is by backup power stored in battery cells such as a lithium-ion battery. Grid-scale batteries operate in the same way that micro-batteries do in consumer products, but in a much bigger size. On a sunny day, the solar panel generates electricity from the sun, the generated DC electricity will flow through an inverter to produce AC electricity and the extra AC electricity will charge up the battery. Then, the battery stores electrical

energy, which is subsequently released when needed. However, this is not the best alternative way as the cost of the battery is expensive.

Next, a study was conducted in northern Algeria to investigate whether installing PV system at certain places beyond 300metres of the transformer can compensate for the voltage drop problem which is caused by the intermittency of solar irradiance. As shown in Figure 2.1, the voltage drop exceeds 10% before the installation of PV system.

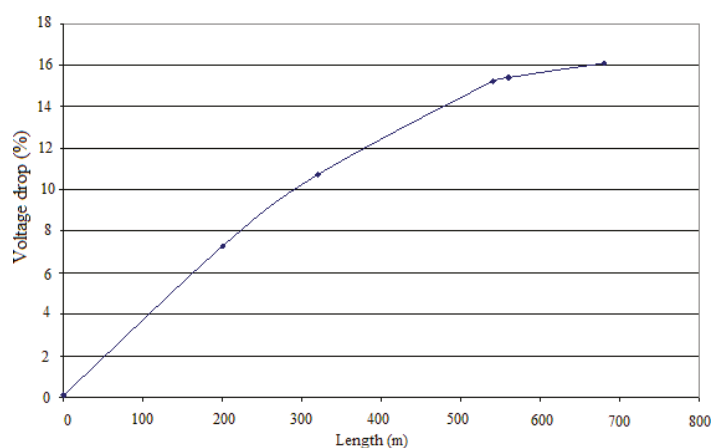


Figure 2.1: Graph of voltage drop before the installation of PV system (Mouheba et al., 2012)

Based on Mouheb's research, the grid is installed with ten sets of solar modules with specifications of $150W_p$ and 24 V. After that, the PV systems were placed at a distance of 300 metres from the transformer. The voltage loss is significantly reduced after the installation of the PV system, as seen in Figure 2.2. PV system installation aids in lowering voltage drops by reducing current flows via line. Figure 2.3 depicts the amount of current flowing through the line at various hours. From the result, it is shown that voltage drop in a low-voltage network is compensated with the installation of PV systems to the grid (Mouheba et al., 2012).

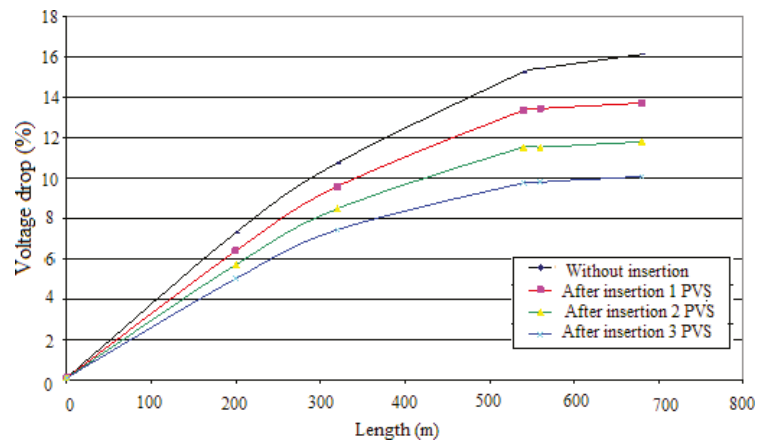


Figure 2.2: Graph of voltage drop after the installation of PV system (Mouheba et al., 2012)

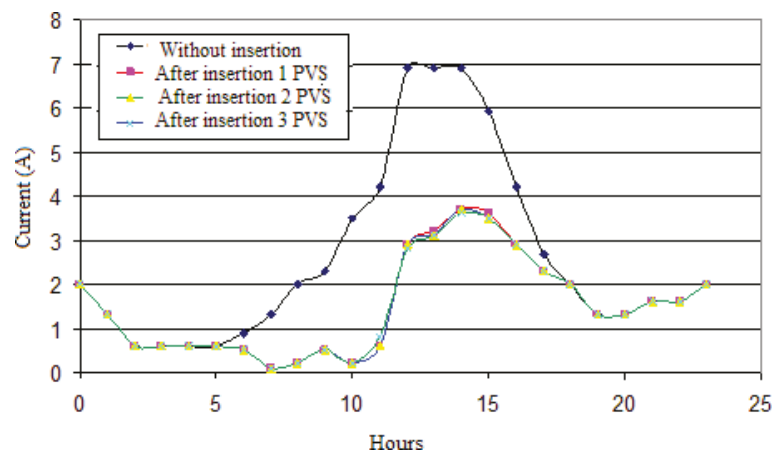


Figure 2.3: Graph of current before and after the installation of PV system (Mouheba et al., 2012)

2.2.5 Models Developed to Predict Solar Irradiance Availability

The Wavelet-base Variability Model (WVM) which is developed by Lava et al. is intended to supply simulated power plant output to grid integration studies that evaluate the effects of installing PV on existing electric feeders. The ratio of the variability of solar irradiance over the area of the power plant can be predicted by using WVM with single irradiance point sensor data, PV density, cloud speed of the day, and a Spatio-temporal correlation function given.

Next, it is a general model developed by Hoff and Perez to investigate the effects when output between two different plants is correlated. The covariance between any two plants is calculated by multiplying the standard deviations of each location by the correlation coefficient between two places.

$$\sigma_{\Delta t}^{fleet} = \sqrt{\sum_{i=1}^N \sum_{j=1}^N \sigma_{\Delta t}^i \sigma_{\Delta t}^j \rho_{\Delta t}^{i,j}} \quad (2.1)$$

According to Hoff and Perez, the output variability can be affected by the distance between plants, the number of plants, the position of the sun and sky clearness. Hoff and Perez then develop a new parameter called the Dispersion factor that links three parameters, the speed of the cloud, V , the distance between plants, d , and the time interval, Δt (Perez et al., 2016).

$$\rho = \frac{1}{1 + \frac{d}{(\Delta t)(V)}} \quad (2.2)$$

Hoff and Perez proved that the relative output variability from multiple PV plants is affected by the Dispersion factor. From the result obtained by Hoff and Perez, three relationships are observed. When the distance between two plants increases the relative output variability decreases; when the time interval increases the speed of decreasing of relative output variability decreases and when the cloud moving speed increases the speed of decreasing of relative output variability decreases as well (Perez et al., 2016).

2.2.6 Research on Forecasting Method and Forecasting Horizon for Solar Irradiance in Different Locations

Monjoly et al. (2017) proposed a new approach for hourly forecasting of global solar radiation using multiscale decomposition methods. They decomposed clear sky index Kc data into different time scales and used classic forecasting models based on linear and non-linear methods. They also proposed a hybrid approach based on multiscale decomposition methods for solar forecasting. The authors verified the effectiveness of their approach by analyzing the predictive performances using different multiscale decompositions and forecast models. The analysis revealed that the proposed hybrid approach resulted in significantly improved accuracy for solar forecasting. They demonstrated the superiority of their approach over classic neural network models, with RMSE (root mean square error) decreasing from 25.86% to 7.86%. Therefore, the study provides a reliable and effective approach for hourly forecasting of global solar radiation.

Royer et al. (2016) proposed a short-term solar radiation forecasting model based on an iterative combination of wavelet artificial neural networks (WANN). The study used solar radiation data from three different locations in Brazil and compared the performance of their model with that of single WANN and persistence models. The results showed that the iterative combination of WANN outperformed the other models in terms of accuracy and precision, as measured by the mean absolute error (MAE) and the mean absolute percentage error (MAPE). The study concluded that the proposed model has the potential to contribute to the development of renewable energy systems by improving the accuracy of solar radiation forecasting.

Lima et al. (2016) conducted a study to produce solar irradiance forecasts for the Brazilian Northeastern region using the Weather Research and Forecasting Model (WRF) combined with a statistical post-processing method and an artificial neural network (ANN) technique. The results showed a significant improvement in the WRF model forecasts when adjusted by the ANNs, yielding lower bias and RMSE, and an increase in the correlation coefficient. The study provides a useful methodology for short-term predictions of energy resources to support the planning and management of electricity generation and distribution systems in the region.

Halabi et al. (2018) developed hybrid adaptive neuro-fuzzy inference system models to predict monthly global solar radiation using different meteorological parameters. The proposed models included particle swarm optimization, genetic algorithm, and differential evolution. The models were evaluated using statistical indicators, such as root mean square error and coefficient of determination, and showed good agreement with measured datasets. Hybrid particle swarm optimization achieved the best statistical indicators in both training and testing models. The developed models were deemed efficient for predicting global solar radiation for various applications.

Malik and Garg (2019) proposed a long-term solar irradiance forecasting method for Indian cities using artificial neural networks (ANNs). The study aimed to predict monthly and yearly solar irradiance values for different Indian cities for up to 20 years in advance. The authors used historical meteorological data from the National Aeronautics and Space Administration (NASA) and the Indian Meteorological Department (IMD) to train and validate the ANN model. The accuracy of the ANN model was evaluated using statistical metrics such as mean absolute error (MAE), root mean square error (RMSE), and coefficient of determination (R^2). The results showed that the proposed ANN model achieved high accuracy in predicting the long-term solar irradiance values, with an overall R^2 of 0.94.

2.2.7 Cloud Detection and Coverage Area using All Sky Imager (all sky camera)

Clouds have a substantial impact on the Earth's radiation balance. One of the factors that are crucial for weather forecasting and climate modelling is cloud coverage. Cloud coverage initially is determined by professional human observers; however, they are replaced by automatic observation equipment. All-sky imaging equipment has the potential to replace human observers as automatic observation instruments.

All sky cameras (ASC) are proposed that an ASC acquire whole sky hemispheric photos and generate correct cloud coverage information. Five major modules we included in this all-sky imaging system. Every 10 minutes, the imaging system takes a full-sky image. The temperature and humidity control module are used to collect temperature and humidity readings from linked sensors to understand real-time internal environment conditions. The data and analysis module is functioning to receive recorded photos, and temperature and humidity signals. The data processing module, which serves as the ASC's command and control centre, recognizes clouds from received whole-sky photos uses an algorithm for cloud detection and calculates the related cloud coverage and recognizes clouds from received whole-sky photos (Fa et al., 2019). The whole sky photos and computed cloud coverage is sent by the data transmission module to a web-based database management system for purpose of data storage. Finally, for easy viewing, all the whole sky images and their respective cloud coverage are shown on the web page.

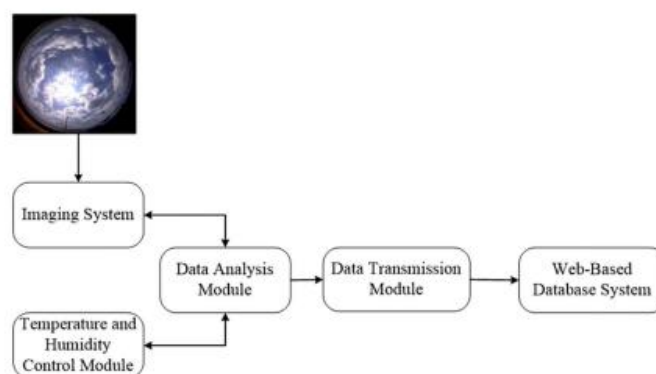


Figure 2.4: Flowchart of All Sky Camera system (Fa et al., 2019)

2.2.8 Solar Irradiance Forecasting with All-Sky Imager

The paper by Wang et al. (2020) proposes a minutely solar irradiance forecasting method based on a real-time sky image-irradiance mapping model. The authors aim to improve the accuracy of solar power forecasting, which is dependent on the accuracy of solar irradiance forecasting. The proposed method uses sky images to predict future solar irradiance at minute-level intervals, ranging from 1 to 10 minutes ahead. The authors start by extracting the red-green-blue (RGB) values and position information of pixels in sky images after background elimination and distortion rectification. The mapping relationship between the sky image and solar irradiance is explored, and a real-time sky image-irradiance mapping model is built, trained, and updated using real-time sky images and solar irradiance. The latest updated surface irradiance mapping model is then used to forecast the future solar irradiance by extracting inputs from the current sky image. The proposed method was tested and evaluated against other benchmarks using mean absolute percentage error (MAPE), root mean square error (RMSE), and mean bias error (MBE) for blocky clouds, thin clouds, and thick clouds. The results showed that the proposed method delivered much higher forecasting accuracy than other benchmarks. The average MAPE, RMSE, and MBE values were 22.66%, 92.72, and -1.26% for blocky clouds, 20.44%, 132.15, and -1.06% for thin clouds, and 18.82%, 120.78, and -0.98% for thick clouds.

Crisosto et al. (2018) proposed a one-hour prediction model for global solar irradiance using artificial neural networks (ANNs) and all-sky images. Accurate prediction of solar irradiance is critical for the optimal operation and control of photovoltaic (PV) systems. The authors used a dataset of 9,000 all-sky images captured by a fish-eye lens camera in Chile. The images were preprocessed to remove the sky background and shadows, and then converted into RGB color space. The RGB values and position information of the pixels in the image were used as inputs to the ANN model, which was trained to predict global solar irradiance. The proposed model achieved high prediction accuracy with an average mean absolute error (MAE) of 65.3 W/m² and root mean squared error (RMSE) of 87.6 W/m² for the one-hour ahead prediction. The authors compared their model with other methods, such as the persistence model

and the clear sky model, and demonstrated that the ANN-based approach outperforms them in terms of accuracy.

Al-Lahham et al. (2020) propose a methodology for predicting solar irradiance using a sky imager and machine learning techniques. The proposed approach involves capturing sky images and then processing them to extract the relevant features. The extracted features are then used to train a machine learning model that can predict solar irradiance for the next 5 minutes. The authors tested their proposed methodology on a dataset of sky images captured in Kuwait and compared their results with those obtained from two other machine learning models, namely, Multi-layer Perceptron (MLP) and Long Short-Term Memory (LSTM). The results obtained from the proposed methodology show that it outperforms the other two models in terms of prediction accuracy. The proposed model achieved an average mean absolute error (MAE) of 25.15 W/m², which is lower than the MAE obtained from the MLP (27.11 W/m²) and LSTM (28.51 W/m²) models. The authors attribute the superior performance of their model to the use of a convolutional neural network (CNN) architecture, which is better suited to processing image data.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This project aims to develop a low-cost all-sky for solar power forecasting purposes. The all-sky imager will capture sky images, while the data logger will record solar irradiance data. The chosen microprocessor for the prototype is the Raspberry Pi 3B+, which will be paired with other components, including the DS18B20 temperature sensor, the SOZ-03 Silicon Solar Irradiance Meter, the ADS1115 Analog to Digital Converter, and the Raspberry Pi camera module.

3.2 Circuit construction

First, Raspberry Pi 3B+ is powered up by using a 240V AC to 5V adapter. Next, the SOZ-03 Silicon Solar Radiation meter is powered up by using a 240V AC to 12V DC adapter. After powering up all the equipment, the data logger starts to collect the data for solar irradiance and temperature. The temperature data is obtained by the DS18B20 temperature sensor and the solar irradiance data is obtained by the SOZ-03 Silicon Solar Radiation meter. The camera module is connected to Raspberry Pi 3B+ at the CSI (Camera Module Interface) port to log the sky images. Since Raspberry Pi 3B+ can only read digital signals, a ADS1115 is used to convert the analog signal from the SOZ-03 Silicon Solar Irradiance Meter into a digital signal to be read by Raspberry Pi 3B+. The logged solar irradiance and temperature data and sky images will be uploaded to Google Drive. The circuit construction for this low-cost all-sky imager is shown in Figure 3.1.

Each Python code has its own function: writetoexcel.py logs data, uploadexcel.py upload Excel files to Google Drive, and imagecapturinganduploading.py captures and upload images to Google Drive. The data logging process for solar irradiance and temperature will begin at 7am every day and continue for 12 hours until 7pm. The data will be logged every 10 seconds during this time, and the information recorded will be saved in an Excel file. This Excel file will contain a log of the recorded data for each 10-second interval. At the end of each day, the Excel file will be uploaded to a

designated Google Drive folder at 7.30pm. This will ensure that the data is securely backed up and easily accessible for analysis or further processing. After uploading the Excel file to Google Drive, the operation will end, and the process of data logging will start up again at 7 am the following day. The flowchart of the data logging process is shown in Figure 3.2. The image capturing and uploading process for the all-sky imager will begin at 7am and continue until 7pm every day. Sky images will be captured and uploaded to a designated Google Drive folder every 5 minutes during this period, starting from 7am and ending at 7pm. Each captured image will be named with the date of that day, followed by the hour, minute, and second of the time taken, and the solar irradiance value. This naming convention will enable easy identification and organization of the images. The process of capturing and uploading images follows the same schedule as the data logging process, ending at 7:00 pm each day and starting up again at 7:00 am the following day. The flowchart of the image capturing and uploading process is shown in Figure 3.3.

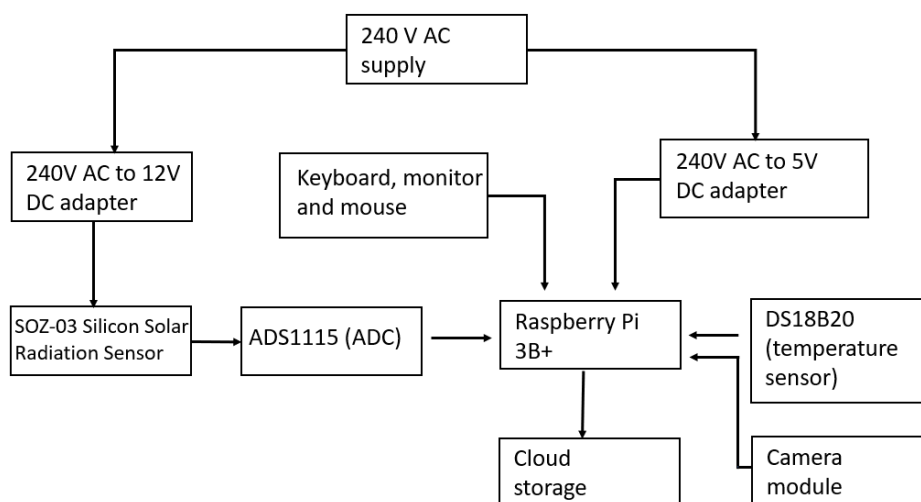


Figure 3.1: Block Diagram of Circuit Construcion

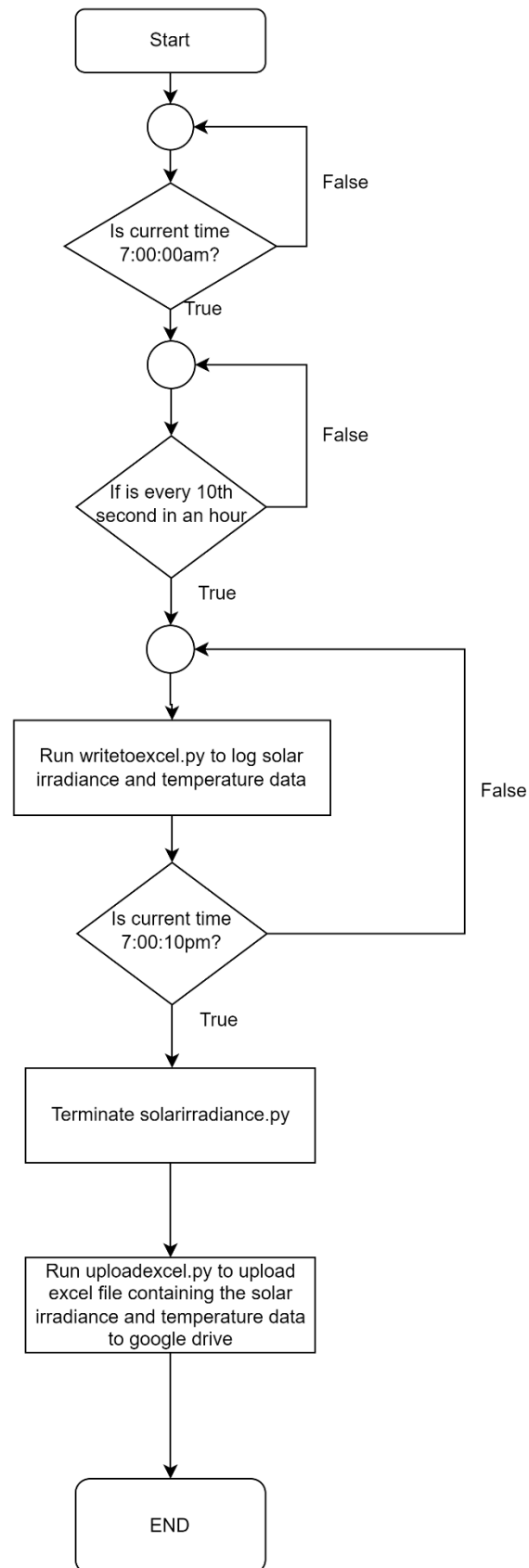


Figure 3.2: Flowchart of Solar Irradiance and Temperature Data Logging Process

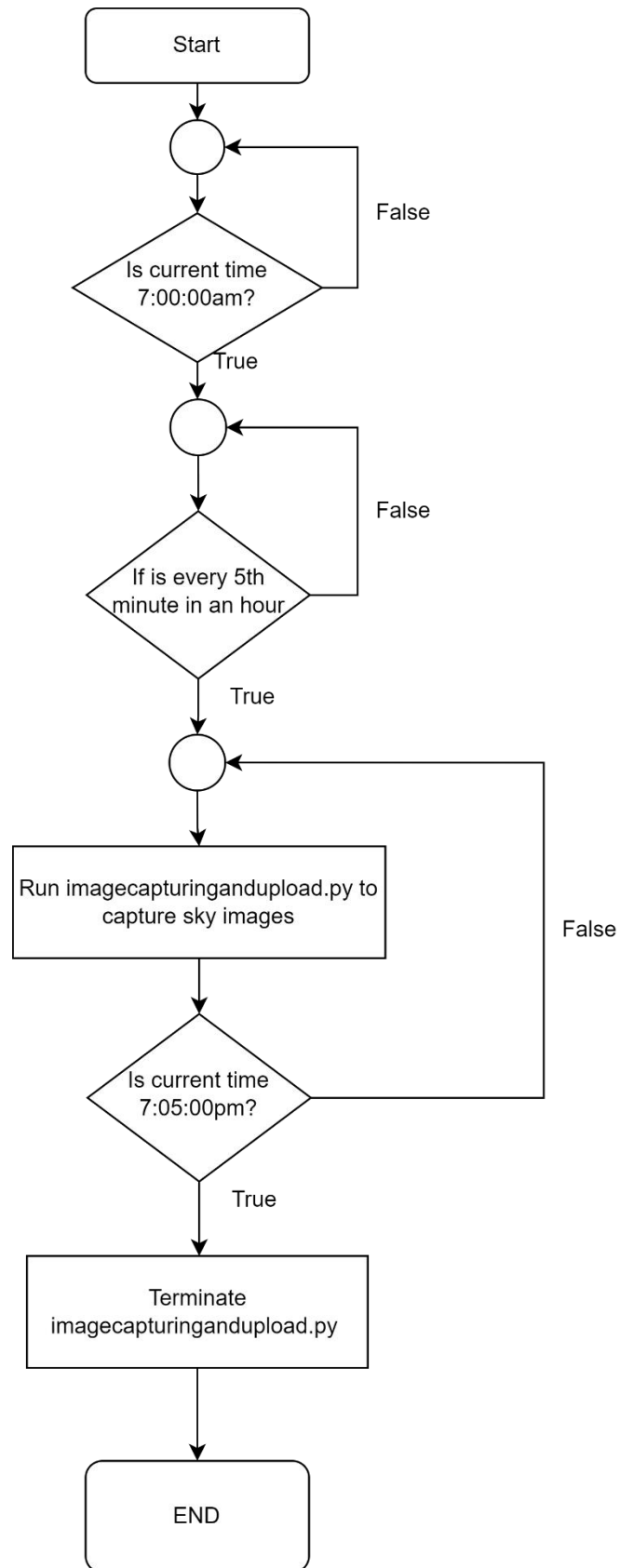


Figure 3.3: Flowchart of Image Capturing and Uploading Process

3.3 Raspberry Pi 3B+ microprocessor

Raspberry pi 3B+ microprocessor has a CPU (central processing unit) which is 6 bit with 1GB RAM (random access memory). It also comes with Broadcom BCM2837B0 chipset, 1.4 GHz Quad-Core ARM Cortex-A53, having 40 pin header, 4 USB 2.0 ports, Gigabit Ethernet, Micro SD socket, HDMI, MircoUSB power connector, Camera Serial Interface and comes with WiFi and Bluetooth. With the WiFi interface, it allows the updating of solar irradiance data and cloud images to the cloud storage from the protypte to the cloud storage. However, this Raspberry Pi 3B+ is not able to read analog signal as it lacks of an ADC (analog to digital converter)(Watson, 2018). The parts of Raspberry Pi 3B+ is shown in Figure 3.5.

Raspberry Pi 3B+ pinout with GPIO function :

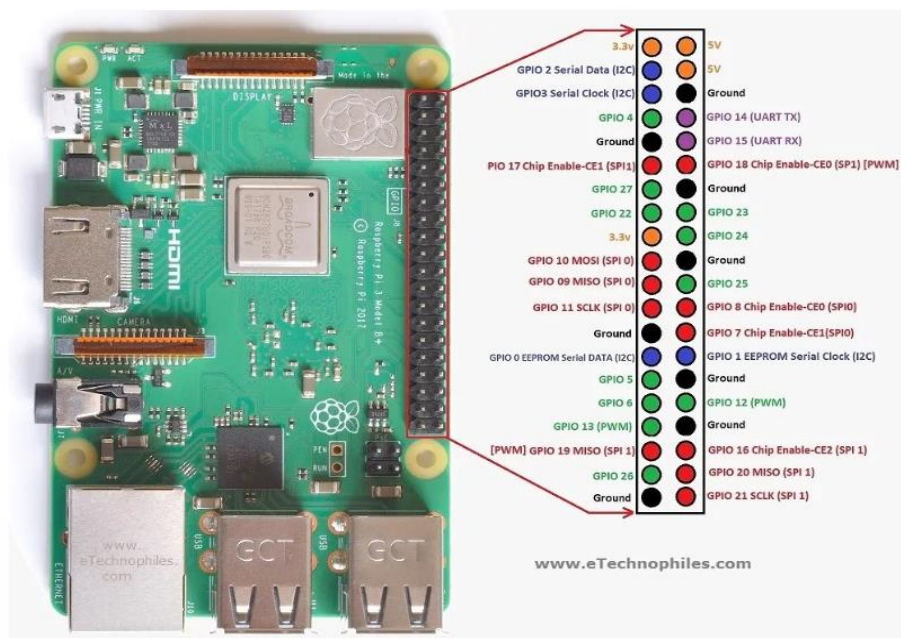


Figure 3.4: GPIO pins of Raspberry Pi 3B+ (Watson, 2018)

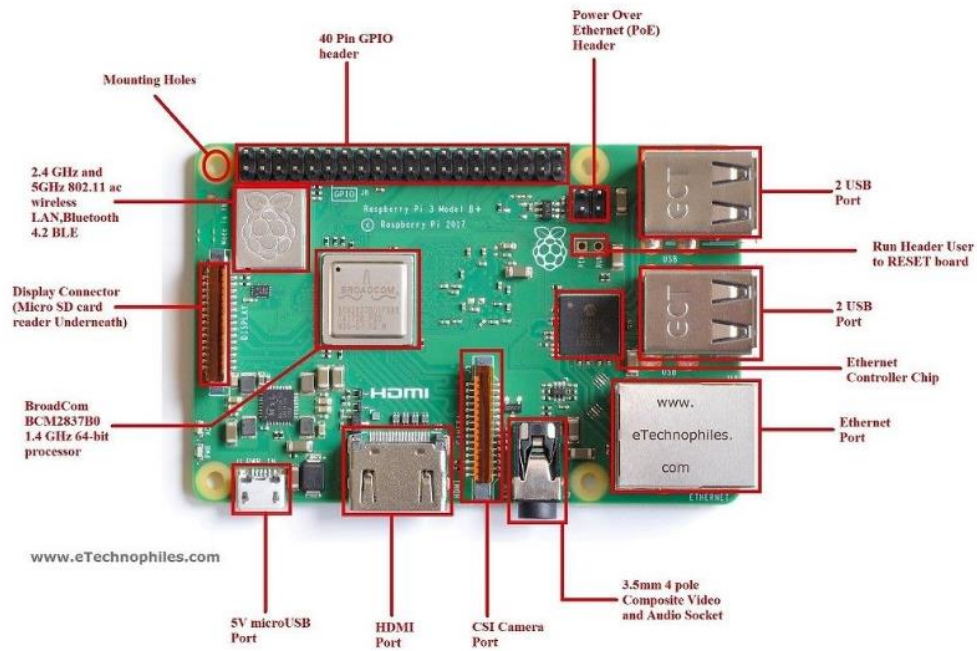


Figure 3.5: Parts of Raspberry Pi 3B+ (Watson, 2018)

3.4 DS18B20 temperature sensor

The temperature sensor selected in this prototype to measure the temperature is the DS18B20 temperature sensor shown in Figure 3.7. It is a robustly designed temperature sensor which can measure a wide range of temperatures, ranging from $-55\text{ }^{\circ}\text{C}$ to $+125\text{ }^{\circ}\text{C}$ with a decent accuracy of $\pm 0.5\text{ }^{\circ}\text{C}$. Besides, a $4.7\text{ k}\Omega$ resistor is required as a pull-up resistor to keep the line in a high state. The connection of the DS18B20 is shown in Figure 3.6 (PiMyLifeUp, 2022)

Table 3.1: The specifications of DS18B20

Specification	Values
Power supply range	3.3 – 5.5 V
Operating temperature	-55°C to $+125^{\circ}\text{C}$
Accuracy	$\pm 0.5^{\circ}\text{C}$ between range of -10 to $+85\text{ }^{\circ}\text{C}$
Resolution	9-12 bits

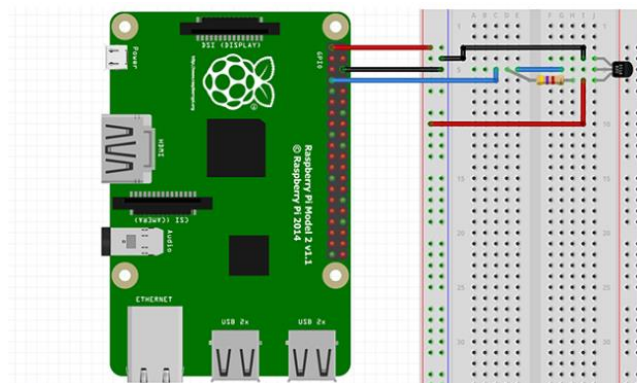


Figure 3.6: Connection of DS18B20 temperature sensor (PiMyLifeUp, 2022)



Figure 3.7: Waterproof version of DS18B20

3.5 ADS1115 (Analog to Digital Converter)

ADS1115 is a 16-bit Analog-to-Digital Converter (ADC) that is used to convert analog signals into digital signals in electronic circuits. The device is capable of converting signals from four different input channels and provides a precision measurement of the voltage levels. Figure 3.8 and Figure 3.9 shows the pinout and the hardware of the ADS1115.

Here is a description of each pin and its function on the ADS1115:

- (i) VDD and GND: These pins provide power and ground to the ADC.
- (ii) SCL: This pin is used for the serial clock signal for communication with the device. It is used in conjunction with the SDA pin.
- (iii) SDA: This pin is used for the serial data signal for communication with the device. It is used in conjunction with the SCL pin.
- (iv) ADDR: This pin is used to set the device address for I2C communication. It can be connected to either VDD or GND to set the address.
- (v) ALERT: This pin is used to indicate when a conversion is complete. It can also be used to generate an interrupt to the microcontroller.

- (vi) A0, 8. A1, 9. A2: These pins are used to select the input channel for the conversion. The device can measure signals from up to four different input channels.
- (vii) REF: This pin is used to set the reference voltage for the ADC. It can be connected to either VDD or GND to set the reference voltage.

To connect the ADS1115 to a microcontroller or other circuit, you will need to connect the VDD and GND pins to a power source and ground, respectively. The SDA and SCL pins should be connected to the corresponding I2C pins on the microcontroller. The ADDR pin should be connected to either VDD or GND to set the device address. The A0, A1, and A2 pins should be connected to the input signals you wish to measure. Finally, the REF pin should be connected to either VDD or GND to set the reference voltage for the ADC.

Table 3.2: Specification of ADS1115

Specifications	Values
Input voltage range	2 – 5.5 V
Operating temperature range	-40°C to +125 °C
Resolution	16 bits

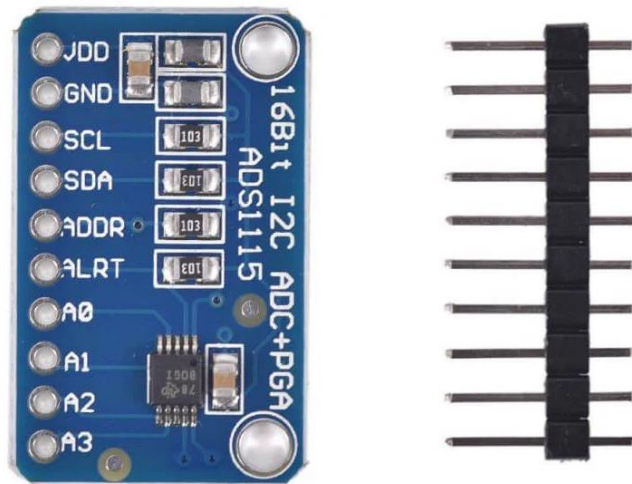


Figure 3.8: ADS1115 (Learning about Electronics, 2018)

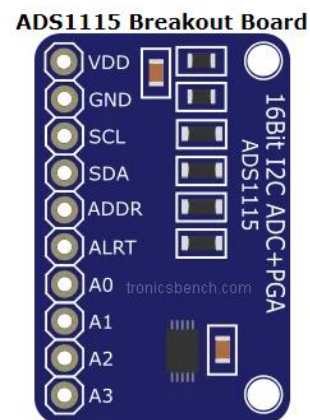


Figure 3.9: Pinout of ADS1115 (Learning about Electronics, 2018)

3.6 NES Silicon Solar Irradiance Meter

The SOZ-03 Silicon Solar Radiation Sensor consists of a solar panel made of mono-crystalline silicon with a special solar glass. It has an advanced waterproof junction box made up of UV-resistant material, hence providing it high UV resistance and long-term stability. Figure 3.10 shows the SOZ-03 Silicon Solar Irradiance Meter. Table 3.3 shows the specification of a SOZ-03 Silicon Solar Irradiance Meter.

Table 3.3: Specification of SOZ-03 Silicon Solar Irradiance Meter

Specifications	Values
Linear output signal range	50 – 1200 W/m ²
Output signal voltage	90 – 110 mV
Storage temperature	-45 °C to +70 °C
Power supply	7-30 V _{DC} / max 5mA



Figure 3.10: SOZ-03 Silicon Solar Radiation Sensor

3.7 Raspberry Pi Camera Module

As Raspberry Pi 3B+ comes with a CSI camera port, the Raspberry Pi camera module can connect directly to the CSI camera port (Raspberry Pi Foundation, n.d.). The connection of the camera module to the CSI camera port of Raspberry Pi is shown in Figure 3.11.

Features of Raspberry Pi camera module:

- (i) It has a Still Picture Resolution of 2592 x 1944
- (ii) It has 5MP Omnivision 5647 Camera Module
- (iii) It is compatible with many versions of Raspberry Pi Model
- (iv) It has a field of view of 62.2 degree

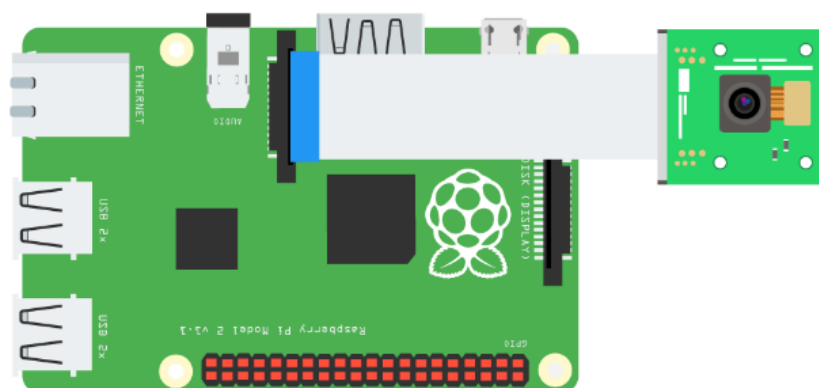


Figure 3.11: Connection of Raspberry Pi camera module to CSI port of Raspberry Pi (Raspberry Pi Foundation, n.d.)

3.8 Waterproof Enclosure for the prototype

The prototype is required to work under extreme weather for a long period of time. The enclosure should protect the components from water ingress and high heat. The weatherproof PVC IP66 junction box which is shown in is selected as the enclosure for the prototype. Furthermore, nylon plastic waterproof cable joints with an IP68 rating which is shown in Figure 3.13 is used to prevent water ingress when wires such as the power supply and the two wires from SOZ-03 solar irradiance meter is required to enter the junction box. The use of a waterproof casing with a silicone waterproof gasket and multiple enclosing screws was an effective solution to prevent water from entering the enclosure. The silicone gasket created a tight seal between the edges of the casing, while the enclosing screws provided additional support to maintain the stability of the waterproofing capability. This was crucial to ensure the components inside were protected from water damage, especially in harsh weather conditions.



Figure 3.12: Weatherproof PVC IP66 junction box



Figure 3.13: Waterproof cable joints IP68

3.9 Heat sink and Raspberry Pi Fan Module

By analyzing an image captured by a thermal camera, such as the one shown in Figure 3.14, it is possible to determine the parts of the Raspberry Pi 3B+ that generate the most heat. The areas with the highest temperatures are indicated by the color red, while blue represents the lowest temperatures (Raspberry Tips, n.d.). The CPU (central processing unit) is typically the hottest spot and appears prominently in the image. Meanwhile, the NIC (network interface controller) located on the right side of the board also generates significant heat. To help dissipate heat and prevent overheating of the Raspberry Pi 3B+ microprocessor, a combination of a heat sink and a fan module can be used. The heat sink increases the surface area available for heat transfer, while the fan module forces air over the heat sink to increase the rate of heat transfer and lower the temperature of the component being cooled. Figure 3.15 shows the heat sink used for Raspberry Pi 3B+ while Figure 3.16 shows the Raspberry Pi Fan Module.

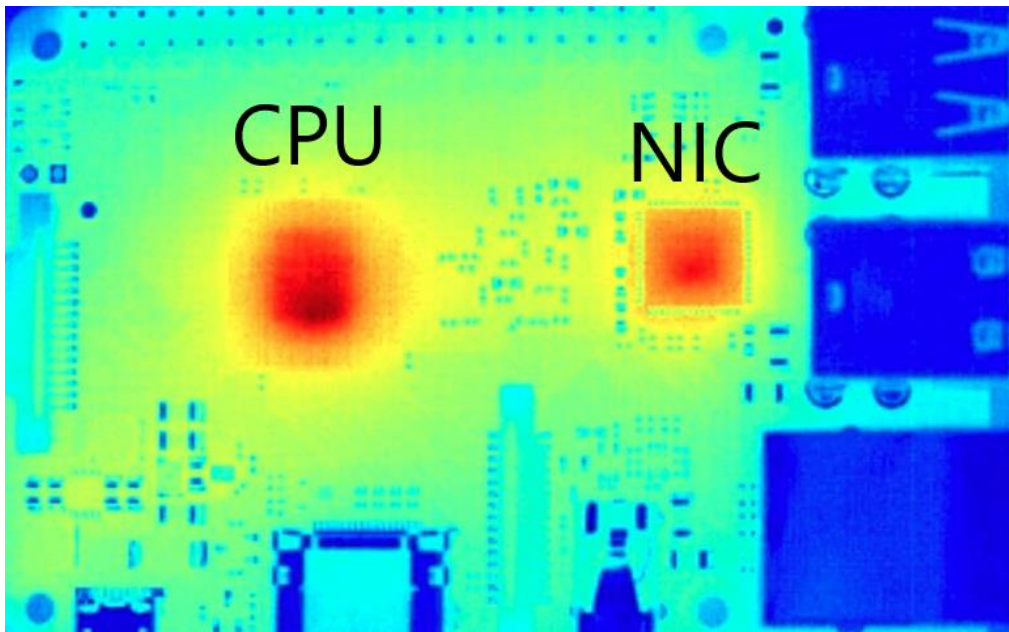


Figure 3.14: Image Captured by Thermal Camera while Raspberry Pi 3B+ Microprocessor is Working (Raspberry Tips, n.d.)

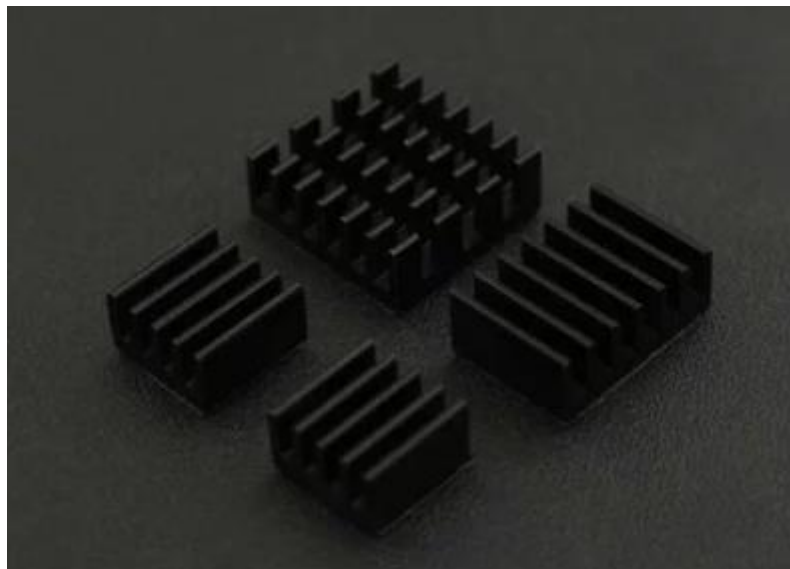


Figure 3.15: Heat Sink for Raspberry Pi 3B+



Figure 3.16: Raspberry Pi Fan Module

3.10 Google Drive API

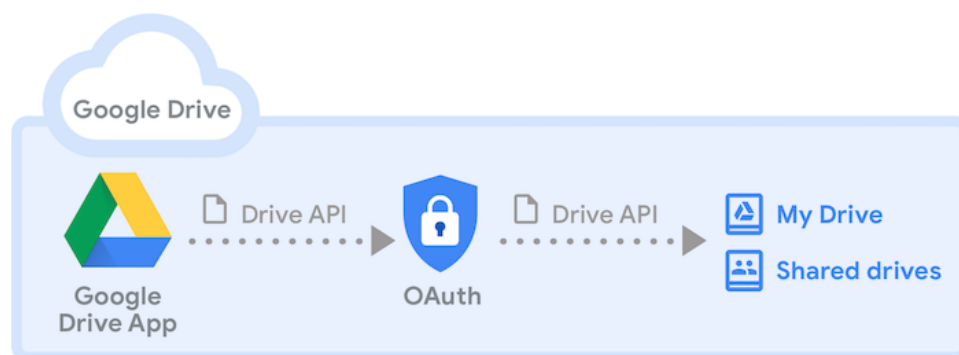


Figure 3.17: Relationship diagram of Google Drive API

The Google Drive API is an interface that developers can use to access and manipulate various Google Drive functions like creating folders, uploading and downloading files, and managing permissions. Figure 3.17 shows the pathway to access into Google Drive Storage.

Through the Google Drive API, developers can easily integrate Google Drive features into their own applications, enabling users to interact with their Google Drive files and folders directly from within the application. For instance, a developer could use the API to allow users to upload files to their Google Drive account from their own application. In our project, we are doing as a developer.

To use the Google Drive API, developers must first create a project in the Google Cloud Console and enable the API for that project. They then need to authenticate with the API using OAuth 2.0 to obtain an access token that they can use to make API requests. The API provides various endpoints to manage files and folders, and developers can use any programming language that supports HTTP requests to interact with it. Figure 3.18 shows the interface where the access token and refresh token is obtained in order to access into Google Drive Storage.

The screenshot shows the Google Developers OAuth 2.0 Playground interface. At the top, there is a header with the Google Developers logo and the text "OAuth 2.0 Playground" with a close button (X). Below the header, there are two steps: "Step 1 Select & authorize APIs" and "Step 2 Exchange authorization code for tokens". The "Step 2" section is expanded, showing instructions: "Once you got the Authorization Code from Step 1 click the Exchange authorization code for tokens button, you will get a refresh and an access token which is required to access OAuth protected resources." Below the instructions, there is a text input field for the "Authorization code" containing the value "4/0AWtgzh43uTfKn6qrebd-I7jQQsL0u1zV_elpXQBQCiv". A blue button labeled "Exchange authorization code for tokens" is positioned below the input field. Below the button, there are two more text input fields: "Refresh token" containing "1//042J9Zr_bP-TECgYIARAAGAQSNwF-L9lrX_qEx4V14Hd" and "Access token" containing "ya29.a0AVvZVsozZIRnIN0FrDZaJFr". A "Refresh access token" button is located to the right of the access token input field. At the bottom of the form, there is a checked checkbox labeled "Auto-refresh the token before it expires." A note at the very bottom states: "Note: The OAuth Playground will automatically revoke refresh tokens after 24h. You can avoid this by specifying your own application OAuth credentials using the Configuration panel."

Figure 3.18: Access token generated from OAuth 2.0 Playground

Other than that, client IDs and client secrets are often utilized as a means of authentication and authorization. A client ID serves as an exclusive identifier that is assigned to an application upon registration with a service provider, such as Facebook or Google. The client ID is utilized to distinguish the application when it submits requests to the provider's API, and is frequently contained within the request headers or URL parameters. This allows the provider to ascertain which application is initiating the request and uphold any access controls or rate limitations that may be in effect.




A client secret, on the other hand, is a confidential key that is furnished to the application along with the client ID. The client secret is typically employed alongside the client ID to authenticate the application during API requests. Its confidentiality is meant to be maintained, and not disclosed to any other party. When an application makes a request to the API, it submits the client ID and client secret in the request headers or URL parameters, and the service provider employs the client secret to verify the legitimacy of the request.

The client ID and client secret are utilized to guarantee that only authorized applications can access a service provider's API. By enlisting applications and assigning distinct client IDs and client secrets, service providers can exercise better control over who can access their APIs and how much access they are granted. Figure 3.19 shows the client ID and client secret for a particular Google Drive account.

Client ID	307749992917-tp29t2s5lhkua9nk8jblodhhjapnlpne.apps.googleusercontent.com
Creation date	2 February 2023 at 20:06:30 GMT+8

Client secrets

If you are in the process of changing client secrets, you can manually rotate them without downtime. [Learn more](#)

Client secret	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	 
Creation date	2 February 2023 at 20:06:30 GMT+8	
Status	 Enabled	

[+ ADD SECRET](#)

Figure 3.19: Client secret and client ID obtained from user account

Overall, the Google Drive API is a powerful tool that helps developers create robust applications that integrate seamlessly with Google Drive functionality. This allows users to effectively manage their files and collaborate with others without having to leave the application. Thus, in this project, we are utilizing Google Drive API to have connection with the Google Drive in order to upload the images of the sky images to Google Drive.

3.11 Cronjob in Raspberry Pi

A cron job, also known as a cron schedule, is a way to automate the execution of a command or script on a Unix-like operating system at specific intervals or times. Cron is a time-based job scheduler used in Unix-like operating systems (OS), such as Linux and macOS, and is derived from the Greek word "chronos" meaning time. Cron jobs can be set to run at any frequency, from every minute to every year, and are commonly used for tasks like backing up data, sending email reminders, updating software, and performing system maintenance. To set up a cron job, a user creates a file using a text editor that contains the command or script to be executed and the schedule for when it should run. The crontab file is used to store a list of all scheduled cron jobs and can be edited using the "crontab" command-line utility. There are 5 parameters in cronjob which is shown in Figure 3.20. It is separated by minute / hour / day of month / month / day of week and command to execute.



Figure 3.20: 5 Parameters of Cronjob

In summary, cron jobs are a powerful tool that automate tasks on Unix-like operating systems, making them a popular choice for system administrators and developers. These are the general steps to set up a cron job to run a Python script everyday from 7am to 7pm:

- (i) Open the crontab file: Use the command "crontab -e" in the terminal to open the crontab file.
- (ii) Write the cron job command: In the crontab file, write the command or script you want to schedule. The format of the command should include five fields, separated by spaces or tabs, which represent the minute, hour, day of the month, month, and day of the week when the command will be executed. Write the cron job command: Use "`* 7-19 * * * /usr/bin/python3 /path/to/script.py`". This command will run the script.py file located at /path/to/ every day from 7am to 7pm.
- (iii) Save and exit: After writing the command in the crontab file, save and exit the file. The cron job will now be scheduled to run at the specified time and date.

3.12 Rain-Proof Radical Coat solution

The rain-proof radical solution is typically made from a combination of water-repellent compounds, such as fluoropolymers, silanes, or siloxanes. These compounds work by bonding with the surface of the dome, creating a layer that repels water droplets. The function of the rain-proof radical solution is to

prevent water droplets from accumulating on the surface of the dome, which can distort or blur the images captured by the camera. The solution works by forming a hydrophobic barrier on the surface of the dome, repelling water and preventing it from adhering to the surface. This ensures that the camera can capture clear and high-quality images, even in wet or rainy conditions. The rain-proof radical coat solution used in this study is shown in Figure 3.21.



Figure 3.21: Rain-proof Radical Coat solution from MR.DIY

3.13 Verifying the Functionality of Prototype

To ensure the functionality of the all-sky imager prototype, a verification process must be conducted to test its ability to operate under different environmental conditions. Under hot sun conditions, the temperature of the Raspberry Pi 3B+ microprocessor must be within the recommended ranges of 0°C to 80°C and Raspberry Pi Camera must be within -20°C to 60°C, respectively. Under rainy conditions, the water resistance of the junction box and cable joints must be tested to prevent water damage to the prototype. Additionally, the sky images taken during both sunny and rainy days must be analyzed for accuracy, and the Wi-Fi connection must be checked for the ability to upload the images and Excel files. In the event of poor connectivity, the backup code should be tested to ensure temporary storage on the SD card is a viable option. The list of the verification process is shown below:

- (i) Set up the all-sky imager prototype in an outdoor location with a clear view of the sky.
- (ii) Power on the all-sky imager and verify that it is functioning correctly, including checking that all hardware components are working
- (iii) Test the prototype at different times of the day and in different weather conditions, such as during hot sun and rainy days.
- (iv) Capture sky images at different times of the day and in different weather conditions, such as during hot sun and rainy days.
- (v) Verify that the captured sky images are of good quality and can be analyzed to obtain accurate data, such as cloud cover or sky brightness.
- (vi) Test the connectivity to Google Drive and verify that sky images and excel files can be uploaded successfully.
- (vii) Simulate a temporary loss of internet connectivity and verify that the backup code temporarily stores the sky images captured on the SD card storage.

3.14 Testing the Prototype Outdoors

Ensuring that the hardware components of the prototype are operating within their recommended temperature ranges is crucial to ensure their proper functionality. Tuning certain parameters of the Raspberry Pi Camera Module was also necessary to improve the quality of sky images captured under sunlight conditions. Testing the strength of the wifi connectivity and implementing a backup code to store images in case of a temporary loss of internet connectivity were important steps in ensuring the reliability of the prototype.

- (i) **Temperature Control:** In order to ensure proper functioning of the Raspberry Pi 3B+ microprocessor and Raspberry Pi Camera module, it is important to operate them within the recommended temperature range. This is particularly crucial when dealing with hot sun conditions, as the microprocessor should operate within 0°C to 80°C, while the camera module should be within -20°C to 60°C.

- (ii) **Parameter Tuning:** To ensure good quality of the sky image captured, it is important to identify which parameters affect the image quality. This involves observing the images captured and identifying any issues such as overexposure or reflection. Once identified, adjustments can be made to the relevant parameters to improve the image quality.
- (iii) **Testing Wi-Fi Connectivity:** It is important to ensure that the prototype has a strong connection to Wi-Fi in order to be able to upload the sky images captured. To test this, the strength of the Wi-Fi connectivity is measured at the location where the prototype is placed. If a weak or no connection is detected, the backupcode.py is tested to ensure that it can function by storing the captured images in the SD card storage.

3.15 Work plan

Gantt Chart

No.	Project Activities	Planned Completion Date	Gantt Chart																
			W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17
1.	Discuss the project details with Dr Lim	2022-06-17	■																
2.	Formulate general project overview	2022-06-24		■															
3.	Write the report overview and prepare for introduction writing	2022-07-08			■	■													
4.	Find information regarding literature review and compile relevant studies into the report	2022-08-19			■	■	■	■	■	■	■	■							
5.	Write the methodology and plan the project flow	2022-08-19								■	■	■							
6.	Record the preliminary results and report writing	2022-09-02										■	■	■					

Figure 3.22: Gantt Chart for FYP 1

Gantt Chart





No.	Project Activities	Planned Completion Date	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	
1.	Buy components, setting up our prototype to upload images to google drive, capture solar irradiance data	2023-03-31	■	■	■	■	■	■	■	■	■									 
2.	Test our prototype on rooftop to capture data and capture images for certain days and poster planning and preparation	2023-04-22										■	■	■	■					 

Figure 3.23: Gantt Chart for FYP 2

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

The use of solar energy has become increasingly popular as a renewable source of energy. In order to optimize the efficiency of solar panels, it is important to gather data on solar irradiance, temperature, and other environmental factors. In this project, a prototype has been developed using a Raspberry Pi 3B+ along with an ADS1115 and temperature sensor, as well as a Raspberry Pi camera module, all housed in a waterproof case.

The design of the prototype includes the collection of solar irradiance data at 10-second intervals from 7am to 7pm daily and record it into an Excel file, upload it to Google Drive at 7.30pm and capture images of the sky every 5 minutes from 7am to 7pm daily and upload them to Google Drive. The prototype is programmed to automate these processes through the use of cronjob. In this part, the results obtained from the prototype and the implications of the data gathered is discussed.

4.2 Automated Process for Data Logging and Image Capturing

The automation of the sky image capturing and data logging process using a cronjob is a crucial step in reducing manual labour and increasing efficiency. The automation process ensured that the data was captured and stored regularly and accurately, without the need for manual intervention.

This study aimed to automate the process of capturing sky images and logging temperature and solar irradiance data using a cronjob. The code `imagecapturingandupload.py` was scheduled to run from 7 am to 7:05 pm to capture sky images and log solar irradiance data every 5 minutes. The captured images were named with the real-time solar irradiance data, date, and time, and then uploaded to Google Drive. The `writetoexcel.py` code was scheduled to log the solar irradiance and temperature data every 10 seconds and write it to an Excel file, which was also scheduled to run from 7 am to 7 pm. Finally, the `uploadexcel.py` code was scheduled to upload the Excel file containing the temperature and solar irradiance data to Google Drive at 7:30 pm. Figure 4.1

shows the list of cronjob for the automation data logging, image capturing and upload as well as the uploading of the excel files containing solar irradiance data for the project.

The automation process also relied heavily on a strong Wi-Fi connection for uploading the Excel file and images to Google Drive. However, in cases where the Wi-Fi connection is weak, the backup code ensured that the data was not lost. The backup code stored the captured images and data in the Raspberry Pi storage, which is the SD card storage, until a stronger Wi-Fi connection was available to upload the data to Google Drive. Figure 4.2 shows the location of the sky images and excel files which are temporary stored if the Wi-Fi connection is weak.

```
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
30 19 * * * /usr/bin/python3 /home/pi/uploadexcel.py
*/5 7-19 * * * /usr/bin/python3 /home/pi/imagecapturingandupload.py
0 7 * * * /usr/bin/python3 /home/pi/writetoexcel.py
```

Figure 4.1: List of Cronjob for the All-Sky Imager Prototype

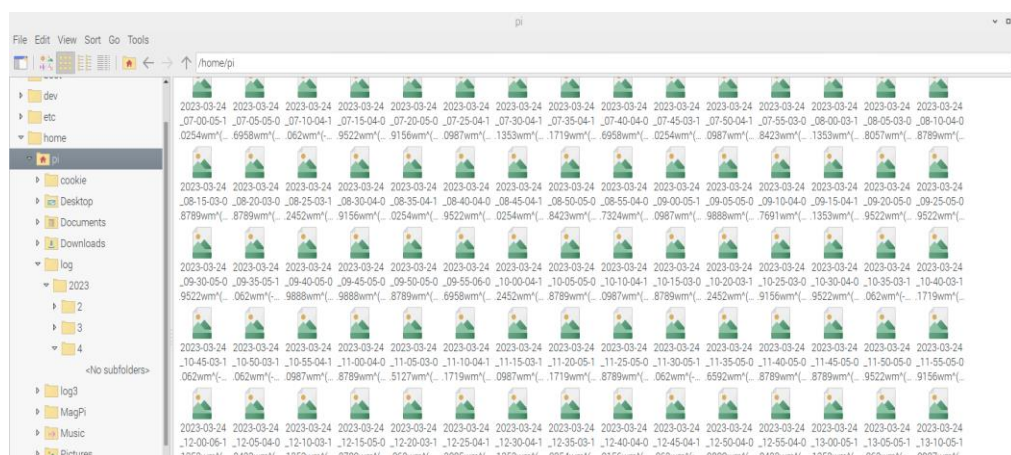


Figure 4.2: Temporary Storage (SD card) when the Wi-Fi Connection is weak

4.3 Tuning the Parameter of Raspberry Pi Camera Module

This study is aimed to tune the parameters of the Raspberry Pi camera module to capture outdoor pictures without overexposure due to sunlight. Initially, the default camera settings is used and the pictures captured were overexposed and showed only white images. To achieve the objective, the tuning process is conducted by considering the scenario of the sunlight from 12 pm to 4 pm.

The Raspberry Pi camera module has various parameters that can be adjusted to optimize image quality for different lighting conditions and purposes. Among the main parameters are resolution, frame rate, exposure compensation, ISO, white balance, image format, sharpness, contrast, saturation, and zoom and cropping. Resolution determines the number of pixels in the image, while frame rate refers to the number of frames per second captured by the camera. Exposure compensation regulates the amount of light entering the camera sensor and can be set automatically or manually through the automatic exposure control (AEC) function. ISO (International Organization for Standardization) sets the camera's sensitivity to light, which can be increased for capturing brighter images in low-light conditions but can also add noise or graininess. White balance adjusts the color temperature of the image and can be set manually or automatically through the automatic white balance (AWB) function. Image format supports various formats such as JPEG and PNG. Sharpness, contrast, and saturation can be manually adjusted through the camera's software to control image quality. Lastly, the camera module can support digital zoom and cropping to focus on a specific part of the image. To optimize image quality for capturing pictures under sunlight and prevent overexposure, it is important to tune the parameters of the Raspberry Pi camera module. While all of the parameters mentioned earlier can be adjusted to optimize image quality, when dealing with sunlight, only certain parameters matter. These parameters include exposure compensation, ISO, and white balance. Exposure compensation needs to be set to allow only the appropriate amount of light to enter the camera sensor, while ISO should be adjusted to avoid overexposure while capturing bright images under sunlight. White balance should be set to ensure that the colours in the image are accurately represented, preventing any unwanted colour casts. By carefully tuning these parameters, users can obtain high-quality images even when capturing under sunlight.

The tuning process was initiated by first setting the camera's ISO to 100 and exposure compensation to 0. However, this resulted in visible pictures but with many reflections and overexposed areas. To address this issue, the ISO was adjusted to 100 and the camera's exposure compensation was set to -25, which resulted in visible pictures with minimal overexposure. Additionally, the camera resolution was set to 1920 x 1080 and the white balancing mode was set to auto. Figure 4.3 illustrates the sky image captured using the Raspberry Pi Camera Module's default settings, while Figure 4.4 shows the same image captured after the ISO is set to 100, Camera Resolution to 1920x1080, Camera Exposure Compensation to -25 and White Balancing Mode to Auto. The comparison of the two figures demonstrates how tuning these important parameters can effectively deal with overexposure caused by sunlight, resulting in a sky image with minimal overexposure.

Figure 4.3: Sky image captured



Figure 4.3: Sky image captured which is overexposed

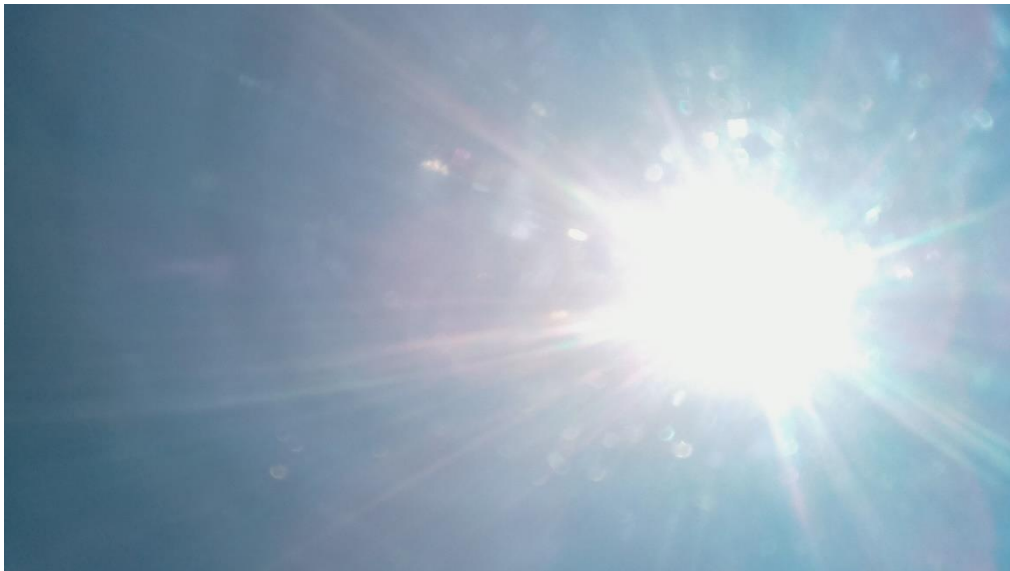


Figure 4.4: Sky image captured with minimal exposure

4.4 Adjusting the Position of the Raspberry Pi Camera

The position of the camera can have a significant impact on the quality of the images captured. When the camera is installed beneath a waterproof case, it can result in significant reflection in the captured images, as shown in Figure 4.4. This is because the light passing through the dome can reflect into the camera lens, leading to decreased image quality. By moving the camera closer to the dome, the reflection can be reduced, and the image quality can be improved. In this case, moving the camera directly on the tip of the dome resulted in the least reflection observed in the captured images, which is the best position for capturing high-quality images. Figure 4.7 shows the sky image with lesser reflection after changing the position of the Raspberry Pi Camera to the tip of the dome. The position of the Raspberry Pi Camera is changed from position shown in Figure 4.5 to Figure 4.6.



Figure 4.5: Initial position of the Raspberry Pi Camera (beneath the waterproof casing)



Figure 4.6: Position of Raspberry Pi Camera when moved to the tip of the dome



Figure 4.7: Sky image with lesser reflection observed

4.5 Waterproofing for the Circuit and Enclosure for the Raspberry Pi Camera

The prototype has been successfully tested outdoors, and the waterproofing of the enclosure has proven to be effective in preventing water from entering the casing during rainy conditions. To ensure that rainwater does not enter the casing through the hole where the camera extends out, plasticine was used to seal the transparent dome chosen as the enclosure for the Raspberry Pi camera, while a Weatherproof PVC IP66 junction box was selected as the waterproof casing for the circuit.. Figure 4.8 shows an image of the All-Sky Imager prototype after a rainstorm.

To further prevent water droplets from accumulating on the surface of the dome, the Radical Rain Coat solution was sprayed on its surface. Figure 4.9 shows the sky image captured while water droplets were accumulating on the surface of the dome. After the application of the Radical Rain Coat solution, the sky image captured during a rainy day which is shown in Figure 4.10 is notably clearer compared to Figure 4.9.

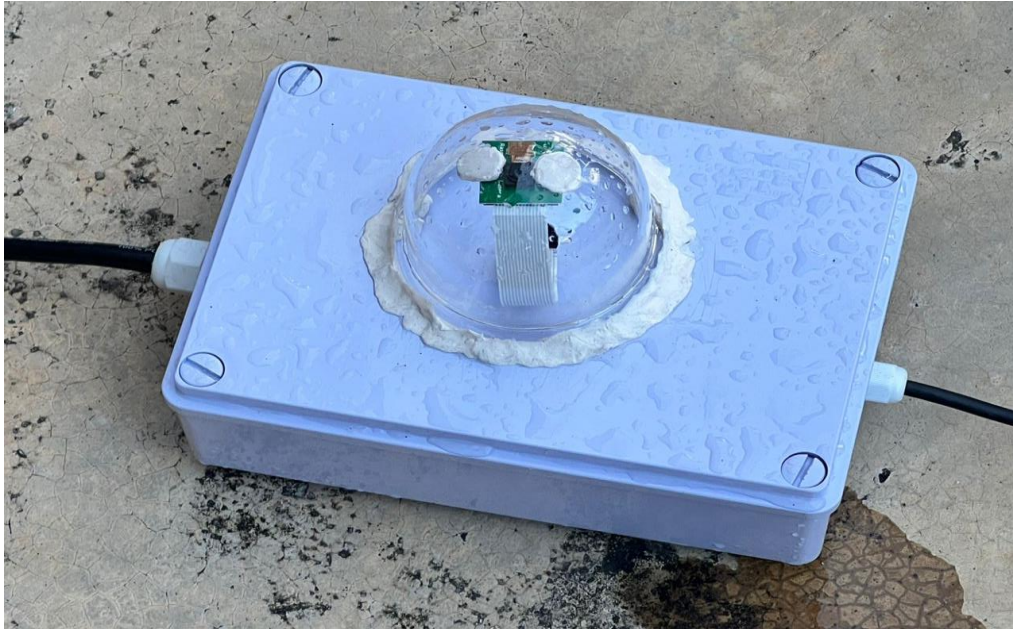


Figure 4.8: Prototype of the All-Sky Imager

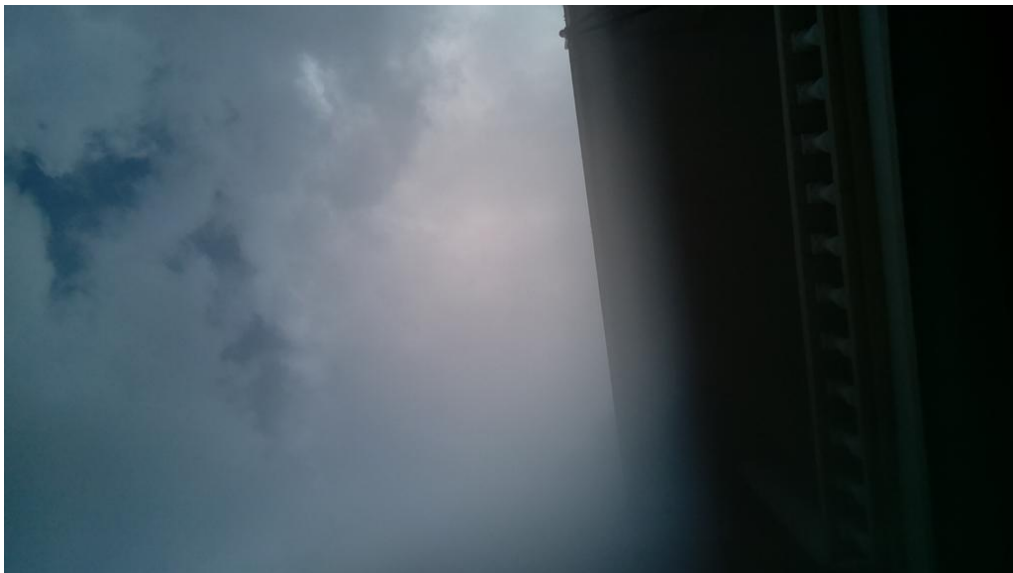


Figure 4.9: Sky image captured when water droplets accumulate on the surface of the dome

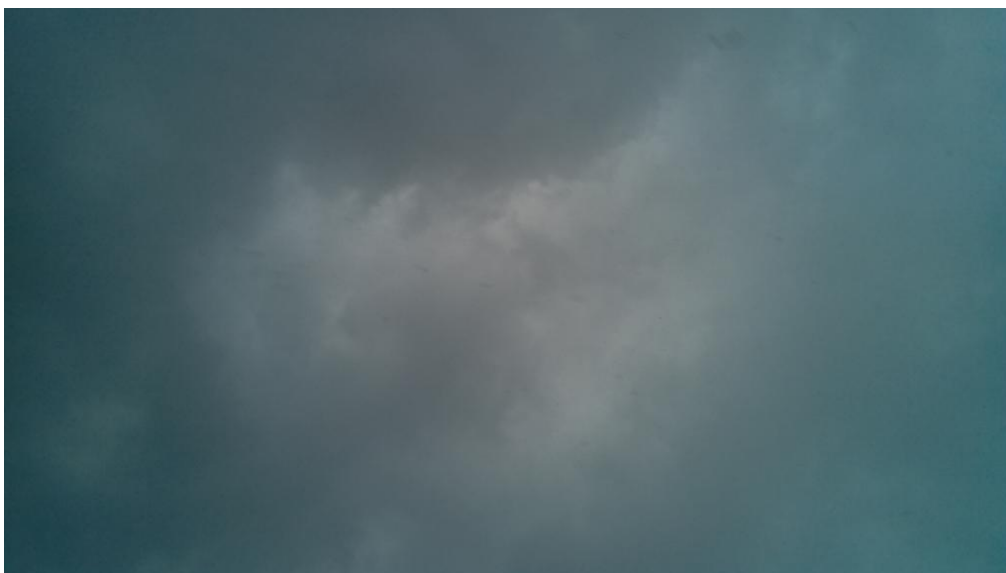


Figure 4.10: Sky image captured after the Rain-proof Radical Coat Solution is applied during rainy day

4.6 Summary on the Process to Capture a Sky Image with least Overposure and Reflection Observed

The process to capture a sky image with least overexposure and reflection involved tuning the default settings of the Raspberry Pi camera, including setting the ISO to 100 and exposure compensation to -25. However, this resulted in visible pictures with many reflections. To solve the reflection issue, the position of the camera was moved to the tip of the dome. This helped in capturing sky images with minimal exposure and reflection observed. Additionally, the use of a Radical Rain Coat solution helped to prevent water droplets from accumulating on the surface of the dome, resulting in clear sky images even during rainy weather conditions.

4.7 Data Logging and Image Uploading with Google Drive

The use of data logging with Google Drive has become an increasingly popular method for storing and accessing data remotely. To access Google Drive, a few parameters are required, including the Google Drive folder ID, Google Drive credentials created using a client ID and client secret, and a functioning refresh token.

One important aspect of using data logging with Google Drive is the need to ensure that the refresh token is up to date and functioning properly. If

the refresh token expires, it will need to be regenerated through the Google Developers platform in order to regain access to Google Drive. Once access to Google Drive has been established, the system can be used to upload data in real time. For example, sky images can be captured at regular intervals and uploaded to Google Drive, with the file names containing important information such as the date, time, and solar irradiance value. Figure 4.11 and Figure 4.12 shows the excel files and sky images that is uploaded to Google Drive.

Overall, data logging with Google Drive provides a reliable and secure way to store and access data remotely, with the ability to upload data in real time and ensure that it is easily accessible from anywhere with an internet connection. To ensure that data can be accessed easily when needed, it is necessary to maintain the refresh tokens in an active state.

To ensure a smooth data collection process, it's essential to estimate the storage requirements of specific tasks, such as capturing sky images every 5 minutes from 7am to 7pm daily, using the free 20 GB storage capacity provided by Google Drive. By performing this estimation, one can better manage storage usage and ensure adequate space for data over time. Assuming each sky image captured is about 1.3 MB in size to calculate the number of images captured per day, we can first calculate the total number of minutes in a 12-hour period:

$$12 \text{ hours} \times \frac{60 \text{ minutes}}{\text{hour}} = 720 \text{ minutes} \quad (4.1)$$

Dividing 720 minutes by the 5-minute interval between captures:

$$\frac{720 \text{ minutes}}{5 \text{ minutes per capture}} = 144 \text{ captures per day} \quad (4.2)$$

To get the usage of the Google Drive storage per day, the number of captures per day is multiplied by the estimated file size per capture and added with the estimated file size of an Excel file of 0.25 MB which containing the solar irradiance data:

$$144 \text{ captures per day} \times 1.3 \text{ MB per capture} = 187.2 \text{ MB per day} \quad (4.3)$$

$$187.2 \text{ MB per day} + 0.25 \text{ MB per day} = 187.45 \text{ MB per day} \quad (4.4)$$

So, the estimated daily storage usage for capturing sky images every 5 minutes from 7 am to 7 pm and uploading an excel file at 7.30pm would be approximately 187.45 MB. Dividing the total storage capacity of 20 GB (20480 MB) by the daily storage usage:

$$\frac{20480 \text{ MB}}{187.45 \text{ MB}} = 109.17 \text{ days} \quad (4.5)$$

Therefore, the 20 GB free Google Drive storage would last approximately 109.17 days, or about 3 months and 19 days if sky images are captured and uploaded every 5 minutes from 7 am to 7 pm, and an excel file is uploaded at 7.30 pm every day.

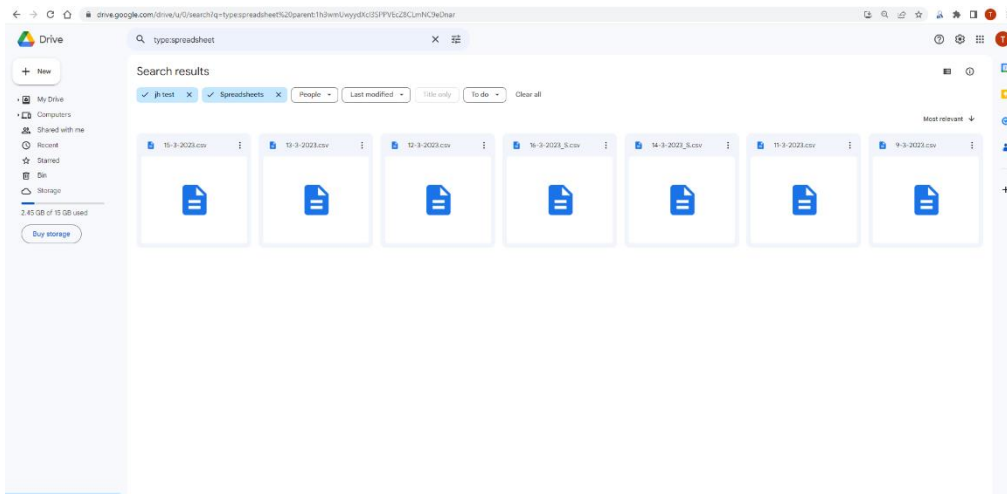


Figure 4.11: Excel Files containing Solar Irradiance Data that is uploaded to Google Drive everyday at 7.30 pm

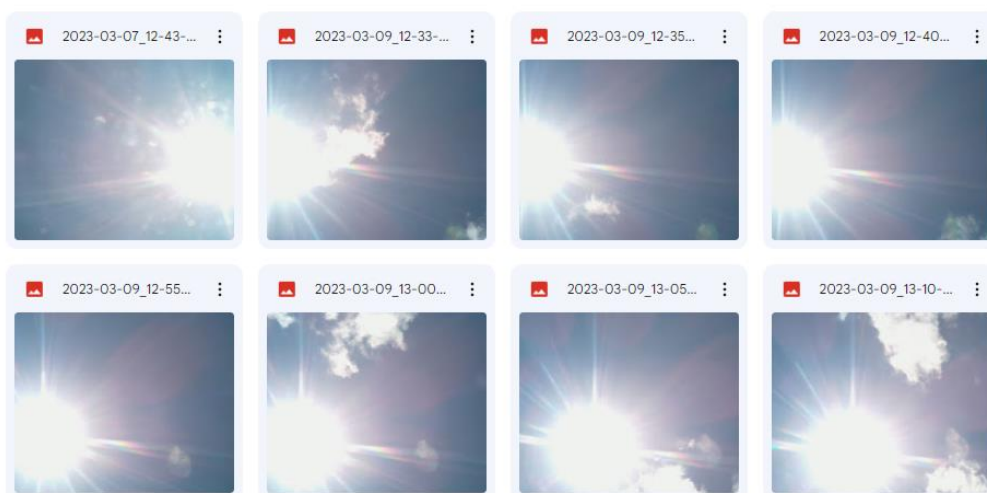


Figure 4.12: Images uploaded each 5 minutes from 7am to 7pm to Google Drive

4.8 Sky Images Captured under Different Sky Conditions

As per the observation of the sky images captured during the study, it is evident that the prototype is able to function under different weather conditions. The images captured during sunny, cloudy and rainy conditions provide insights into the effectiveness of the prototype in capturing the sky images. The sky images captured under different weather conditions (sunny, cloudy, and rainy) are presented below. Figure 4.13 to Figure 4.15 depict the sky images captured under sunny conditions, while Figure 4.16 to Figure 4.18 show the sky images captured under cloudy conditions. Lastly, Figure 4.19 to Figure 4.21 depict the sky images captured under rainy conditions.

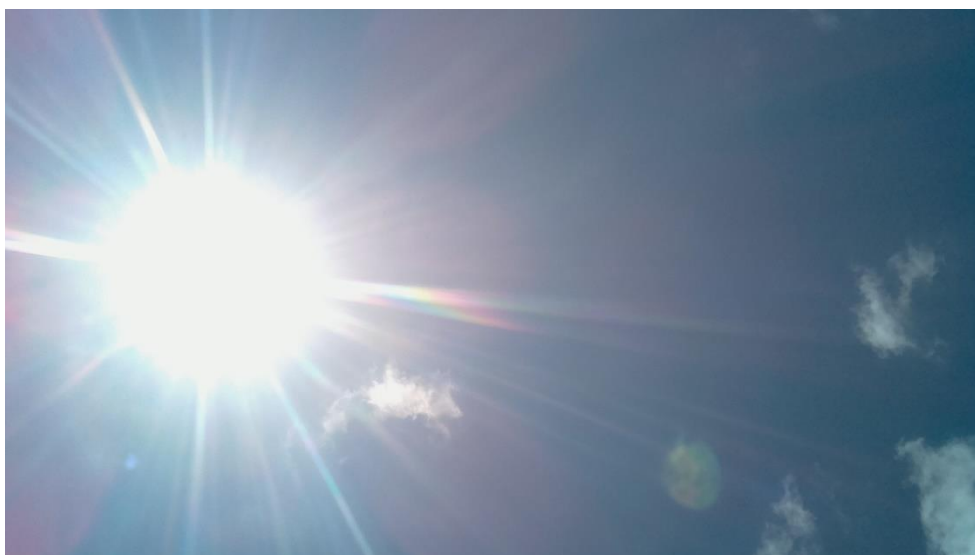


Figure 4.13: Sky Image Captured on 2023-03-09_12:35



Figure 4.14: Sky Image Captured on 2023-04-19_14:20



Figure 4.15: Sky Image Captured on 2023-04-22_14:25



Figure 4.16: Sky Image Captured on 2023-03-11_16:40



Figure 4.17: Sky Image Captured on 2023-04-20_17:20



Figure 4.18: Sky Image Captured on 2023-04-21_11:00

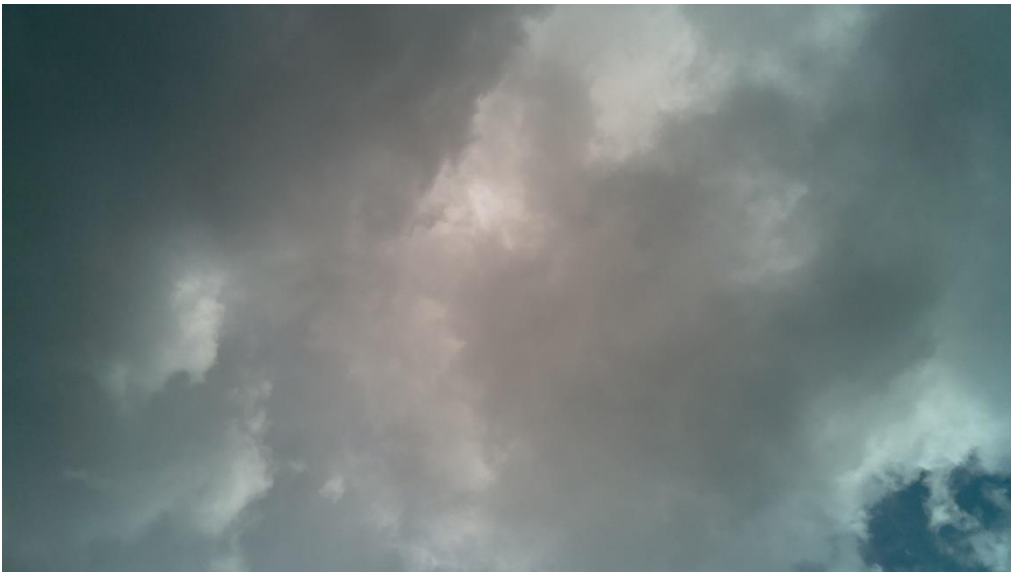


Figure 4.19: Sky Image Captured on 2023-03-09_17:30



Figure 4.20: Sky Image Captured on 2023-03-09_17:35

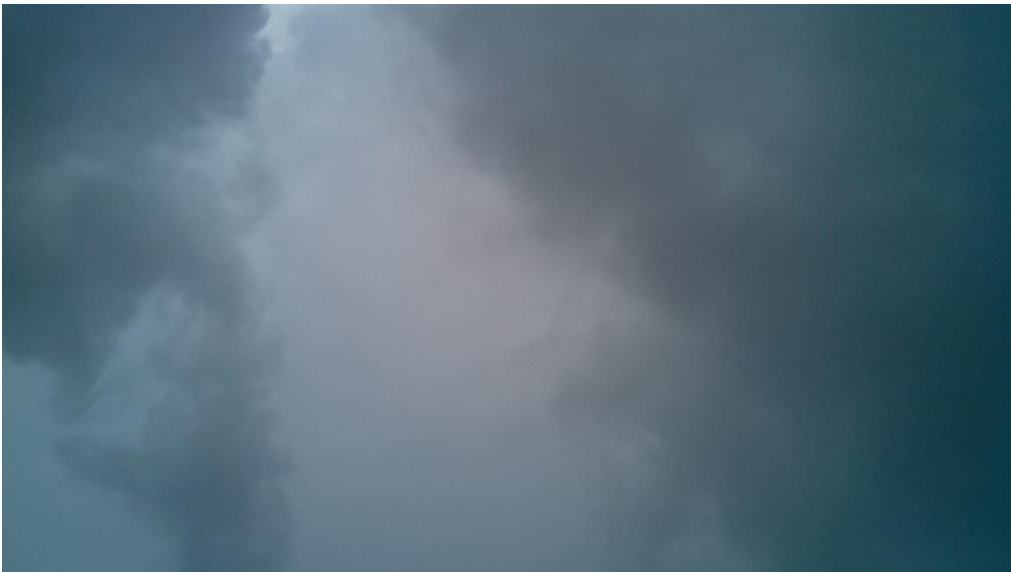


Figure 4.21: Sky Image Captured on 2023-04-29_15:00

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The prototype successfully automated the process of capturing and uploading sky images to Google Drive every 5 minutes and logging solar irradiance data every 10 seconds from 7am to 7 pm every day. Additionally, the Excel file containing the solar irradiance data will be uploaded to Google Drive every day at 7:30 pm. This automation significantly reduced manual labor and ensured the accurate capture and storage of data without any need for manual intervention. The prototype relied on a strong Wi-Fi connection for uploading the data to Google Drive, and a backup code was implemented to prevent data loss in case of a weak connection.

Initially, the sky images captured had an overexposure issue which was resolved by tuning the default settings of the Raspberry Pi camera to an ISO of 100 and an exposure compensation of -25. However, this resulted in visible pictures with many reflections. To address this issue, the Raspberry Pi camera was repositioned to the tip of the dome, resulting in sky images with the least exposure and reflections observed. Lastly, the three objectives of this project had been met which are building a data logger to log solar irradiance data using Raspberry Pi 3B+ as a microprocessor, capture the sky images using a Raspberry Pi camera module and upload the collected solar irradiance data which stored in excel files and sky images to Google Drive effectively

5.2 Recommendations for future work

For future improvements, consider purchasing a camera with a higher field of view to replace the current one. This would enable capturing a wider area of the sky, providing more comprehensive data. Additionally, exploring alternative methods for water-proofing the prototype, such as using silicone or specialized waterproofing materials to seal the dome instead of using plasticine, may be beneficial to increase the durability of the device in harsh weather conditions.

To further optimize the device's performance, placing it at a higher position could help reduce the shading effect caused by nearby. It's also recommended to test different locations to ensure a stronger Wi-Fi connection, which is crucial for the automated uploading of data to Google Drive.

REFERENCES

- Albadi, M., 2019. Solar PV Power Intermittency and Its Impacts on Power Systems – An Overview. *The Journal of Engineering Research [TJER]*, 16(2), p.142.
- Al-Lahham, A. et al., 2020. Sky imager-based forecast of solar irradiance using machine learning. *Electronics (Switzerland)*, 9(10), pp.1–14.
- Crisosto, C., Hofmann, M., Mubarak, R. and Seckmeyer, G., 2018. One-hour prediction of the global solar irradiance from all-sky images using artificial neural networks. *Energies*, 11(11).
- Fa, T., Xie, W., Wang, Y. and Xia, Y., 2019. Development of an all-sky imaging system for cloud cover assessment. *Applied Optics*, 58(20), p.5516.
- Fondriest Environmental, I., 2014, *Fundamentals of Environmental Measurements* [Online]. Available at: <https://www.fondriest.com/environmental-measurements/parameters/weather/photosynthetically-active-radiation/> [Accessed: 7 September 2022].
- Halabi, L.M., Mekhilef, S. and Hossain, M., 2018. Performance evaluation of hybrid adaptive neuro-fuzzy inference system models for predicting monthly global solar radiation. *Applied Energy*, 213, pp.247–261.
- Lannoye, E., Flynn, D. and O'Malley, M., 2011. The role of power system flexibility in generation planning. *IEEE Power and Energy Society General Meeting*. 2011, pp. 1-6.
- Lave, M., Quiroz, J., Reno, M.J. and Broderick, R.J., 2017. High temporal resolution load variability compared to PV variability. *2017 IEEE 44th Photovoltaic Specialist Conference, PVSC 2017*, pp.1–6.
- Learning about Electronics, 2018, *How to Connect an MP3002 ADC Chip to a Raspberry Pi* [Online]. Available at: <http://www.learningaboutelectronics.com/Articles/MCP3002-analog-to-digital-converter-ADC-to-Raspberry-Pi.php> [Accessed: 7 September 2022].
- Lima, F.J.L. et al., 2016. Forecast for surface solar irradiance at the Brazilian Northeastern region using NWP model and artificial neural networks. *Renewable Energy*, 87, pp.807–818.
- Malik, H. and Garg, S., 2019. Long-Term Solar Irradiance Forecast Using Artificial Neural Network: Application for Performance Prediction of Indian Cities. In: pp. 285–293.

Monjoly, S., André, M., Calif, R. and Soubdhan, T., 2017. Hourly forecasting of global solar radiation based on multiscale decomposition methods: A hybrid approach. *Energy*, 119, pp.288–298.

Mouheba, M., Hamidatb, A. and Loukarfic, L., 2012. Impact of PV compensation in improving the voltage drop in electrical networks LV. *Energy Procedia*. 2012 Elsevier BV, pp. 751–761.

Perez, R. et al., 2016. Spatial and Temporal Variability of Solar Energy. *Foundations and Trends® in Renewable Energy*, 1(1), pp.1–44.

PiMyLifeUp, 2022, *Raspberry Pi Temperature Sensor using the DS18B20* [Online]. Available at: <https://pimylifeup.com/raspberry-pi-temperature-sensor/> [Accessed: 7 September 2022].

Raspberry Pi Foundation, *Getting Started With Camera Module* [Online]. Available at: <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera> [Accessed: 25 April 2023].

Raspberry Tips, *How to install heat sinks on a Raspberry Pi 3 B+ ?* [Online]. Available at: <https://raspberrytips.com/install-heat-sinks-raspberry-pi/#:~:text=Heat%20sinks%20should%20be%20installed,back%20of%20the%20Raspberry%20Pi>. [Accessed: 26 April 2023].

Royer, J.C., Wilhelm, V.E., Teixeira Junior, L.A. and Franco, E.M.C., 2016. SHORT-TERM SOLAR RADIATION FORECASTING BY USING AN ITERATIVE COMBINATION OF WAVELET ARTIFICIAL NEURAL NETWORKS. *Independent Journal of Management & Production*, 7(1).

SEDA MALAYSIA, 2021, *NEM 3.0 Delivering A Greener Future* [Online]. Available at: <https://www.seda.gov.my/reportal/nem/> [Accessed: 7 September 2022].

Wang, F. et al., 2020. A minutely solar irradiance forecasting method based on real-time sky image-irradiance mapping model. *Energy Conversion and Management*, 220.

Watson, D., 2018, *Introduction to Raspberry Pi 3B+* [Online]. Available at: <https://www.theengineeringprojects.com/2018/07/introduction-to-raspberry-pi-3-b-plus.html> [Accessed: 7 September 2022].

APPENDICES

Appendix A: writetoexcel.py

```
import os
import geocoder
from datetime import datetime
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
import time
import board
import busio
from w1thermsensor import W1ThermSensor

i2c=busio.I2C(board.SCL, board.SDA)
# Initialize ADC1115
ads = ADS.ADS1115(i2c)

def read_soz03():
    ads.gain=2/3
    chan0=AnalogIn(ads, ADS.P0)
    sensor=W1ThermSensor()
    temp=sensor.get_temperature()
    voltage = float(chan0.voltage)
    voltage=round(voltage,4)

    if voltage <= 0.004:
        voltage=0
    else:
        pass
    # Calculate solar irradiance
    solar_irradiance = (voltage*240)
    solar_irradiance=round(solar_irradiance,4)
```

```
print(voltage,temp,solar_irradiance)

f.write(str(now)+",")
f.write(str(temp)+","+str(voltage)+","+str(solar_irradiance)+",")
#print('Temp (Cel): {}'.format(temp))
#print('Voltage (mV): {}'.format(volt))
#print('Irradiance : {}'.format(irr))

while True:

    now = datetime.now()
    year = str(now.year)
    month = str(now.month)
    day = str(now.day)
    hour = str(now.hour)

    exists_year = os.path.isdir('/home/pi/log/'+year)
    if exists_year:
        pass
    else:
        os.chdir('/home/pi/log/')
        os.mkdir(year)

    exists_month = os.path.isdir('/home/pi/log/'+year+'/'+month)
    if exists_month:
        pass
    else:
        os.chdir('/home/pi/log/'+year+')
        os.mkdir(month)
```

```

file = (day+"-"+month+"-"+year)
exists = os.path.isfile('/home/pi/log/'+year+'/'+month+'/'+
+'_S'+'.csv')
if exists:
    pass
else:
    f = open ('/home/pi/log/'+year+'/'+ month +'/' + file +'_S'+'.csv','a+')
    f.write(",","\n")
    f.write("Time"+", "+"Temp"+", "+"Voltage (V)"+"+", "+"Irradiance
(W/m2)"+"+",")
    f.close

now = time.strftime("%H:%M:%S")
sec = int(time.strftime("%S"))
current_time = datetime.now().time()
if datetime.strptime("04:00", "%H:%M").time() <= current_time <=
datetime.strptime("19:00:10", "%H:%M:%S").time():
    # Run your code here

if (sec% 10 == 0):
    f= open ('/home/pi/log/'+year+'/'+month+'/'+ file +'_S'+'.csv','a+')
    f.write(",","\n")
    read_soz03()
    f.close()
    time.sleep(9)
else:
    print("Not within time range")
    break

```


Appendix B: uploadexcel.py

```

# Import libraries
import os
from datetime import datetime
from googleapiclient.discovery import build
from google.oauth2.credentials import Credentials
from googleapiclient.http import MediaFileUpload

# Get current date information
now = datetime.now()
year = str(now.year)
month = str(now.month)
day = str(now.day)
hour = str(now.hour)
file = (day+"-"+month+"-"+year)
# Replace with your Google Drive folder ID
folder_id = '1h3wmUwyydXcI3SPPVEcZ8CLmNC9eDnar'
file_name = file + ".csv"
file_path = '/home/pi/log/'+year+'/'+month+'/'+ file + '_S'+'.csv'

def upload_to_drive(file_path, file_name):
    # Authenticate with Google Drive API
    # Create credentials object using client ID and client secret
    creds = Credentials.from_authorized_user_info(info={
        "refresh_token": "1//04LvPCbjEP2GHCgYIARAAGAQSNwF-
L9IrPnvaECoYSYGyXUtYPxj06gR0YkbBEz4nn86nYyOr4EFPYNI-
rEDgXisc7cLdpTcmfjw",
        "client_id": "307749992917-
tp29t2s5lhkua9nk8jblodhhjapnlpne.apps.googleusercontent.com",
        "client_secret": "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I",
        "token_uri": "https://oauth2.googleapis.com/token"
    })
    service = build('drive', 'v3', credentials=creds)

```

```
# Upload the file to Google Drive
file_metadata = {'name': file_name, 'parents': [folder_id]}
media = MediaFileUpload(file_path,
                        mimetype='text/csv')
file = service.files().create(body=file_metadata, media_body=media,
                             fields='id').execute()
print(f'File ID: {file["id"]}')

if __name__ == '__main__':
    while True:
        current_time = datetime.now().time()
        if datetime.strptime("05:00", "%H:%M").time() <= current_time
<= datetime.strptime("20:00", "%H:%M").time():
            print("Within time range.")
            upload_to_drive(file_path, file_name)
            break
        else:
            break
```

Appendix C: imagecapturinganduploading.py

```
import sys
import picamera
import pytz
import requests
import time
import os
import geocoder

from datetime import datetime
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
import board
import busio

from google.oauth2.credentials import Credentials
from googleapiclient.discovery import build
from googleapiclient.http import MediaFileUpload

i2c=busio.I2C(board.SCL, board.SDA)
# Initialize ADC1115
ads = ADS.ADS1115(i2c)

def read_soz03():
    ads.gain=2/3
    chan0=AnalogIn(ads, ADS.P0)

    solar_irradiance=chan0.voltage*240

    print(chan0.value,solar_irradiance)
```

```

#print('Temp (Cel): {}'.format(temp))
#print('Voltage (mV): {}'.format(volt))
#print('Irradiance : {}'.format(irr))
return solar_irradiance

# Replace with your Google Drive folder ID
folder_id = '1h3wmUwydXcI3SPPVEcZ8CLmNC9eDnar'

def upload_to_drive(file_path, file_name):
    # Authenticate with Google Drive API
    # Create credentials object using client ID and client secret
    creds = Credentials.from_authorized_user_info(info={
        "refresh_token": "1//04LvPCbjEP2GHCgYIARAAGAQSNwF-
L9IrPnvaECoYSYGyXUtYPxj06gR0YkbBEz4nn86nYyOr4EFPYNI-
rEDgXisc7cLdpTcmfjw",
        "client_id": "307749992917-
tp29t2s5lhkua9nk8jblodhhjapnlpne.apps.googleusercontent.com",
        "client_secret": "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",
        "token_uri": "https://oauth2.googleapis.com/token"
    })
    service = build('drive', 'v3', credentials=creds)

    # Upload the file to Google Drive
    file_metadata = {'name': file_name, 'parents': [folder_id]}
    media = MediaFileUpload(file_path,
                             mimetype='image/jpeg')
    file = service.files().create(body=file_metadata, media_body=media,
                                  fields='id').execute()
    print(f'File ID: {file["id"]}')

```

```
def capture_and_upload(solar_irradiance):
    # Set timezone to Kuala Lumpur
    tz = pytz.timezone('Asia/Kuala_Lumpur')
    now = datetime.now(tz)
    solar_irradiance_name=str(solar_irradiance)+'wm^(-2)'

    # Name the image file using current date and time
    file_name          =          now.strftime('%Y-%m-%d_%H-%M-%S'+
'+solar_irradiance_name+'.jpg')
    file_path = '/tmp/' + file_name

    # Capture an image using Raspberry Pi camera
    with picamera.PiCamera() as camera:
        # Set the camera resolution to 1920x1080
        camera.resolution = (1920, 1080)

        # Set the exposure compensation to -6 to reduce overexposure
        camera.exposure_compensation = -25

        # Set the ISO to 100 to reduce sensitivity to light
        camera.iso = 100

        # Set the white balance to auto
        camera.awb_mode = 'auto'

        # Wait for 5 seconds to allow the camera to adjust to the lighting conditions
        time.sleep(5)

        camera.led=False
        camera.capture(file_path)

    # Upload the image to Google Drive
```

```
upload_to_drive(file_path, file_name)

if __name__ == '__main__':
    while True:
        current_time = datetime.now().time()
        if datetime.strptime("04:00", "%H:%M").time() <= current_time <=
datetime.strptime("19:05", "%H:%M").time():
            print("Within time range.")
            #Run your code here
            solar_irradiance=read_soz03()
            capture_and_upload(solar_irradiance)
            break

        else:
            print("Not within time range.")
            # Stop running your code here
            break
    #time.sleep(300) # Sleep for 60 seconds before checking again
```