DESIGN AND CONSTRUCTION OF COST SAVING SOLAR IRRADIANCE DATA COLLECTION SYSTEM

SOI SHENG LEONG

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

Faculty of Engineering and Green Technology
Universiti Tunku Abdul Rahman

August 2019

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	:	SOI SHENG LEONG
ID No.	:	16AGB05778
Date	:	___________________________

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DESIGN AND CONSTRUCTION OF COST SAVING SOLAR IRRADIANCE DATA COLLECTION SYSTEM"** was prepared by **SOI SHENG LEONG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons) Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _________________________

Supervisor : Dr. Yew Tiong Keat

Date : _________________________

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, Dr Yew Tiong Keat and moderator, Mr Lee Yu Jen for their invaluable advice, guidance and enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement.

Last but not least, I would like to thank the lab assistants for their help and allowing me to conduct experiment and testing using the equipment available in the laboratory.

DESIGN AND CONSTRUCTION OF COST SAVING SOLAR IRRADIANCE DATA COLLECTION SYSTEM

ABSTRACT

Photovoltaic system is one common alternative in reducing usage of fossil fuel to generate electricity. Photovoltaic system requires low cost of maintenance and is less location-dependent. However, the cost to install a photovoltaic system is relatively high. Furthermore, not all locations are suitable to build a photovoltaic system. Therefore, solar irradiance data is crucial in determining feasibility to install a photovoltaic system. In practical, solar irradiance data is collected using data logger and cannot be collected remotely. To tackle these problems, a cost saving irradiance data collection system which is able to perform solar tracking mechanism and IoT data collection, is designed and constructed. Arduino UNO was implemented as the core in performing solar tracking mechanism, while ESP32 was implemented as the core in performing IoT data collection system. In this project, LDRs were calibrated in order to achieve high precision in tracking the Sun. High precision current-to-voltage circuity for both pyrheliometer and pyranometer were constructed in order to achieve accurate current reading in ESP32, which will indirectly help in achieving high precision pyrheliometer and pyranometer. Low cost pyrheliometer and pyranometer were designed and constructed in this project. Extra features were included in this project. One of the features includes the initialization of solar tracker when the solar tracker is powered on. Another extra feature included was the ability of Blynk user interface platform in displaying the current status of solar tracker besides being able to display real-time solar irradiance data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AADAST	azimuth-altitude dual-axis solar tracker
ADC	analog-to-digital converter
DAST	dual-axis solar tracker
DNI	Direct Normal Irradiance
GPIO	General-Purpose Input/Output
HTTP	HyperText Transfer Protocol
I/O	Input/Output
IoT	Internet of Things
LCOE`	levelized cost of energy generation
LDR	light dependent resistor
LED	light emitting diode
MJSC	Multi-Junction solar cell
NC	normally closed
NO	normally open
PV	photovoltaic
PWM	Pulse Width Modulation
PCB	printed circuit board
RSSI	Received Signal Strength Indicator
SAST	single-axis solar tracker
SSS	static solar system
SPDT	Single Pole Double Throw
TTDAST	tip-tilt dual-axis solar tracker

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background Study

Renewable energy sources are utilized in generation of electricity to reduce dependency upon fossil fuels. As compared to other renewable energy sources, PV system requires lower cost of maintenance and is less location-dependent. However, the cost to install a solar power plant is relatively high. Furthermore, not all locations are suitable and feasible to build photovoltaic system. Therefore, feasibility analysis is crucial to determine suitability for PV system installation.

Solar irradiance data is useful to determine feasibility of installing a PV system. Solar irradiance data can help to predict future energy yield, performance, efficiency and maintenance. These crucial factors will help in deciding whether to start up a PV system at a particular site.

1.2 Problem Statement

Many solar power stations perform data collection by using data logger. With data logger, data has to be collected at the power station. It is inconvenient to perform data collection as data cannot be collected remotely. Furthermore, data logger requires initial investment cost and is expensive for small tasks (RF Wireless World, 2012). In a worst case scenario, if the data logger malfunctions, the data will be lost and not recorded.

1.3 Aims and Objectives

The aims and objectives of the project are:

i) To construct a cost effective active dual-axis solar tracking system
ii) To calibrate the constructed solar irradiance sensor with the actual solar irradiance meter
iii) To design high precision amplifying circuit and current-to-voltage converter for the sensors
iv) To implement the application of IoT in collecting data from sensors remotely via Internet

1.4 Project Scope

The project basically consists of two major parts, which are solar tracking mechanism and IoT data collection system. For solar tracking mechanism, calibration of light sensor consisting four LDRs (light dependent resistors) was done to ensure accurate tracking of the solar tracker. Arduino software was implemented to enable solar tracking mechanism controlled by Arduino UNO. For IoT data collection system, ESP32 WiFi module was used to perform data transmission to Blynk apps and Google Spreadsheet using Arduino software. Construction of solar meters which are pyrheliometer and pyranometer, and calibration of ADC (analog-to-digital converter) module were done as well. Circuitry for both solar tracking mechanism and IoT data collection system were constructed and combined.

There are two main extra features included in the project. One of the features included is the initialization of solar tracker. The position of y-axis or vertical axis of the solar tracker will be initialized when it is powered on. Another extra feature done is that enable current status of solar tracker to be displayed in Blynk apps besides only displaying real-time data. The current status will indicate current illumination level (sunny or cloudy) and indicate whether the solar tracking is currently tracking or stop.

Lastly, cable management is crucial to ensure the project is presentable. The project was done on 20th August 2019.

1.5 Summary of the Entire Project

The entire project is summarized in five different chapters. The first chapter of the project briefly introduces the importance of solar irradiance data in determining the feasibility to install a PV system, and states the problem faced in the current solar irradiance data collection system. In this chapter, aims, objectives and scope of project are illustrated as well.

Literature review is done in the second chapter of the project. Areas of studies related to the project are summarized. Online articles and journals from different authors were reviewed. In chapter three of the project, procedures done in designing and constructing the cost saving solar irradiance data collection system were summarized. The block diagram and algorithm of the entire project were illustrated in this chapter. The procedures in enabling accurate solar tracking system and IoT data collection system were explained in details.

The fourth chapter of the project analyses the results obtained and discusses on improvement done. In this chapter, problems encountered and solutions for respective problems were discussed. Some precautions were mentioned in this chapter as well. Lastly, the fifth chapter concludes the entire project done and suggests future improvements that can be done.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter summarizes various areas of studies applied in the project. Studies and research done by authors from websites and journal are reviewed and used as references throughout the project.

2.2 Solar Irradiance

Solar radiation is where the Sun emits radiant energy to the Earth. Solar radiant energy reaches the Earth in different ways. Direct solar radiation is the sunlight travelled directly to the surface of the Earth. Diffuse solar radiation is the sunlight being scattered or diffused, but still able to reach the Earth's surface. Reflected radiation is the sunlight reflected back to the atmosphere by the surface of the Earth.

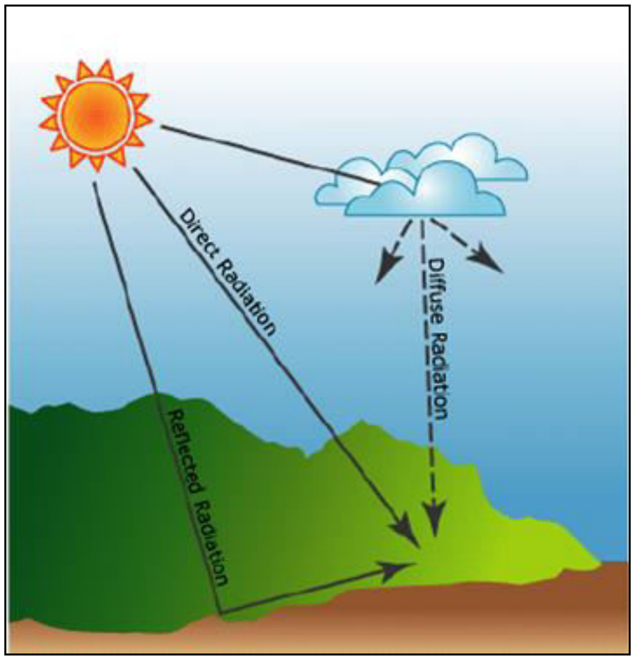


Figure 2.1: Different Ways on How Solar Radiant Energy Reaches the Earth

Solar irradiance is the measurement of solar radiation, which is the total amount of solar power received per unit area of a surface from the Sun with the unit of measurement, W/m^2.

There are two types of sensors to measure solar irradiance, which are pyranometer and pyrheliometer. Pynarometer is used to measure the total global irradiance which includes direct and diffuse radiation. Pyrheliometer is used to measure direct radiation to its sensing element, which is known as direct normal irradiance (DNI).

Figure 2.2: Pynarometer

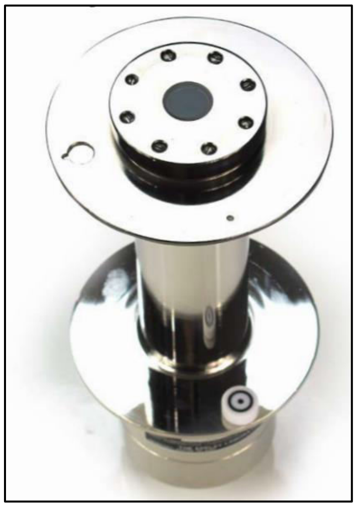


Figure 2.3: Pyrheliometer

The design and mechanism of both pyrheliometer and pyranometer are further discussed in the subsections respectively.

2.2.1 Pyranometer

Pyranometer is the solar meter implemented to measure the total irradiance which includes direct sunlight and diffused sunlight collected. The design of the standard pyranometer is shown in the next page.

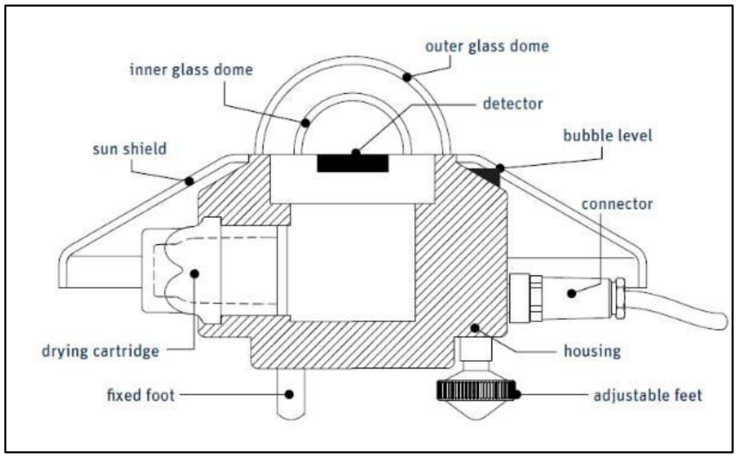


Figure 2.4: Design of the Standard Pyranometer (Omni Instruments, 2009)

Pyranometer has two glass domes, which are outer and inner glass domes. Both of the glass domes help to concentrate the sunlight within the field of view of 180° to the detector of the pyranometer. The connector will be connected to the detector, and the low voltage signal generated by the detector will need to be converted to its irradiance value according to the formula provided in Equation 2.1.

$$TI = V \times sensitivity \tag{2.1}$$

where

TI = total irradiance measured by pyranometer, W/m^2

V = voltage generated by the detector of pyranometer, mV

sensitivity = sensitivity of pyranometer, μV/(W.m^{-2})

A standard pyranometer weighs approximately 0.9kg and costs around £1,690.00 (RM 8600.00).

2.2.2 Pyrheliometer

Pyrheliometer is the solar meter which measures DNI which includes only direct sunlight collected. The design of the standard pyrheliometer is shown in the picture below.

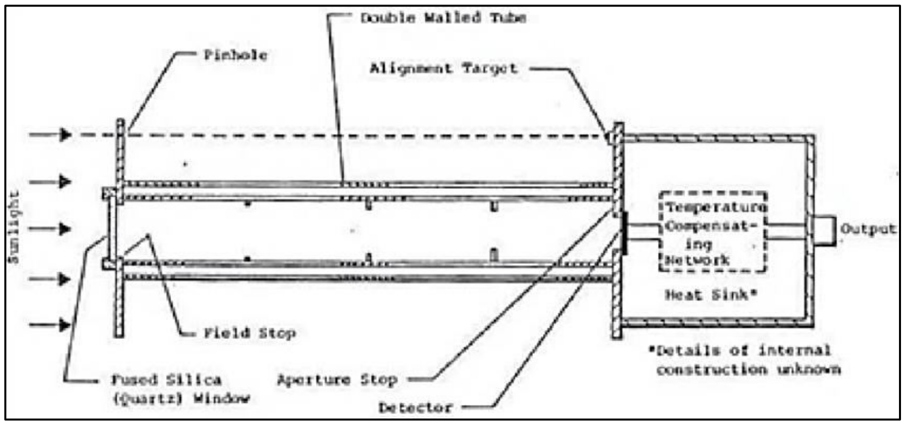

Double Walled Tube
Pinhole
Alignment Target
Sunlight
Temperature Compensat-ing Network
Output
Heat Sink*
Field Stop
*Details of internal construction unknown
Fused Silica (Quartz) Window
Aperture Stop
Detector

Figure 2.5: Design of the Standard Pyrheliometer

A standard pyrheliometer is designed with a detector covered with a hollow tube to accept only direct sunlight. The length of the hollow tube of a standard pyrheliometer is decided by three factors, which are the focal length of the quartz window, the width of the opening and the area of the detector. The inner layer of the tube is coated with a non-reflective layer. To determine whether the pyrheliometer is pointing to the Sun, the alignment target of the pyrheliometer is the indicator. If it points to the Sun, the sunlight will pass through the pinhole and a spot of light can be seen on the alignment target.

The field of view or acceptance angle of a standard pyrheliometer is 5° as shown in the picture in next page. In another words, the pyrheliometer has to point directly to the Sun due to its minimal acceptance angle (AmritaVirtual Lab, 2019). Pyrheliometer is usually attached to DAST for higher efficiency and accuracy in measuring DNI. DNI values are important because DNI is about 80 % of solar energy received by a PV system (EKO Instruments, 2017). DNI can be calculated according to the small voltage signal generated by the detector as shown in Equation 2.2.

$$DNI = V \times sensitivity \tag{2.2}$$

where

DNI = direct normal irradiance measured by pyrheliometer, W/m^2

V = voltage generated by the detector of pyrheliometer, mV

sensitivity = sensitivity of pyreliometer, μV/(W.m^{-2})

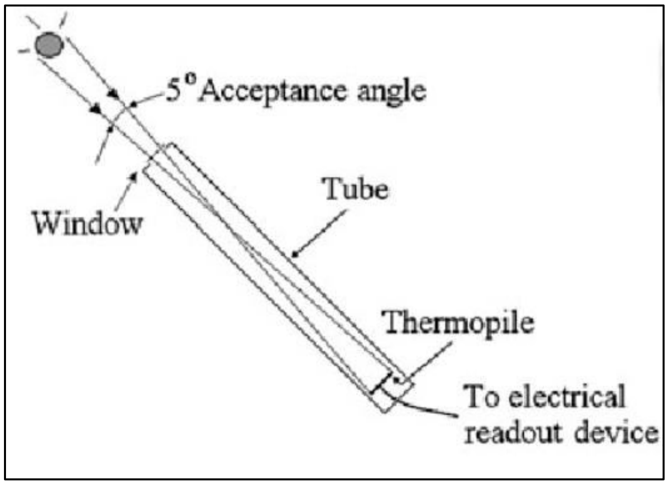


Figure 2.6: Acceptance Angle of Pyrheliometer

A standard pyrheliometer weighs approximately 1kg and costs around RM10,000.

2.3 Types of Solar Irradiance Collection System

There are three major method to collect solar energy, which are static solar system (SSS), single-axis solar tracker (SAST) and dual-axis solar tracker (DAST).

2.3.1 Static Solar System (SSS)

SSS is also known as fixed-mounted solar system. For the design of SSS, the solar panel is placed with one fixed angle of orientation. According to Landau (2017), there are specific formulas for angle of orientation of SSS based on latitude of the location. If the latitude of the location is below 25°, the angle of orientation for SSS can be calculated according to Equation 2.3. If the latitude of the location is in between 25° and 50°, the angle of orientation can be calculated according to Equation 2.4. For the latitude of location above 50°, the calculation of angle of orientation is complicated.

$$\textit{orientation angle} = \textit{latitude} \times 0.87 \tag{2.3}$$

where

orientation angle = angle of orientation of SSS, °

latitude = latitude of the location, °

$$\textit{orientation angle} = \textit{latitude} \times 0.76 + 3.1° \tag{2.4}$$

where

orientation angle = angle of orientation of SSS, °

latitude = latitude of the location, °

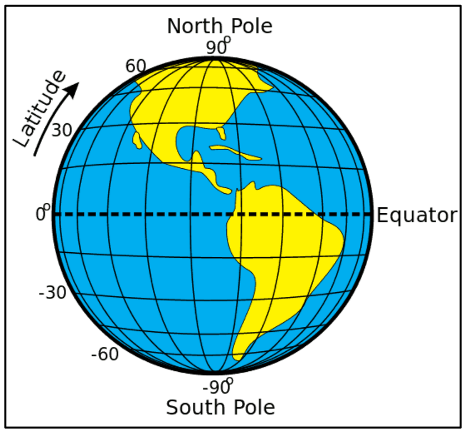


Figure 2.7: Latitude of the Earth

SSS is the cheapest method for solar energy collection compared to SAST and DAST. Besides, it is easy to install SSS, where this system can be placed on the roof or on the ground (Naked Solar Ltd, 2019). However, the drawback of using SSS is that the optimal angle of orientation has to be selected in order to optimize the efficiency in solar energy collection.

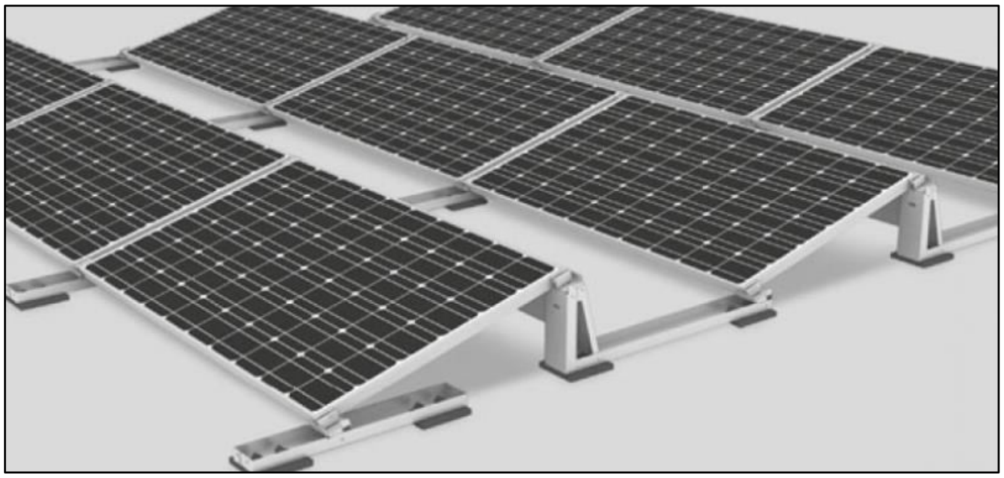


Figure 2.8: Static Solar System

2.3.2 Single-Axis Solar Tracker (SAST)

SAST is a device that move in one axis to track the Sun. SAST has more degree of freedom compared to SSS. There are four different configurations of SAST, which are horizontal SAST, vertical SAST, tilted SAST and polar-aligned SAST.

For the design of horizontal SAST, an actuator to rotate the solar panel is placed horizontal to the ground. In another words, the solar panel will tilt perpendicular to the ground for tracking purpose.

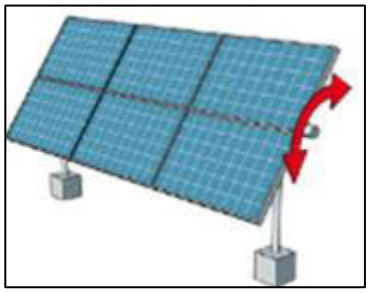


Figure 2.9: Horizontal Single-Axis Solar Tracker

For the design of vertical SAST, an actuator is placed vertical to the ground. In another words, the solar panel will move either clockwise or counterclockwise parallel to the ground. The solar panel will not be placed flat, but will be tilted at a fixed angle of orientation as shown in the picture below.

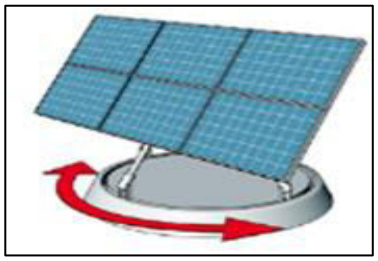


Figure 2.10: Vertical Single-Axis Solar Tracker

Tilted SAST is designed where the solar panel is tilted at a certain angle of elevation and rotates side by side as shown in the picture below.

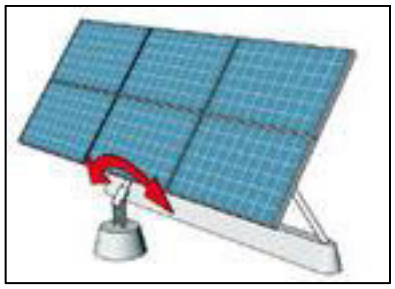


Figure 2.11: Tilted Single-Axis Solar Tracker

Polar-aligned SAST has the same mechanism as tilted SAST. The only difference is that the solar panel is tilted aligned to the polar star. The polar star is known as Polaris star, and can be identified by drawing a line through the outer bowl of the Big Dipper as shown in Figure 2.13 (SolarSystemQuick.com, 2010).

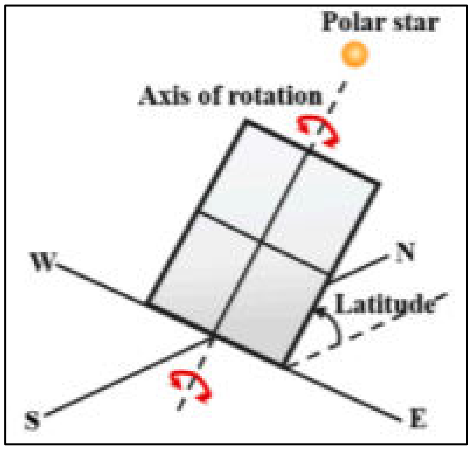


Figure 2.12: Polar-Aligned Single-Axis Solar Tracker

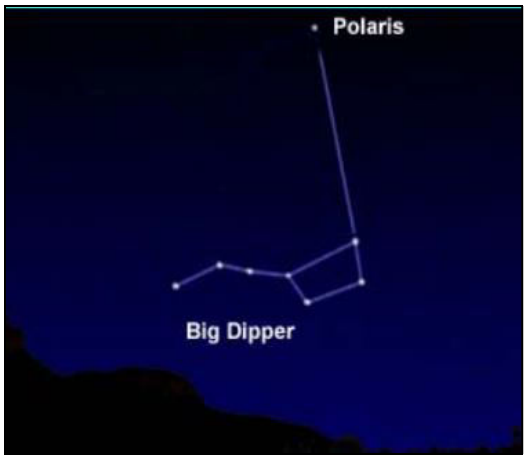


Figure 2.13: Identification of Polaris Star From Big Dipper (SolarSystemQuick.com, 2010)

2.3.3 Dual-Axis Solar Tracker (DAST)

DAST has more degree of freedom compared to both SSS and SAST, where it is able to perform both elevation and azimuth movement. Elevation is a movement of tilting upwards and downwards, while azimuth is a movement of rotating clockwise and counterclockwise parallel to the horizon.

There are three major types of DAST, which are passive solar tracker (PST), active solar tracker (AST) and open-loop solar tracker (OLST). All of them perform tracking mechanism automatically, but their design differs respectively.

For both PST and AST, they perform their own mechanism depending on the condition of the environment. Further explanation and comparison among these two trackers are discussed in Section 2.4.2.

Unlike PST and AST, OLST is preset with computer controlled algorithms to determine the location of the Sun (harshi1990, 2013). The computer controlled algorithms are done by performing calculation based on astronomical data or sun position algorithm.

2.4 Dual-Axis Solar Tracking System

DAST is implemented in collecting solar irradiance data. The reason of implementing this system and the type of the system implemented are further analyzed in the subsections respectively.

2.4.1 Reason to Implement Dual-Axis Solar Tracking System

The main reason of implementing DAST is that it performs better than SSS according to the research conducted by Lee and Rahim (2013).

According to the research done, it captures more solar energy daily throughout a month as shown in Figure 2.4. Furthermore, Lee and Rahim (2013) found that it has higher efficiency and generates more energy under both sunny and cloudy days as shown in Table 2.1. Therefore, it is concluded that DAST is the better option.

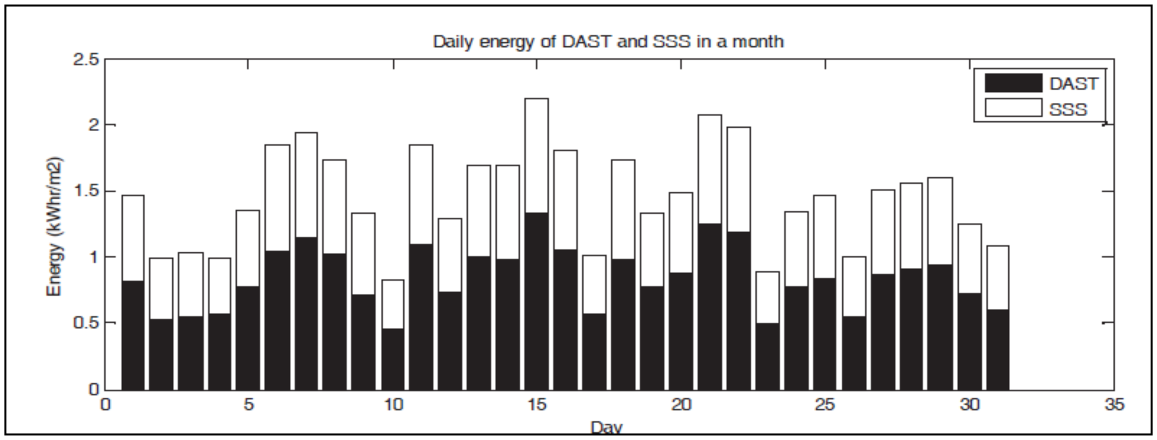


Figure 2.14: Daily Energy Captured by Dual-Axis Solar Tracker (DAST) and Static Solar System (SSS) in a Month (Lee and Rahim, 2013)

Table 2.1: The Efficiency (η) and Energy Gained by the Dual-Axis Solar Tracker (DAST) over Static Solar System (SSS) under Cloudy and Sunny Day (Lee and Rahim, 2013)

	Efficiency, η (%)	Energy Generated, (kW·hr/m^2)
Cloudy	26.91	0.108
Sunny	82.12	0.603

DAST is also a better choice compared to SAST. This is because it tracks the Sun in both horizontal and vertical direction, while SAST is limited to track either in horizontal or vertical direction depending on its configuration.

2.4.2 Type of Dual-Axis Solar Tracking System Implemented

There are two major methods implemented in autonomous solar tracking system depending on the conditions of the environment, which are active and passive tracking method. For active tracking method, external electronic components are required. The controller of the solar tracker controls the movement of the motors according to the light intensities detected by light sensors. On the other hand, passive tracking method uses low boiling point fluid in the canisters to move the solar tracker.

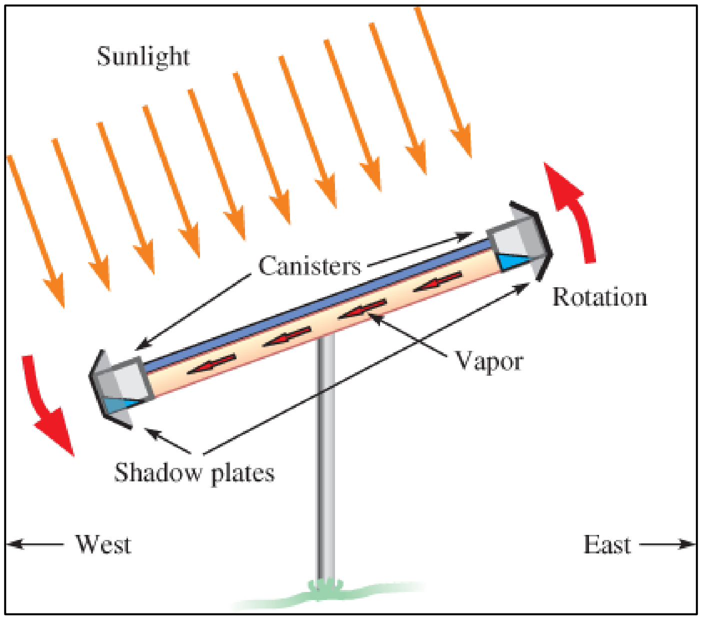


Figure 2.15: Solar Tracker Using Passive Tracking Method

Although passive solar tracking system requires lower cost, active tracking method is recommended because passive tracking method depends on ambient temperature, whereas active tracking method depends on light intensity. Due to dependency upon ambient temperature, solar tracker with passive tracking method rarely points directly to the Sun as the temperature will vary from day to day (harshi1990, 2013). Therefore, to ensure the solar tracker tracks the Sun accurately, active tracking system is recommended as it track according to light intensities instead of ambient temperature.

2.5 Solar Data Collection System

There are different designs for solar data collection system. However, they have a common weakness, which is data collection cannot be done remotely. Furthermore, the cost to design the system is relatively high for small task in performing data collection.

According to Koutroulis and Kalaitzakis (2003), there are two main designs in performing solar data collection system. One of the designs is a microcontroller-based system. In this design, the sensors will be connected to the ADC for the microcontroller to read the irradiance values from the sensors. The microcontroller will then update the values to the computer via RS-232 cable to perform data collection and storage.

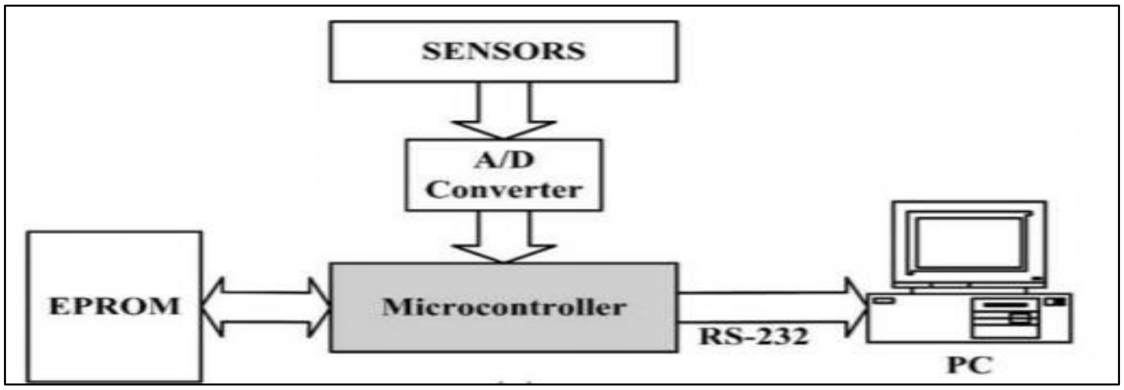


Figure 2.16: Solar Data Collection Using Microcontroller-Based System (Koutroulis and Kalaitzakis, 2003)

Another design according to Koutroulis and Kalaitzakis (2003) is by implementing data logging system. In this design, the data logger is implemented to store the data from the sensors, and the computer will collect the data from the data logger via RS-232 cable.

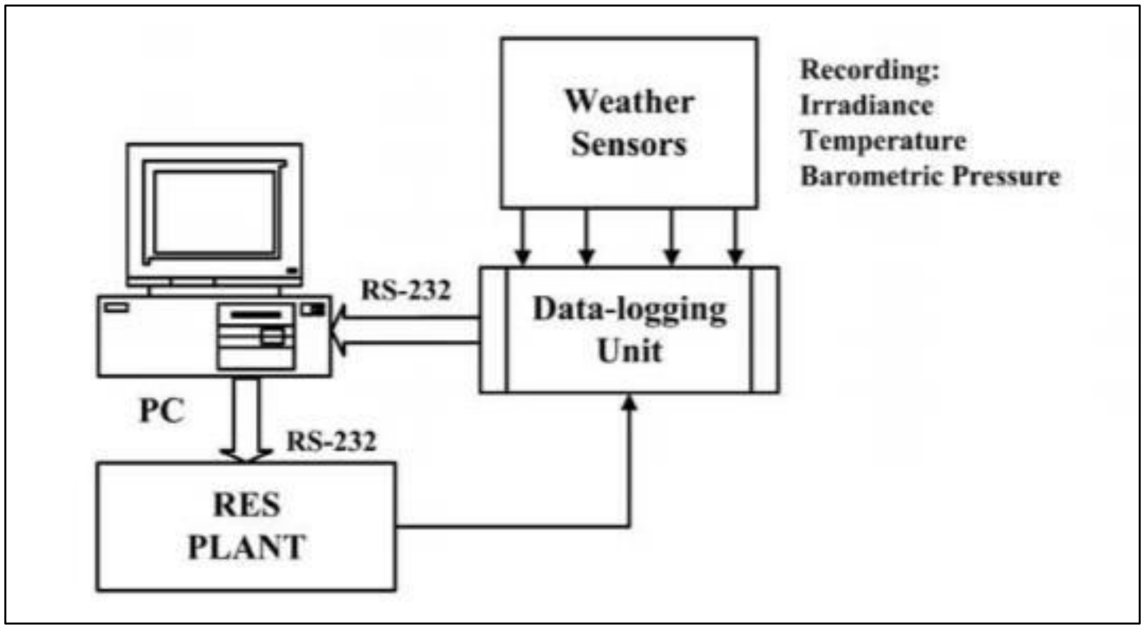


Figure 2.17: Solar Data Collection Using Data Logging System (Koutroulis and Kalaitzakis, 2003)

2.6 Internet of Things (IoT)

IoT is where devices are interconnected via the Internet. In the world of IoT, all connected devices can share and collect information remotely anytime and anywhere without the need of human intervention.

IoT is widely implemented in businesses and various fields of industries as it eases the process of data collection and analysis. There are four major layers of IoT on how the data of sensors or devices is transmitted and analyzed via the Internet.

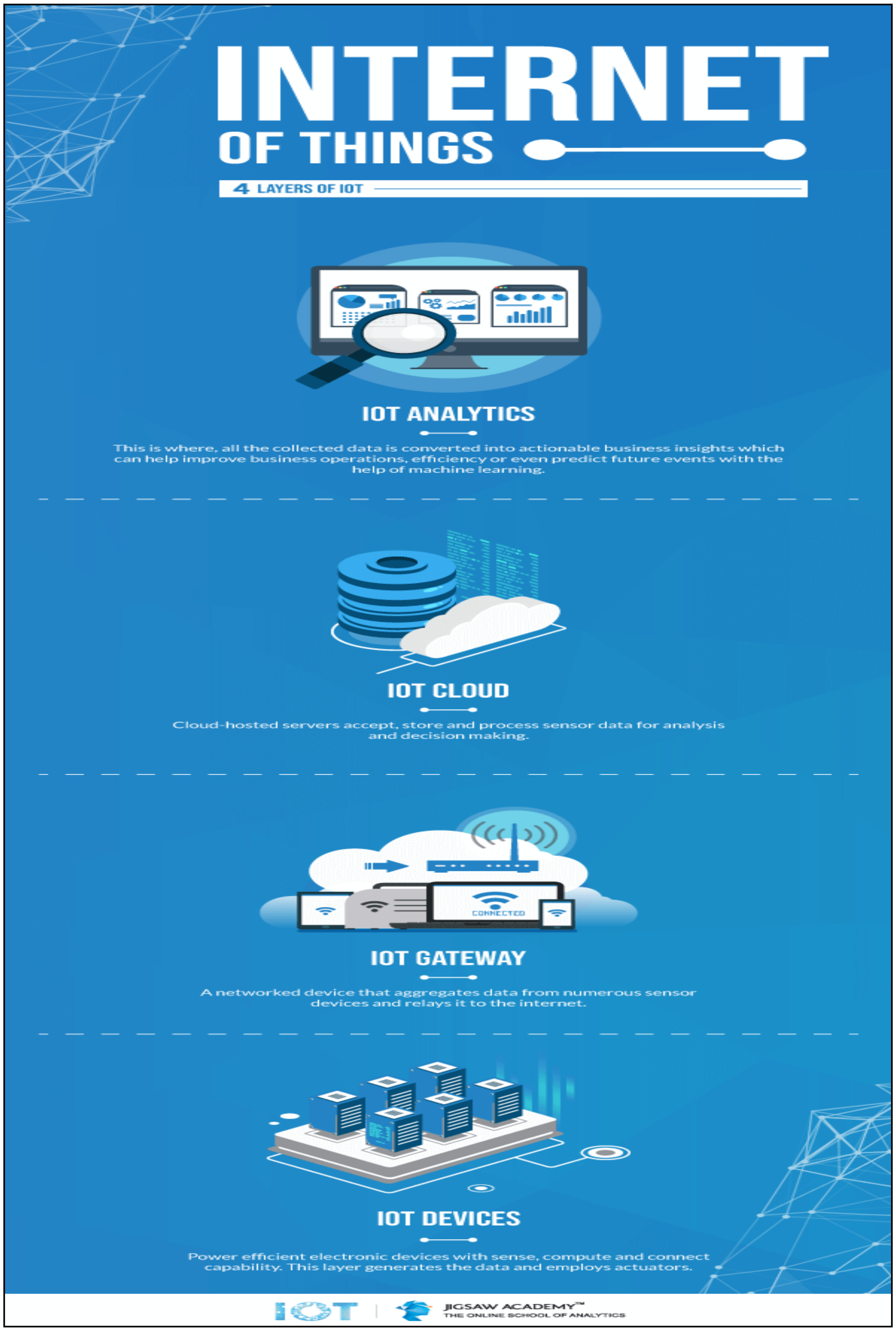

INTERNET
OF THINGS
4 LAYERS OF IOT
IOT ANALYTICS
This is where, all the collected data is converted into actionable business insights which can help improve business operations, efficiency or even predict future events with the help of machine learning.
IOT CLOUD
Cloud-hosted servers accept, store and process sensor data for analysis and decision making.
CONNECTED
IOT GATEWAY
A networked device that aggregates data from numerous sensor devices and relays it to the internet.
IOT DEVICES
Power efficient electronic devices with sense, compute and connect capability. This layer generates the data and employs actuators.
IOT
JIGSAW ACADEMY™
THE ONLINE SCHOOL OF ANALYTICS

Figure 2.18: Four Layers of IoT (Gupta, 2018)

The first layer of IoT consists of sensor-connected IoT devices (Gupta, 2018). In this layer, the sensors will detect the physical parameters such as temperature, humidity, etc., and convert the parameters into a signal which can be measured electrically. The IoT devices will help the sensors to send signal wirelessly to the gateway device as the sensors themselves cannot send data wirelessly.

In the world of IoT, data of the sensing devices are transmitted remotely for data collection and analysis. However, the devices themselves are unable to upload the data via Internet. Hence, the second layer of IoT which is the gateway device, can help to solve this problem. IoT gateway device can be a Wi-Fi router or any device that has Internet connection, and acts as a networking device connecting the sensing devices to the Cloud. In another words, IoT gateway device receives the data from sensing devices and then transmit the data to the Cloud in the second layer of IoT (Gupta, 2018).

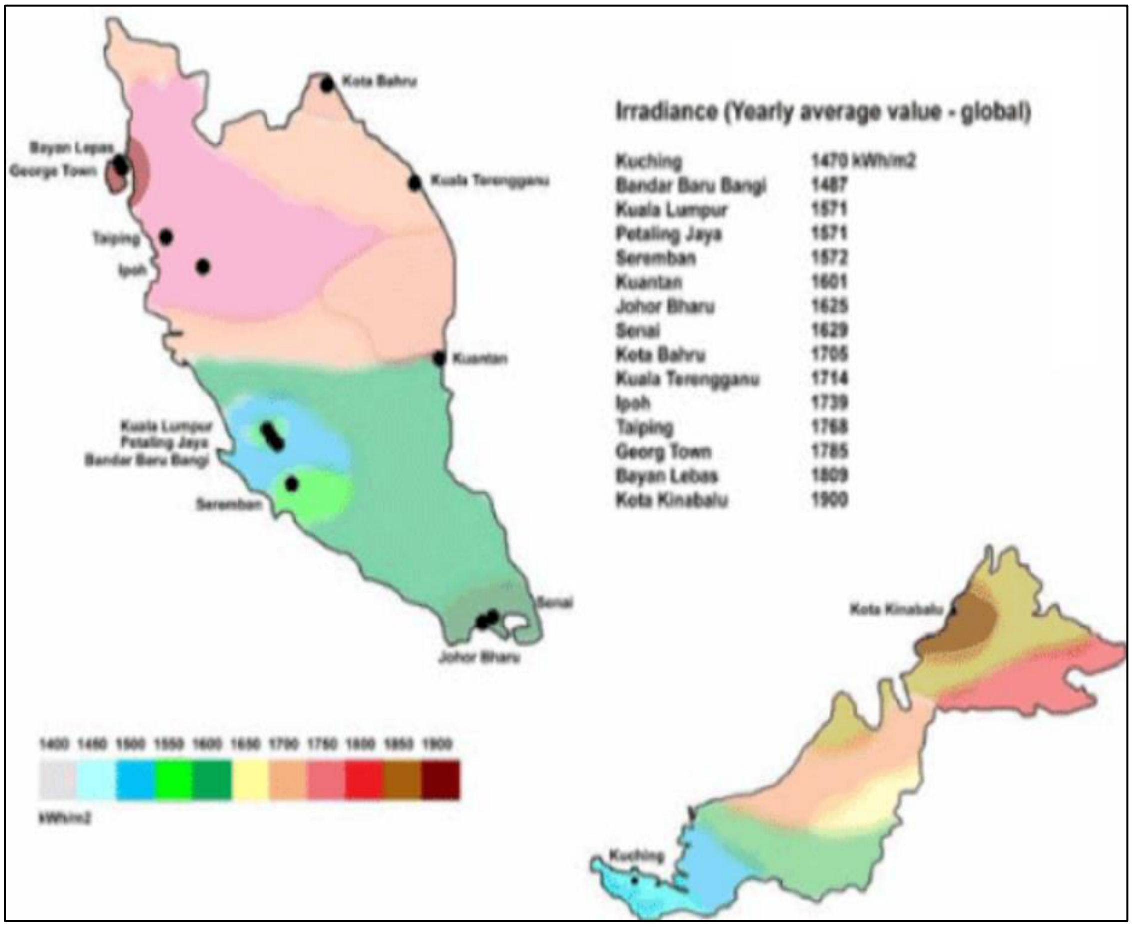
The third layer of IoT is the Cloud (Gupta, 2018). In this layer, the Cloud server accepts the data transmitted by the IoT gateway device. The Cloud server will store and process the data received for monitoring and analytic purposes.

The fourth layer of IoT is IoT analytic (Gupta, 2018). In this layer, the users will analyse and monitor the data of the sensing devices by accessing into IoT platform obtained from the Cloud server. The user can access the Cloud platform user interface using mobile phone or computer as long these devices are connected to the Internet.

2.7 Solar Analysis in Malaysia

Malaysia is located at the equatorial region of the Earth and has a hot tropical rainforest climate throughout a year. Basically, there are only sunny and raining season throughout a year in Malaysia due to its tropical climate.

According to Aziz, et al (2016), Malaysia is a country with high potential in installing a PV system as the average annual radiation in Malaysia is relatively high, which is around 1643 kWh/m^2.

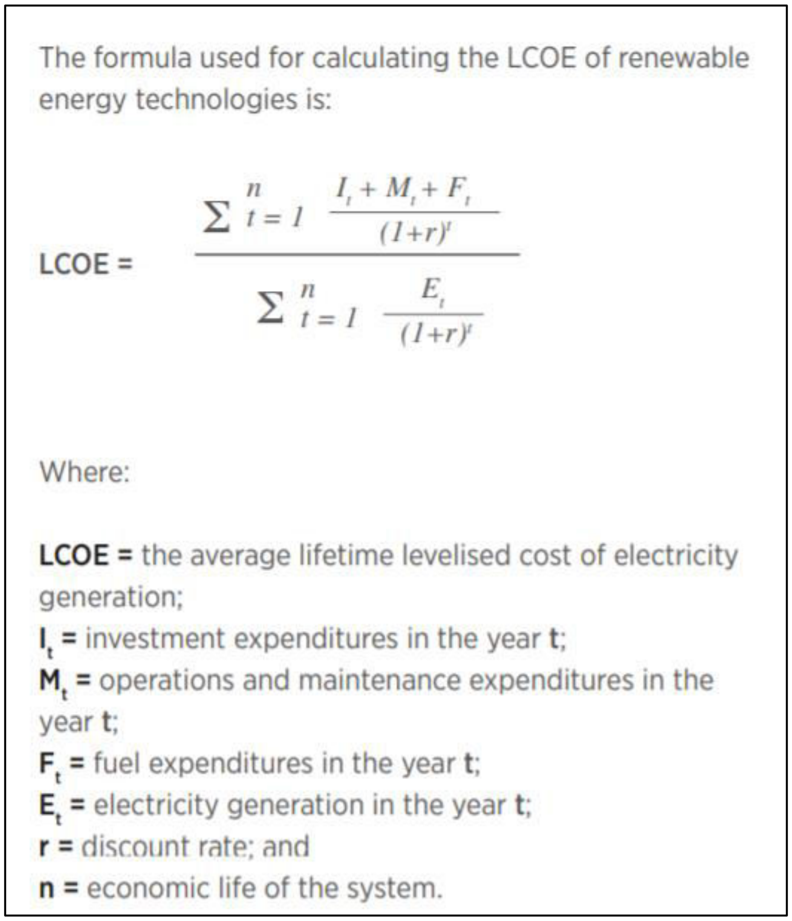


Irradiance (Yearly average value - global)

Kuching	1470 kWh/m2
Bandar Baru Bangi	1487
Kuala Lumpur	1571
Petaling Jaya	1571
Seremban	1572
Kuantan	1601
Johor Bharu	1625
Senai	1629
Kota Bahru	1705
Kuala Terengganu	1714
Ipoh	1739
Taiping	1768
Georg Town	1785
Bayan Lebas	1809
Kota Kinabalu	1900

Figure 2.19: Solar Radiation in Malaysia (Aziz, et al., 2016)

Although the average annual radiation is relatively high, feasibility study has to be done before the installation of PV system due to high cost of installation. Furthermore, the performance of commercialized solar panel will degrade over time (SunPower Corporation, 2019). To conduct feasibility study, solar irradiance data will be collected monthly or yearly, which will help in the calculation of levelised cost of electricity generation (LCOE). LCOE is crucial to determine the suitability to start a PV system at the particular site.



The formula used for calculating the LCOE of renewable energy technologies is:

$$\text{LCOE} = \frac{\sum_{t=1}^{n} \frac{I_t + M_t + F_t}{(1+r)^t}}{\sum_{t=1}^{n} \frac{E_t}{(1+r)^t}}$$

Where:

LCOE = the average lifetime levelised cost of electricity generation;
I_t = investment expenditures in the year **t**;
M_t = operations and maintenance expenditures in the year **t**;
F_t = fuel expenditures in the year **t**;
E_t = electricity generation in the year **t**;
r = discount rate; and
n = economic life of the system.

Figure 2.20: Calculation of LCOE of Renewable Energy Technologies (IRENA, 2012)

2.8 Conclusion

Based on the literature review done, Malaysia is a suitable to implement solar energy for generation of electricity. However, the cost to install a PV system is high and not every location is suitable to build a PV system. Therefore, solar irradiance data is needed to be collected to analyze the feasibility and suitability to install PV system at the particular site.

However, the major issue faced in the present solar irradiance data collection system is that the data cannot be collected remotely. Furthermore, the cost to set up the system is relatively high for small task in collecting irradiance data. To solve this issue, application of IoT will be implemented in this project. Besides its ability to save cost in setting up data collection system, data can be collected remotely in the world of IoT as long Internet network is provided.

Another issue observed from the literature review, the cost of solar irradiance meters are high. To solve this issue, both cost saving pyrheliometer and pyranometer will be constructed, and construction of these meters will be further discussed in Chapter 3.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter illustrates the design and mechanism of the solar irradiance data collection system. The components and software implemented for the project are briefly described in this chapter. Besides, budget and milestone of the project are shown in this chapter.

3.2 Design of Solar Irradiance Data Collection System

The solar irradiance data collection system constructed is implemented for both solar tracking purpose and remote data collection. The design architecture of the entire project and algorithms of both solar tracking mechanism and IoT data collection system are explained in the subsections respectively.

3.2.1 Design Architecture of Solar Irradiance Data Collection System

The main functions of the project are solar tracking mechanism and IoT data collection system. The block diagram is shown in the picture below.

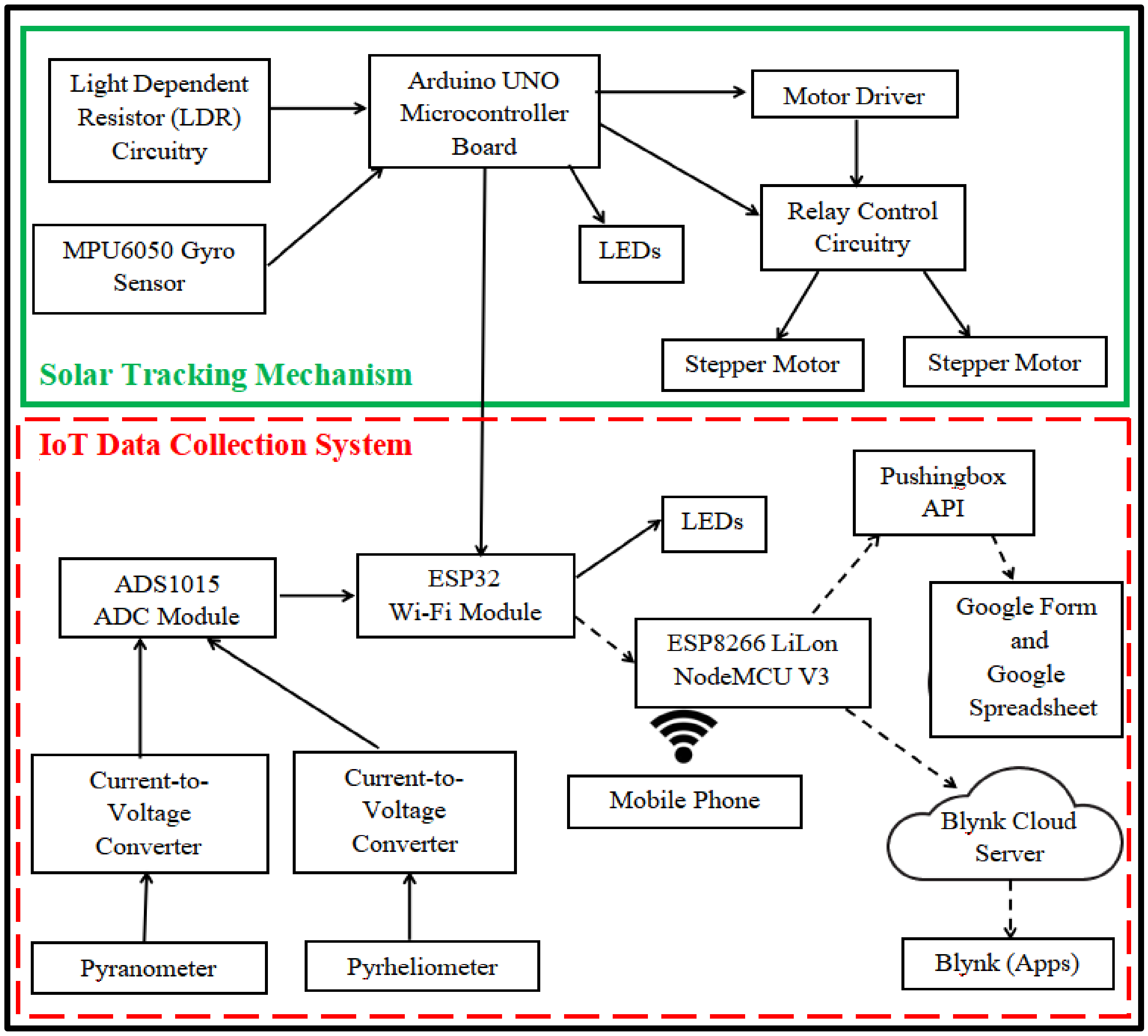


Figure 3.1: Block Diagram of Solar Irradiance Data Collection System

For solar tracking mechanism, Arduino UNO acts as the core to get inputs from both light dependent resistor (LDR) circuitry and MPU6050 gyro sensor and perform operations according the input signal received.

The LDR circuitry consists of four LDRs and each LDR receives the ambient light intensity respectively. Arduino UNO will compare all different light intensities from the LDR circuitry, and send signal to both motor driver and relay control circuitry to control the stepper motors in moving the solar tracker.

MPU6050 gyro sensor is a robust gyroscope module and is implemented to initialize the solar tracker to the desired angle of elevation when it is powered on. The gyro sensor is also used to limit the elevation angle to prevent the plate of the solar tracker from hitting the entire circuitry. The LEDs are implemented to indicate the current status of the solar tracker.

In IoT data collection system, ESP32 is implemented as the IoT device, while ESP8266 LiLon NodeMCU V3 is implemented as the IoT gateway device. Since ESP8266 LiLon NodeMCU V3 itself does not have Internet network, mobile phone is used to provide Internet hotspot to enable the ESP8266 module to transfer data from ESP32 to both Pushingbox API and Blynk Cloud server. Pushingbox API will upload the data to Google Form and Google Spreadsheet for IoT database purpose, while Blynk Cloud server will receive and show real-time data in mobile phone using Blynk apps.

Both pyrheliometer and pyranometer are constructed and used as sensors to detect solar irradiance, which are direct normal irradiance and global irradiance. Both of the sensors generate current signal according to the solar irradiance detected. Since ADS1015 ADC module is only able to detect voltage signal, therefore, current-to-voltage converters were needed in order to amplify and convert low current signals from both sensors to voltage signals to the ADC module. The LEDs were added to indicate Wi-Fi connection status and data transmission status.

ESP32 will calculate current signals according to voltage signals measured by the ADC module. The calculated current signals will then be converted to solar irradiance values for both sensors. In a meanwhile, ESP32 will receive the current status of solar tracker from Arduino UNO via UART serial communication. The solar irradiance values and current status of solar tracker will be data for the IoT data collection system. ESP32 will be connected to ESP8266 LiLon NodeMCU V3 wirelessly, and the data will then be uploaded through the Internet.

3.2.2 Algorithm of Solar Tracking Mechanism

The algorithm of solar tracking mechanism is shown in the pictures in next three pages. The coding uploaded to Arduino UNO as shown in Appendix N can be referred for more details on the exact operation of the mechanism.

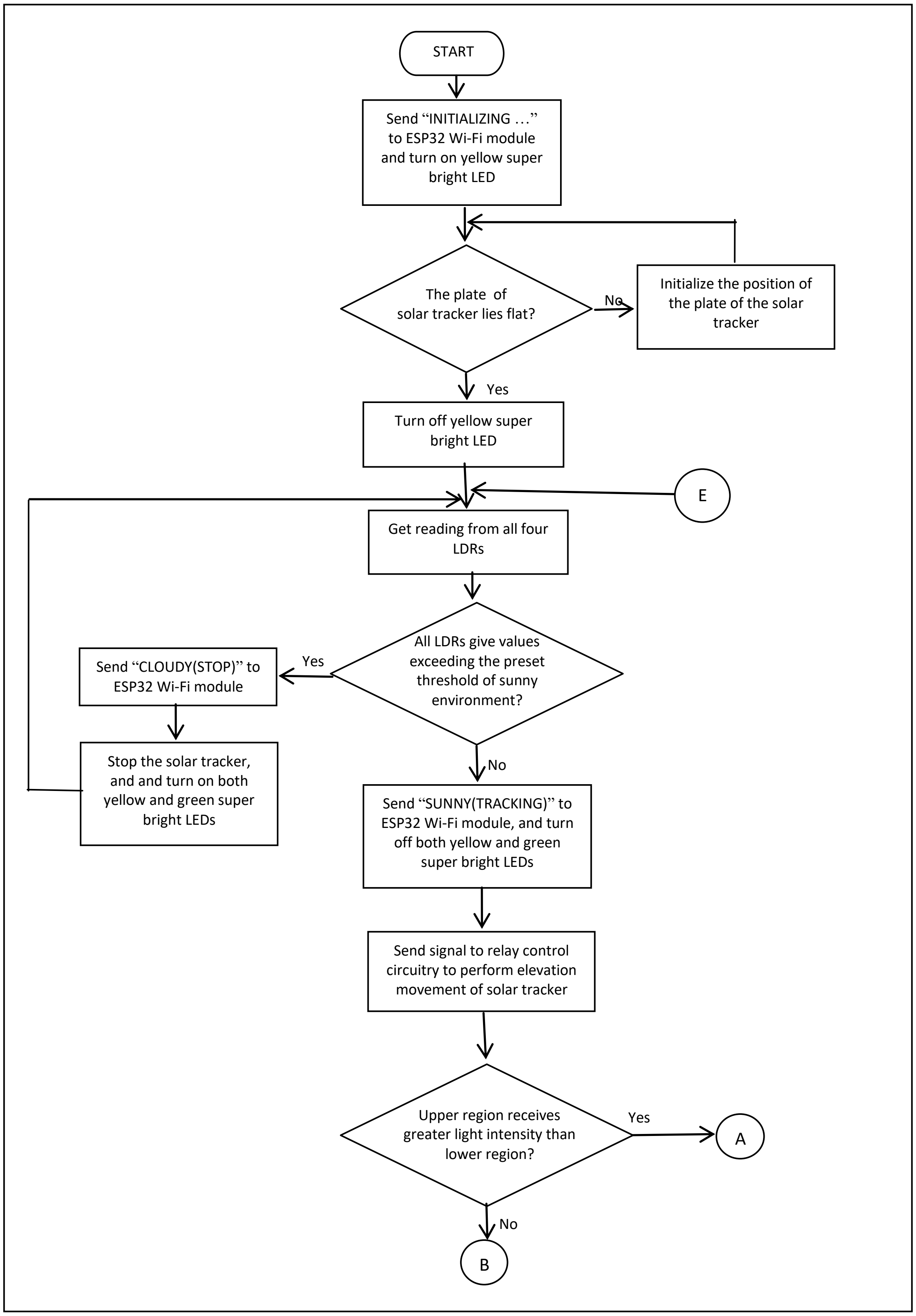

START
Send "INITIALIZING …" to ESP32 Wi-Fi module and turn on yellow super bright LED
The plate of solar tracker lies flat?
No
Initialize the position of the plate of the solar tracker
Yes
Turn off yellow super bright LED
E
Get reading from all four LDRs
All LDRs give values exceeding the preset threshold of sunny environment?
Yes
Send "CLOUDY(STOP)" to ESP32 Wi-Fi module
Stop the solar tracker, and and turn on both yellow and green super bright LEDs
No
Send "SUNNY(TRACKING)" to ESP32 Wi-Fi module, and turn off both yellow and green super bright LEDs
Send signal to relay control circuitry to perform elevation movement of solar tracker
Upper region receives greater light intensity than lower region?
Yes
A
No
B

Figure 3.2: Algorithm of Solar Tracking Mechanism

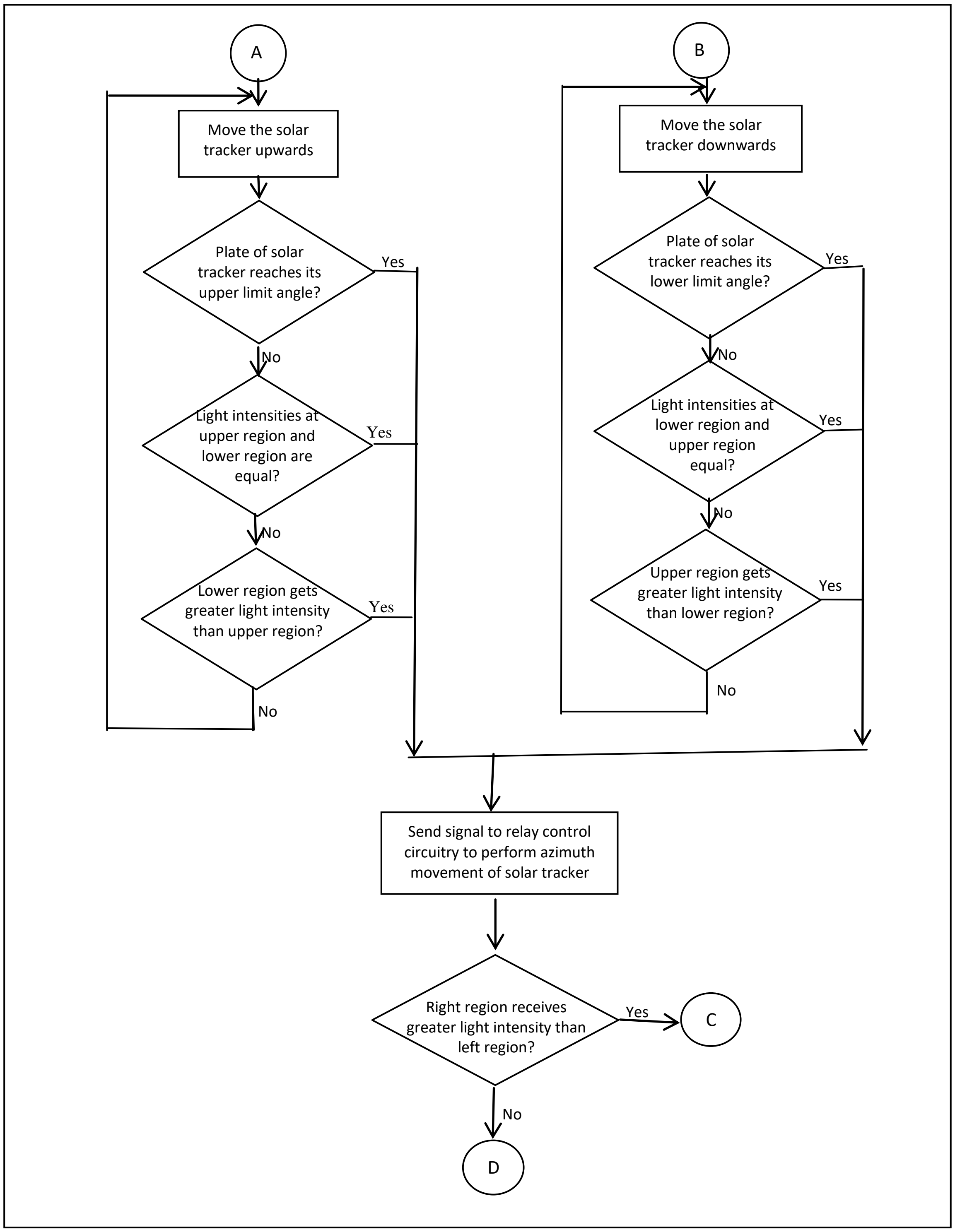

A
B
Move the solar tracker upwards
Plate of solar tracker reaches its upper limit angle?
Yes
No
Light intensities at upper region and lower region are equal?
Yes
No
Lower region gets greater light intensity than upper region?
Yes
No
Move the solar tracker downwards
Plate of solar tracker reaches its lower limit angle?
Yes
No
Light intensities at lower region and upper region equal?
Yes
No
Upper region gets greater light intensity than lower region?
Yes
No
Send signal to relay control circuitry to perform azimuth movement of solar tracker
Right region receives greater light intensity than left region?
Yes
C
No
D

Figure 3.3: Algorithm of Solar Tracking Mechanism (Continued)

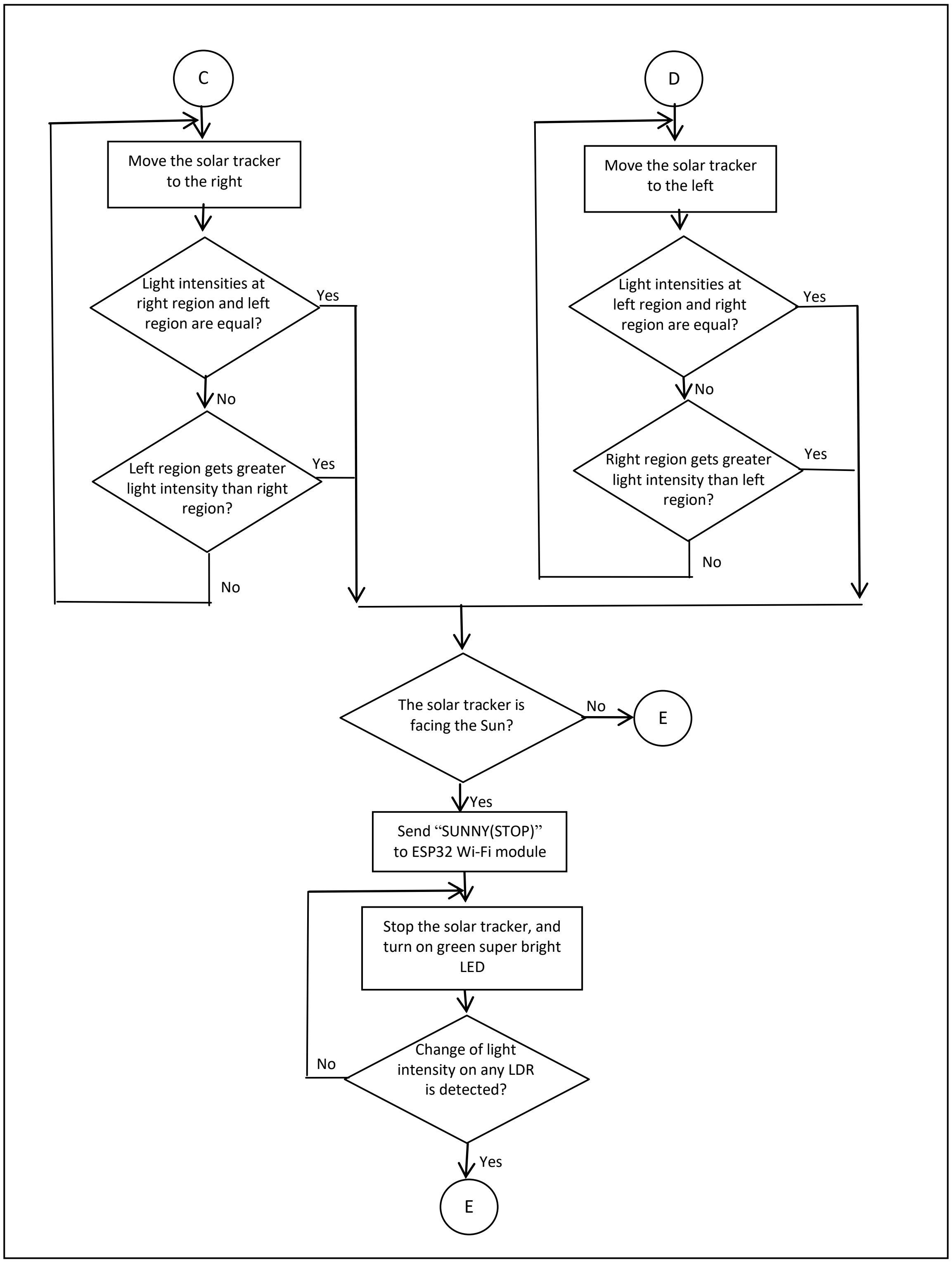

C
D
Move the solar tracker to the right
Light intensities at right region and left region are equal?
Yes
No
Left region gets greater light intensity than right region?
Yes
No
Move the solar tracker to the left
Light intensities at left region and right region are equal?
Yes
No
Right region gets greater light intensity than left region?
Yes
No
The solar tracker is facing the Sun?
No
E
Yes
Send "SUNNY(STOP)" to ESP32 Wi-Fi module
Stop the solar tracker, and turn on green super bright LED
Change of light intensity on any LDR is detected?
No
Yes
E

Figure 3.4: Algorithm of Solar Tracking Mechanism (Continued)

Arduino UNO is implemented as a controller to perform operations depending on the input signals from both LDR circuitry and MPU6050 gyro sensor. Yellow and green super bright LEDs are added as indicators for the current status of the solar tracker. Arduino UNO will update the current status of solar tracker to ESP32 via UART serial communication.

When the solar tracker is powered on, the plate of the solar tracker will initialize its position to lay parallel to the ground. MPU6050 gyro sensor will play its role to determine whether the plate of solar tracker is lying parallel to the ground by calculating the angle of elevation of the plate. When the solar tracker is initializing its position, Arduino UNO will update the current status of "INITIALIZING…" which indicates the solar tracker is currently initializing its position. In a meanwhile, yellow super bright LED will light up to indicate that the plate of solar tracker is currently initializing its position. Once the plate of solar tracker is lying parallel to the ground, the yellow super bright LED will turn off.

Next, all four LDRs in the LDR circuitry will start to measure the ambient light intensity detected respectively and send signal to Arduino UNO to perform comparison among light intensities measured by the LDRs. To determine whether the environment is cloudy or sunny, a threshold of sunny environment is preset into Arduino UNO. If light intensities from all four LDRs exceed the preset threshold, Arduino UNO will update the current status of "CLOUDY(STOP)", and both green and yellow super bright LEDs will light up to indicate cloudy environment. The solar tracker will stop if the environment is cloudy.

If the light intensities obtained do not exceed the preset threshold, the solar tracker will start to track the Sun. Arduino UNO will update current status of "SUNNY(TRACKING)" to indicate the environment is sunny and the solar tracker starts to track. Both yellow and green super bright LEDs will turn off when the solar tracker is tracking.

In this project, the solar tracker will perform its elevation movement followed by its azimuth movement. Firstly, Arduino UNO will turn on the relay control circuitry to enable the solar tracker to perform elevation movement which is to tilt upwards or downwards. LDRs at the upper region and lower region are then calculated in average respectively and compared. The solar tracker will tilt upwards if the upper region gets greater light intensity, and vice versa. Next, Arduino UNO will turn off the relay control circuitry to enable azimuth movement of solar tracker which is to move left or right. LDRs at the right region and left region are then calculated in average respectively and compared. The solar tracker will move to the right if the right region gets greater light intensity, and vice versa.

After performing both elevation and azimuth movement, Arduino UNO will check the values of LDRs to ensure that the solar tracker is facing the Sun. If it faces the Sun, ESP32 will receive the current status of “SUNNY(STOP)” from Arduino UNO to indicate the solar tracker stops and points towards the Sun. The green super bright LED will also light up for indication purpose. However, if it does not face the Sun, it will keep tracking until it faces the Sun. If there is any changes in the light intensity detected by any one of the LDRs, the green super bright LED will turn off to indicate the solar tracker starts to track.

3.2.3 Algorithm of IoT Data Collection System

The algorithm of IoT data collection system is shown in the pictures in next two pages. The coding uploaded to ESP32 can be referred for more details on the exact operation shown in Appendix O.

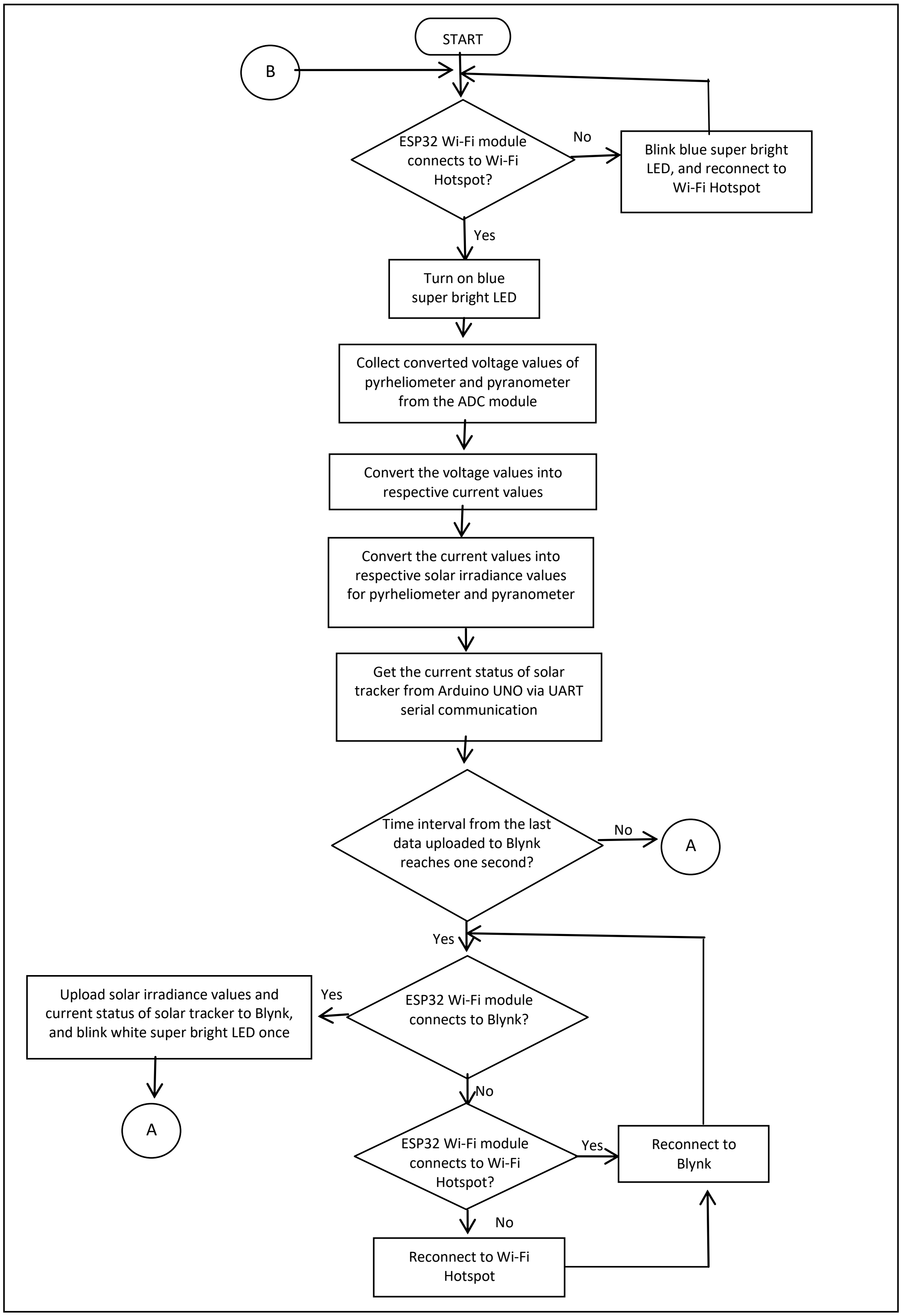

START
B
ESP32 Wi-Fi module connects to Wi-Fi Hotspot?
No
Blink blue super bright LED, and reconnect to Wi-Fi Hotspot
Yes
Turn on blue super bright LED
Collect converted voltage values of pyrheliometer and pyranometer from the ADC module
Convert the voltage values into respective current values
Convert the current values into respective solar irradiance values for pyrheliometer and pyranometer
Get the current status of solar tracker from Arduino UNO via UART serial communication
Time interval from the last data uploaded to Blynk reaches one second?
No
A
Yes
ESP32 Wi-Fi module connects to Blynk?
Yes
Upload solar irradiance values and current status of solar tracker to Blynk, and blink white super bright LED once
A
No
ESP32 Wi-Fi module connects to Wi-Fi Hotspot?
Yes
Reconnect to Blynk
No
Reconnect to Wi-Fi Hotspot

Figure 3.5: Algorithm of IoT Data Collection System

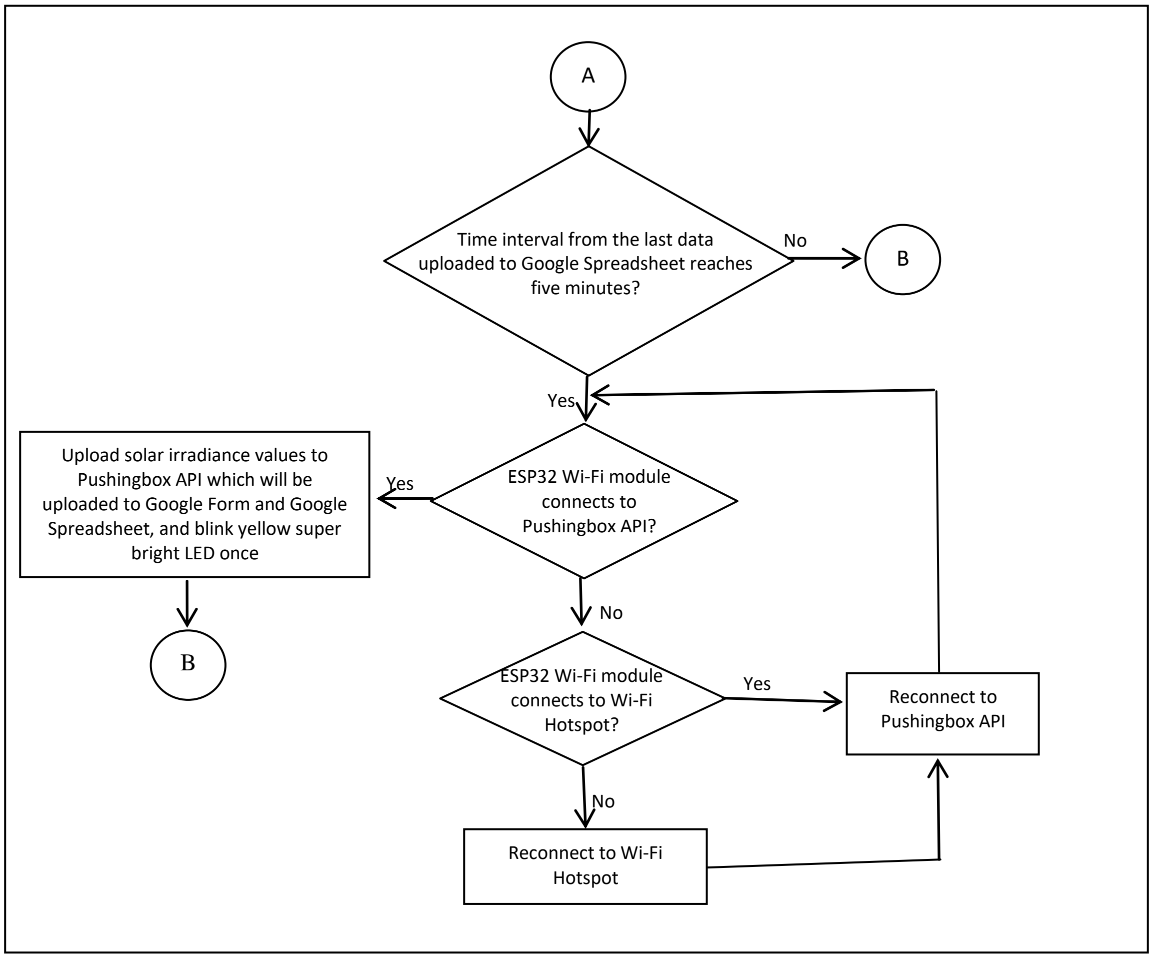

A
Time interval from the last data uploaded to Google Spreadsheet reaches five minutes?
No
B
Yes
Upload solar irradiance values to Pushingbox API which will be uploaded to Google Form and Google Spreadsheet, and blink yellow super bright LED once
Yes
ESP32 Wi-Fi module connects to Pushingbox API?
No
B
ESP32 Wi-Fi module connects to Wi-Fi Hotspot?
Yes
Reconnect to Pushingbox API
No
Reconnect to Wi-Fi Hotspot

Figure 3.6: Algorithm of IoT Data Collection System (Continued)

ESP866 LiLon NodeMCU V3 is implemented as Wi-Fi extender to boost the range of Wi-Fi network from mobile phone. In IoT data collection system, ESP32 is the core to receive inputs from ADS1015 ADC module and Arduino UNO, and then upload data to Blynk and Google Spreadsheet. There are three super bright LEDs, which are blue, white and yellow in colours, acting as indicators.

To perform IoT data transmission, ESP32 connects to the hotspot of ESP8266 LiLon NodeMCU V3 which has the Internet network from mobile phone. The blue super bright LED will keep blinking if ESP32 is not connected to the hotspot from ESP8266 LiLon NodeMCU V3. If ESP32 connects to the hotspot, the blue super bright LED will light up and stop blinking.

Once the hotspot is connected, ESP32 will get the converted voltage values of both pyrheliometer and pyranometer from ADS1015 ADC module. ESP32 will convert the voltage values obtained to respective current values. This step is done when testing the accuracy of the ADC module in reading the voltage values from both current-to-voltage converters with respect to input current signals from both pyrheliometer and pyranometer. The current values obtained will then be converted to respective solar irradiance values. The solar irradiance values are direct normal irradiance from pyrheliometer and global irradiance from pyranometer. Next, ESP32 will get the current status of solar tracker from Arduino UNO via UART serial communication.

Once the data are collected, ESP32 will send the data to Blynk every second and to Google Spreadsheet every five minutes. Solar irradiance values and current status of solar tracker will be uploaded to Blynk Cloud server which will then be displayed in Blynk mobile apps. On the other hand, only solar irradiance values will be uploaded to Pushingbox API server which will then be uploaded to Google Form and Google Spreadsheet. The white super bright LED blinks once if the data are uploaded to Blynk, while yellow super bright LED blinks once if the data are uploaded to Google Spreadsheet. Once the data are sent to Blynk and Google Spreadsheet, ESP32 will repeat the steps in checking Wi-Fi connection, and obtaining solar irradiance values and current status of solar tracker.

3.3 Setup of Solar Tracking Mechanism

This subsection discusses about the setup of the solar tracking mechanism. The process of building the mechanism is illustrated and explained in details.

3.3.1 Design of Dual-Axis Solar Tracker

The dual-axis solar tracker designed can track the Sun automatically without the need of human intervention. Active solar tracking method is implemented where microcontroller will decide the movement of solar tracker according to the light intensities detected by light sensors.

Azimuth-altitude dual-axis solar tracker (AADAST) and tip-tilt dual-axis solar tracker (TTDAST) are the common designs of dual-axis solar tracker. Both of these designs are similar in terms of tracking mechanism, where they are able to perform both azimuth movement and elevation movement. Figure 3.7 shows a clear picture on azimuth and elevation. From Figure 3.7, azimuth movement is the movement of turning to the left or to the right along the horizontal axis, while elevation movement is the movement of tilting upwards or downwards along the vertical axis.

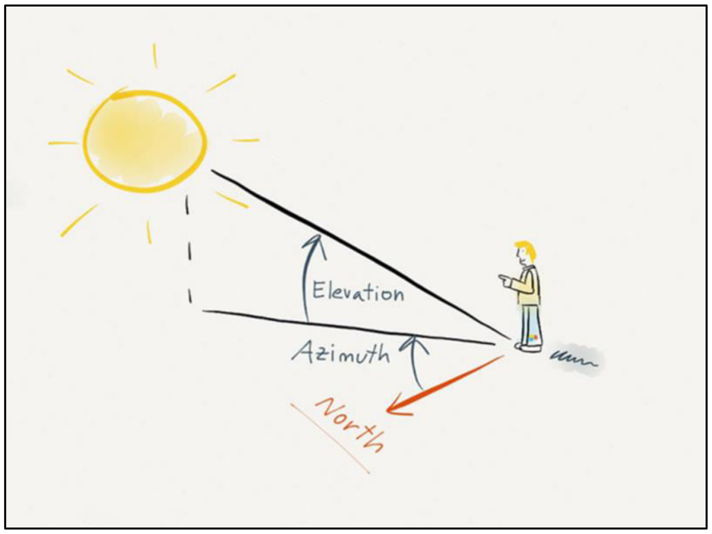


Figure 3.7: A Representation of the Azimuth and the Elevation of the Sun (Pons, 2019)

There is distinct difference in design of AADAST and TTDAST. TTDAST is constructed with swivel gear actuator for horizontal movement and a linear actuator for vertical rotation (Whitlock, 2016). AADAST normally implements two actuators or motors to perform both azimuth and elevation movement (Ray and Tripathi, 2016). In the project, the design of AADAST is implemented.

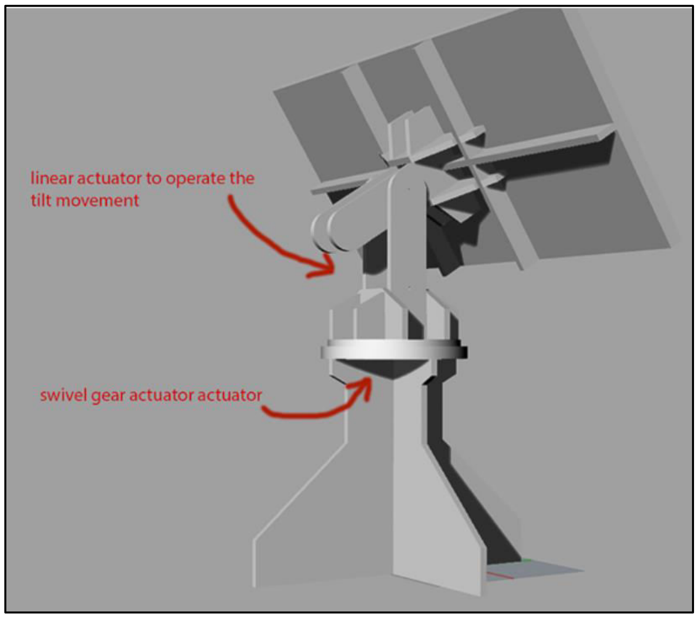

linear actuator to operate the tilt movement
swivel gear actuator actuator

Figure 3.8: Mechanical Design of Tip-Tilt Dual-Axis Solar Tracker

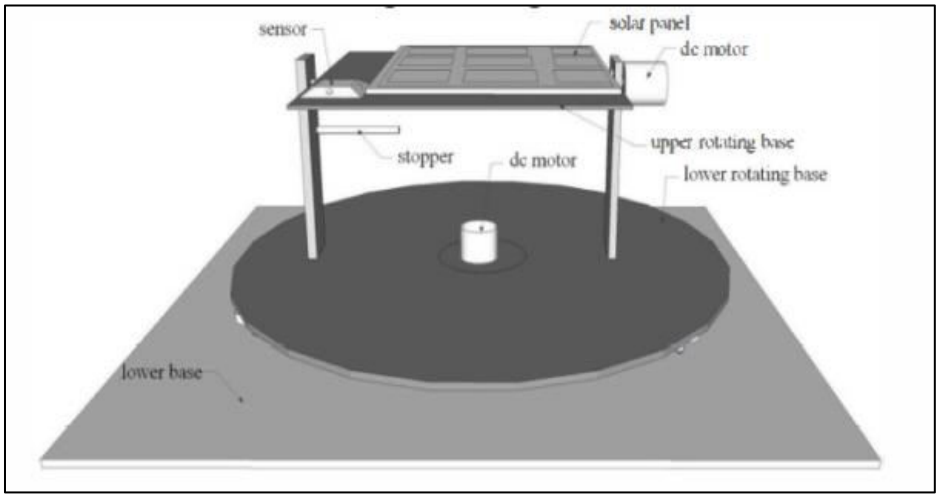


Figure 3.9: Mechanical Design of Azimuth Altitude Dual Axis Solar Tracker (Ray and Tripathi, 2016)

3.3.2 Microcontroller and Software Implemented

Arduino UNO microcontroller board is implemented for the solar tracking mechanism. It is ATmega328 based microcontroller and can be powered on with DC power supply in the range of 7V to 12V. Besides receiving input signals, it can be a mini power supply where it can supply 5V.

Arduino UNO can perform its operation according to the source code uploaded to it. It is easy to setup the Arduino IDE software. The IDE software is free and open-source for users to download Arduino IDE official website. Once the software is download, the source code can be created, compiled and then uploaded to Arduino UNO via USB cable. The programming language used in Arduino IDE software is in C or C++, which is easy to understand. Besides that, the source code can be rewrite and upload again to the microcontroller easily.

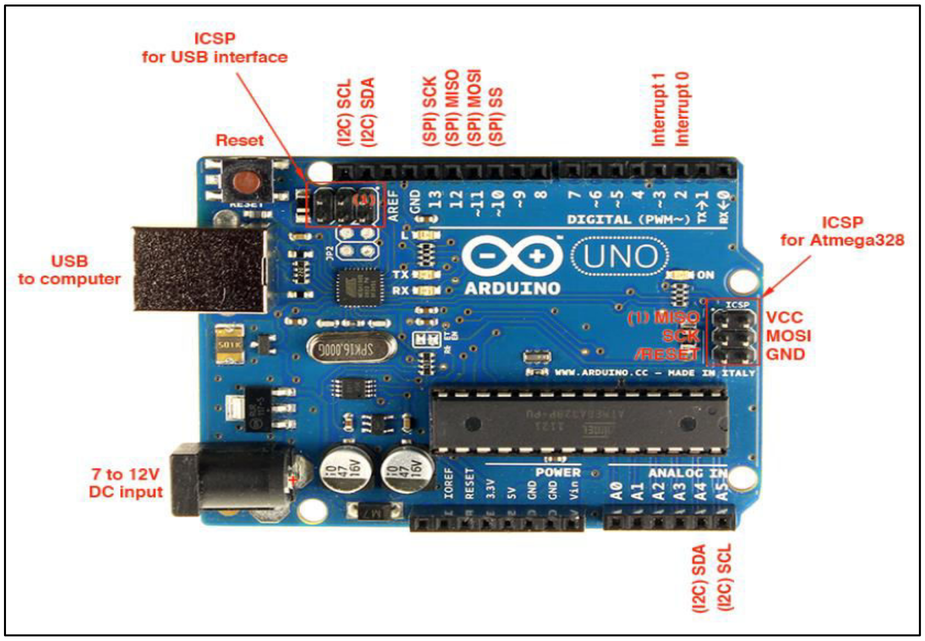


Figure 3.10: Arduino UNO Microcontroller Board

ARDUINO
Genuino
AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits

Figure 3.11: Arduino IDE Software

3.3.3 Design of LDR Circuitry

To enable accurate solar tracking mechanism, LDR circuitry is the crucial component of the solar tracker. Light dependent resistor (LDR) depends on light intensity received to decide its resistance value (Electrical4U, 2019). With this characteristic, LDR is also known as photoresistor. LDR works on the principle of photoconductivity, which is a phenomenon where the material conductivity increases as the amount of photons increases (Electrical4U, 2019). When conductivity increases, more current will start to flow through the LDR causing its resistance to drop. In another words, the resistance of LDR will drop if the ambient light intensity is high, and vice versa.

In this project, four LDRs are implemented to detect ambient light intensities from four different regions. The four LDRs are placed closely, but separated by using opaque material to form a '+' shape as shown in Figure 3.12 (Hosain, et al., 2015). For the material to form the '+' shape, black impraboard was selected because it is opaque and hard. The impraboard was then cut and stick to form a '+' shape as shown in Figure 3.13. Once the '+' shaped partition for LDR circuitry was done, the partition was then stick to the circuitry of LDRs as shown in Figure 3.10. The purpose of LDR sensor array design in Figure 3.12 is to enable shading effect for the solar tracker to track the Sun as shown in Figure 3.14. The presence of shadow on any region will move the solar tracker until all regions receive equal light intensity or the solar tracker faces the Sun. The shading effect principle shown in Figure 3.12 can help in increasing the sensitivity of LDR circuitry in detecting difference of light intensities among all regions which can lead to high precision solar tracking mechanism. High precision solar tracking mechanism is needed because the pyrheliometer only detects the sunlight which directly strikes to it. Furthermore, the small sensing cell of the pyrheliometer is covered by a long hollow tube.

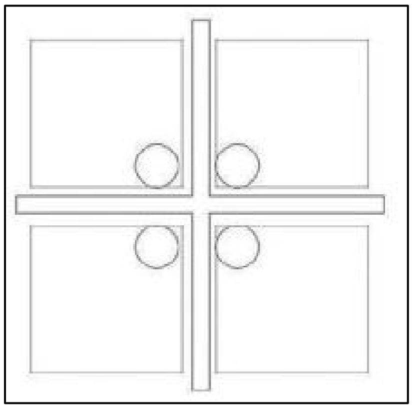


Figure 3.12: LDR Sensor Array Design (Hossain, et al., 2015)

Figure 3.13: LDR Circuitry Partition Constructed Using Impraboard

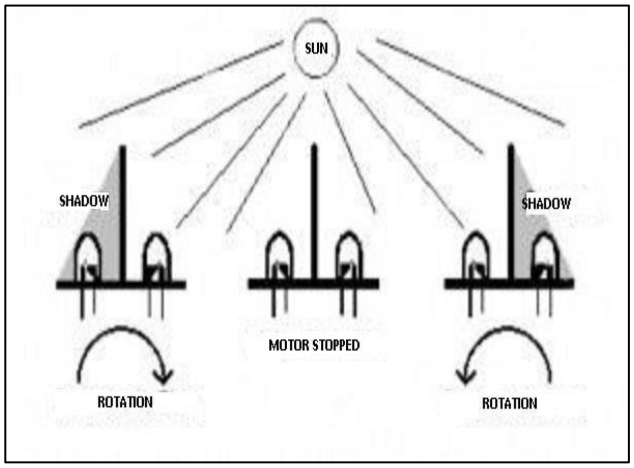


Figure 3.14: How Does Shading Effect Helps the Solar Tracker to Track the Sun

The LDR circuitry is designed by combining the four voltage divider circuits as shown in the picture below. The four voltage divider circuits have a common Vcc of 5V supplied from Arduino UNO, and a common ground. The analog input pins of Arduino UNO are connected in between the LDR and resistor of the voltage divider circuits accordingly to avoid confusion when perform LDR test code in Appendix J.

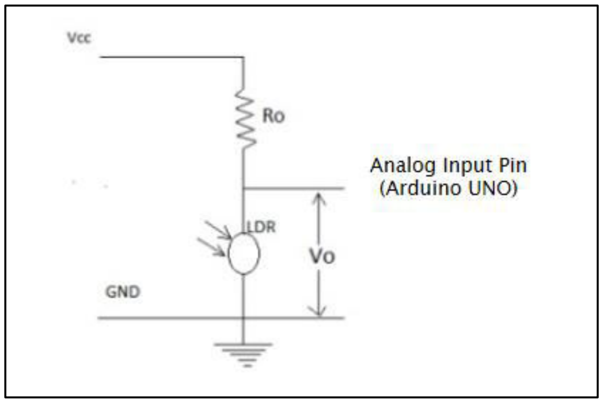


Figure 3.15: Voltage Divider Circuit for Each LDR

3.3.4 Asynchronous Dual-Axis Solar Tracking System

In this project, CRD5107P stepper motor driver and PK543AW-P36 stepper motors were implemented. However, one motor driver can only control one stepper motor. Therefore, relay control circuitry was constructed to enable the motor driver to control two stepper motors asynchronously.

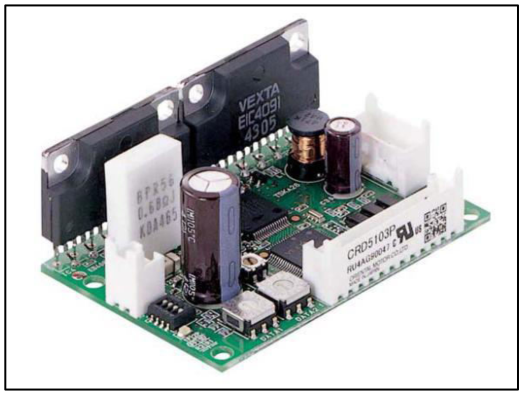


Figure 3.16: CRD5107P Stepper Motor Driver

Figure 3.17: PK543AW-P36 Stepper Motor

Relay is an electronic component which acts as a switch. In this project, SPDT relays are implemented to construct the relay control circuitry. The internal design and the working mechanism of SPDT relay are shown in the pictures in next page respectively. When the copper wire windings or the electromagnetic coil is not energized, the movable armature in the relay remains at its initial position, where the common terminal is connecting NC terminal. When there is voltage supplied to the coil, the coil is energized, attracting the movable armature towards NO terminal. In another words, the common terminal will be connected to NO terminal when there is voltage supplied to the coil.

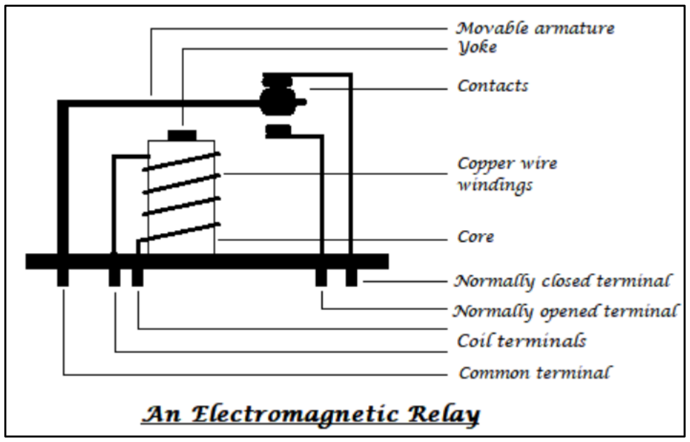


Figure 3.18: Internal Design of SPDT Relay (Thonti, 2017)

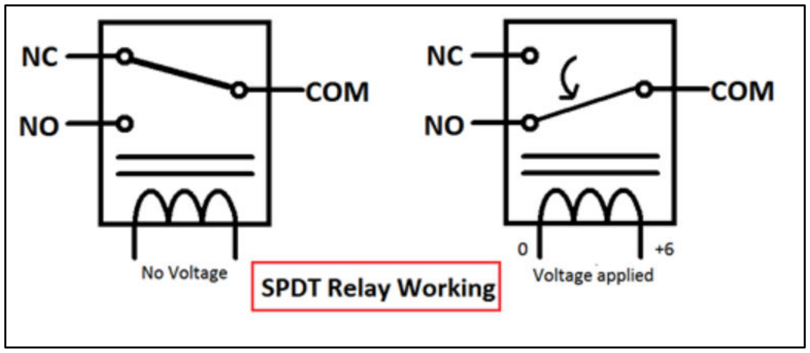


Figure 3.19: Working Mechanism of SPDT Relay (Thonti, 2017)

Each stepper motor has five wires with different colours which are blue, red, orange, green and black. Therefore, five relays are used in the circuitry. The common terminals of the relays are connected to motor connector terminals of the motor driver. The NO terminals of the relays are connected to the stepper motor for vertical movement, while NC terminals of the relays are connected to the stepper motor for the horizontal movement. However, do take note that the wires from motor connector terminals of motor driver have to be connected according to the colour of the wires from the stepper motors, which is shown in Figure 3.20.

To enable the switching mechanism of the relays, Arduino UNO is implemented to control the relay control circuitry. However, the current signal from Arduino UNO is too low to energize the coil of the 5 V relays eventhough it produces 5 V to the relays. Therefore, a pair of transistors is formed into darlington transistor to have higher current gain (Elprocus, 2013). The connection of darlington transistor in the relay control circuitry is shown in Figure 3.20.

Besides deciding the switching mechanism of the relays, Arduino UNO plays an important role in deciding the stepper motor in rotating clockwise or counterclockwise by sending pulse to the motor driver. Pin D9 of Arduino UNO connects to CW pulse input of motor driver for clockwise direction, while pin D10 of Arduino UNO connects to CCW pulse input of motor driver for counterclockwise direction. To activate the motor driver, 24V power supply is needed. For more details on wiring for the motor driver, the datasheet attached in Appendix C can be referred.

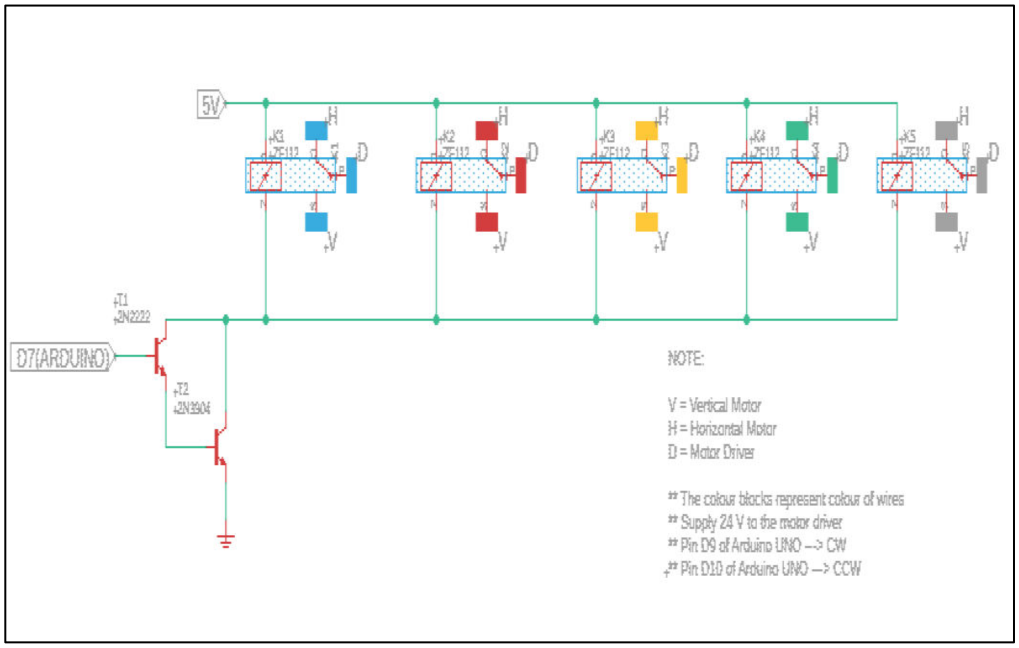

5V
K1 ZF112
K2 ZF112
K3 ZF112
K4 ZF112
K5 ZF112
H
D
V
T1 2N2222
D7(ARDUINO)
T2 2N3904
NOTE:
V = Vertical Motor
H = Horizontal Motor
D = Motor Driver
** The colour blocks represent colour of wires
** Supply 24 V to the motor driver
** Pin D9 of Arduino UNO --> CW
** Pin D10 of Arduino UNO --> CCW

Figure 3.20: Schematic Diagram of Relay Control Circuitry

After schematic diagram of the relay control circuitry is done, the PCB layout was then drawn using Eagle software as shown in the picture below. The PCB board for the relay control circuitry is shown in the picture in next page.

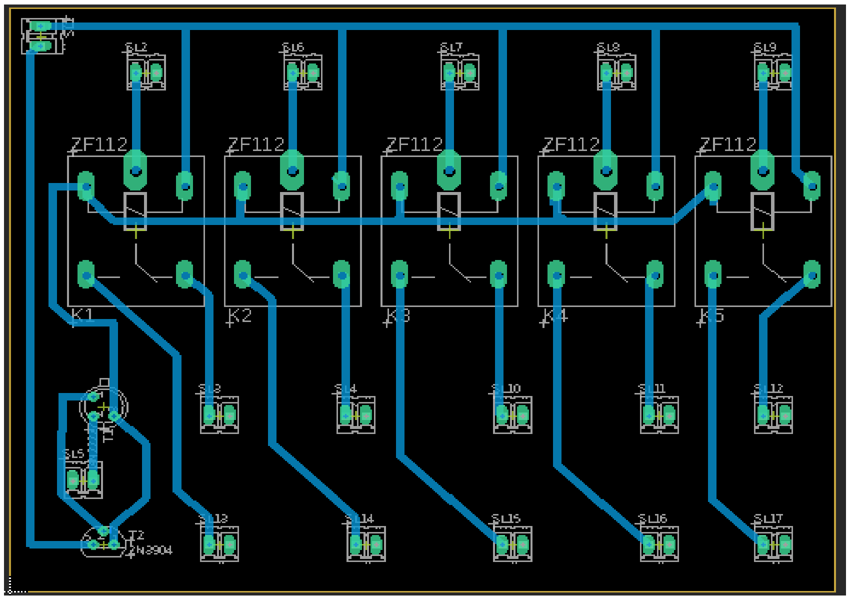


Figure 3.21: Eagle PCB Layout for Relay Control Circuitry

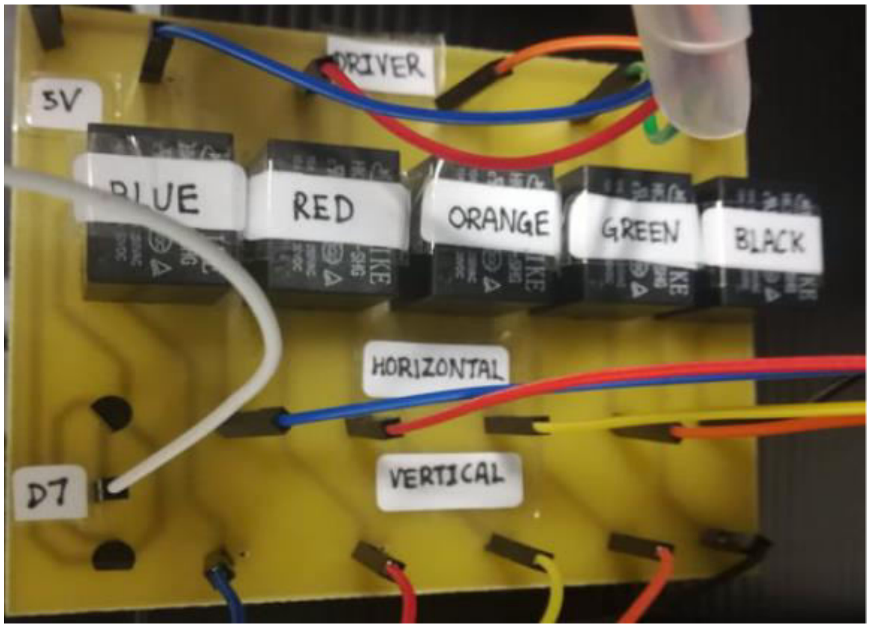


Figure 3.22: PCB Board for Relay Control Circuitry

After the components and wires are soldered on PCB board for relay control circuitry, the movement of the solar tracker is tested using test codes in Appendices F, G, H and I. The mechanism in moving the solar tracker is shown in the table below.

Table 3.1: Mechanism for Movement of the Solar Tracker

Arduino UNO pin			**Movement of Solar Tracker**
D7	**D9**	**D10**	
HIGH	LOW	1 pulse	UP
HIGH	1 pulse	LOW	DOWN
LOW	LOW	1 pulse	LEFT
LOW	1 pulse	LOW	RIGHT

3.3.5 MPU6050 Gyro Sensor

The extra feature included is the initialization of the solar tracker when it is powered on. The plate of the solar tracker will initialize its position, so that it lies parallel to the ground. According to Pandian (2019), the Sun will be at North or South depending on the seasons throughout a year and will rise from East to West throughout a day regardless seasonal changes. Therefore, initialization of solar tracker is important. Lying the plate of solar tracker flat to be the initial position is important to help the solar tracker to decide whether to track at the North or South region throughout the day. Hence, MPU6050 gyro sensor is useful in initializing the position of the plate of the solar tracker by calculating its elevation angle.

Another function of the gyro sensor is to limit the range of elevation angle to prevent the plate of the solar tracker from hitting the circuitry of the solar tracker. With MPU6050 gyro sensor, usage of limit switches can be eliminated and hence, reduce the number of components and wires used.

3.3.6 LED Indicators in Solar Tracking Mechanism

There are two LEDs used as indicators in solar tracking mechanism. The purpose of adding indicators is to reduce the dependency of connecting Arduino UNO using USB cable to perform serial monitoring. Furthermore, it is inconvenient to perform serial monitoring on the moving solar tracker. The operation on how the indicators work is shown in the table in next page.

Table 3.2: Operation of Indicators in Solar Tracking Mechanism

Current Status of Solar Tracker	**Yellow super bright LED**	**Green super bright LED**
Initializing the position of plate of solar tracker	ON	OFF
Stop during cloudy environment	ON	ON
Stop and facing the Sun	OFF	ON
Tracking the Sun	OFF	OFF

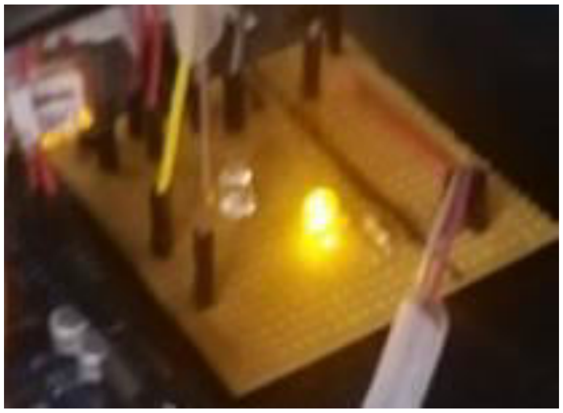


Figure 3.23: Yellow LED Turns On When Solar Tracker is Initializing

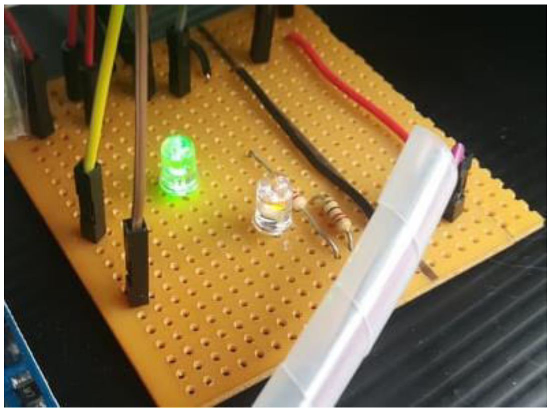


Figure 3.24: Both Green and Yellow LEDs Turn On In Cloudy Environment

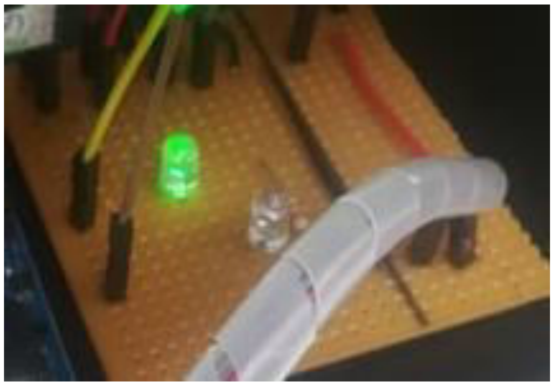


Figure 3.25: Green LED Turn On When Solar Tracker Faces the Sun

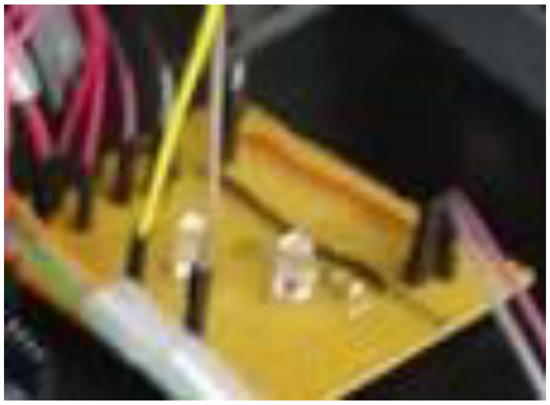


Figure 3.26: Both Green and Yellow LEDs Turn Off When Solar Tracker is Tracking

3.3.7 Schematic Diagram for Solar Tracking Mechanism

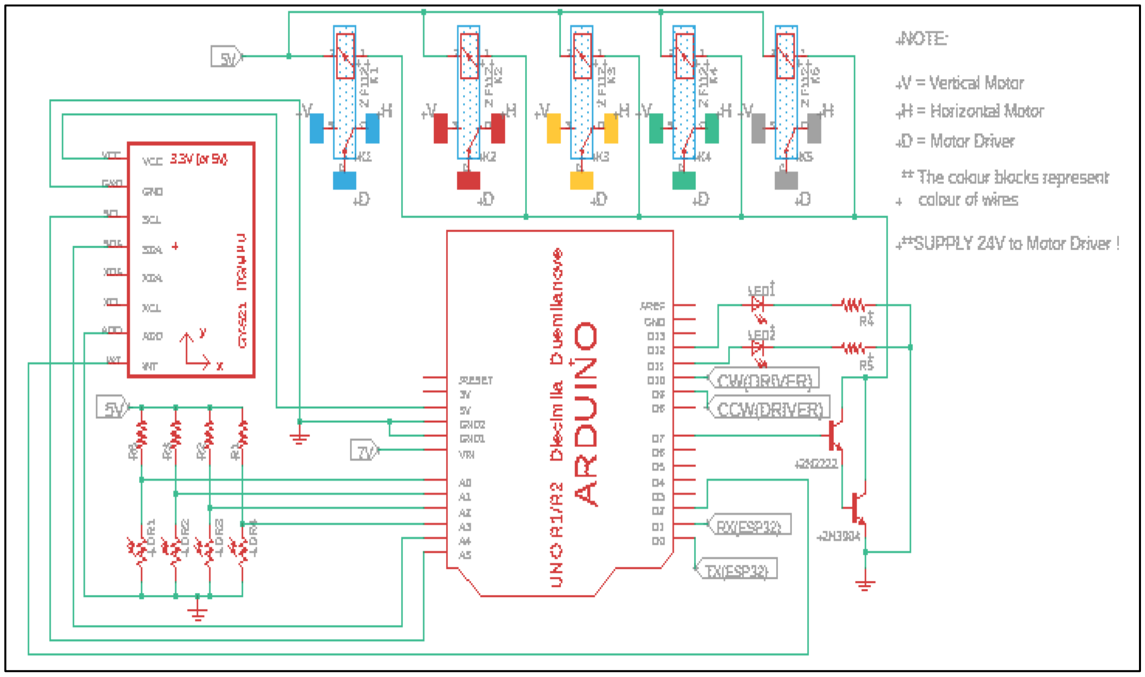


Figure 3.27: Schematic Diagram for Solar Tracking Mechanism

3.4 Setup of IoT Data Collection System

This subsection illustrates and explains the procedures in setting up the IoT data collection system. The circuitry for the IoT data collection system is illustrated in this subsection. Hardware and software selection are discussed as well.

3.4.1 Design and Construction of Low Cost Pyrheliometer and Pyranometer

This subsection illustrates the sensing cells implemented for both sensors, and the designs of both pyrheliometer and pyranometer.

3.4.1.1 Multi-Junction Solar Cell (MJSC)

MJSC has higher efficiency in absorbing photons compared to single junction cells as there are three different semiconductor materials arranged in different layers to absorb different wavelengths of sunlight (Maas, 2012) . The semiconductor with the largest bandgap is placed at the top to absorb short wavelength, followed by the semiconductor with medium bandgap, and the semiconductor with the narrowest bandgap at the bottom for long wavelength absorption, which is shown as in Figure 3.28. Concentrated light is supplied on the solar cell to enhance its efficiency, and it can be done using optical light collector such as lenses (Fedkin, 2018). The lens will collect the sunlight and concentrate it to a small area of solar cell as shown in Figure 3.29.

Two MJSCs as shown in Figure 3.30, are implemented as the sensing cell for both pyrheliometer and pyranometer.

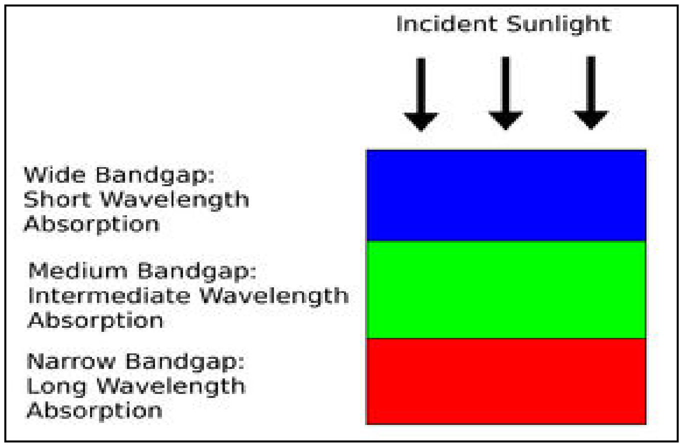

Incident Sunlight
Wide Bandgap: Short Wavelength Absorption
Medium Bandgap: Intermediate Wavelength Absorption
Narrow Bandgap: Long Wavelength Absorption

Figure 3.28: How Multi-Junction Solar Cell Absorb Different Wavelengths of Sunlight (Maas, 2012)

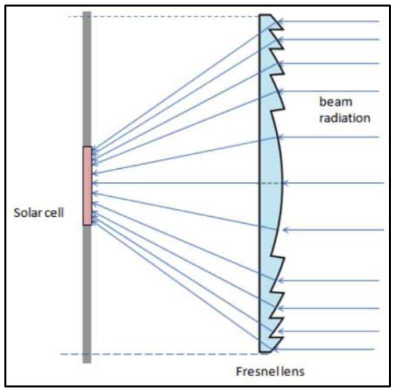


Figure 3.29: How Lens Concentrate the Sunlight to the Solar Cell (Fedkin, 2018)

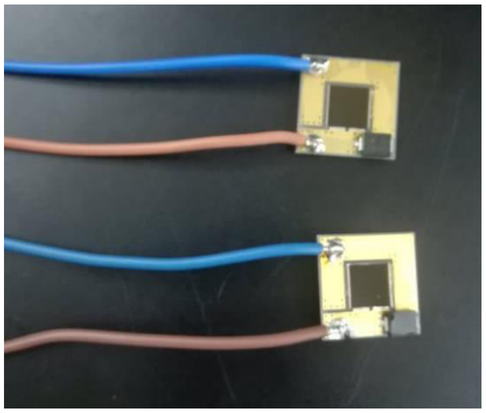


Figure 3.30: Multi-Junction Solar Cell Implemented For The Project

3.4.1.2 Design of Cost Effective Pyrheliometer

Pyrheliometer is a sensing device measuring direct normal irradiance, which is the direct sunlight hitting the sensing element. Pyrheliometer is a long tube which only allow the sunlight passing through it with an acceptance angle of 5°. The inner layer of the tube is blackened and is not reflective to ensure only acceptance angle of 5°. The design of the pyrheliometer is shown in the picture in next page.

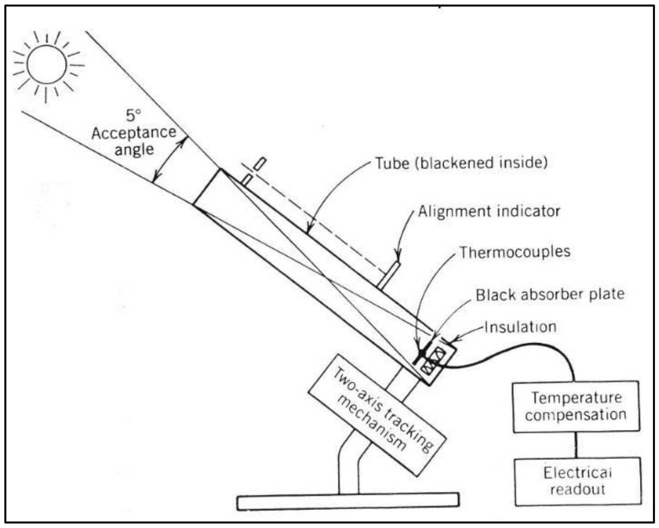
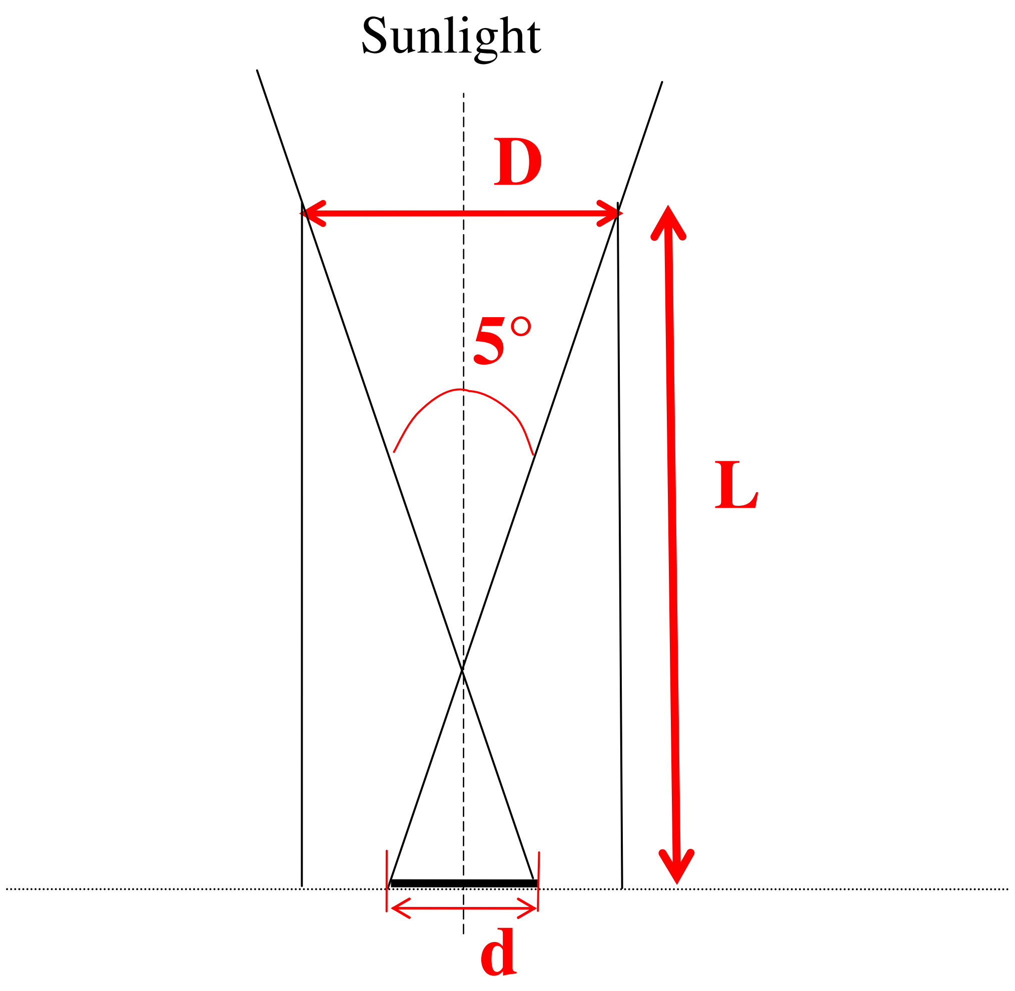


Figure 3.31: Design of Standard Pyrheliometer

However, cost of the standard pyrheliometer is relatively high. Therefore, a cost effective pyrheliometer is designed and constructed in this project. One of the solar cells shown in Figure 3.30 will be implemented as sensing element for the pyrheliometer. Before constructing the body of the pyrheliometer, an outline shown in next page was drawn and simple mathematical calculation was done.



Note:

- D = Diameter of the opening of pyrheliometer
- L = Length of pyrheliometer
- d = Diameter of the solar cell

Figure 3.32: Outline for the Construction of Pyrheliometer

From the outline drawn in the previous page, a formula for construction of pyrheliometer was formed as shown in Equation 3.1.

$$\tan(\frac{5°}{2}) = \frac{[(\frac{D}{2})+(\frac{d}{2})]}{L} \tag{3.1}$$

where

D = diameter of the opening of pyrheliometer, cm

d = diameter of the solar cell, cm

L = length of pyrheliometer, cm

From Equation 3.1, the diameter of solar cell, d was known, which is 1 cm. However, the diameter of the opening of pyrheliometer, D and length of pyrheliometer, L were unknown. Therefore, the diameter of the opening of pyrheliometer, D was predefined to calculate the length of pyrheliometer, L. The diameter of the opening, D has to be larger than the diameter of the solar cell. In this project, the diameter of the opening of pyrheliometer was set to be 2.6 cm. Hence, the length of pyrheliometer, L would be 41.2 cm according to Equation 3.1 together with given values of diameter of the opening of pyrheliometer, D and diameter of the solar cell, d.

Impraboard was selected as material for the body of pyrheliometer. However, impraboard is reflective and it is not desirable to have inner layer of the body to be reflective. Therefore, black T-shirt was cut and stick on the surface of the impraboard. The shirt acts as a non-reflective coating for inner layer of the body. Impraboard itself and impraboard covered with black T-shirt were tested under the light as shown in Figure 3.33. From Figure 3.33, impraboard covered with black T-shirt can be seen not to be reflective compared to impraboard itself. The impraboard was cut into four rectangles and covered with black T-shirt respectively. The four rectangles were stick to form a hollow cuboid, and the outer body of the cuboid covered with PVC tape. The body of pyrheliometer was done as shown in Figure 3.34. The body of the pyrheliometer was then tested under the light by looking through the opening of the body. The inner layer of the body of pyrheliometer was proved to be not reflective when the result obtained was as in Figure 3.35.

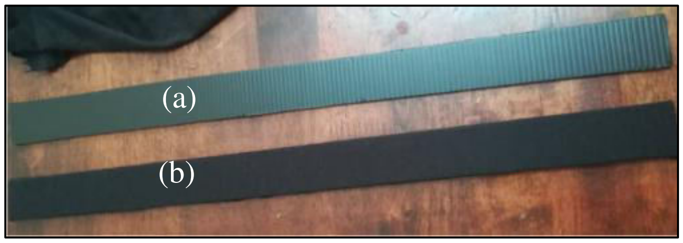


Figure 3.33: Testing Impraboard itself (a) and Impraboard Covered With Black T-Shirt (b) Under the Light

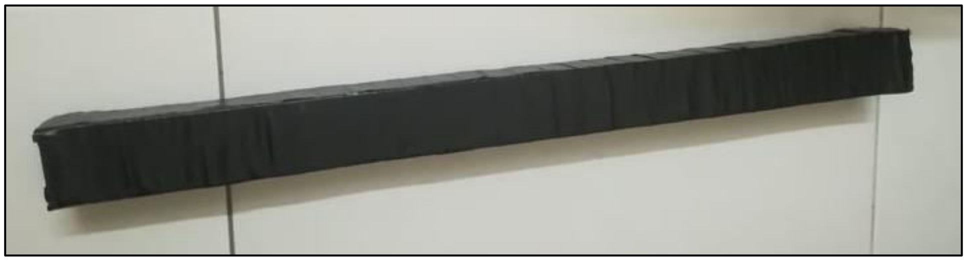


Figure 3.34: Body of the Pyrheliometer

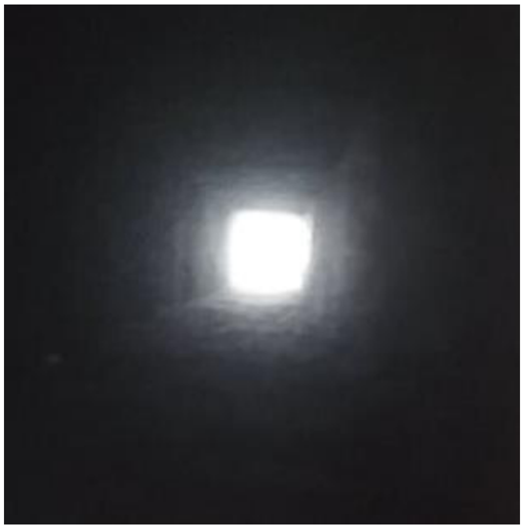


Figure 3.35: View through the Opening of Pyrheliometer When the Body is Pointing to the Light

Once the inner layer of the pyrheliometer was verified to be not reflective, the solar cell was stick at the base of the pyrheliometer and covered with the body constructed. The pyrheliometer was done constructed and stick on the plate of the solar tracke. The calculation done for the construction of the pyrheliometer is proved to be accurate if the solar cell can be seen through the body of pyrheliometer when the solar tracker is facing the Sun.

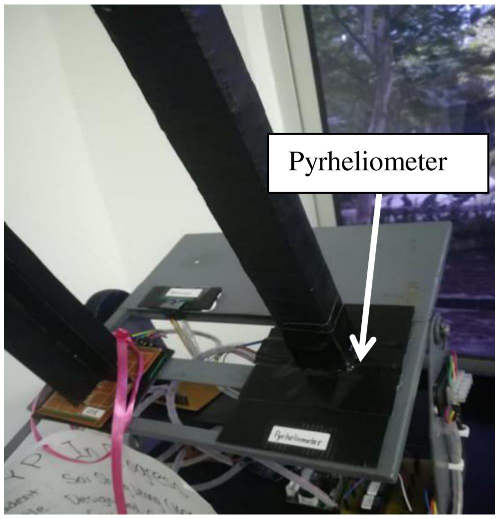


Figure 3.36: Pyrheliometer Constructed For The Project

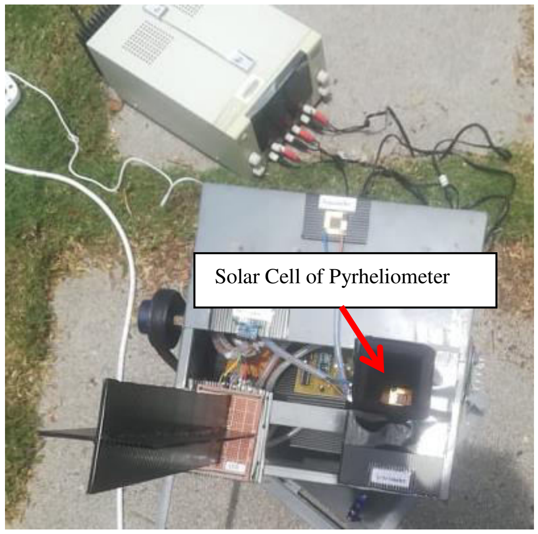


Figure 3.37: Solar Cell is Visible Through the Body of Pyrheliometer If The Solar Tracker is Facing the Sun

The constructed pyrheliometer is not only able to work as a standard pyrheliometer, but it is also lighter compared to the standard pyrheliometer. Furthermore, the cost incurred for pyrheliometer constructed is much more cheaper compared to the standard pyrheliometer.

After the construction of pyheliometer was done, comparison was done with the actual normal incidence pyrheliometer as in Appendix E. The results of comparison will be reviewed and analyzed in Chapter 4.

3.4.1.3 Design of Cost Effective Pyranometer

Pyranometer is a sensing device which measures global irradiance. A pyranometer consists of two glassdomes which help to concentrate the sunlight to the sensing element. The sensing element will then convert the radiation received to a differential voltage signal proportional to the amount of radiation received. However, the cost of a standard pyranometer is relatively high.

Since pyranometer measures global irradiance which is the solar radiation around it, therefore, one of the solar cells is implemented as the pyranometer, where it is directly stick on the plate of the solar tracker. The solar cell is also able to collect the solar radiation around it, which has the same functionality as the standard pyranometer. The solar cell is not only lighter and smaller in size, but it also helps to save cost as well.

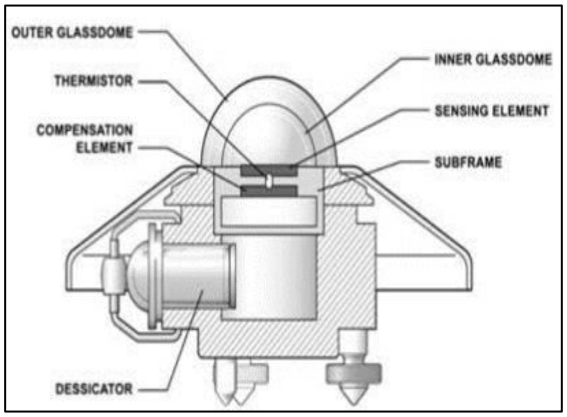


Figure 3.38: Design of Standard Pyranometer

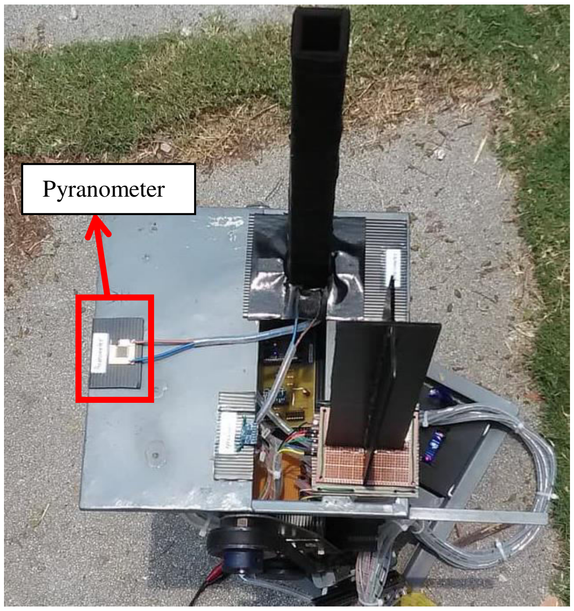


Figure 3.39: Design of Pyranometer Constructed

3.4.2 Current-to-Voltage Converter Circuitry

Since the solar cells given produce current signal when receive radiation from the Sun, therefore, current-to-voltage conversion is needed as both ESP32 and ADS1015 ADC module are able to read only voltage input signals.

Two current-to-voltage converters are needed as there are two solar cells given. The procedures of constructing the current-to-voltage conversion circuitry was done and tested before the construction of both pyrheliometer and pyranometer. There are two types of current-to-voltage converter, which are active and passive current-to-voltage converters shown in the pictures below respectively.

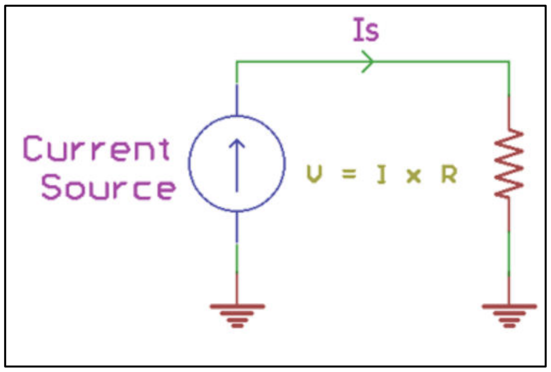


Figure 3.40: Passive Current-to-Voltage Converter

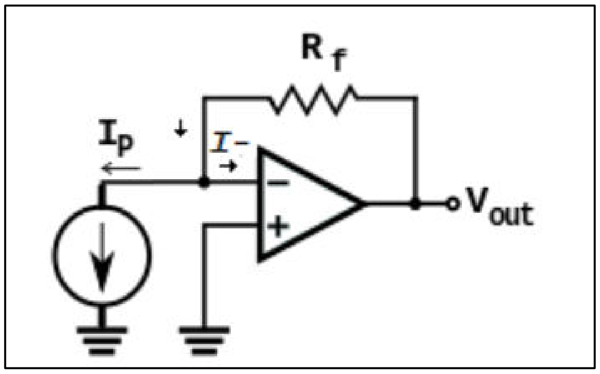


Figure 3.41: Active Current-to-Voltage Converter

Passive current-to-voltage converters might work well theoretically, but the performance will not be as desired in practical due to poor gain and insufficient noise-to-signal ratio (Gupta, 2019). Active current-to-voltage converters will be the solution for the problem of passive current-to-voltage converters, and are implemented for current-to-voltage conversion purpose. Two LM324 ICs are used as the current-to-voltage converters for both pyrheliometer and pyranometer.

Before the circuitry for current-to-voltage conversion was constructed, direct voltage input signal testing was done using ESP32 itself and ADS1015 ADC module connecting to ESP32. After the testing was done, the method of using ADS1015 ADC module connecting to ESP32 was preferred. The comparison between these two methods and reason to implement ADS1015 ADC module to read voltage signal are analyzed and discussed in Chapter 4.

To build a circuitry for current-to-voltage conversion, it is important to determine the correct resistance value for the feedback resistor, R_f shown in Figure 3.41. From Figure 3.41, the value for feedback resistor, R_f can be obtained using the formula shown in Equation 3.2.

$$V_{out} = I_p R_f \tag{3.2}$$

where

V_{out} = output voltage of current-to-voltage converter, V

I_p = input current to the current-to-voltage converter, A

R_f = feedback resistance, Ω

The maximum current produced by the solar cells given is 20 mA, and the maximum voltage signal measurable by ADS1015 ADC module is set at 3V. With the values of these two variables given, the feedback resistor for each current-to-voltage converter will be 150 Ω.

3.4.3 ESP32 Wi-Fi Module and Software Implemented

ESP32 is implemented to transfer data wirelessly to both IoT user interface platform and IoT database platform for data collection and analysis. This Wi-Fi module board uses ESP-WROOM-32 chip consisting of thirty GPIO pins, one enable pin, two common ground pins, one pin which supply 3.3V and one power input pin.

Arduino IDE software is compatible with ESP32. It is easy to install ESP32 board in Arduino IDE software regardless the operating system implemented and steps of installation are followed as the pictures shown accordingly (RandomNerdTutorial.com, 2013).

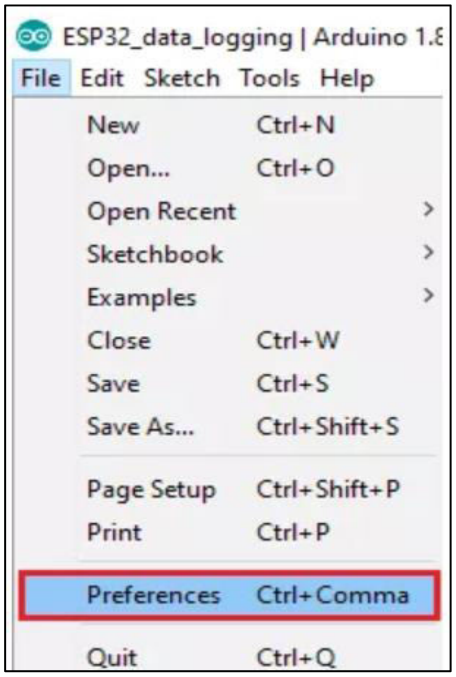
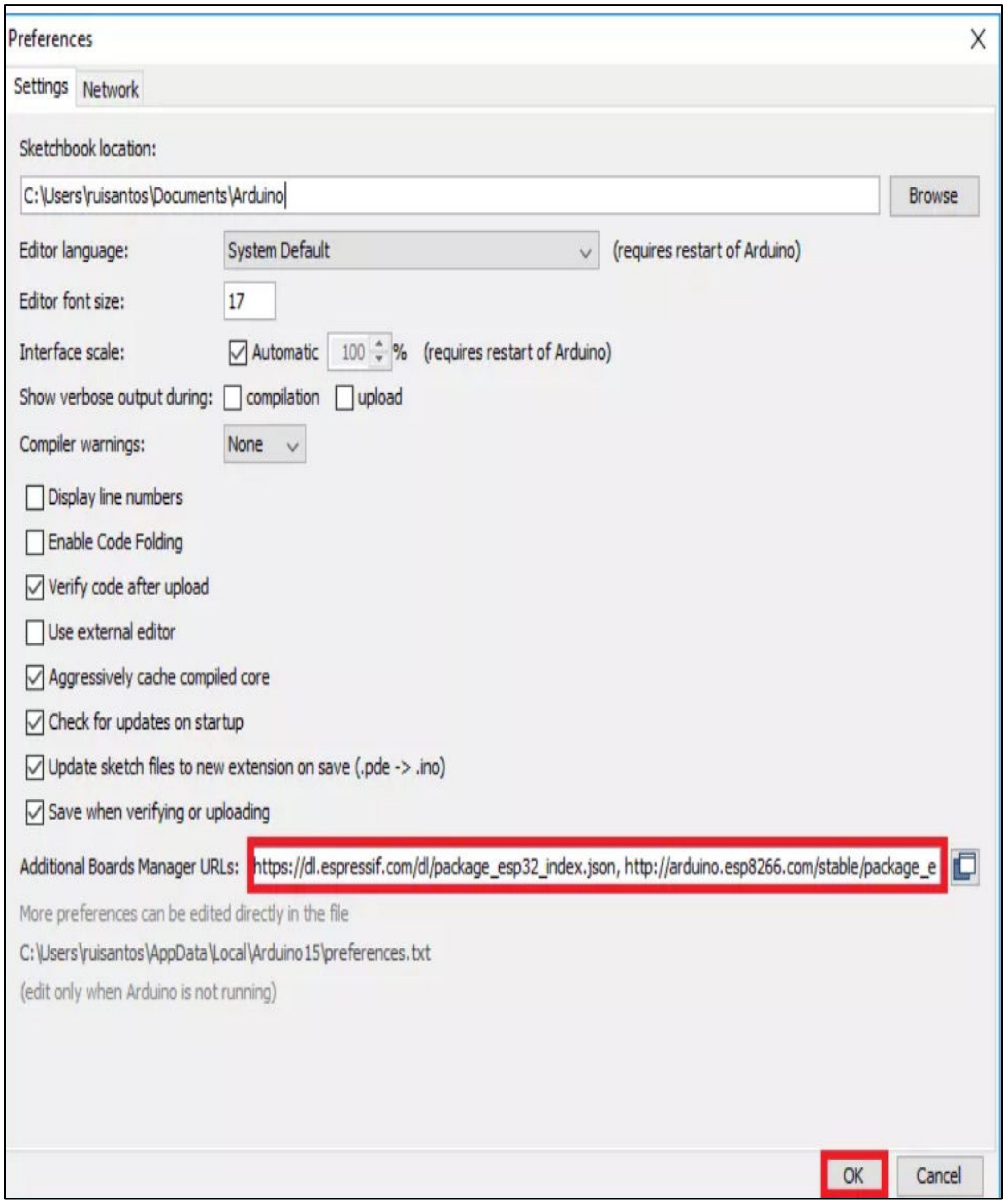


Figure 3.42: Go to “File > Preferences” (RandomNerdTutorial.com, 2013)



Preferences

Settings | Network

Sketchbook location:

C:\Users\ruisantos\Documents\Arduino

Browse

Editor language: System Default (requires restart of Arduino)

Editor font size: 17

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers

☐ Enable Code Folding

☑ Verify code after upload

☐ Use external editor

☑ Aggressively cache compiled core

☑ Check for updates on startup

☑ Update sketch files to new extension on save (.pde -> .ino)

☑ Save when verifying or uploading

Additional Boards Manager URLs: https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_e

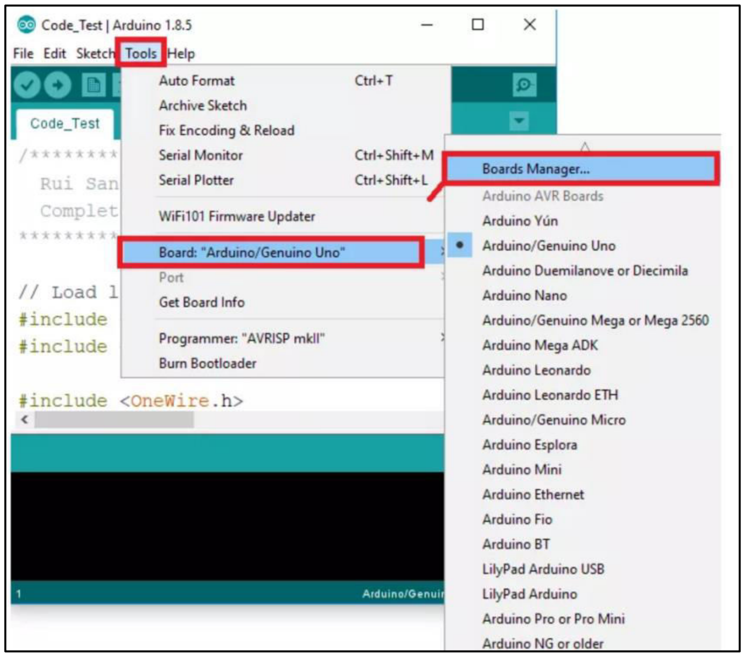
More preferences can be edited directly in the file

C:\Users\ruisantos\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

OK Cancel

Figure 3.43: Edit Preferences Tab (RandomNerdTutorial.com, 2013)

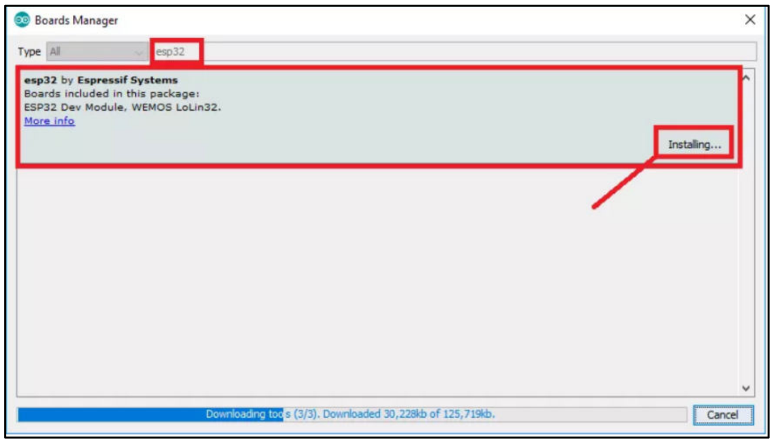


```
/*********
  Rui San
  Complet
*********

// Load l
#include
#include

#include <OneWire.h>
```

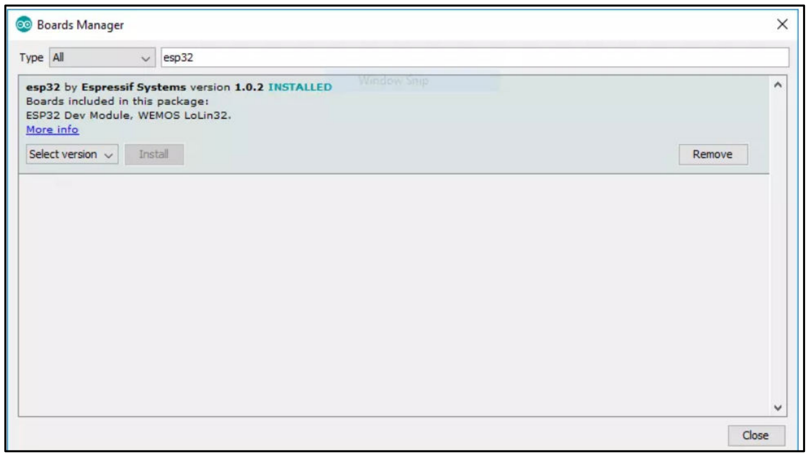
Figure 3.44: Go to “Tools > Board > Boards Manager…” (RandomNerdTutorial.com, 2013)



esp32 by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.

Downloading tools (3/3). Downloaded 30,228kb of 125,719kb.

Figure 3.45: Installation of ESP32 Board in Arduino IDE Software (RandomNerdTutorial.com, 2013)



esp32 by Espressif Systems version 1.0.2 INSTALLED
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.
More info

Figure 3.46: Installation of ESP32 Board Done (RandomNerdTutorial.com, 2013)

3.4.4 Configuration of Wi-Fi Extender

Internet network is a must for IoT data collection system. ESP32 itself does not have Internet network, and hence, mobile phone is used as a router to provide Internet network. However, mobile phone provides short range of Wi-Fi network. Therefore, ESP8266 LiLon NodeMCU V3 is implemented as the Wi-Fi extender to increase the range of Wi-Fi network.

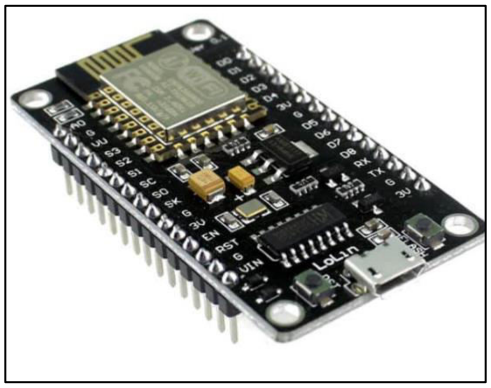
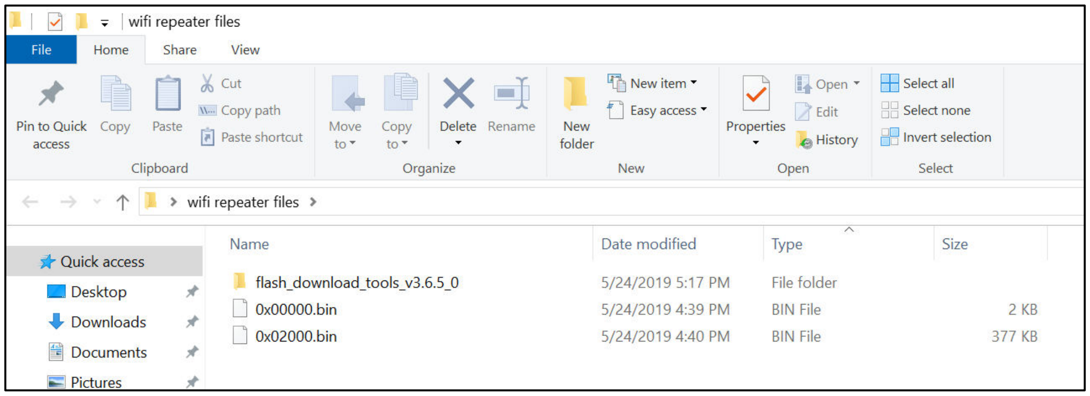


Figure 3.47: ESP8266 LiLon NodeMCU V3

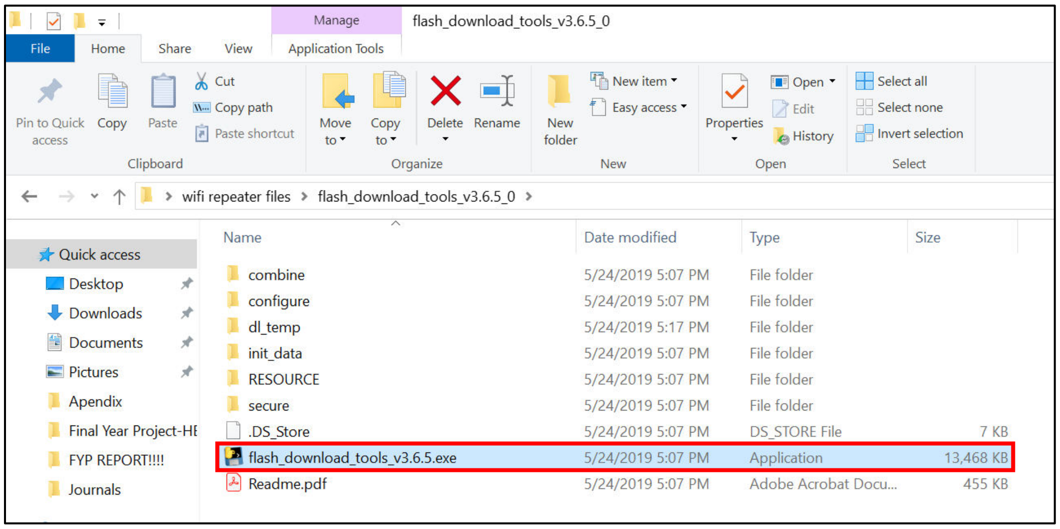
Before configuring the Wi-Fi extender, a zip file, “wifi extender files” from MediaFire (2019) has to be downloaded. Next, extract and open the zip file downloaded as shown in the picture below. At the same time, connect ESP8266 LiLon NodeMCU V3 to the computer via USB cable for configuration of Wi-Fi extender.



Name	Date modified	Type	Size
flash_download_tools_v3.6.5_0	5/24/2019 5:17 PM	File folder	
0x00000.bin	5/24/2019 4:39 PM	BIN File	2 KB
0x02000.bin	5/24/2019 4:40 PM	BIN File	377 KB

Figure 3.48: Files In “wifi repeater files”

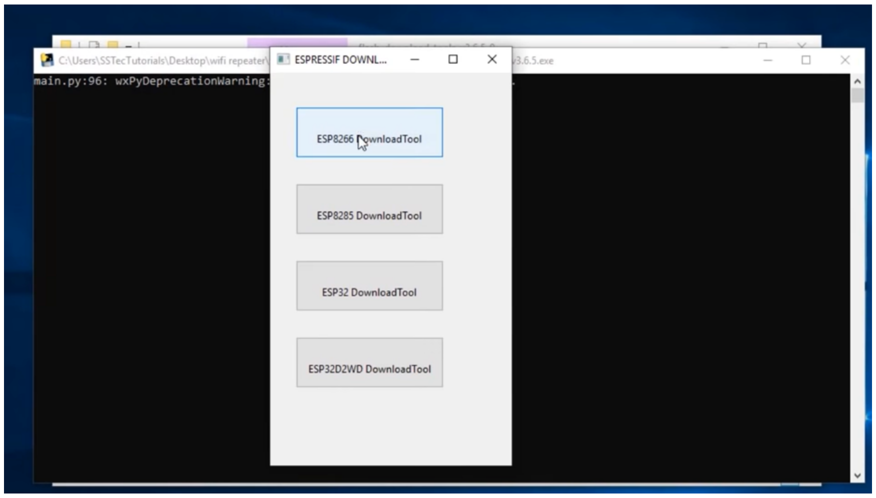
Open “flash_download_tools_v3.6.5_0” folder from extracted zip file shown in the picture above. Then, click on “flash_download_tools_v3.6.5.exe” as shown in the picture below.



Name	Date modified	Type	Size
combine	5/24/2019 5:07 PM	File folder	
configure	5/24/2019 5:07 PM	File folder	
dl_temp	5/24/2019 5:17 PM	File folder	
init_data	5/24/2019 5:07 PM	File folder	
RESOURCE	5/24/2019 5:07 PM	File folder	
secure	5/24/2019 5:07 PM	File folder	
.DS_Store	5/24/2019 5:07 PM	DS_STORE File	7 KB
flash_download_tools_v3.6.5.exe	5/24/2019 5:07 PM	Application	13,468 KB
Readme.pdf	5/24/2019 5:07 PM	Adobe Acrobat Docu...	455 KB

Figure 3.49: Click On “flash_download_tools_v3.6.5.exe”

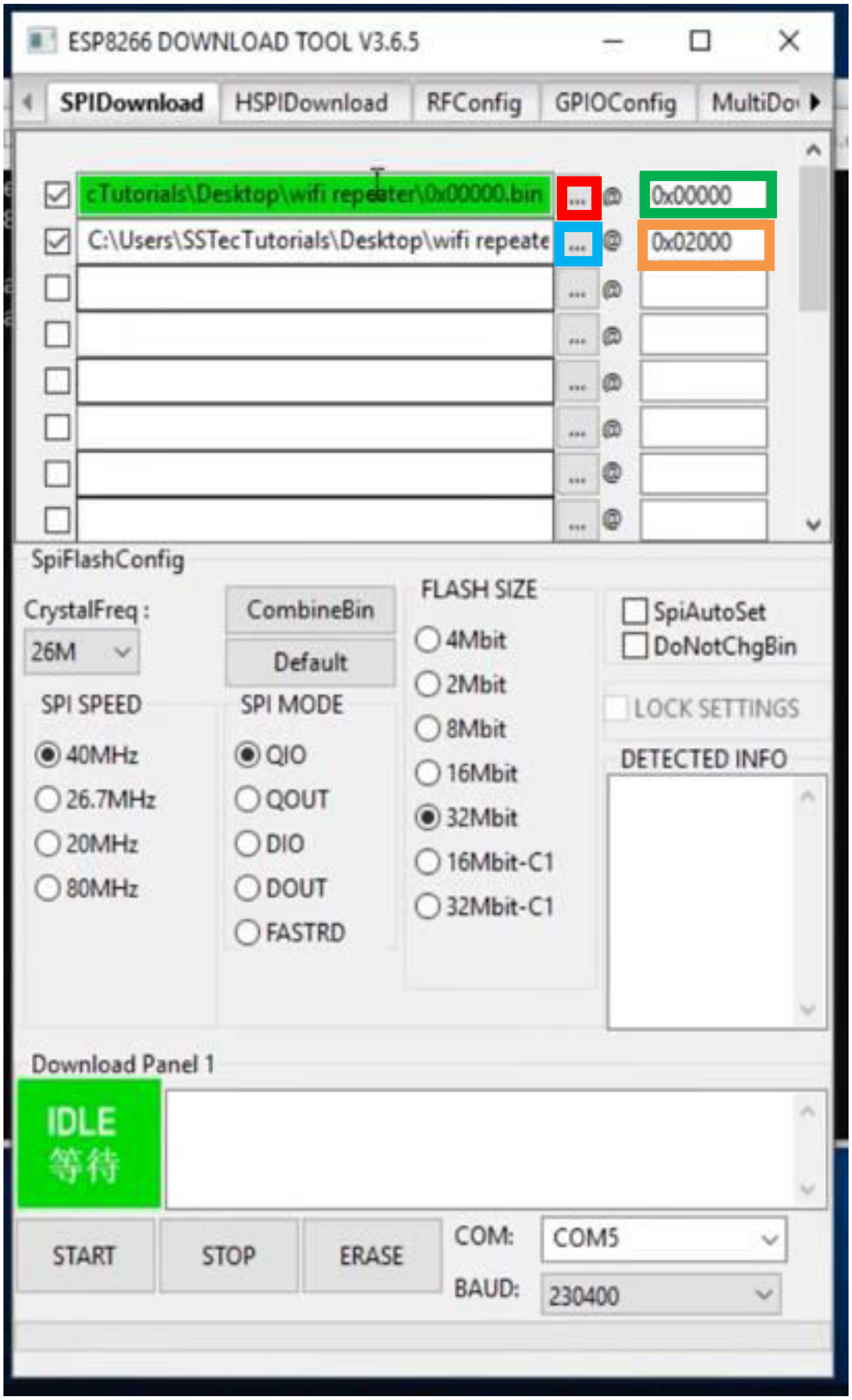
After clicking on the execution file, wait until it is as shown in the picture below, and then click on the button “ESP8266 DownloadTool”.



```
main.py:96: wxPyDeprecationWarning:
```

Figure 3.50: Click on “ESP8266 DownloadTool”

ESP8266 Download Tool firmware is executed as shown in next page. In the section “SPIDownload”, the column in both green and orange frame are typed as follow. Before checking the two radio buttons at the upper left corner, the button in the red frame is clicked, and then double click on “0x0000.bin” when a tab appears as in Figure 3.48. Next, the button in the blue frame is clicked, and then double click on “0x02000.bin” when a tab appears as in Figure 3.48. For “SpiFlashConfig”, the properties of “CrystalFreq”, “SPI SPEED”, “FLASH SIZE” and “SPI MODE” are set as follow. The “COM:” must be according to the name of the serial communication port which can be checked in Control Panel, while the “BAUD:” is set to 115200 since baud rate of data transmission for the IoT data collection system is 115200 bits per second.



[✓] cTutorials\Desktop\wifi repeater\0x00000.bin @ 0x00000
[✓] C:\Users\SSTecTutorials\Desktop\wifi repeate @ 0x02000

CrystalFreq : 26M

SPI SPEED: [● 40MHz] 26.7MHz 20MHz 80MHz

SPI MODE: [● QIO] QOUT DIO DOUT FASTRD

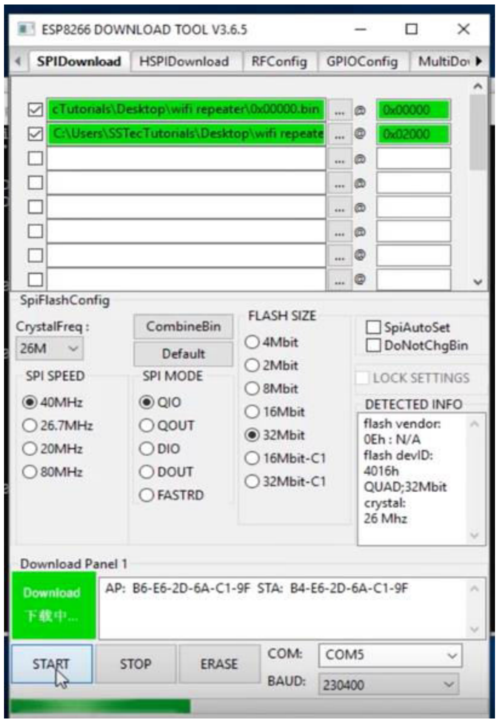
FLASH SIZE: 4Mbit 2Mbit 8Mbit 16Mbit [● 32Mbit] 16Mbit-C1 32Mbit-C1

☐ SpiAutoSet
☐ DoNotChgBin

COM: COM5
BAUD: 230400

Figure 3.51: ESP8266 Download Tool Firmware

Once the setting for flashing is done, flash the ESP8266 module by clicking "START" button as shown in Figure 3.52. The green square box at the lower left corner will show "Download". Once the ESP8266 is flashed, the green square box at the lower left corner in Figure 3.53 will show "FINISH".



☑ cTutorials\Desktop\wifi repeater\0x00000.bin @ 0x00000
☑ C:\Users\SSTecTutorials\Desktop\wifi repeate @ 0x02000

CrystalFreq : 26M
SPI SPEED: [✓ 40MHz] 26.7MHz 20MHz 80MHz
SPI MODE: [✓ QIO] QOUT DIO DOUT FASTRD
FLASH SIZE: 4Mbit 2Mbit 8Mbit 16Mbit [✓ 32Mbit] 16Mbit-C1 32Mbit-C1
☐ SpiAutoSet
☐ DoNotChgBin

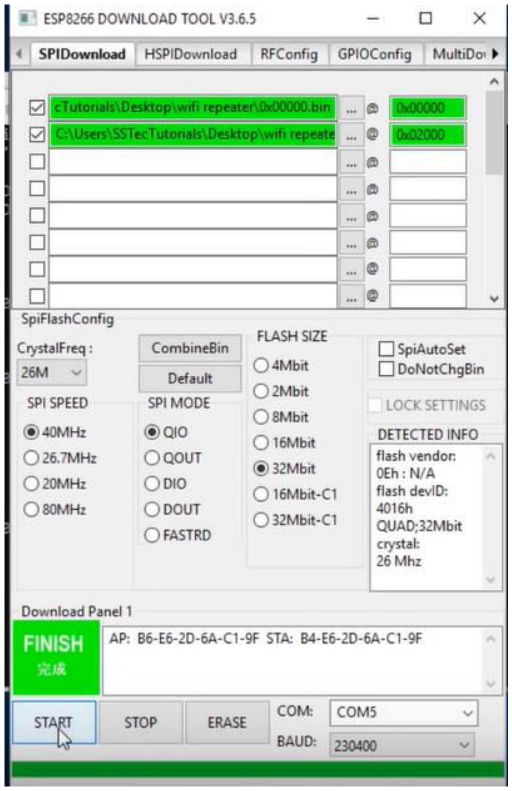
DETECTED INFO:
flash vendor:
0Eh : N/A
flash devID:
4016h
QUAD;32Mbit
crystal:
26 Mhz

Download
下载中...

AP: B6-E6-2D-6A-C1-9F STA: B4-E6-2D-6A-C1-9F

COM: COM5
BAUD: 230400

Figure 3.52: Flash ESP8266 LiLon NodeMCU V3



☑ cTutorials\Desktop\wifi repeater\0x00000.bin @ 0x00000
☑ C:\Users\SSTecTutorials\Desktop\wifi repeate @ 0x02000

CrystalFreq : 26M

SPI SPEED: (●) 40MHz ○ 26.7MHz ○ 20MHz ○ 80MHz

SPI MODE: (●) QIO ○ QOUT ○ DIO ○ DOUT ○ FASTRD

FLASH SIZE: ○ 4Mbit ○ 2Mbit ○ 8Mbit ○ 16Mbit (●) 32Mbit ○ 16Mbit-C1 ○ 32Mbit-C1

☐ SpiAutoSet
☐ DoNotChgBin

DETECTED INFO: flash vendor: 0Eh : N/A flash devID: 4016h QUAD;32Mbit crystal: 26 Mhz

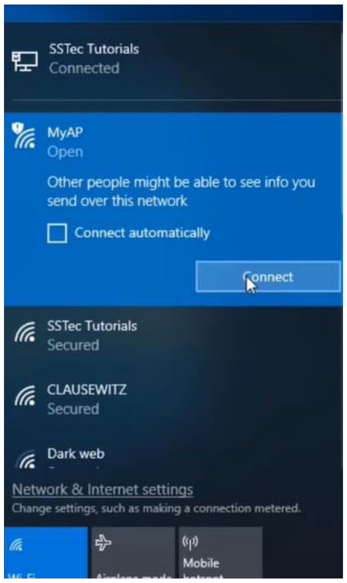
AP: B6-E6-2D-6A-C1-9F STA: B4-E6-2D-6A-C1-9F

COM: COM5
BAUD: 230400

Figure 3.53: ESP8266 LiLon NodeMCU V3 is Done Flashing

Once ESP8266 LiLon NodeMCU V3 is flashed, the configuration of Wi-Fi extender has to be done. Remove the USB cable from the computer after flashing the ESP8266 module. After that, supply the ESP8266 module by connecting it to 5V adapter. The access point of the ESP8266 module with its default SSID of “MyAP” can be seen when turn on Wi-Fi connection as shown in Figure 3.54. The security of the SSID is open and it requires no password to connect.

After connected to the ESP8266 module, open the web browser and type “192.168.4.1”. The Wi-Fi extender configuration page will appear as shown in Figure 3.55. Firstly, change the “SSID” and “Password” in “STA Settings” according to the SSID and password of hotspot from mobile phone, and then click on the “Connect” button. Next, change the “AP Settings” for configuration of access point of ESP8266 Wi-Fi extender. In this project, the SSID for the hotspot to ESP32 is “Solar_Tracker” and the password of the hotspot is “UTAR_1234”. The security mode of the hotspot is set to “WPA2”. Once the hotspot is configured, a new SSID “Solar_Tracker” can be seen as shown in Figure 3.56.


SSTec Tutorials
Connected

MyAP
Open

Other people might be able to see info you send over this network

Connect automatically

Connect

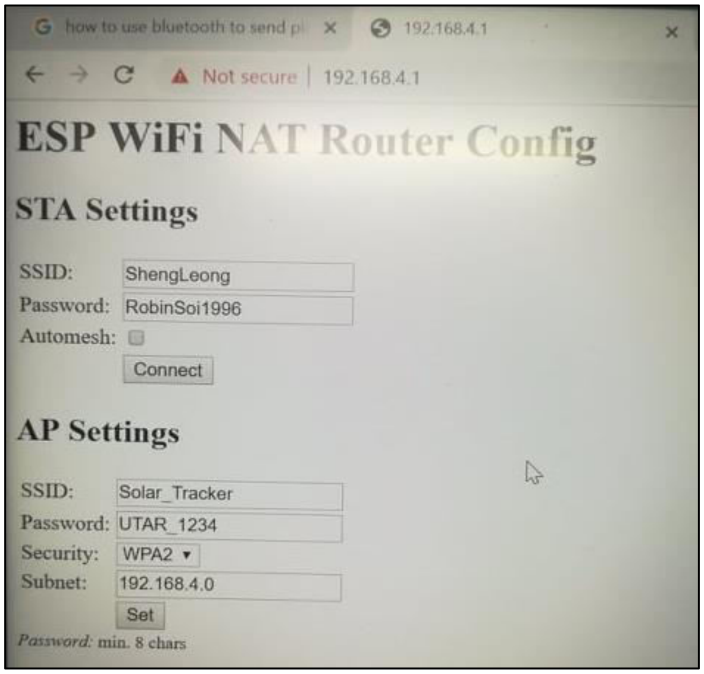
SSTec Tutorials
Secured

CLAUSEWITZ
Secured

Dark web

Network & Internet settings
Change settings, such as making a connection metered.

Figure 3.54: Connect the Preset Hotspot For Configuration of Wi-Fi Extender



ESP WiFi NAT Router Config

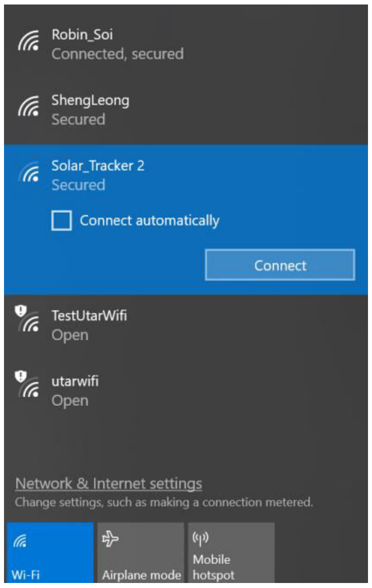
STA Settings

SSID: ShengLeong
Password: RobinSoi1996
Automesh:
Connect

AP Settings

SSID: Solar_Tracker
Password: UTAR_1234
Security: WPA2
Subnet: 192.168.4.0
Set
Password: min. 8 chars

Figure 3.55: Wi-Fi Extender Configuration Page


Robin_Soi
Connected, secured

ShengLeong
Secured

Solar_Tracker 2
Secured

Connect automatically

Connect

TestUtarWifi
Open

utarwifi
Open

Network & Internet settings
Change settings, such as making a connection metered.

Figure 3.56: Configuration of Wi-Fi Extender Completed

To enable the connection of the ESP32 to ESP8266 LiLon NodeMCU V3 Wi-Fi extender, the code line as shown in the picture below has to be included in the source code shown in Appendix O.

```
const char* ssid = "Solar_Tracker"; //Wi-Fi network you want to connect
const char* password = "UTAR_1234";
```

Figure 3.57: Code Line to Configure Wi-Fi Connection Between ESP32 Wi-Fi Module and ESP8266 LiLon NodeMCU V3 Wi-Fi Extender

3.4.5 IoT User Interface Platform

In this project, Blynk is implemented as the IoT user interface platform where the users can view the real-time data in it. In this project, the IoT data collection system requires only real-time data display together with a data storage system. Furthermore, this simple IoT data collection system does not require sophisticated software operation. Therefore, Blynk is implemented as it is easy to use and is powerful in performing IoT data collection (Blynk Inc., 2019). It can display real-time data for the user to visualize, and is able to store data. Furthermore, Arduino IDE software is compatible to perform IoT data transmission to Blynk for data visualization (Blynk Inc., 2019).

It is easy to set up IoT data collection system using Blynk. There are three main elements needed, which are a smartphone, an IoT hardware and Internet connection (Blynk Inc., 2019). First of all, download Blynk app in the smartphone. Blynk can be downloaded in both iOS and Android operating system. Once Blynk app is download, execute it and create a new account for creation of IoT user interface platform.

Figure 3.58: Blynk App Icon

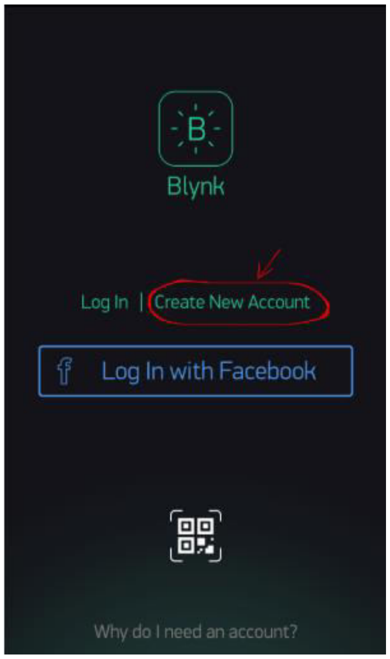


Figure 3.59: Create New Blynk Account

After log into the created account, click on “New Project” icon as shown in Figure 3.60 to create a new user interface platform. Next, set the project name, and select device and connection type as shown in Figure 3.61. Once the project name, device and connection type are set as in Figure 3.61, click on “Create” button to create the new user interface platform. Auth Token will be received by the creator via e-mail, and there will be a message box appear in Blynk app as shown in Figure 3.62. Auth Token is a unique identifier for connection of IoT device with the smartphone. Once the Auth Token is received from the e-mail, copy the Auth Token and replace the highlight part of the code line of Appendix O with the received Auth Token as shown in Figure 3.63.

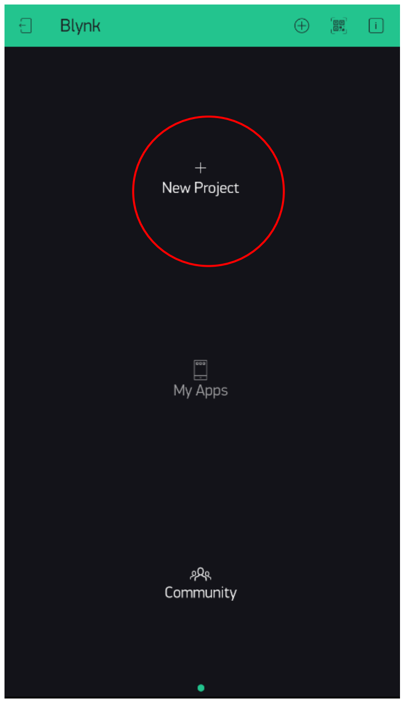
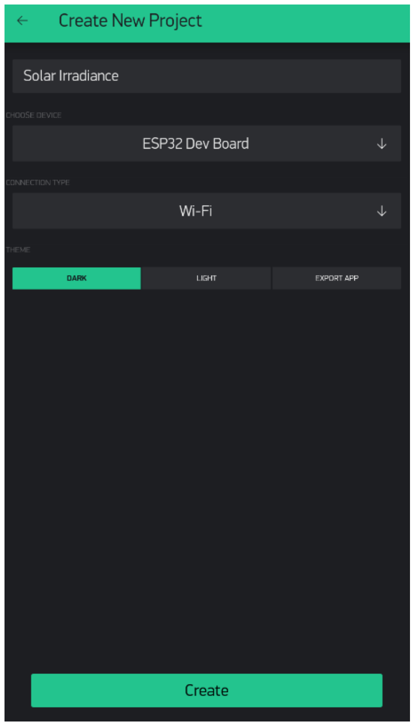


Figure 3.60: Click On “New Project” to Create New Project In Blynk


Create New Project

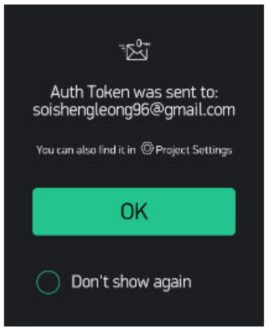
Solar Irradiance

CHOOSE DEVICE: ESP32 Dev Board

CONNECTION TYPE: Wi-Fi

THEME: DARK

Figure 3.61: Creation of New User Interface in Blynk


Auth Token was sent to:
soishengleong96@gmail.com

You can also find it in Project Settings

Figure 3.62: Auth Token Sent to E-mail

```
const char auth[] = "951c69db77154adaa6c754fcbb90ed8d";
```

Figure 3.63: Arduino Code Line to Declare the Auth Token of Blynk Project

The new user interface platform should be empty by default, and widgets can be added by clicking “+” icon shown in Figure 3.64. The widget menu bar will be displayed. The required widgets for user interface of the IoT data collection system are added until the user interface platform looks as shown in Figure 3.65. The white indicator box functions as a display to indicate the current status of solar tracker. The “Database Access” button is the button which allow users to view the data of both pyrheliometer and pyranometer stored in Google Spreadsheet when it is pressed. In the user interface platform, two value displays and two graphs are implemented to display live irradiance data measured by both pyrheliometer and pyranometer.

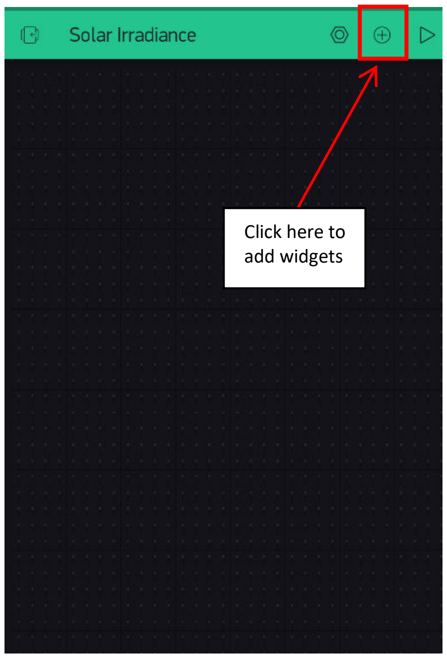


Figure 3.64: Click On “+” Icon to Add Widgets to the User Interface

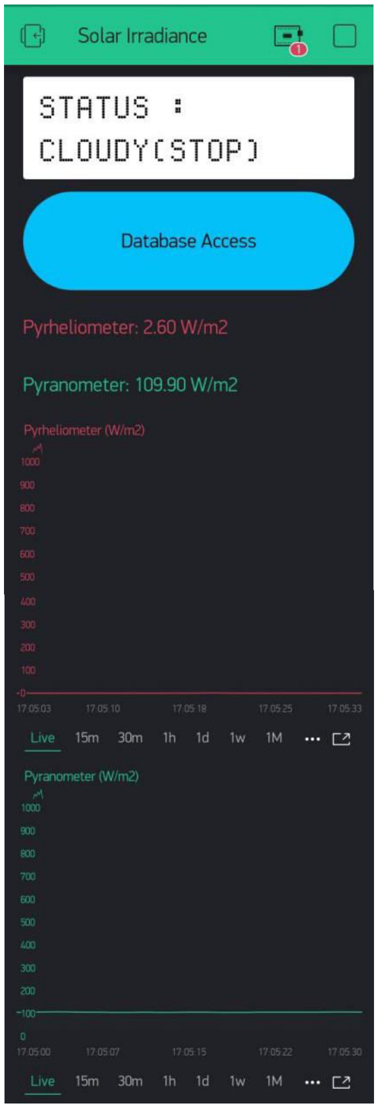


Figure 3.65: Blynk IoT User Interface Platform

Blynk user interface itself does have an indicator of connection status. The indicator will tell whether ESP32 is connecting the hotspot as shown in both Figure 3.66 and Figure 3.67.

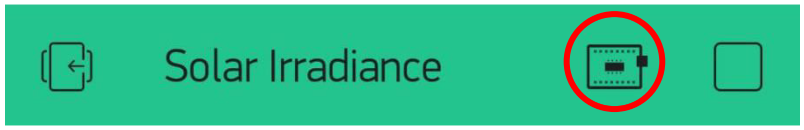


Figure 3.66: Indicator When ESP32 is Connected to Hotspot

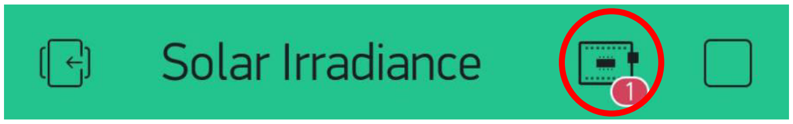


Figure 3.67: Indicator When ESP32 is Not Connected to Hotspot

In this project, the user interface platform will be accessible to everyone. To make the user interface platform accessible, click on "Project Setting" to enable shared access. In "Shared Access", turn on the shared access and then click on "Generate Link" button as shown in Figure 3.68. The QR code will be generated as shown in Figure 3.69.

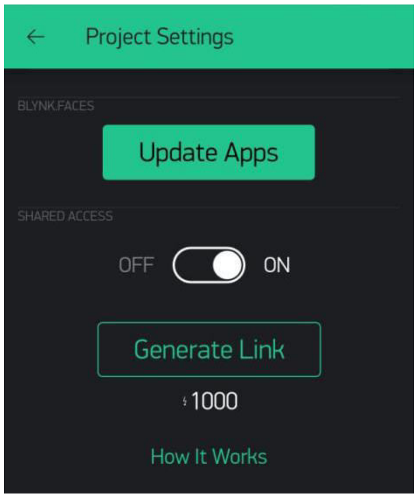
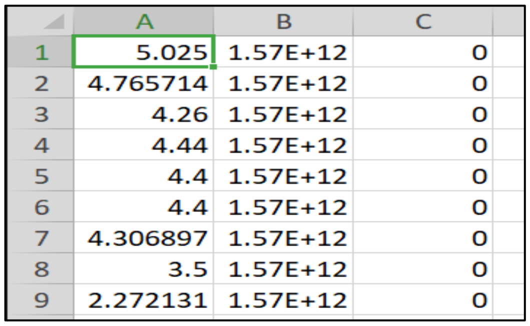


Figure 3.68: Generate Link For Shared Access

Figure 3.69: QR Code Created For Blynk IoT User Interface Platform

3.4.6 IoT Database

Although real-time database can exported from the real-time graphs in Blynk, the CSV files exported are hard to understand as shown in the picture below. Therefore, a user-friendly IoT database has to be created.



	A	B	C
1	5.025	1.57E+12	0
2	4.765714	1.57E+12	0
3	4.26	1.57E+12	0
4	4.44	1.57E+12	0
5	4.4	1.57E+12	0
6	4.4	1.57E+12	0
7	4.306897	1.57E+12	0
8	3.5	1.57E+12	0
9	2.272131	1.57E+12	0

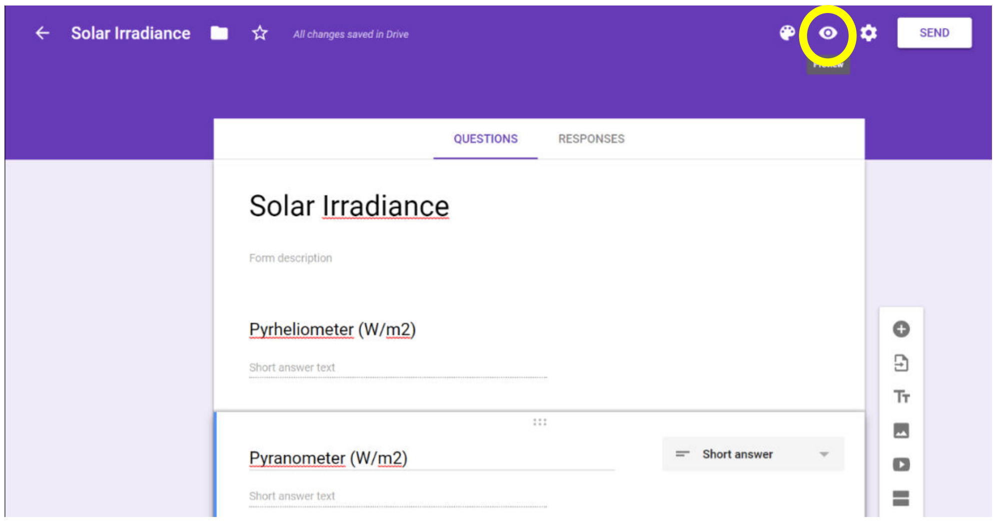
Figure 3.70 : Example of Real-Time Database Exported From Blynk

In this project, Google Spreadsheet is implemented as the database for IoT data collection system. Database is crucial because it is useful for users to analyze data stored from time to time.

The data of both pyrheliometer and pyranometer will be collected every five minutes. Data are not collected instantaneously due to the limited memory space available in Google Spreadsheet. Google Spreadsheet can hold only up to five million cells (Google, 2019). Google Spreadsheet itself has nine rows by default, and hence, there are only 555,555 columns available. In another words, Google Spreadsheet can only collect up to 555,555 sets of data. If the data are collected every second, there will be 86,400 data per day, and the data can be stored for only 6 days. Therefore, data collection per second is not recommended. Besides, it is quite instantaneous to save solar irradiance data every five minutes. Throughout a day, the Sun rises from the East to the West, and it takes hours for the Sun to change its position. Therefore, it is reasonable to save irradiance data every five minutes.

In this project, Pushingbox API is implemented to retrieve irradiance data from ESP32 and upload them to Google Form and Google Spreadsheet for data storage purpose. Pushingbox API is a cloud service to send notifications based on HTTP request such as GET or POST. In this project, Pushingbox API will GET the data from ESP32, and then POST to Google Form and Google Spreadsheet. Pushingbox API is a free cloud service and it allows 1000 request per day, which is enough for data to be uploaded every five minutes throughout a day.

To enable data transmission from ESP32 to Google Spreadsheet, online tutorial from Microcontroller Tutorials (2019) was referred. Firstly, create a form using Google Form as shown in Figure 3.71. Next, click the eye icon which is in the yellow circle shown in Figure 3.71. A preview of the form created will be shown as in Figure 3.72.



Solar Irradiance

Form description

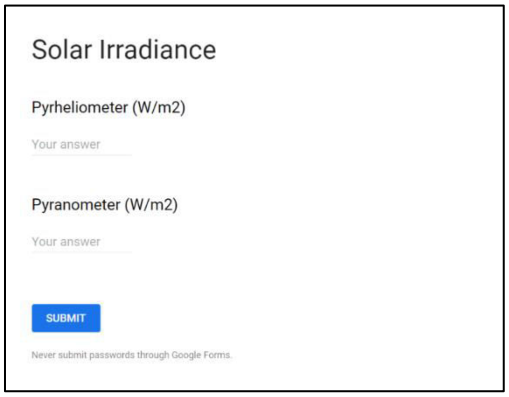
Pyrheliometer (W/m2)

Short answer text

Pyranometer (W/m2)

Short answer text

Figure 3.71: Create a Form in Google Form


Solar Irradiance

Pyrheliometer (W/m2)

Your answer

Pyranometer (W/m2)

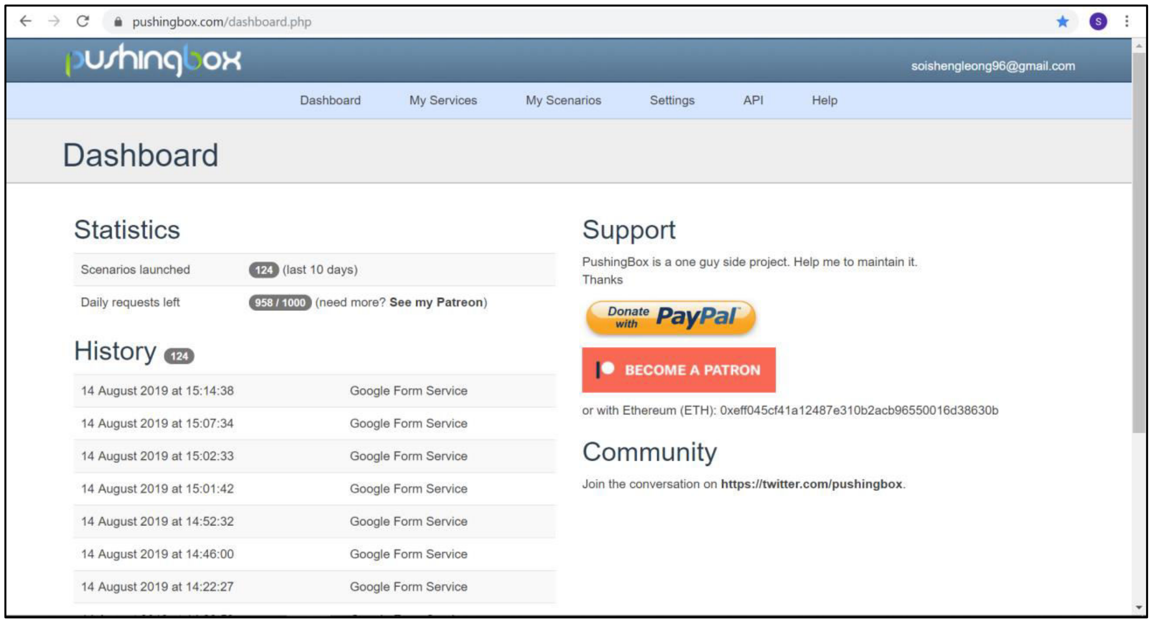
Your answer

SUBMIT

Never submit passwords through Google Forms.

Figure 3.72: Preview of the Form Created

Next, configuration of Pushingbox API service is done. To configure Pushingbox API, go to pushingbox.com and create a new account. Once the new account is created, the dashboard should be similar as in Figure 3.74. From the dashboard, click on "My Services" and then click on "Add a Service". The list of services will be shown as in Figure 3.74. From the list of services, select "CustomURL". In CustomURL Service tab shown in Figure 3.75, select the method as "GET" for Google Form to retrieve data upload to Pushingbox API. The URL of the Google Form preview shown in Figure 3.73 is copied and pasted into the "Root URL" section, but remember to replace the keyword "viewform" in the URL to "formResponse". The step of changing the keyword is to let Pushingbox API to fill up the short answers created (Pyrheliometer(W/m2) and Pyranometer(W/m2)) in Google Form shown in Figure 3.72. The name of the CustomURL service is set to "Google Form Service" in this project as shown in Figure 3.75. Click on the "Update" button once the configuration of CustomURL service is done.


pushingbox.com/dashboard.php

pushingbox

soishengleong96@gmail.com

Dashboard | My Services | My Scenarios | Settings | API | Help

Dashboard

Statistics

Scenarios launched	124 (last 10 days)
Daily requests left	958 / 1000 (need more? **See my Patreon**)

History 124

14 August 2019 at 15:14:38	Google Form Service
14 August 2019 at 15:07:34	Google Form Service
14 August 2019 at 15:02:33	Google Form Service
14 August 2019 at 15:01:42	Google Form Service
14 August 2019 at 14:52:32	Google Form Service
14 August 2019 at 14:46:00	Google Form Service
14 August 2019 at 14:22:27	Google Form Service

Support

PushingBox is a one guy side project. Help me to maintain it.
Thanks

Donate with PayPal

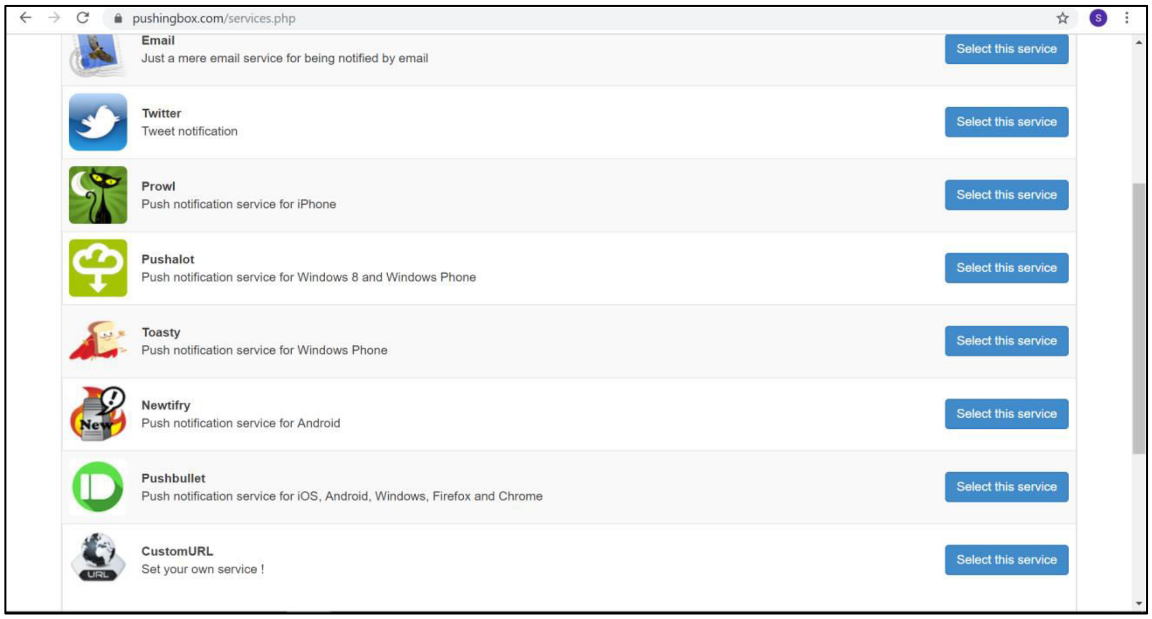
BECOME A PATRON

or with Ethereum (ETH): 0xeff045cf41a12487e310b2acb96550016d38630b

Community

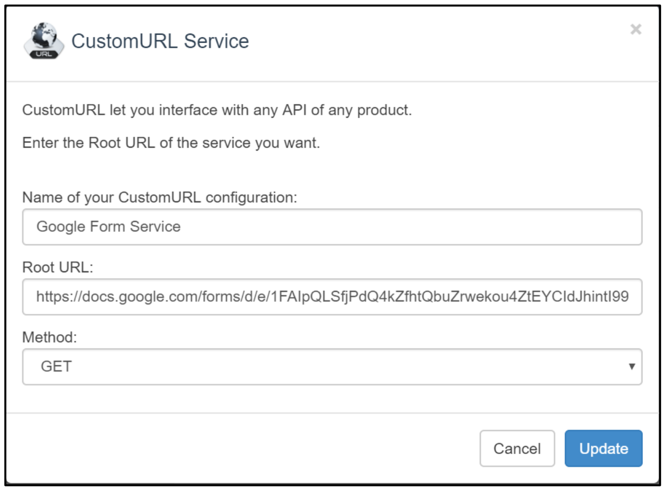
Join the conversation on **https://twitter.com/pushingbox**.

Figure 3.73: Pushingbox API Dashboard


pushingbox.com/services.php

Email Just a mere email service for being notified by email	Select this service
Twitter Tweet notification	Select this service
Prowl Push notification service for iPhone	Select this service
Pushalot Push notification service for Windows 8 and Windows Phone	Select this service
Toasty Push notification service for Windows Phone	Select this service
Newtifry Push notification service for Android	Select this service
Pushbullet Push notification service for iOS, Android, Windows, Firefox and Chrome	Select this service
CustomURL Set your own service !	Select this service

Figure 3.74: List of Services Available in Pushingbox API


CustomURL Service

CustomURL let you interface with any API of any product.

Enter the Root URL of the service you want.

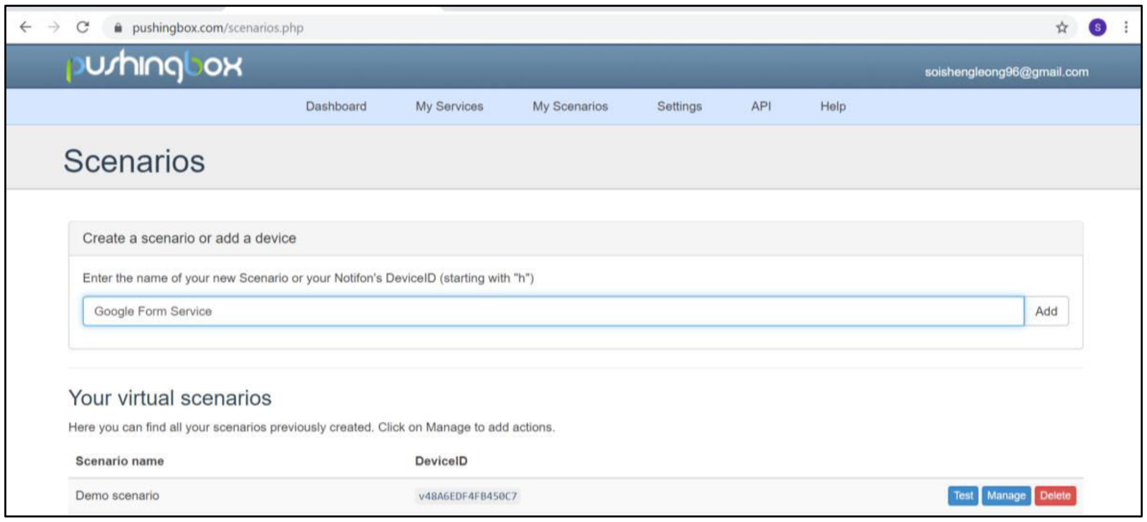
Name of your CustomURL configuration: Google Form Service

Root URL: https://docs.google.com/forms/d/e/1FAIpQLSfjPdQ4kZfhtQbuZrwekou4ZtEYCIdJhintI99

Method: GET

Figure 3.75: Configuration of CustomURL Service

After the configuration of Pushingbox API service is done, click on “My Scenarios” shown in Figure 3.73 to set a notification ID. In “Create a scenario or add a device” tab, enter the name for the scenario and then click on “Add” button to create the scenario as shown in Figure 3.76. For this project, the name of the scenario is set to “Google Form Service”. Next, click on “Add an Action” and then add the CustomURL Service created as action of the scenario shown in Figure 3.77. A message box shown in Figure 3.78 will then appear.


pushingbox.com/scenarios.php

soishengleong96@gmail.com

Scenarios

Create a scenario or add a device

Enter the name of your new Scenario or your Notifon's DeviceID (starting with "h")

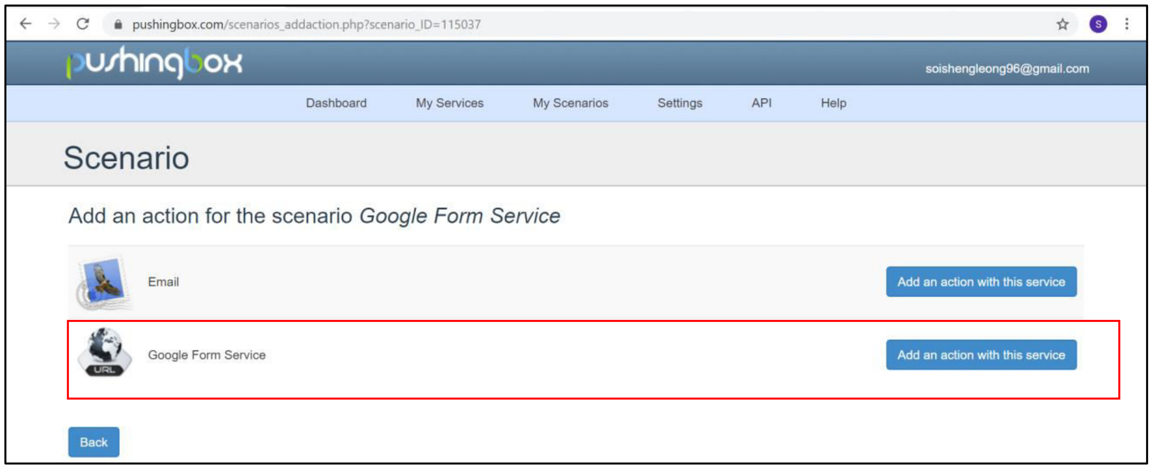
Google Form Service

Your virtual scenarios

Here you can find all your scenarios previously created. Click on Manage to add actions.

Scenario name	DeviceID	
Demo scenario	v4BA6EDF4FB450C7	Test Manage Delete

Figure 3.76: Create a scenario in Pushingbox API


pushingbox.com/scenarios_addaction.php?scenario_ID=115037
pushingbox
soishengleong96@gmail.com
Dashboard
My Services
My Scenarios
Settings
API
Help

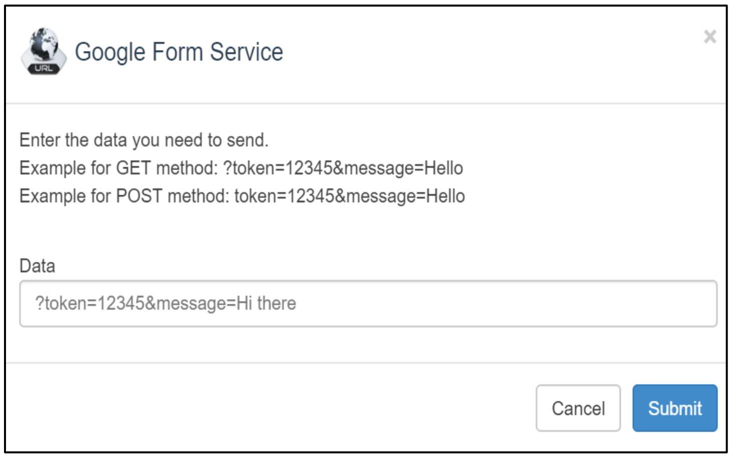
Scenario

Add an action for the scenario *Google Form Service*

Email — Add an action with this service

Google Form Service — Add an action with this service

Figure 3.77: Add CustomURL Service (Google Form Service) as the Action of Scenario


Google Form Service

Enter the data you need to send.
Example for GET method: ?token=12345&message=Hello
Example for POST method: token=12345&message=Hello

Data: ?token=12345&message=Hi there

Figure 3.78: Message Box After Action of Scenario Created

In the message box above, notification ID or task of Pushingbox API to perform its operation has to be set by adding a command line in the column under “Data”. In this project, two solar irradiance data are included in data transmission from ESP32 to Google Form via Pushingbox API, and Google Form will perform GET request to receive data from ESP32. Therefore, the format of the command line must be in the structure shown in Figure 3.79.

```
?name of input field 1 in Google Form=$name of variable 1 in ESP32 Arduino$&name of input field 2 in Google Form=$name of variable 2 in ESP32 Arduino$
```

Figure 3.79: Format of Command Line

According to Appendix O, the name of variable 1 is set as "Pyrheliometer(W/m2)" and the name of variable 2 is set as "Pyranometer(W/m2)". The input fields are the short answers of the preview form in Figure 3.72, which are highlighted in Figure 3.79. To check the names of input fields in Google Form, right click on the respective input field then click "Inspect" to check the name of input field. The "Elements" tab will appear as shown in Figure 3.81. In the blue highlighted region, search the keyword with the format of " name= "entry.xxxxxxx" " and then copy the name of the input field with the format of "entry.xxxxxxx". For this project, the name of input field "Pyrheliometer (W/m2)" is "entry.686762019" and the name of input field "Pyranometer (W/m2)" is "entry.765147090". Therefore, the command line will be "?entry.686762019=$Pyrheliometer(W/m2)$&entry.765147090=$Pyranometer(W/m2)$ " and the message box should be as shown in Figure 3.82. Next, click "Update" button and the action of the scenario is updated.

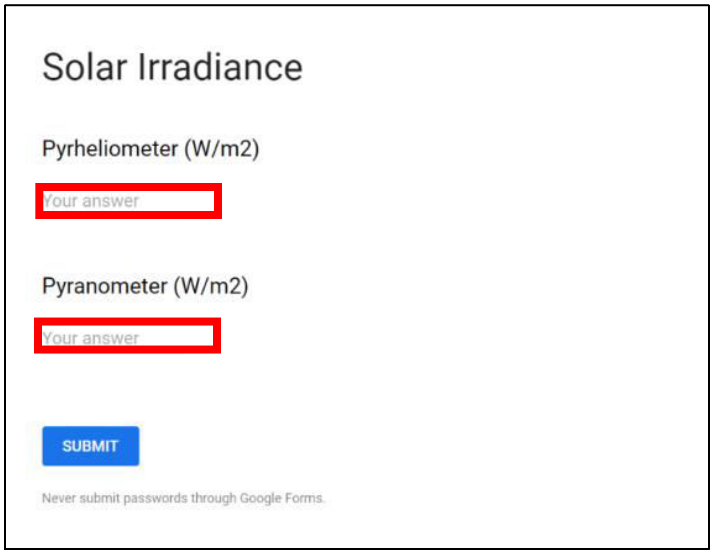
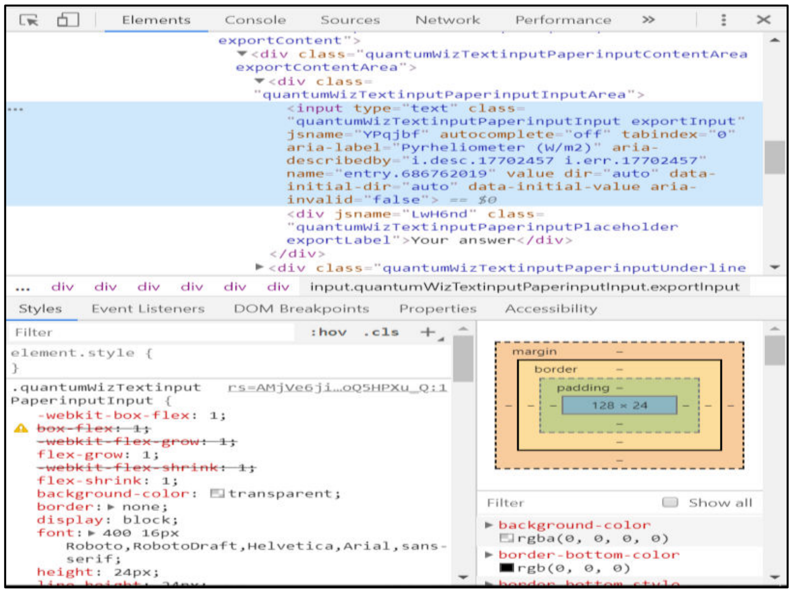


Figure 3.80: Input Fields in the preview form in Google Form



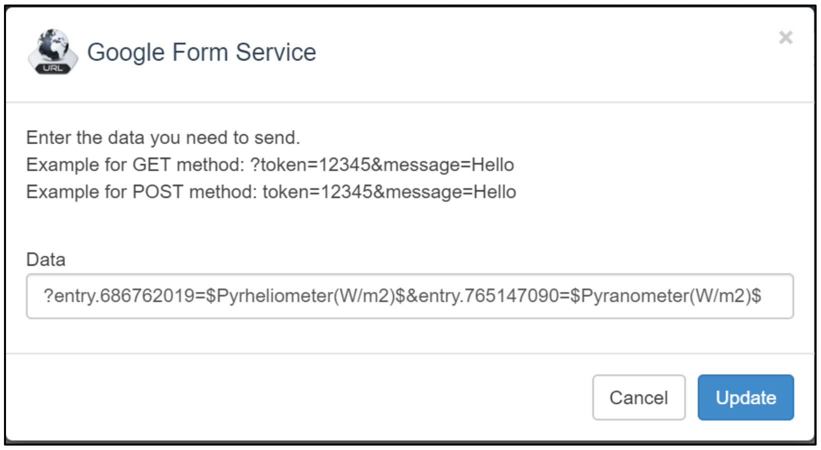
```
exportContent">
  ▼<div class="quantumWizTextinputPaperinputContentArea
  exportContentArea">
    ▼<div class=
    "quantumWizTextinputPaperinputInputArea">
        <input type="text" class=
        "quantumWizTextinputPaperinputInput exportInput"
        jsname="YPqjbf" autocomplete="off" tabindex="0"
        aria-label="Pyrheliometer (W/m2)" aria-
        describedby="i.desc.17702457 i.err.17702457"
        name="entry.686762019" value dir="auto" data-
        initial-dir="auto" data-initial-value aria-
        invalid="false"> == $0
        <div jsname="LwH6nd" class=
        "quantumWizTextinputPaperinputPlaceholder
        exportLabel">Your answer</div>
      </div>
    ►<div class="quantumWizTextinputPaperinputUnderline
```

... div div div div div div input.quantumWizTextinputPaperinputInput.exportInput

```
element.style {
}
.quantumWizTextinput   rs=AMjVe6ji…oQ5HPXu_Q:1
PaperinputInput {
   -webkit-box-flex: 1;
   ~~box-flex: 1;~~
   ~~-webkit-flex-grow: 1;~~
   flex-grow: 1;
   ~~-webkit-flex-shrink: 1;~~
   flex-shrink: 1;
   background-color: ▨transparent;
   border: ► none;
   display: block;
   font: ► 400 16px
      Roboto,RobotoDraft,Helvetica,Arial,sans-
      serif;
   height: 24px;
```

```
► background-color
  ▨rgba(0, 0, 0, 0)
► border-bottom-color
  ■rgb(0, 0, 0)
```

Figure 3.81: “Elements” Tab



Enter the data you need to send.
Example for GET method: ?token=12345&message=Hello
Example for POST method: token=12345&message=Hello

Data

?entry.686762019=$Pyrheliometer(W/m2)$&entry.765147090=$Pyranometer(W/m2)$

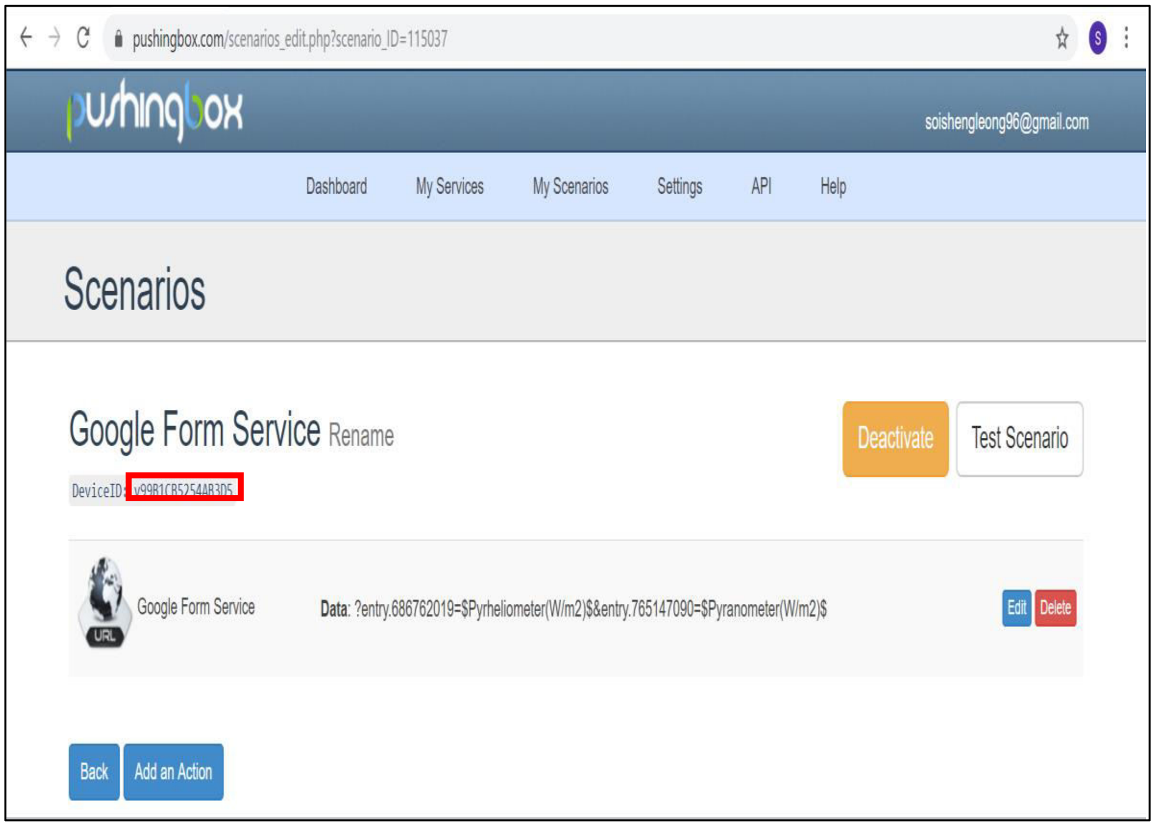
Figure 3.82: Fill in the Command Line and Update the Action of Scenario

Once the action of scenario is updated, the “Scenarios” page should be as shown in Figure 3.84. From the page, copy DeviceID from the scenario of Pushingbox API and then paste it on the highlighted part in the code section of Appendix N shown in Figure 3.83. The configuration of Pushingbox API cloud service is done. To enable ESP32 to send data to Pushingbox API, the name of server host and HTTP port of Pushingbox API has included in global declaration section of Arduino IDE source code in Appendix

O as shown in Figure 3.85. The name of server host of Pushingbox API is “api.pushingbox.com” and the HTTP port is “80”.

```
// We now create a URL for the request
String url = "/pushingbox?";
url += "devid=";
url += "v99B1CB5254AB3D5";
url += "&Pyrheliometer(W/m2)=" + String(Pyrheliometer);
url += "&Pyranometer(W/m2)=" + String(Pyranometer);
```

Figure 3.83: Code Section For ESP32 Wi-Fi module to Transfer Data to Pushingbox API


pushingbox.com/scenarios_edit.php?scenario_ID=115037

soishengleong96@gmail.com

Scenarios

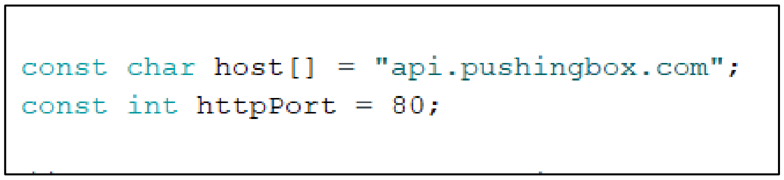
Google Form Service Rename

DeviceID: v99B1CB5254AB3D5

Google Form Service

Data: ?entry.686762019=$Pyrheliometer(W/m2)$&entry.765147090=$Pyranometer(W/m2)$

Figure 3.84: Copy DeviceID of the Scenario



```
const char host[] = "api.pushingbox.com";
const int httpPort = 80;
```

Figure 3.85: Code Section to Declare the Server Host and HTTP Port of Pushingbox API

Once the configuration of Pushingbox API is done, click on "RESPONSES" in Figure 3.71. Next, click on the icon in the red circle shown in Figure 3.86, and create a new spreadsheet. The created spreadsheet is linked to Google Form. The created spreadsheet is empty by default which is shown in Figure 3.87, and the data are ready to be stored.

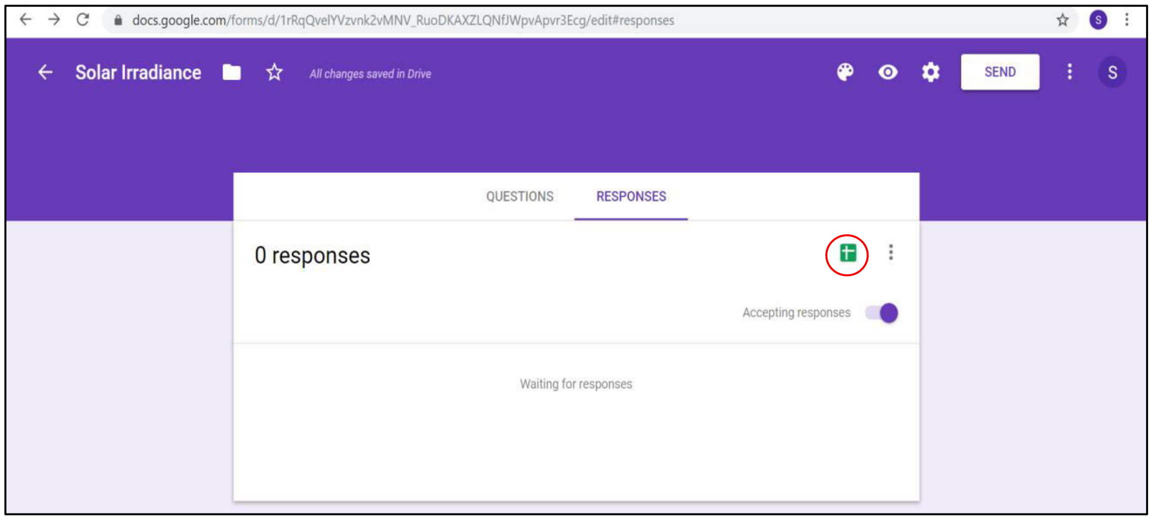

docs.google.com/forms/d/1rRqQvelYVzvnk2vMNV_RuoDKAXZLQNfJWpvApvr3Ecg/edit#responses
Solar Irradiance
All changes saved in Drive
SEND
QUESTIONS
RESPONSES
0 responses
Accepting responses
Waiting for responses

Figure 3.86: Create a Spreadsheet To Link With Google Form

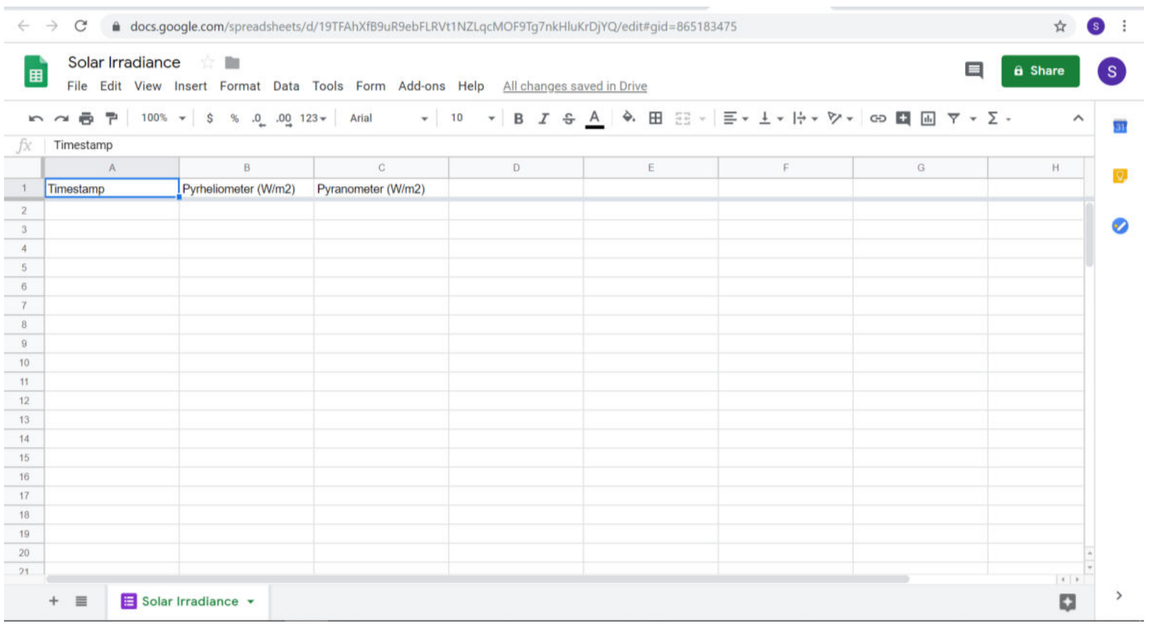

docs.google.com/spreadsheets/d/19TFAhXfB9uR9ebFLRVt1NZLqcMOF9Tg7nkHluKrDjYQ/edit#gid=865183475
Solar Irradiance
File Edit View Insert Format Data Tools Form Add-ons Help
All changes saved in Drive
Share
Timestamp
Pyrheliometer (W/m2)
Pyranometer (W/m2)
Solar Irradiance

Figure 3.87: Google Spreadsheet Created

In this project, the database of IoT data collection system is accessible to public to view the data. To enable this property, click on "Share" button and a message box shown in Figure 3.88 will appear. Click on "Get shareable link" to enable public access to the database, and then get the shareable link. The shareable link will be useful for configuration of "Database Access" button in Blynk user platform shown in Figure 3.65.

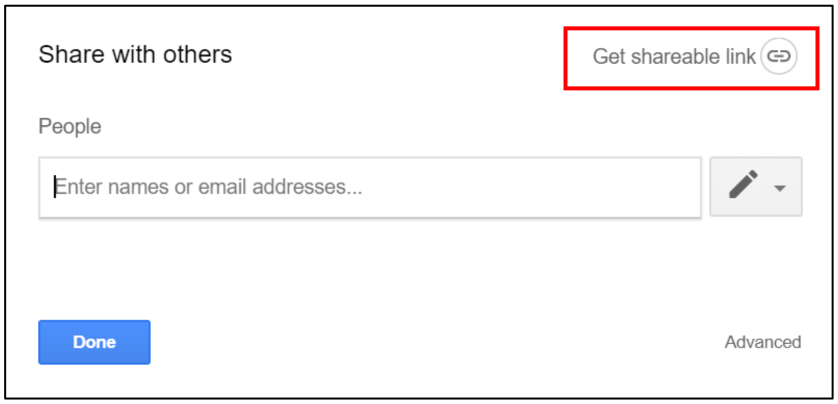


Figure 3.88: Get Shareable Link For Public Access

3.4.7 LED Indicators In IoT Data Collection System

There are three LEDs used as indicators in IoT data collection system. The purpose of adding indicators is to indicate Wi-Fi connection status and data transmission status. The LEDs are super bright LEDs with different colours, which are blue, white and yellow. When ESP32 is not connected to the hotspot from ESP8266 Wi-Fi extender, the blue super bright LED will keep blinking while the rest of the LEDs are ‘OFF’. Once the ESP32 is connected to the hotspot, the blue super bright LED will stop blinking and light up. Both yellow and white super bright LEDs function as data transmission indicators. The white super bright LED blinks once if data is uploaded to Blynk, while the yellow super bright LED blinks once if data is uploaded to Google Spreadsheet.

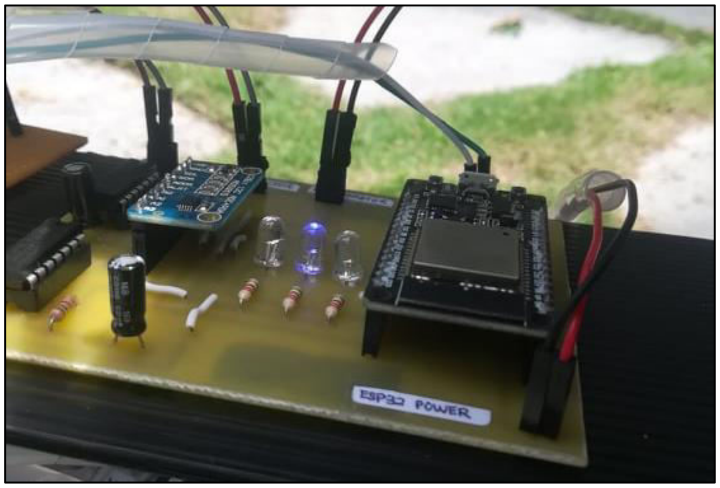


Figure 3.89 : Blue Super Bright LED Lights Up When Connected to Hotspot

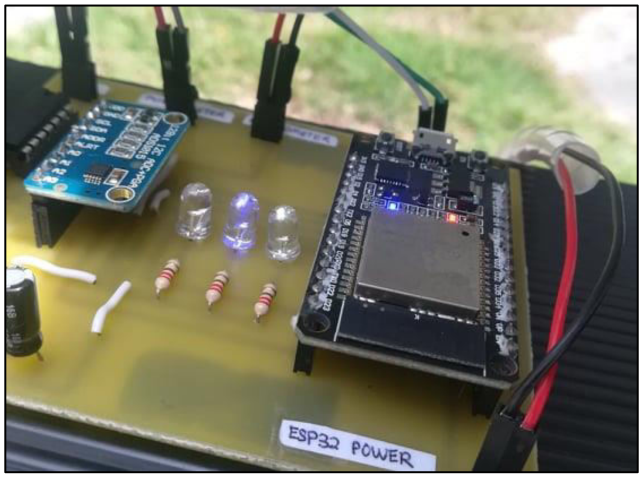


Figure 3.90 : White Super Bright LED Blinks Once When Data Uploaded To Blynk

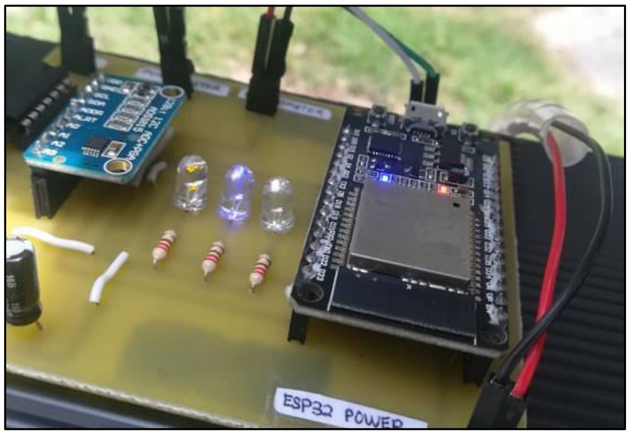


Figure 3.91 : Yellow Super Bright LED Blinks Once When Data Uploaded To Google Spreadsheet

3.4.8 Schematic Diagram and PCB Design for IoT Data Collection System

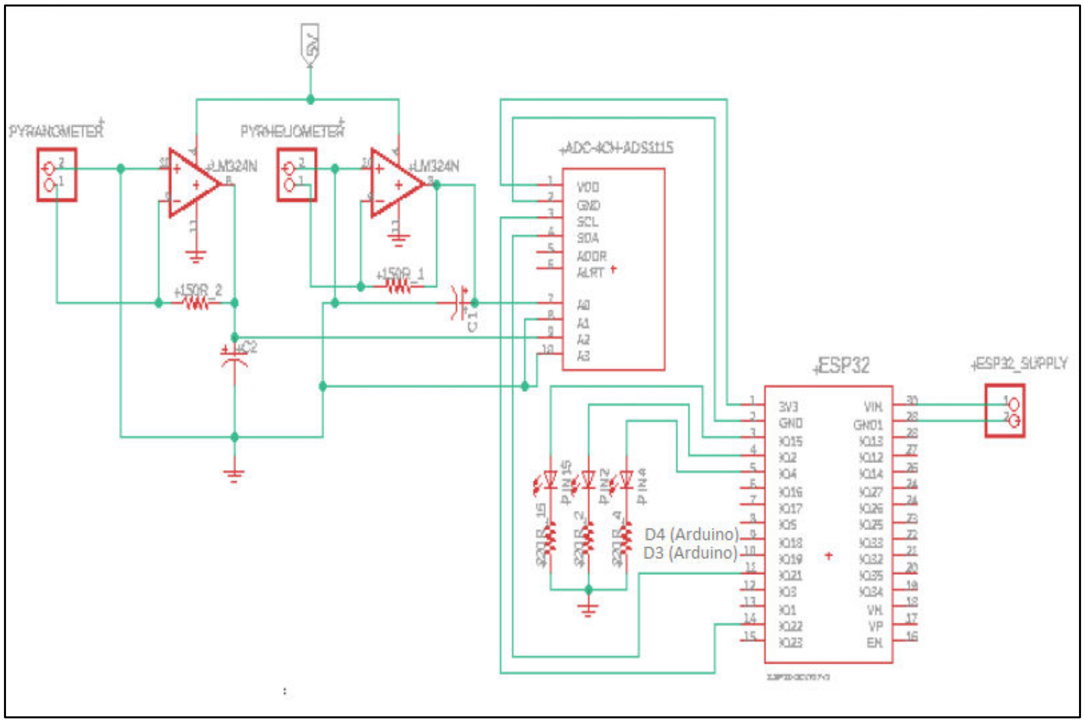


Figure 3.92 : Schematic Diagram for IoT Data Collection System

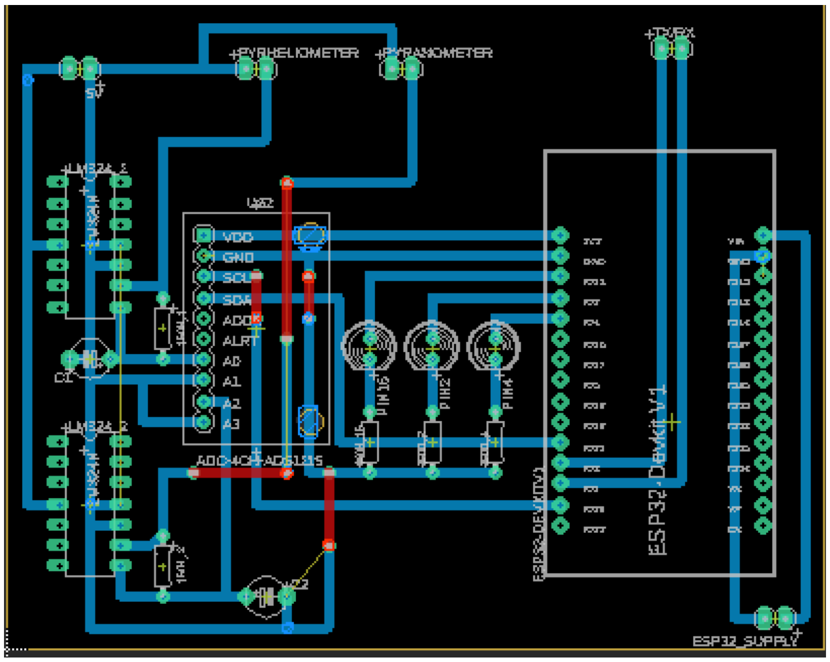


Figure 3.93 : Eagle PCB Layout for IoT Data Collection System

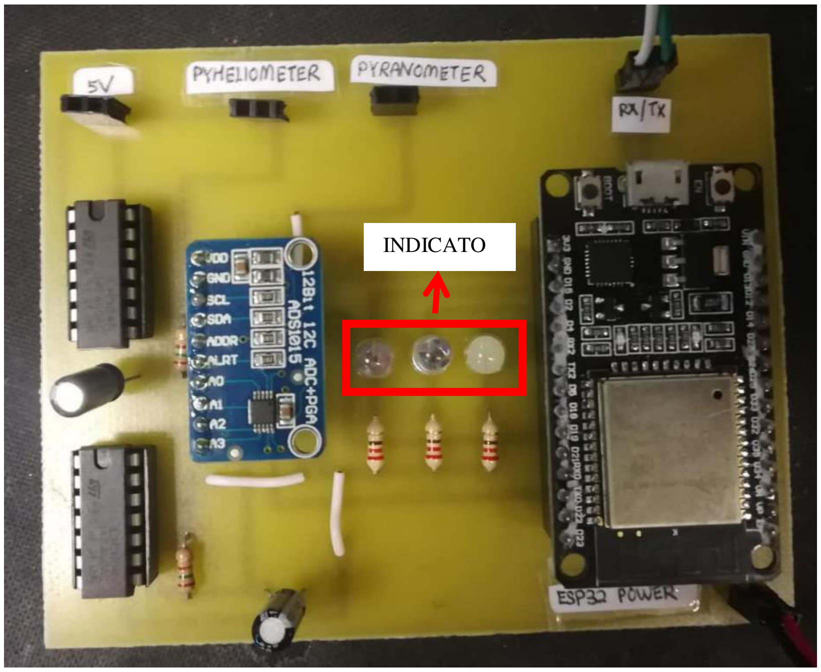


Figure 3.94 : PCB Board for IoT Data Collection System

3.5 Equipment and Cost Analysis

Table 3.3: Cost Analysis for Components and Equipment

No.	Component / Equipment	Quantity	Unit Price (RM)	Total Price (RM)	Remarks
1	L Bracket 30*30mm	1	1.70	1.70	From MR. D.I.Y.
2	Cable Clip 10mm - 70S	1	2.08	2.08	From MR. D.I.Y.
3	3" S/S L Bracket	2	4.90	9.80	From MR. D.I.Y.
4	8pc Bolt Nut	1	1.50	1.50	From MR. D.I.Y.
5	Terminal Block 3A 80003	3	2.10	6.30	From MR. D.I.Y.

6	Bulb 12V 55WA-H4	1	12.70	12.70	From MR. D.I.Y.
7	Pet Necklace	2	3.60	7.20	From MR. D.I.Y.
8	IVON Charger Set XX-V8	1	8.74	8.74	From MR. D.I.Y.
9	Plastic Plug With Hole 3A	2	0.94	1.88	From MR. D.I.Y.
10	IVON Charger AD14-V8	2	5.90	11.80	From MR. D.I.Y.
11	LWD 5Y 2G Cable Box	1	29.90	29.90	From MR. D.I.Y
12	Impra Board	3	7.90	23.70	From Super Galaksi Enterprise and Miracle Stationary & Sport Centre
13	LM324 IC	2	2.30	4.60	From Destiny Electronic Centre
14	14 Pin IC Socket	2	0.40	0.80	From Destiny Electronic Centre
15	Veroboard	1	3.80	3.80	From Destiny Electronic Centre
16	16 Pin Single Row Female	2	2.10	4.20	From Destiny Electronic Centre
17	PVC Cloth	1	20.11	20.11	From Kamdar Sdn Bhd
18	Mens Black T-shirt	1	19.90	19.90	From Tesco Kampar
19	Jumper Wire Male to Male 20cm	1	3.50	3.50	From FabGear Engineering Enterprise

20	Jumper Wire Male to Female 20cm	1	3.50	3.50	From FabGear Engineering Enterprise
21	Jumper Wire Male to Female 30cm	2	4.50	9.00	From FabGear Engineering Enterprise
22	Jumper Wire Male to Male 30cm	1	4.50	4.50	From FabGear Engineering Enterprise
23	LDR	4	1.00	4.00	From Tonsin Component
24	8mm Spiral Wrap	1	10.50	10.50	From Tonsin Component
25	7/0.2mm Tinned PVC SGL Black - 100 meter	1	27.00	27.00	From States Electronic Sdn Bhd
26	Nylon Spiral Wrapping - White / 8mm	1	6.00	6.00	From States Electronic Sdn Bhd
27	ESP32 Wi-Fi module	1	43.90	43.90	From SG Robot Technology Enterprise
28	Arduino UNO R3	1	26.50	26.50	From SG Robot Technology Enterprise
29	ADS1015 12-bit Precision ADC module	1	21.90	21.90	From SG Robot Technology Enterprise
Total Cost:				**RM 331.01**	

3.6 Project Milestone

Gantt Chart for FYP 1:

Table 3.4: Gantt Chart for FYP 1

Week **Task**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Title Selection	■	■												
Preliminary Work and Literature Review			■	■										
Component Selection			■	■	■							■	■	■
Coding for the tracker to move automatically and troubleshooting					■	■	■	■	■	■	■			
Progress Report Writing												■		
Submission of Progress Report													■	
FYP 1 Oral Presentation														■

Gantt Chart for FYP 2:

Table 3.5: Gantt Chart for FYP 2

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCB design and soldering components	▨														
Calibration and data collection of solar irradiance sensors	▨	▨	▨	▨	▨	▨	▨								
Create IoT user interface	▨	▨	▨	▨	▨	▨	▨								
Troubleshooting the whole project								▨	▨	▨	▨	▨			
Report Writing							▨	▨	▨	▨	▨	▨			
Submission of Final Report													▨		
Poster Presentation													▨		
FYP 2 Oral Presentation														▨	
Hard Bound Submission															▨

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

In this chapter, results from various stages done were obtained and analyzed. The problems faced and improvement done were illustrated and discussed in this chapter as well.

4.2 Accuracy in Reading Input Current Signals

To enable ESP32 in determining the current produced by both solar cells given, current-to-voltage converters have to be constructed as ESP32 is able to read only voltage input signals.

Before constructing current-to-voltage converters, the accuracy of voltage signal reading by ESP32 was tested. Although ESP32 has analog input pins with 12-bit resolution according to RandomNerdTutorials.com (2013), the accuracy of ESP32 in reading input voltage signal was relatively low due to its non-linear behaviour in reading voltage input signal as shown in Figure 4.1. Therefore, ADS1015 ADC module was added and connected to ESP32. Accuracy of ADS1015 ADC module was relatively high as it has very high voltage reading resolution of 2 mV per bit. The accuracy of ESP32 itself and ESP32 with ADS1015 ADC module in reading input voltage signal were recorded as shown in Table 4.1 and Table 4.2 respectively.

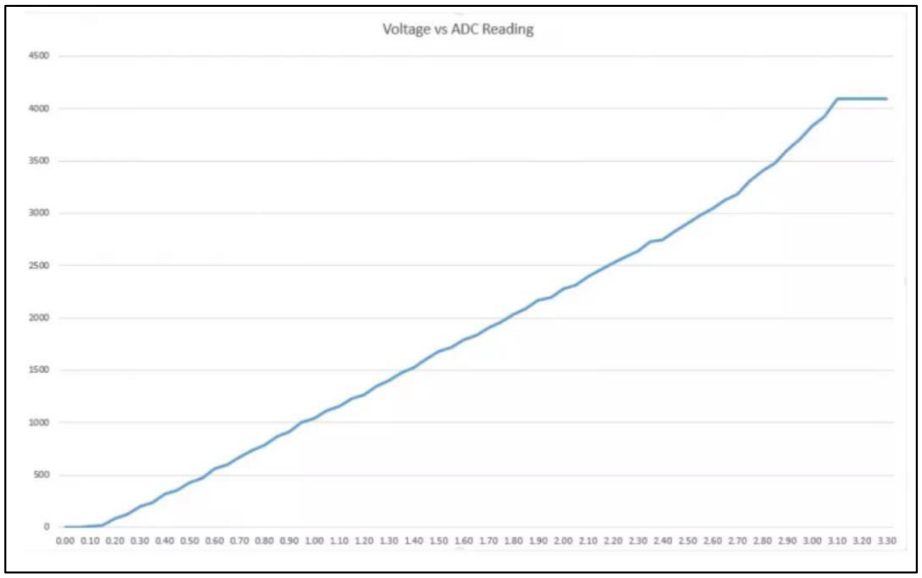


Figure 4.1: Performance of ESP32 Wi-Fi module in Reading Input Voltage Signal (RandomNerdTutorials.com, 2013)

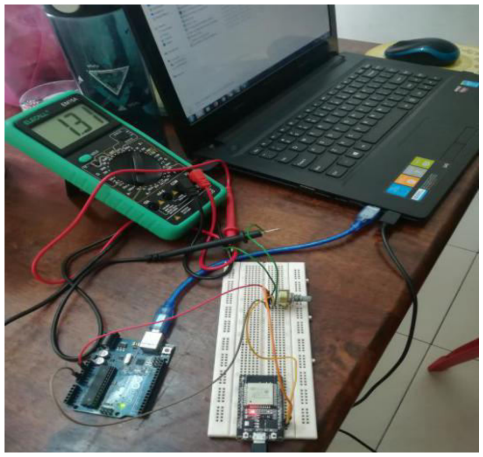


Figure 4.2: ESP32 Wi-Fi Module in Reading Input Voltage Signal

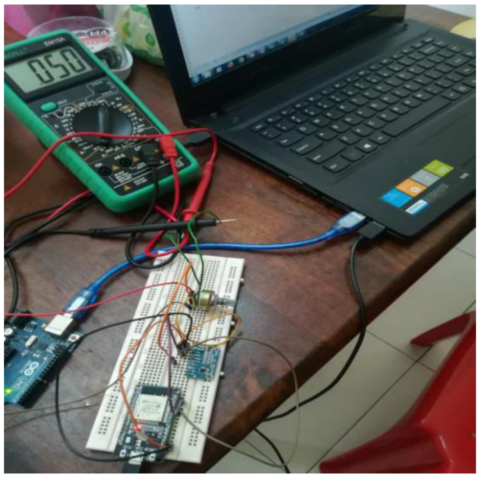


Figure 4.3: ESP32 Wi-Fi Module with ADS1015 ADC Module in Reading Input Voltage Signal

Table 4.1: Accuracy of ESP32 Wi-Fi Module In Reading Input Voltage Signal

Actual Voltage (V)	Measured Voltage (V)	Difference (V)	Accuracy(%)
0.002	0	0.002	0
0.003	0	0.003	0
0.01	0	0.01	0
0.05	0	0.05	0
0.1	0	0.1	0
0.5	0.35 ~ 0.39	0.11 ~ 0.15	70 ~ 78
1	0.87 ~ 0.9	0.1 ~ 0.13	87 ~ 90
1.5	1.35 ~ 1.38	0.12 ~ 0.15	90 ~ 92
2	1.85 ~ 1.88	0.12 ~ 0.15	92.5 ~ 94
2.5	2.34 ~ 2.38	0.12 ~ 0.16	93.6 ~ 95.2
3	2.9 ~ 3.04	0.04 ~ 0.1	96.67 ~ 98.67
3.3	3.3	0	100

Table 4.2: Accuracy of ESP32 Wi-Fi Module with ADS1015 ADC Module In Reading Input Voltage Signal

Actual Voltage (V)	**Measured Voltage (V)**	**Difference (V)**	**Accuracy(%)**
0.002	0.002	0	100
0.003	0.002	0.001	66.66666667
0.01	0.01	0	100
0.05	0.056	0.006	88
0.1	0.104	0.004	96
0.5	0.502	0.002	99.6
1	1.006	0.006	99.4
1.5	1.502	0.002	99.86666667
2	2.008	0.008	99.6
2.5	2.506	0.006	99.76
3	3.01	0.01	99.66666667
3.3	3.3	0	100

From both of the tables above, ADS1015 ADC module was far more accurate compared to ESP32 in reading input voltage signal, where the percentage of accuracy is higher than 90 % for almost every voltage value recorded. From Table 4.1, ESP32 was able to read the input voltage signal starting from 0.5 V, which was relatively ineffective compared to ADS105 ADC module. Furthermore, ESP32 was not stable in reading voltage values as the measured values fluctuate upon a constant input voltage, and was only able to give a stable reading when the input voltage was at 3.3 V. Therefore, ADS1015 ADC module was connected to ESP32 due to its high accuracy in reading input voltage signal.

The current-to-voltage converter circuitry for both solar cells were then constructed and connected to ADS1015 ADC module. The ADC module would then measure the converted voltage signals of both solar cells and send to ESP32 for conversion of voltage signals to current respectively. The calculation for current signals by ESP32 according to converted voltage signals were done with the formula shown in Equation 4.1.

$$i = \frac{V}{R} \times 1000 \tag{4.1}$$

where

i = Current calculated by ESP32, mA

V = Converted voltage measured by ADS1015 ADC module, V

R = Resistance of feedback resistor of current-to-voltage converter, Ω

Although 150 Ω resistors were used for the current-to-voltage converters respectively, it would be better to include the actual resistance value for respective values of R in Equation 4.1. The reason of including the actual resistance was that the resistors have tolerance itself and will not be exactly 150 Ω. Furthermore, considering the actual resistance value can help to improve the accuracy in obtaining the current values.

To obtain current signal from the solar cells, both 12 V light bulb and 12 V halogen light bulb were used to shine on the solar cells. These light bulbs were bright enough for the solar cells to produce current. With the method implemented, the author did not need to test the current-to-voltage circuitry under the Sun. The actual current and converted voltage were measured using multimeters, and the voltage received and the current calculated by ESP32 were recorded as well. Comparisons of voltage and current reading for both solar cells were done as shown in the tables below.

Figure 4.4: 12 V Light Bulb

Figure 4.5: 12 V Halogen Light Bulb

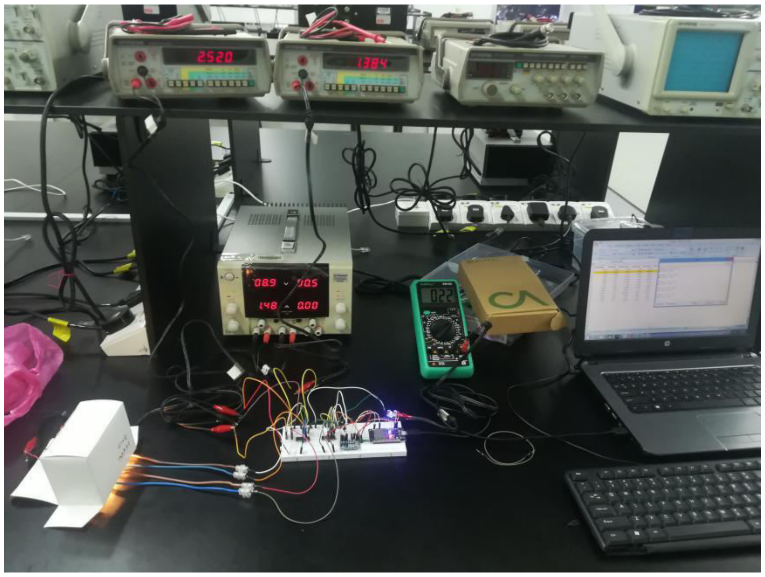


Figure 4.6: Comparison Done on Actual and Measured Voltage and Current Values of Solar Cells

Table 4.3: Comparison of Actual and Measured Voltage Values of First Solar Cell

No.	Actual Voltage (V)	Measured Voltage (V)	Difference (V)	Accuracy (%)
1	0.002	0.002	0	100
2	0.006	0.008	0.002	66.66666667
3	0.018	0.02	0.002	88.88888889
4	0.08	0.08	0	100
5	0.155	0.154	0.001	99.35483871
6	0.24	0.244	0.004	98.33333333
7	0.35	0.354	0.004	98.85714286
8	0.92	0.928	0.008	99.13043478
9	1.18	1.19	0.01	99.15254237
10	1.5	1.502	0.002	99.86666667
11	2.01	2.016	0.006	99.70149254
12	2.26	2.26	0	100
13	2.7	2.702	0.002	99.92592593

Table 4.4: Comparison of Actual and Calculated Current Values of First Solar Cell

No.	Actual Current (mA)	Calculated Current (mA)	Difference (mA)	Accuracy (%)
1	0.02	0.1215	0.1015	-407.5
2	0.05	0.054	0.004	92
3	0.132	0.135	0.003	97.72727273
4	0.536	0.5403	0.0043	99.19776119
5	1.046	1.0401	0.0059	99.43594646
6	1.644	1.648	0.004	99.756691
7	2.39	2.391	0.001	99.958159
8	6.266	6.268	0.002	99.96808171
9	8.043	8.0378	0.0052	99.93534751
10	10.151	10.1452	0.0058	99.94286277
11	13.621	13.617	0.004	99.97063358
12	15.343	15.2651	0.0779	99.49227661
13	18.3	18.2505	0.0495	99.7295082

Table 4.5: Comparison of Actual and Measured Voltage Values of Second Solar Cell

No.	Actual Voltage (V)	Measured Voltage (V)	Difference (V)	Accuracy (%)
1	0.01	0.01	0	100
2	0.014	0.008	0.006	57.14285714
3	0.018	0.018	0	100
4	0.076	0.076	0	100
5	0.153	0.154	0.001	99.34640523
6	0.22	0.226	0.006	97.27272727
7	0.37	0.38	0.01	97.2972973
8	0.95	0.954	0.004	99.57894737

9	1.21	1.22	0.01	99.17355372
10	1.57	1.572	0.002	99.87261146
11	2.03	2.034	0.004	99.80295567
12	2.28	2.198	0.082	96.40350877
13	2.69	2.698	0.008	99.70260223

Table 4.6: Comparison of Actual and Calculated Current Values of Second Solar Cell

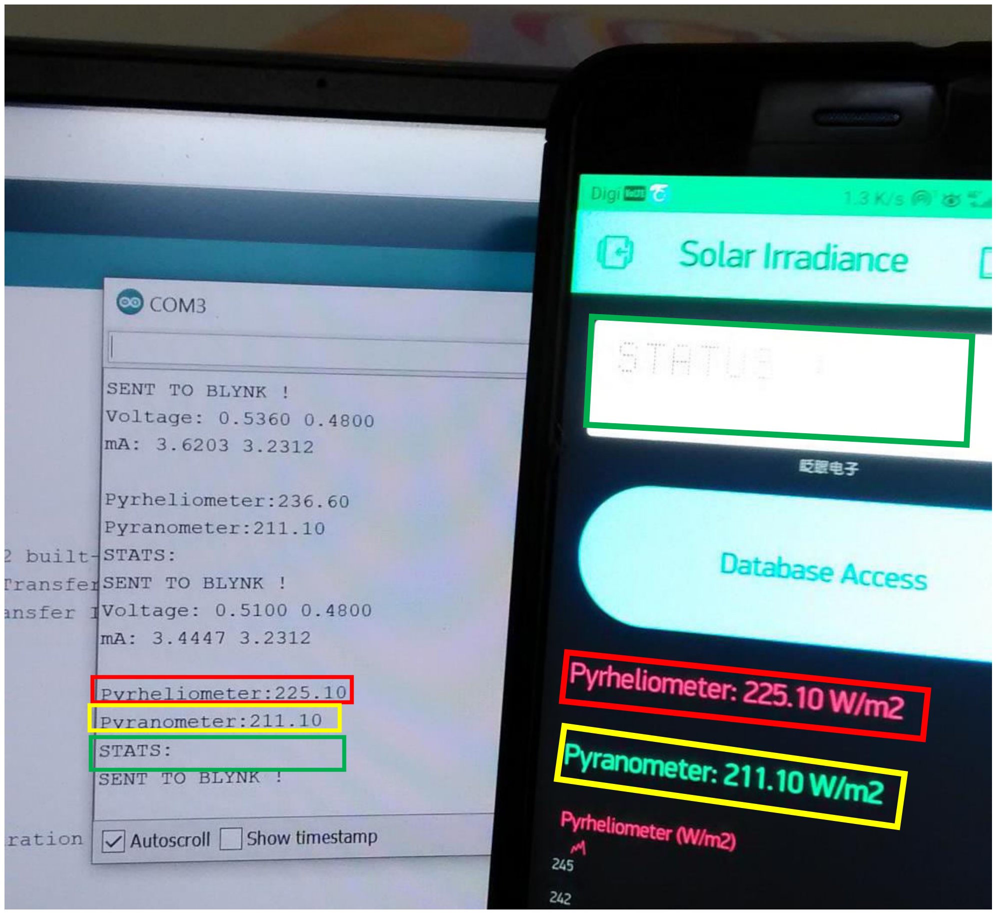
No.	**Actual Current (mA)**	**Calculated Current (mA)**	**Difference (mA)**	**Accuracy (%)**
1	0.015	0.0673	0.0523	-248.6666667
2	0.055	0.054	0.001	98.18181818
3	0.117	0.1211	0.0041	96.4957265
4	0.507	0.5116	0.0046	99.09270217
5	1.026	1.0366	0.0106	98.9668616
6	1.512	1.5213	0.0093	99.38492063
7	2.544	2.558	0.014	99.44968553
8	6.406	6.422	0.016	99.75023416
9	8.185	8.2127	0.0277	99.66157605
10	10.59	10.5822	0.0078	99.92634561
11	13.69	13.6923	0.0023	99.98319942
12	15.4	14.7963	0.6037	96.07987013
13	18.17	18.1622	0.0078	99.9570721

From the tables above, 13 sets of data were recorded and compared for two solar cells. To determine the accuracy in reading current values, the calculated current values were compared with the actual current values respectively. The corresponding measured output voltages were also measured to verify whether the calculated current values were correct according to the formula in Equation 4.1, and to find out the reason of inaccuracy in current reading for the current values less than 0.05 mA as shown in both Table 4.4 and Table 4.6. The accuracy of the calculated current values in both Table 4.4 and Table 4.5 were relatively high, where almost all current values recorded have percentage of accuracy exceeding 90 %. However, inaccuracies occurred for current values lower than 0.05 mA as shown in both Table 4.4 and Table 4.5. This phenomena happened due to small output offset voltages from both LM324 ICs which are implemented as the current-to-voltage converter.

In conclusion, ADS1015 ADC module helped ESP32 in reading input voltage signals accurately, and thus helped in achieving the objective of building a high precision current sensing circuitry.

4.3 Accuracy of IoT Data Transmission

Before started to build up the IoT data collection system, source code shown in Appendix O was uploaded to ESP32, and the accuracy of IoT data transmission was tested by performing serial monitor using USB cable. From the serial monitor test, it was found that the data transmitted were precise as long ESP32 was connected to the hotspot provided the ESP8266 LiLon NodeMCU V3 Wi-Fi extender. This can be proven with the pictures in next two pages.

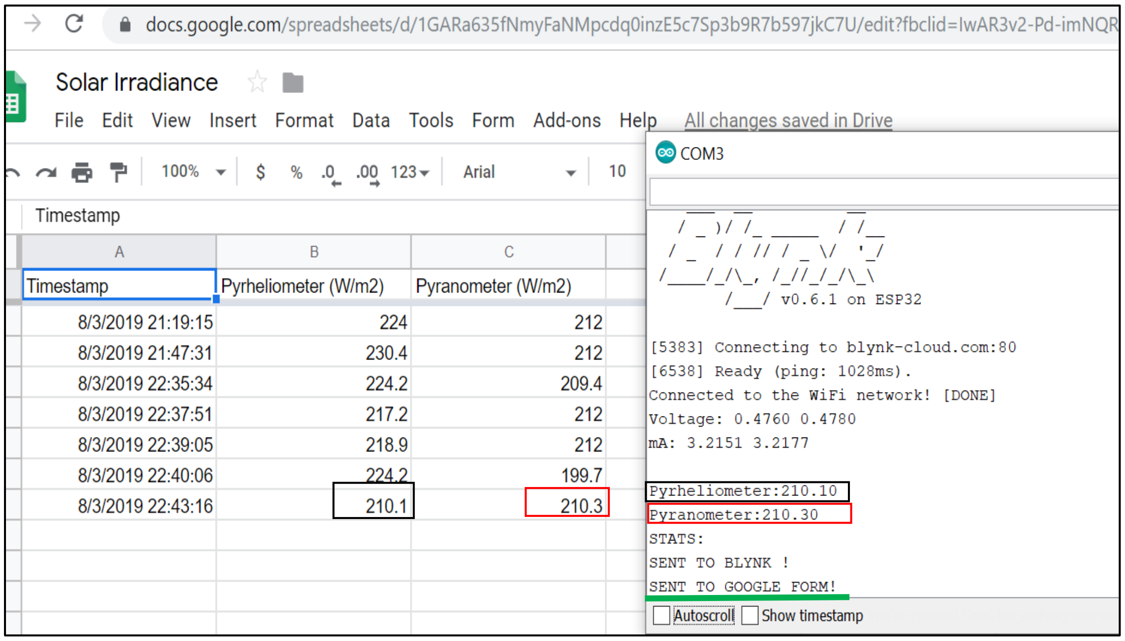


```
SENT TO BLYNK !
Voltage: 0.5360 0.4800
mA: 3.6203 3.2312

Pyrheliometer:236.60
Pyranometer:211.10
STATS:
SENT TO BLYNK !
Voltage: 0.5100 0.4800
mA: 3.4447 3.2312

Pyrheliometer:225.10
Pyranometer:211.10
STATS:
SENT TO BLYNK !
```

Figure 4.7: The Real Time Data Displayed on Blynk App Matched The Data Displayed on the Serial Monitor in Arduino IDE Software


docs.google.com/spreadsheets/d/1GARa635fNmyFaNMpcdq0inzE5c7Sp3b9R7b597jkC7U/edit?fbclid=IwAR3v2-Pd-imNQR

Solar Irradiance

Timestamp	Pyrheliometer (W/m2)	Pyranometer (W/m2)
8/3/2019 21:19:15	224	212
8/3/2019 21:47:31	230.4	212
8/3/2019 22:35:34	224.2	209.4
8/3/2019 22:37:51	217.2	212
8/3/2019 22:39:05	218.9	212
8/3/2019 22:40:06	224.2	199.7
8/3/2019 22:43:16	210.1	210.3

COM3

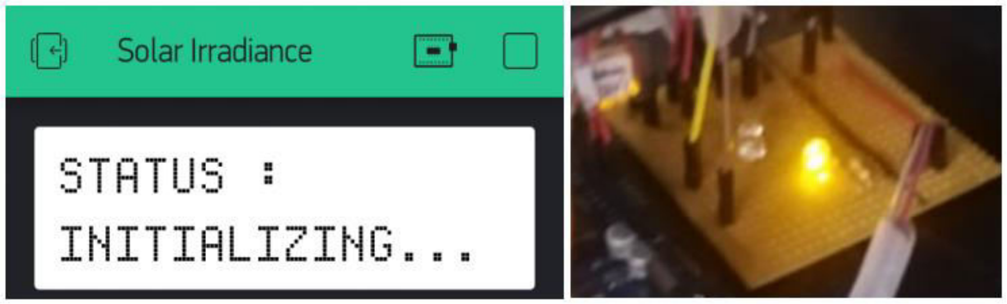
```
    ___  __          __
   / _ )/ /_ _____  / /__
  / _  / / // / _ \/  '_/
 /____/_/\_, /_//_/_/\_\
        /___/ v0.6.1 on ESP32

[5383] Connecting to blynk-cloud.com:80
[6538] Ready (ping: 1028ms).
Connected to the WiFi network! [DONE]
Voltage: 0.4760 0.4780
mA: 3.2151 3.2177

Pyrheliometer:210.10
Pyranometer:210.30
STATS:
SENT TO BLYNK !
SENT TO GOOGLE FORM!
```

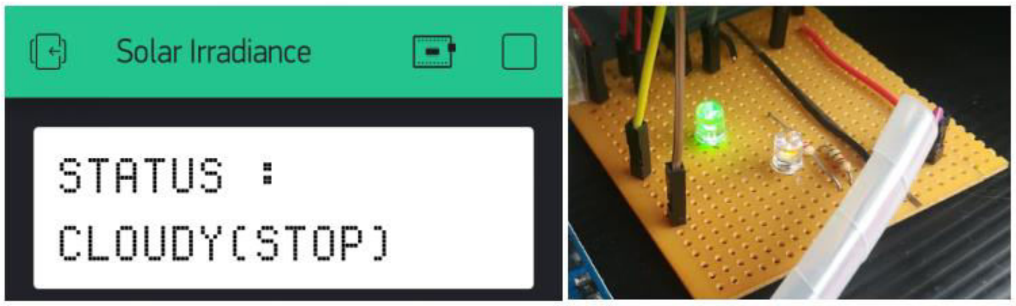
Figure 4.8: The Data in Google Spreadsheet Matched The Data Displayed on Serial Monitor in Arduino IDE Software When Data were Uploaded to Google Spreadsheet

After serial monitoring using Arduino IDE software, the IoT data collection system was then constructed and combined with the circuitry for solar tracking mechanism. After the combination of circuits was done, another test on IoT data transmission was performed. In this test, the display of indicator box in Blynk user interface platform was compared with the LED indicators for the solar tracking mechanism. The LED indicators indicate four different status of the solar tracker. If the solar tracker is initializing, only yellow super bright LED will light up. If the solar tracker stops in the cloudy environment, both yellow and green super bright LEDs will light up. If the solar tracker stops and faces towards the Sun, only green super bright LED will light up. If the solar tracker is tracking, both super bright LEDs will turn off. From this test, it was verified that IoT data transmission was precise as long ESP32 was connected to the hotspot from the Wi-Fi extender. The pictures shown in next two pages proof the accuracy of IoT data transmission.



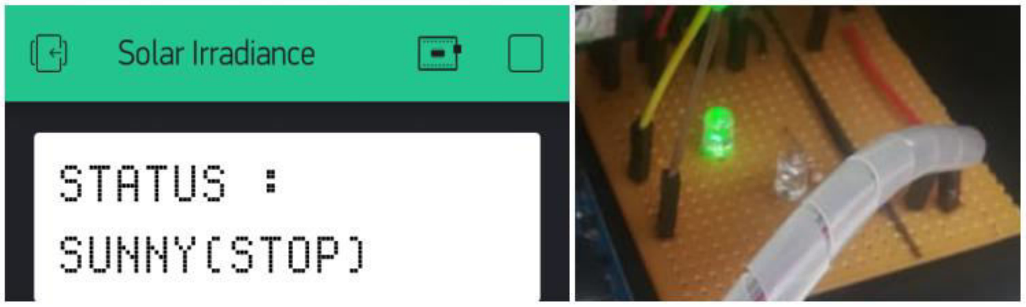
STATUS :
INITIALIZING...

Figure 4.9: Blynk Shows “INITIALIZING…” When The Solar Tracker is Initializing Its Position



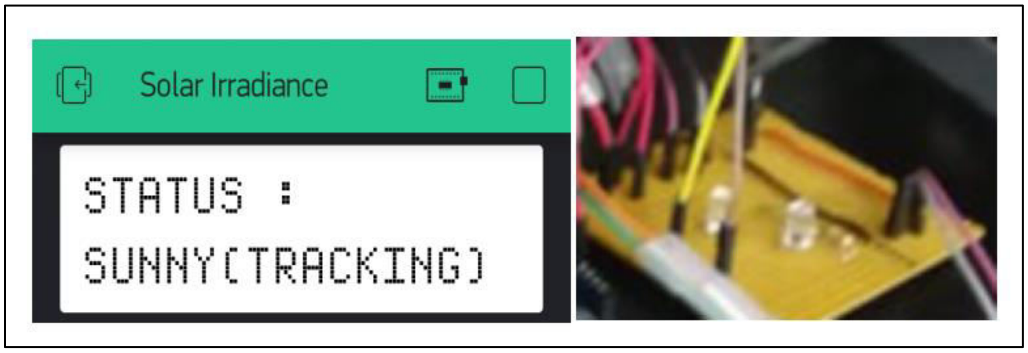
STATUS :
CLOUDY(STOP)

Figure 4.10: Blynk Shows “CLOUDY(STOP)” When The Solar Tracker Stops in Cloudy Environment



STATUS :
SUNNY(STOP)

Figure 4.11: Blynk Shows “SUNNY(STOP)” When The Solar Tracker is Facing The Sun



STATUS :
SUNNY(TRACKING)

Figure 4.12: Blynk Shows "SUNNY(TRACKING)" When The Solar Tracker is Tracking

From the two tests done, it can be concluded that the data will be transmitted precisely as long the IoT device is connected to the Internet.

4.4 Accuracy of Solar Tracking Mechanism

The mechanism of solar tracker is easy to understand theoretically. However, it is not easy to ensure the solar tracker to track accurately using LDRs. This is because the resistance of LDR is not linearly proportional to illumination as shown in Figure 4.13. Furthermore, LDR is a type of resistor and will have tolerance as normal resistors. In another words, the four LDRs implemented will not give the same resistance values. It is even worse when the LDRs are connected in series with resistors to form voltage-divider circuits respectively as shown in Figure 3.15. The colour code for the resistors implemented are same, but the respective resistance value will differ due to tolerance. Hence, this brings great challenge in doing calibration on LDR circuitry.

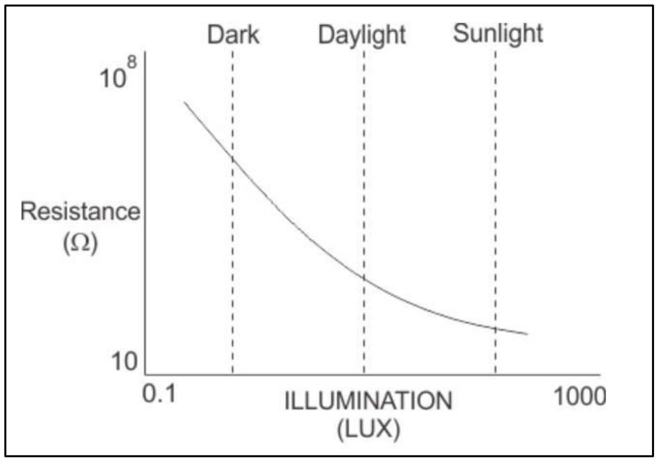


Figure 4.13: Resistance VS Illumination Curve of LDR (Electrical4U, 2019)

To determine the accuracy of solar tracking mechanism, the array of LDRs in LDR circuitry as shown in Figure 3.12 will be observed. If there is any shadow falling on the LDR circuitry board, it indicates that the solar tracker is not facing the Sun precisely. Installation of pyrheliometer is the key factor that requires high precision solar tracking mechanism. This is because the pyrheliometer only receives direct sunlight parallel to its body.

At the beginning of the project, big LDRs were implemented as the light sensors. However, the solar tracker is unable to track the Sun precisely after iterations of calibration. There was shadow visible on the LDR circuitry board of the solar tracker eventhough it was during the sunny day. To solve this problem, the big LDRs were replaced by smaller LDRs. The LDR circuitry was reconstructed, and the LDRs were placed close to each other at the centre of the circuitry board. With the modification done, the solar tracker was able to track precisely during the sunny day. Hence, smaller LDRs are recommended to be implemented as light sensors compared to big LDRs.

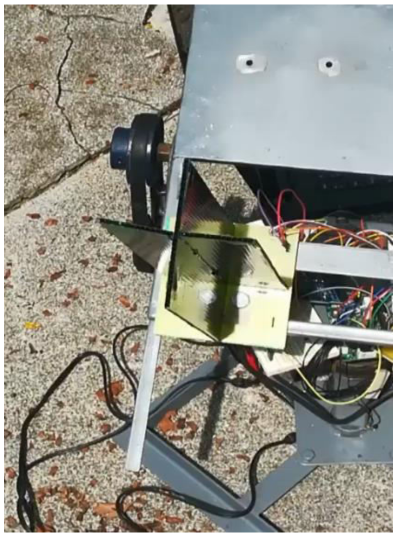


Figure 4.14: Inaccurate Solar Tracking Mechanism During Sunny Day Using Big LDRs

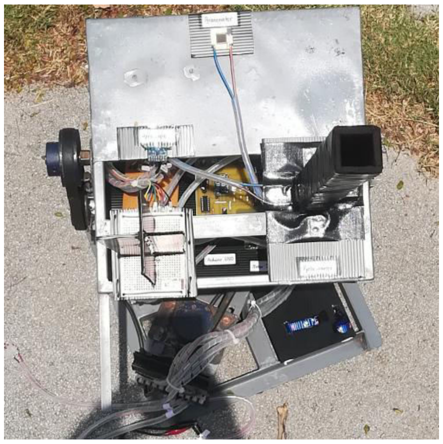


Figure 4.15: Accurate Solar Tracking Mechanism During Sunny Day Using Small LDRs

Before performing calibration for LDR circuitry, the analog input values for all four LDRs in the LDR circuitry were recorded using the LDR test code provided in Appendix J. The analog input values were recorded using serial monitor under surrounding with different illuminance levels, and the average of input values were calculated which would be included in Table 4.7. The illuminance levels include sunny, mostly sunny, partly sunny and cloudy. Sunny is a condition when there is no opaque cloud covering the Sun, mostly sunny is a condition when there is opaque cloud covering a small region of the Sun, partly sunny is a condition when half of the Sun is covered by opaque cloud, and cloudy is a condition when opaque cloud covers the Sun completely (Sonaik, 2017). The exact illuminance values cannot be determined and hence, the illuminance level can only be determined according to the conditions of the sky.

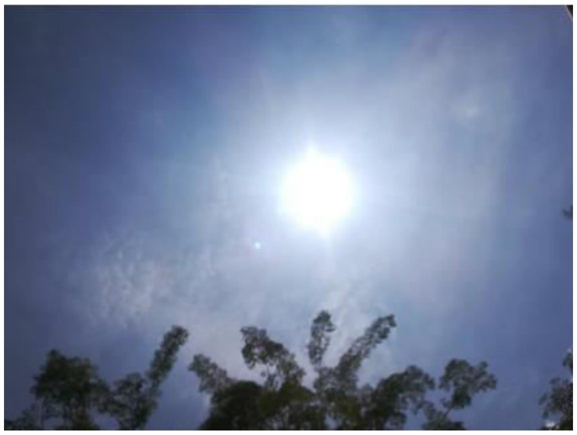


Figure 4.16: Sunny

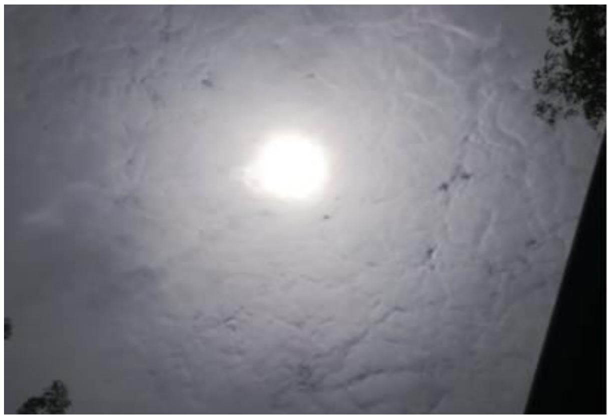


Figure 4.17: Mostly Sunny

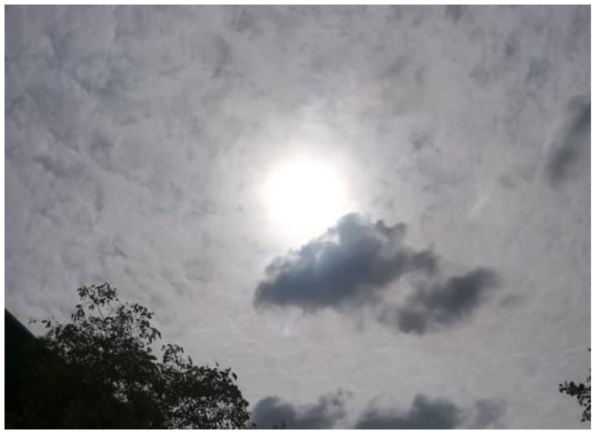


Figure 4.18: Partly Sunny

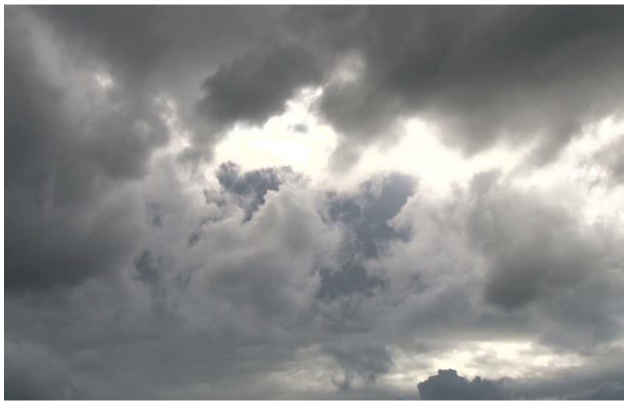


Figure 4.19: Cloudy

Table 4.7: Average Arduino Analog Input Values Recorded From LDRs According To Different Illuminance Level

Illuminance Level	**Arduino Analog Input Value**			
	Top Left LDR	**Top Right LDR**	**Bottom Left LDR**	**Bottom Right LDR**
Cloudy	90	110	90	90
Partly Sunny	10	62	9	10
Mostly Sunny	5	32	3	6
Sunny	2	27	1	3

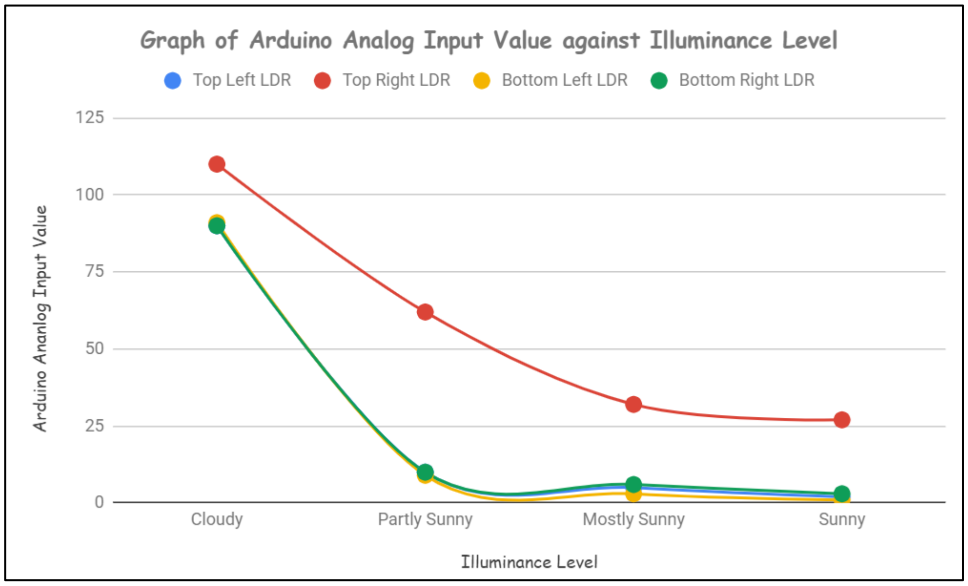


Figure 4.20: Graph of Arduino Analog Input Value against Illuminance Level for the LDRs

From the table in Table 4.7 and graph in Figure 4.20, the top right LDR would give the highest analog input regardless illuminance level. Furthermore, the analog input values recorded were not linearly proportional to illuminance level, and it was a great challenge in doing calibration for the LDR circuitry. Fortunately, only the top right LDR has the distinct difference compared to the other three LDRs which provide similar analog input values. Therefore, the top right LDR was the only LDR needed to be calibrated.

Although small LDRs could help the solar tracker to track more precise compared to big LDRs, there were several scenarios of inaccuracy in the solar tracking mechanism shown in the pictures in two next pages.

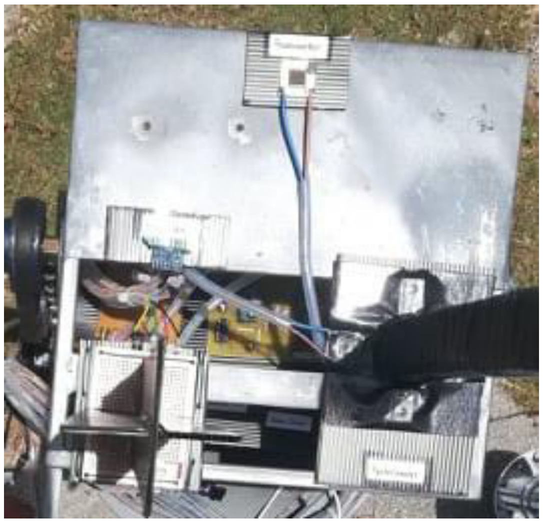


Figure 4.21: Inaccuracy (Scenario 1)

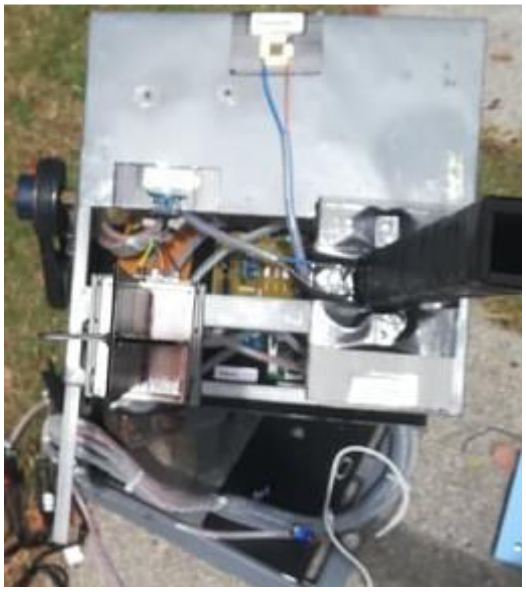


Figure 4.22: Inaccuracy (Scenario 2)

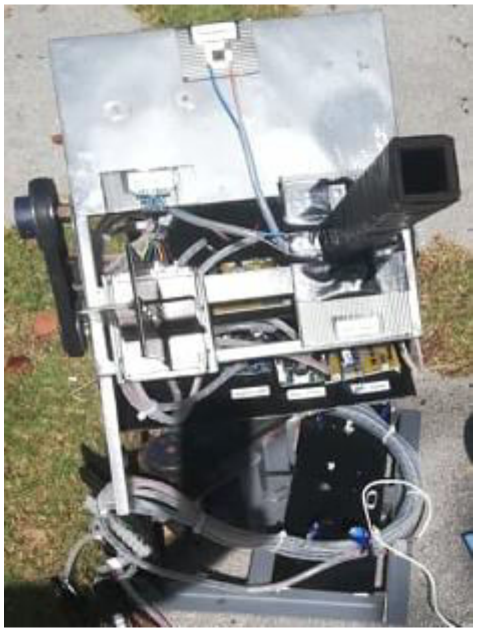


Figure 4.23: Inaccuracy (Scenario 3)

From these scenarios of inaccuracy, the top right region of the LDR circuitry board was never covered by shadow. This means that the calibration of the top right LDR was not precise. From the graph shown in Figure 4.20, the slope of curve for top right LDR differed a lot compared to the other three LDRs. Besides, it was uncertain to determine the exact amount of illuminance although there were conditions of sky to determine the illuminance level. These factors contributed to high uncertainty in presetting a threshold to calibrate the top right LDR. Therefore, these cases of inaccuracies in solar tracking mechanism were unavoidable.

4.5 Comparison of Constructed Pyrheliometer with the Actual Pyrheliometer

Constructed pyrheliometer was compared with the actual pyrheliometer. Calibration would be needed if there was great difference between the construced and actual pyrheliometer. Before comparing both pyrheliometers, the formulas of Direct Normal

Irradiance (DNI) for both constructed and actual pyrheliometer were given as shown in Equation 4.2 and Equation 4.3 respectively.

$$\text{DNI}_{\text{constructed}} = I \times \frac{1000}{15.3} \tag{4.2}$$

where

DNI $_{\text{constructed}}$ = Direct Normal Irradiance of constructed pyrheliometer, W/m^2

I = Calculated current of the solar cell in pyrheliometer, mA

$$\text{DNI}_{\text{actual}} = V \times \frac{1000}{sensitivity} \tag{4.3}$$

where

DNI $_{\text{actual}}$ = Direct Normal Irradiance of actual pyrheliometer, W/m^2

V = Voltage measured from actual pyrheliometer, mV

$sensitivity$ = Sensitivity of actual pyrheliometer, μV/ (W.m^{-2})

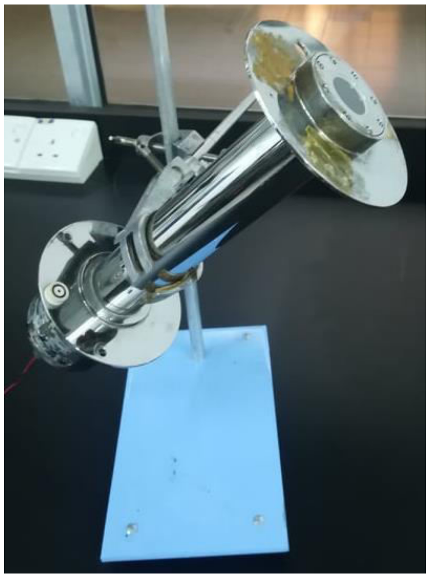


Figure 4.24: Actual Pyrheliometer Used

Comparison of constructed pyrheliometer with actual pyrheliometer was done as shown in Figure 4.25. Both of the pyrheliometers were placed under the Sun and the irradiance values were measured respectively. The irradiance value of constructed pyrheliometer was recorded via Blynk user interface as the formula in Equation 4.2 was included into the source code shown in Appendix O uploaded to ESP32. For the actual pyrheliometer, the small voltage signal was measured using multimeter as shown in Figure 4.26, and then the irradiance value was calculated according to Equation 4.3, where the sensitivity of pyrheliometer is 8.11 μV/ Wm^{-2}. To obtain the exact voltage value of the actual pyrheliometer, the pyrheliometer would have to be adjusted its position until there was one small dot of light falls on the bull-eye of the sensing element of the pyrheliometer as shown in Figure 4.27.

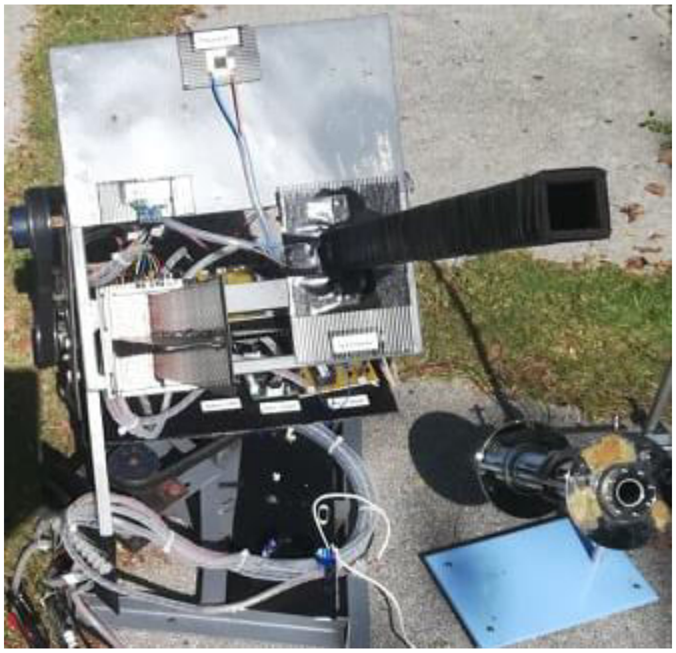


Figure 4.25: Comparison of Constructed Pyrheliometer with Actual Pyrheliometer

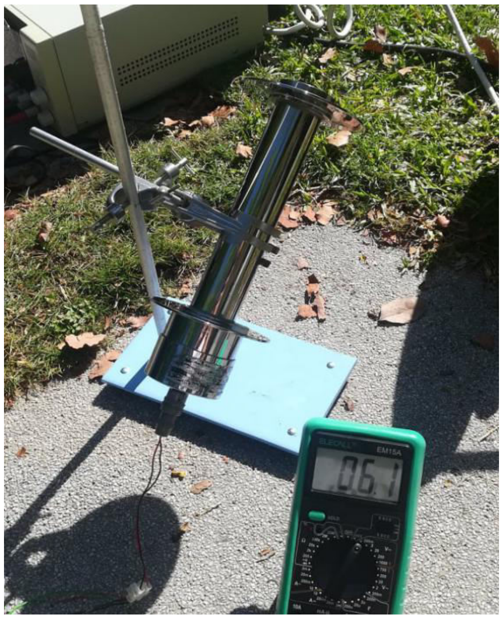


Figure 4.26: Measurement of Small Voltage Signal From The Actual Pyrheliometer Using Multimeter

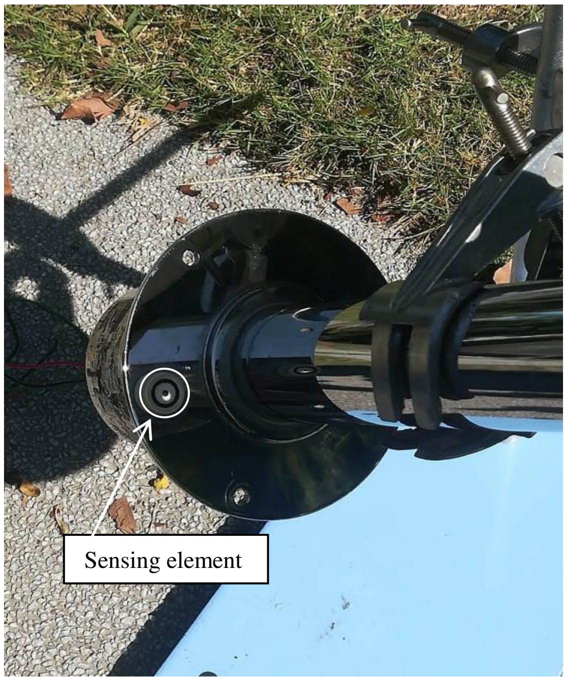


Figure 4.27: Indication of Getting Exact Voltage Value From The Actual Pyrheliometer

The results of comparison between both constructed and actual pyrheliometers were recorded in the table below.

Table 4.8: Comparison of Constructed Pyrheliometer and Actual Pyrheliometer

Actual Pyrheliometer		**Constructed Pyrheliometer**		**Accuracy of Constructed Pyrheliometer in Measuring Irradiance Values (%)**
Measured Voltage (mV)	**Irradiance Value (W/m^2)**	**Calculated Current (mA)**	**Irradiance Value (W/m^2)**	
0.02	2.47	0.0267	1.75	70.85020243
1.4	172.63	2.6331	172.1	99.692985
1.9	234.28	3.3813	221	94.33156906
2	246.61	3.7102	242.5	98.33340092
3.1	382.24	5.7222	374	97.84428631
4.3	530.21	7.7571	507	95.6224892
4.9	604.19	8.7057	569	94.17567322
5	616.52	9.4554	618	99.75994291
5.3	653.51	10.7253	701	92.73308748
6.2	764.49	11.2302	734	96.01172023
6.5	801.48	11.9538	781.3	97.48215801

From the table above, accuracy of the constructed pyrheliometer was relatively high as the constructed pyrheliometer had more 90 % of accuracy for all data recorded during sunny day which having the irradiance values above 100 W/m^2. The accuracy of the constructed pyrheliometer was slightly low during cloudy day, which having an accuracy around 70 %. Therefore, a low cost pyrheliometer with high precision was constructed.

A graph of irradiance values of actual pyrheliometer against current of constructed pyrheliometer was plotted according to Table 4.7. The graph plotted is shown in the picture below.

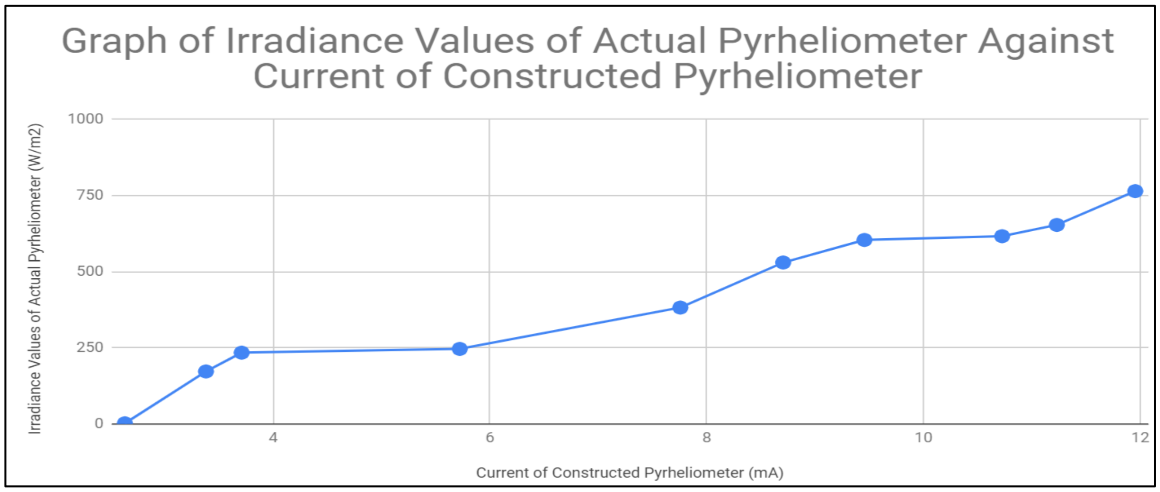


Figure 4.28: Graph of Irradiance Values of Actual Pyrheliometer Against Current of Constructed Pyrheliometer

From the graph above, the range of current produced by the solar cell in the pyrheliometer was in between 4 mA to 12 mA during sunny day. If the day was extremely bright, the solar cell would produce high current values between 8 mA and 12 mA. If it was not very sunny or cloudy, the current values would be smaller than 4 mA.

4.6 Analysis of Solar Irradiance Data Collected

Solar irradiance values from pyranometer and pyrheliometer can help to determine the conditions of the environment which are either cloudy or sunny, and to determine the accuracy of solar tracking mechanism.

Pyranometer is the sensor which measures the overall solar radiation around it which includes direct sunlight and sunlight reflected onto it. Hence, the total irradiance measured by the pyranometer can help to indicate the conditions of the environment. The environment is known to be sunny if the irradiance values of pyranometer exceeds 400 W/m^{-2}. On the other hand, pyrheliometer is the sensor which accepts and measures the radiation of direct sunlight onto the sensing cell. The values of direct normal irradiance measured by the pyrheliometer will determine the accuracy in tracking mechanism. If the environment is sunny and the solar tracker is tracking the Sun accurately, the value of direct normal irradiance by the pyrheliometer will exceed 100 W/m^{-2}. If the day is extremely bright, the value of irradiance measured by the pyrheliometer will be approximately 50 % or almost the same as the irradiance values from the pyranometer.

The solar irradiance data were collected at the common area between Block D and E in Universiti Tunku Abdul Rahman. The data were collected for four days, which were on 9th, 13th, 14th and 16th August 2019. The data were download from Google Spreadsheet. Graphs for respective days were then plotted and analyzed accordingly.

4.6.1 Solar Irradiance Data On 9th of August 2019

On 9th of August 2019, solar irradiance data of both pyrheliometer and pyranometer were collected and the graph was plotted. Data collection was done from 9:00 am to 2:00 pm.

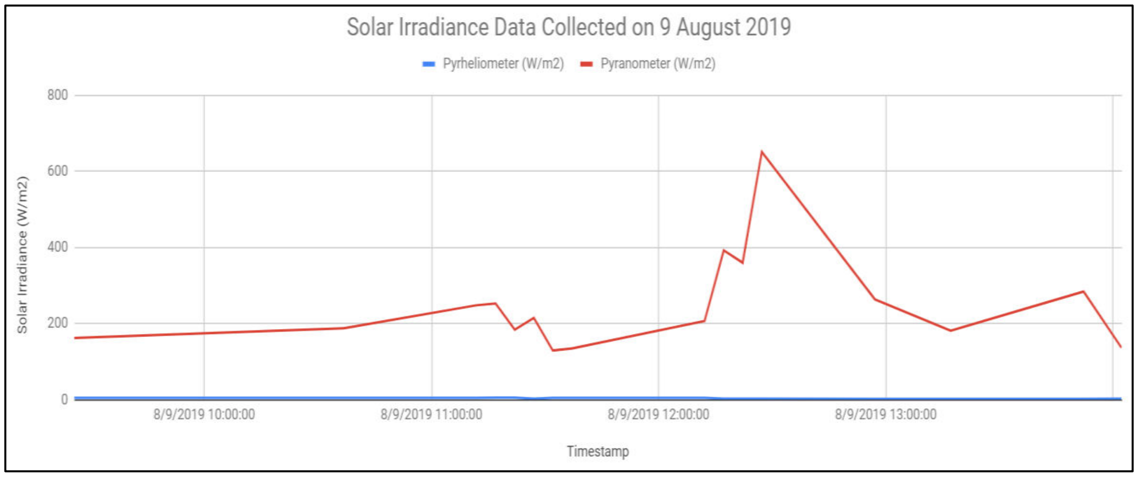


Figure 4.29: Graph of Solar Irradiance Data Collected on 9th of August 2019

According to the graph, the ambient light intensity was relatively low and remained consistent from 9:00 am to 12:00 pm according to total irradiance measured by the pyranometer. This indicated that it was cloudy within this period of time, and hence, the values of direct normal irradiance from pyrheliometer was extremely low due to low ambient light intensity.

The ambient light intensity started to raise at noon and reached its peak around 12:30pm according to total irradiance measured by the pyranometer. This phenomena happened when the Sun was not covered by the cloud during this period of time. However, the values of direct normal irradiance from pyrheliometer did not increase accordingly. This indicated that the solar tracker was tracking the Sun accurately. The values of direct normal irradiance should exceed 100 W/m^{-2} if the solar tracker is facing the Sun accurately.

After the ambient light intensity reached its peak, it dropped drastically and then remained low until 2:00 pm. The values of total irradiance from the pyranometer fluctuated within the range of 200 W/m^{2}, while the values of direct normal irradiance from the pyrheliometer remained extremely low.

In conclusion, it was cloudy throughout the day, except when it was at noon. The Sun was not covered by opaque cloud for around 30 minutes at noon. Therefore, it was basically cloudy on 9th of August 2019.

4.6.2 Solar Irradiance Data On 13th of August 2019

On 13th of August 2019, solar irradiance data were collected from 10:00 am to 5:00 pm, and the graph of solar irradiance data was plotted.

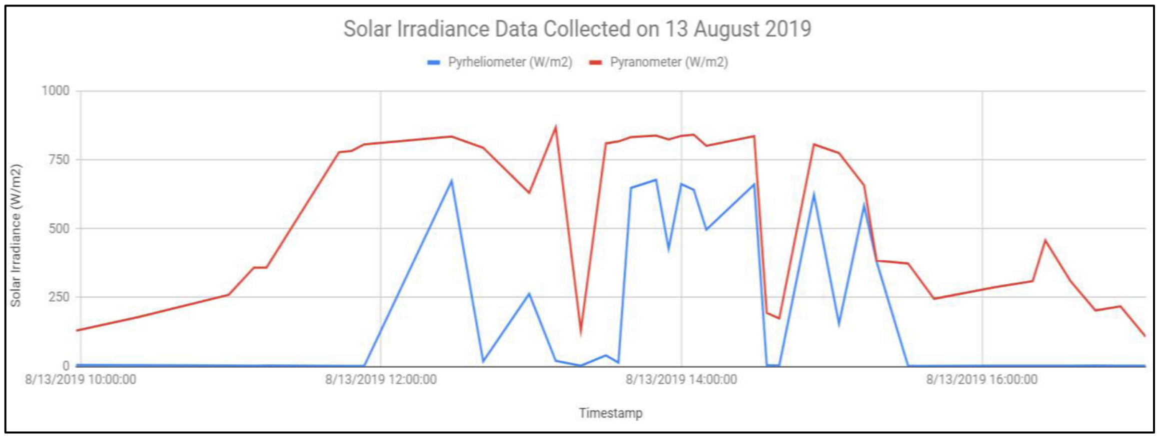


Figure 4.30: Graph of Solar Irradiance Data Collected on 13th of August 2019

From the graph, the ambient light intensity started increasing from morning and then remained high in the afternoon, and dropped when it was in the evening according to total irradiance measured by the pyranometer. Theoretically, the shape of direct normal irradiance from pyrheliometer in the graph should look similar to the total irradiance from pyranometer with lower or slightly lower values during sunny day from 11:00 am to 3:30 pm. From the values of direct normal irradiance obtained, the solar tracker did track the Sun accurately, but there were cases of inaccuracy in solar tracking mechanism when the value of direct normal irradiance was extremely low during sunny day. For the cloudy condition in the morning and evening, the direct normal irradiance measured by the pyrheliometer was extremely low, while the values of total irradiance from the pyranometer were within the range of 250 W/m^2.

In conclusion, it was basically a clear day on 13th August 2019, where it was sunny in the afternoon and no opaque cloud was blocking the Sun.

4.6.3 Solar Irradiance Data On 14th of August 2019

Solar irradiance data were collected from 10:00 am to 5:00 pm on 14th of August 2019. The graph of solar irradiance data was plotted.

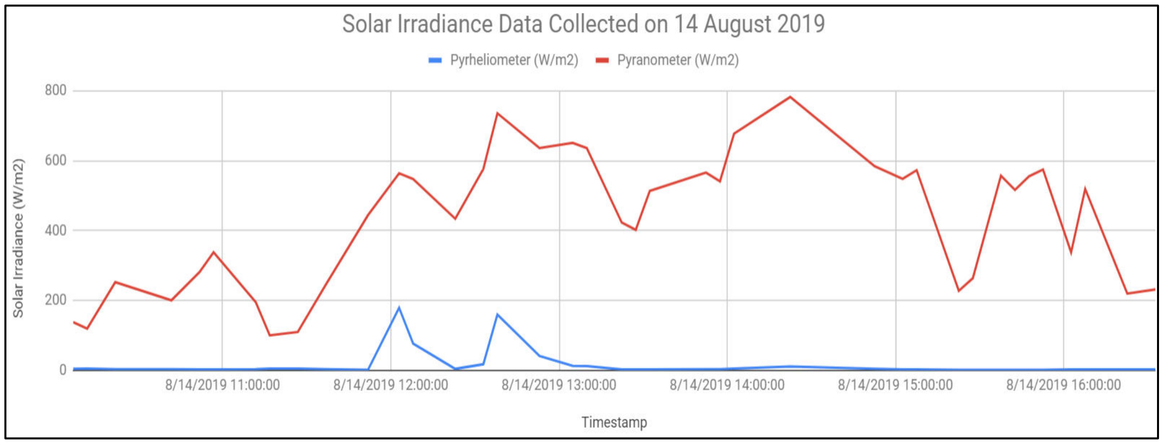


Figure 4.31: Graph of Solar Irradiance Data on 14th August 2019

From the graph, it was basically sunny on 14th August 2019, where most of the values of total irradiance from the pyranometer exceeded 200 W/m^2. However, the solar tracking mechanism on this day was not accurate. From the two peaks of direct normal irradiance measured by the pyrheliometer, the solar tracker is facing the Sun with slight inaccuracy, where there was small region of shadow on the LDR circuitry board as shown in Figure 4.23. For the rest of time, the pyrheliometer was not pointing to the Sun, which indicated inaccuracy in solar tracking mechanism.

4.6.4 Solar Irradiance Data On 16th of August 2019

On 16th of August 2019, solar irradiance data were collected from 10:00 am to 5:00 pm. The graph of solar irradiance data was plotted.

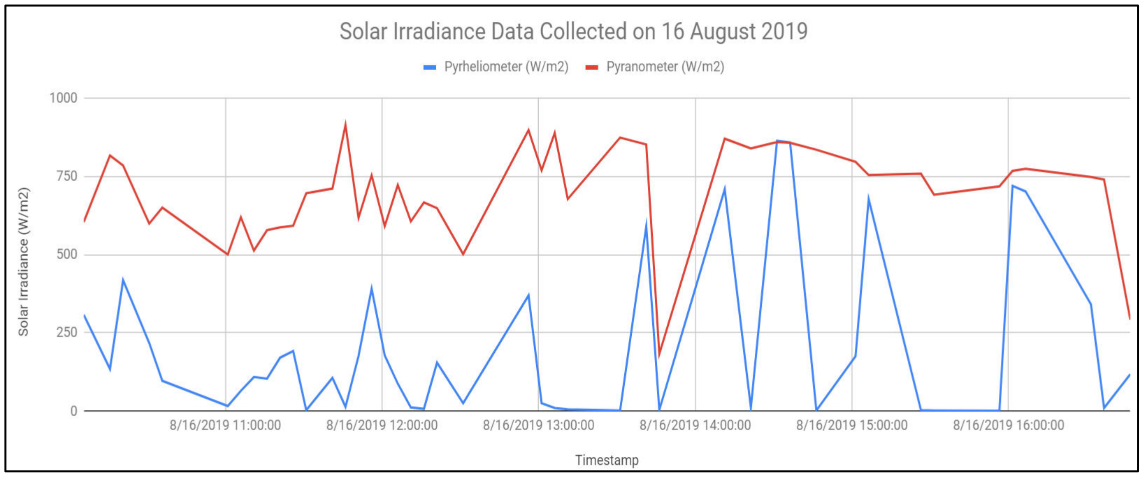


Figure 4.32: Graph of Solar Irradiance Data on 16th of August 2019

From the graph, it was sunny throughout the day on 16th of August 2019, where all values of total irradiance measured by the pyranometer were above the range of 250 W/m^2. The accuracy of solar tracking mechanism was relatively high on that day as the values of direct normal irradiance exceeded 100 W/m^2. There were also a few cases of inaccuracy in solar tracking mechanism when the values of direct normal irradiance were extremely low.

4.6.5 Conclusion Based On The Solar Irradiance Data Collected

In conclusion, it was cloudy on 9th of August 2019, and was sunny on 13th, 14th and 16th of August 2019. From the solar irradiance data analyzed for these four days, the accuracy of tracking mechanism was relatively low. Therefore, enhancement on solar tracking mechanism is required.

Solar irradiance data is useful to determine whether the particular site is suitable to install a PV system. However, solar irradiance data collection for four days is not enough to determine the feasibility to start up a PV system at the particular site. Usually, solar irradiance data are collected monthly or yearly to decide the installation of PV system.

4.7 Improvement Done To Make Solar Tracker More Presentable

To make the solar tracker look presentable, cable management was done by using spiral cable wrap tube. The circuitry of solar tracker was soldered on PCB board and stripboard. The solar tracker is more presentable after cable management and arrangement of circuitry as shown in the picture below.

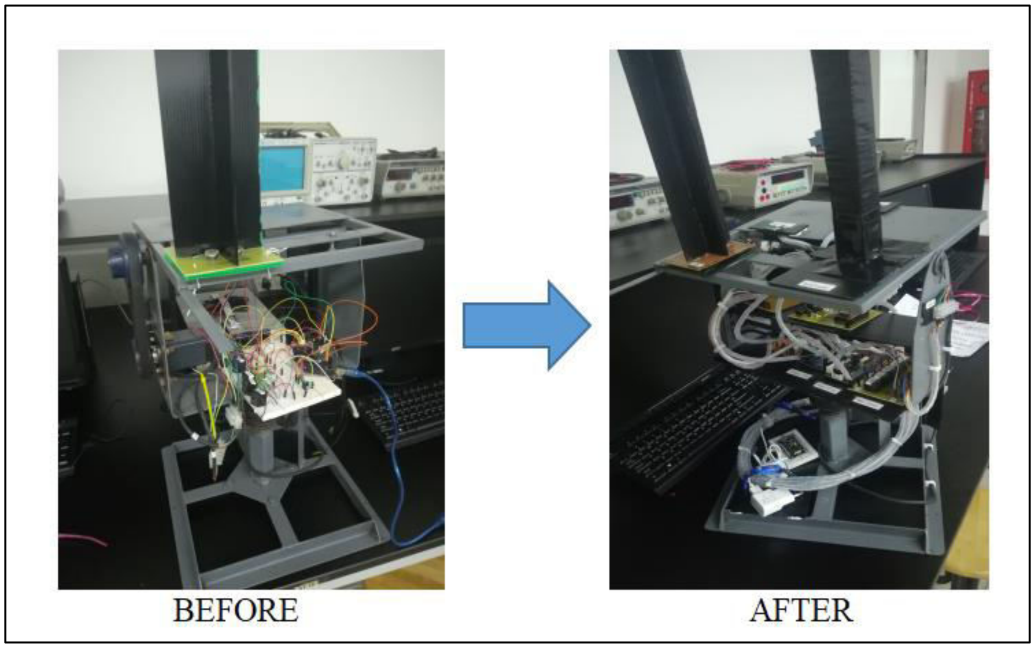


Figure 4.33 : Improvement Done to Make Solar Tracker More Presentable

4.8 Limitations of the Project

This subsection discusses about the constraints and limitations of the project.

4.8.1 Dependency on Weather

The greatest challenge faced in this project was high dependency of weather to test the accuracy of solar tracking mechanism. Solar tracker tracks the Sun throughout the day, and it is useless to test the solar tracker during cloudy day. In another words, testing of solar tracker can be done when it is sunny day. To increase the probability for the solar tracker to track accurately, iterations of testing on solar tracking mechanism under sunny day were required. In another words, if that particular day was either rainny day or cloudy day, it would affect and delay the progression in testing the accuracy of solar tracking mechanism.

4.8.2 Limitation of Arduino IDE Software in Performing Software Operation

At the initial stage of the project, Blynk was not opted to be the IoT user interface platform due to its weakness and limitation which will be discussed in Section 4.8.4. MQTT protocol was applied for IoT data collection purpose at the beginning of the project. MQTT protocol is one of the most easiest and yet the most effective IoT protocol in transmitting real-time data using Publish/Subscribe method. Publish in MQTT refers to an event when the device uploads or publishes data under the topic created to the MQTT broker, while Subscribe in MQTT refers to an event when the device or application subscribes the topic from the MQTT broker and retrieves data from the topic subscribed. The data transmission flow of MQTT protocol is shown in Figure 4.34.

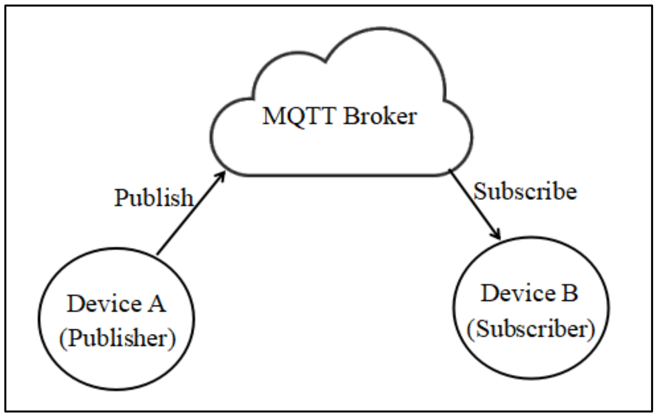


Figure 4.34 : Block Diagram of MQTT Protocol Data Transmission

Thingsboard Community Edition was implemented to create an MQTT platform to visualize the real-time data and charts. Thingsboard was a powerful IoT platform where it was user-friendly and was able to review historical data besides the ability to display live data upload the MQTT broker created. Thingsboard dashboard was created as shown in the picture below. However, Thingsboard Community Edition was unable to export the data of the graphs for data analysis purpose.

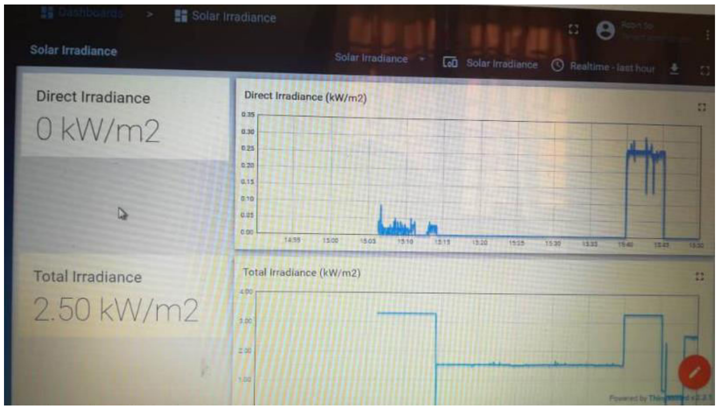


Figure 4.35 : Thingsboard Dashboard Created Before Implementing Blynk

Like Thingsboard Community Edition, most of the free IoT platforms available can only display real-time data. Therefore, an IoT database had to be created, and Google Spreadsheet was opted to serve as the IoT database.

To enable Arduino IDE software to transmit data to Google Spreadsheet, HTTPS client had to be created. However, this caused conflict in creating clients from different open source libraries. PubSub library was needed for creation of PubSub client to perform MQTT data transmission, while HTTPS library was needed for creation of HTTPS client to perform data transmission to Google Spreadsheet. Arduino IDE software got confused as two different clients were created simultaneously. This phenomena happened due to limitations and restrictions of Arduino open-source libraries created. Furthermore, it is hard and tedious to troubleshoot the code line in open-source libraries. In another words, Arduino IDE software is not flexible in dealing with software development.

Due to the limitations of Arduino IDE software, Blynk was then opted to be the IoT user interface platform. This is because Blynk has its own specific library for Arduino IDE software and does not need declaration of client in Arduino IDE software. With these features of Blynk, the problem of having conflict in creating client can be avoided. Thus, ESP32 can upload data to both Blynk and Google Spreadsheet in IoT data collection system of the project.

4.8.3 Limited Range of Wi-Fi Network

Wi-Fi hotspot from the mobile phone has small network coverage, and thus, ESP8266 LiLon NodeMCU V3 is implemented as the Wi-Fi extender to expand the network coverage. Although ESP8266 LiLon NodeMCU V3 is able to extend the Wi-Fi network range more than 10 meter, however, the connection of ESP32 to ESP8266 LiLon NodeMCU V3 is found out to be weak due to low value of RSSI. In another words, the distance of Wi-Fi connection is inversely proportional to RSSI value of the network device as shown in Figure 4.36. RSSI or known as “Received Signal Strength Indicator” is applied to indicate the strength of signal received by a device from the access point. In this project, the device refers to ESP32 and the access point refers to ESP8266 LiLon NodeMCU Wi-Fi extender.

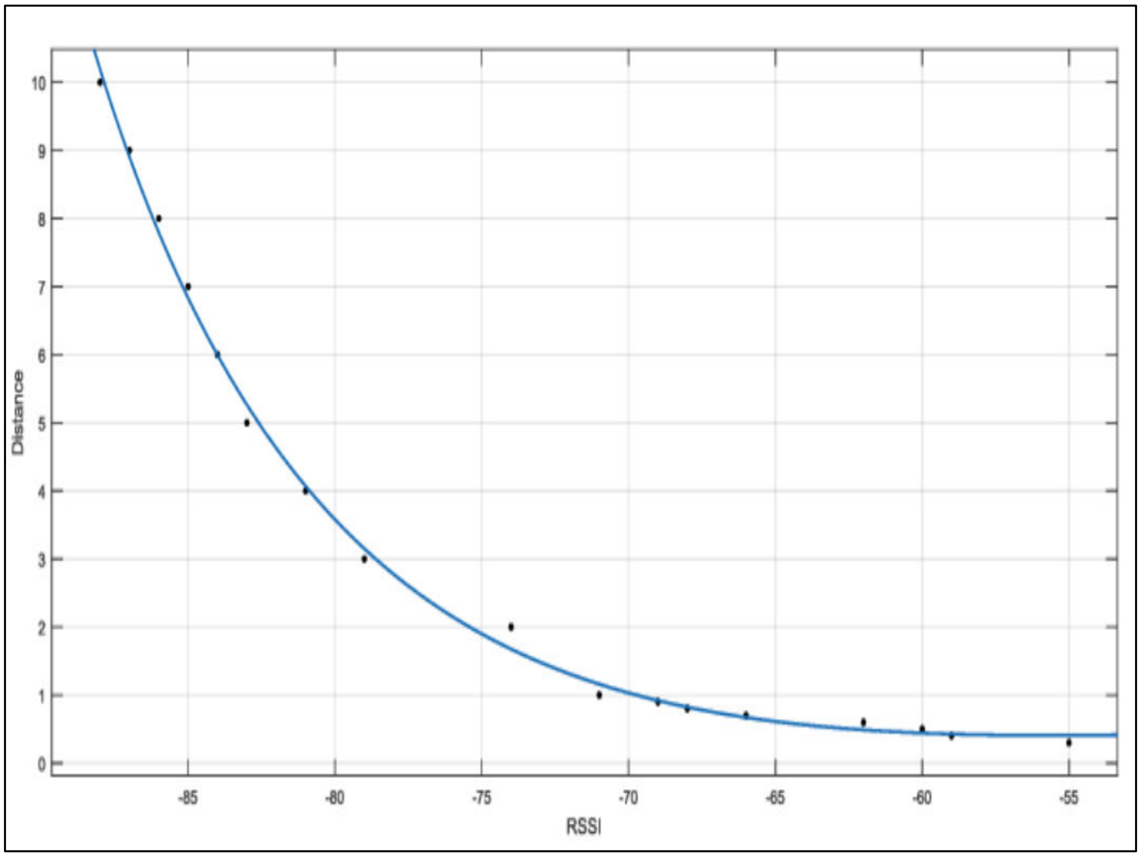


Figure 4.36: Graph of Distance Against RSSI

Therefore, ESP8266 LiLon NodeMCU V3 together with mobile phone has to be transferred nearer the solar tracker for stable Wi-Fi connection as shown in Figure 4.37. A shading for both mobile phone and ESP8266 LiLon NodeMCU V3 was done to avoid direct sunlight.

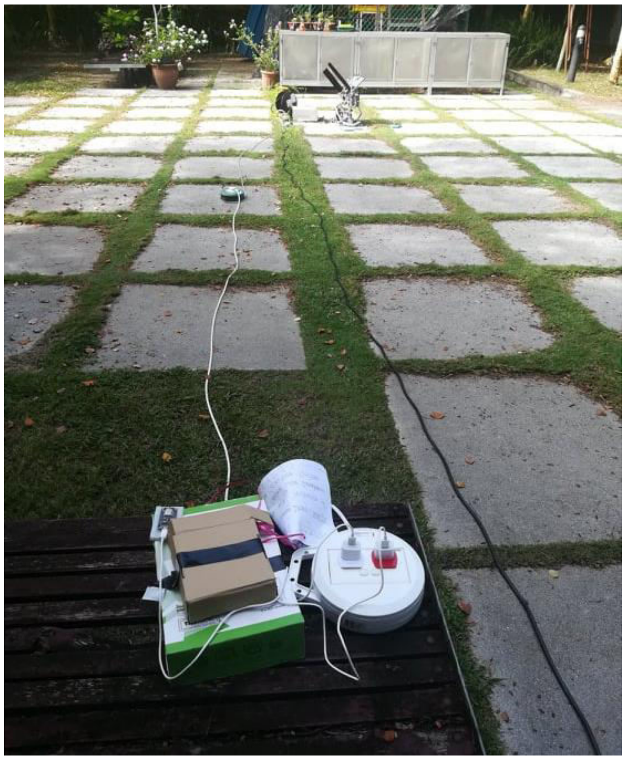


Figure 4.37: Setup Of Wi-Fi Connection

4.8.4 Weakness of Blynk

Blynk is an easy alternative for IoT data collection system and provides an user-friendly interface for end users to visualize real-time data and to utilize the functions available in the user interface. Furthermore, the Blynk user interface created can be accessed in different mobile phones by scanning on the QR code generated.

However, the major drawback of using Blynk is that the mobile phone has to remain turned on while using Blynk to visualize the real-time data. Blynk cloud server will disconnect itself once the mobile phone is not executing the Blynk app, and will require a long time to reconnect if Blynk app is executed again. Therefore, the mobile phone will have to keep on charging to prevent battery drainage due to high power consumption of Blynk app.

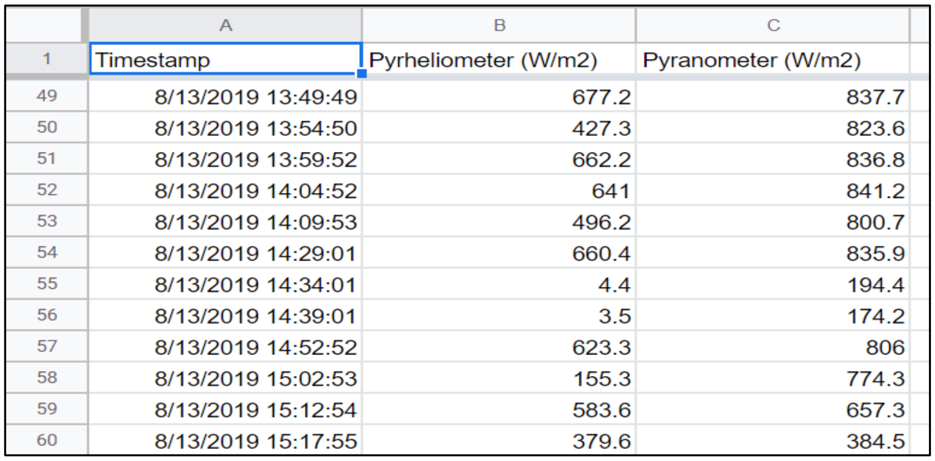
4.8.5 Weakness of Pushingbox API Cloud Service

Pushingbox API cloud service is implemented as a medium to transfer solar irradiance data from ESP32 to Google Form and Google Spreadsheet. It is easy to configure the setting of Pushingbox API cloud service in IoT data transmission.

However, there was limitation found in Pushingbox API cloud service when performing IoT data collection. When performing IoT data collection, it was found that the data were not uploaded to Google Spreadsheet every five minutes consistently according to the algorithm of IoT data collection system discussed in Chapter 3.

There were two possibilities proposed for inconsistency of data collection in Google Spreadsheet, which were mistakes in source code for IoT data collection system and poor Wi-Fi connection of ESP32. Hence, troubleshooting was done upon the source code in Appendix O, and Wi-Fi connection status of ESP32 was checked using Blynk app. After troubleshooting, none of the two possibilities are the factors of inconsistency in data collection as shown in Figure 4.38. The algorithm of the source code in Appendix O was checked and it matched with the algorithm of IoT data collection system illustrated in Chapter 3. Inconsistency of data collection in Google Spreadsheet still occurred eventhough the connection of ESP32 to Wi-Fi hotspot from the Wi-Fi extender was stable throughout the entire session of IoT data collection.

Therefore, it was considered to be the limitation of Pushingbox API cloud service in performing data transmission. Furthermore, the Pushingbox API is a predefined function set for data transmission, and the backend process and algorithm of Pushingbox API is not accessible for troubleshooting.



	A	B	C
1	Timestamp	Pyrheliometer (W/m2)	Pyranometer (W/m2)
49	8/13/2019 13:49:49	677.2	837.7
50	8/13/2019 13:54:50	427.3	823.6
51	8/13/2019 13:59:52	662.2	836.8
52	8/13/2019 14:04:52	641	841.2
53	8/13/2019 14:09:53	496.2	800.7
54	8/13/2019 14:29:01	660.4	835.9
55	8/13/2019 14:34:01	4.4	194.4
56	8/13/2019 14:39:01	3.5	174.2
57	8/13/2019 14:52:52	623.3	806
58	8/13/2019 15:02:53	155.3	774.3
59	8/13/2019 15:12:54	583.6	657.3
60	8/13/2019 15:17:55	379.6	384.5

Figure 4.38: Result of inconsistency of Pushingbox API in transferring data to Google Spreadsheet

4.8.6 Insufficient Time for IoT Data Collection

The solar irradiance data collection system will be useful if the solar tracker tracks the Sun accurately. In another words, solar tracking mechanism is crucial in determining the effectiveness of solar irradiance data collection system. Inaccurate solar tracking mechanism will cause IoT data collection to be inaccurate eventhough high precision sensing circuitry is constructed.

Therefore, most of the time were spent in troubleshooting the solar tracking mechanism. It took a long period of time to achieve high precision solar tracking mechanism although there were possibilities of inaccuracy. Hence, there was a few days left to perform IoT data collection and analysis before the deadline of the project.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In conclusion, the objectives of the project are achieved. A cost effective active dual-axis solar tracking system is constructed. Cost saving pyrheliometer and pyranometer are constructed which can help to save a lot of cost to buy the actual solar meters. Furthermore, the constructed solar meters are much lighter and smaller compared to the actual solar meters. According to the comparison between constructed and actual pyrheliometer, the accuracy of constructed pyrheliometer was relatively high. To ensure the constructed solar meters to function as the actual meters, high precision current-to-voltage converter circuits are needed for ESP32 to read accurate current values for conversion to respective solar irradiance values of both pyrheliometer and pyranometer. In this project, high precision current-to-voltage converter circuitry for the sensors were successfully built. The solar irradiance data can be collected and visualized remotely using Blynk app and Google Spreadsheet as long ESP32 connects to the Internet. Although all objectives in the project are achieved, there are several problems and challenges faced. Firstly, uncertainty in achieving high precision solar tracking mechanism is relatively high. Iterations of calibration were done in order to high precision solar tracking mechanism due to uncertain difference in resistance and tolerance among different LDRs under the same environment. However, cases of inaccuracy in solar tracking mechanism were unavoidable although the solar tracker with small LDRs is better in solar tracking mechanism compared to big LDRs. Besides, the range of Wi-Fi network is relatively short for stable connection between ESP32 and ESP8266 LiLon NodeMCU V3. Furthermore, Arduino IDE software was not flexible in IoT software development due to limitations of open-source libraries. Extra features were implemented in this project as well. MPU6050 gyro sensor was implemented to initialize the position of solar tracker when the solar tracker was powered on. Besides, the current status of solar tracker can be displayed in Blynk user interface platform.

5.2 Future Work and Enhancement

There are weaknesses and limitations in the project. Therefore, this subsection lists down several recommendations for future enhancement of the project.

5.2.1 Solar Tracker With Camera

In this project, the accuracy of solar tracker in tracking the Sun using small LDRs was improved compared to big LDRs. However, there were still several cases of inaccuracy due to great difference in Arduino analog input of the top right LDR from another three LDRs in LDR circuitry. The problem was found but it was hard to troubleshoot and calibrate the top right LDR. This is because the resistance of LDR is not linearly proportional to illuminance level. Furthermore, illuminance level was predefined according to the condition of the sky, and the exact amount of illuminance was unknown.

To solve this problem, a camera can be added as the ‘eye’ of the solar tracker since the LDRs cannot ‘see’ the exact position of the Sun. To ensure the camera is able to ‘recognize’ the Sun, knowledge of image processing is required. Method of edge detection together with image thresholding can be implemented to seek for the Sun according to light intensity within the view of camera.

The proposed algorithm of the recommended idea is that the solar tracker will track according to comparison within LDRs, and then camera will check after the solar tracker according to reading of LDRs. The camera will send a signal to adjust the solar tracker if the pyrheliometer is not pointing the Sun. The camera is recommended to place close to the pyrheliometer to ensure high precision in solar tracking mechanism.

5.2.2 Enhancement on Remote Data Collection System

One of the major problems faced when performing IoT data collection system is limited range of Wi-Fi network. Wi-Fi provides a small coverage of network (Guillemette, 2019). The strength of Wi-Fi signal can be affected and weakened if there are obstacles. To overcome these problems, cellular network is recommended for its wider network coverage. Furthermore, the connection of cellular network is much more stable compared to Wi-Fi network.

Arduino IDE software is powerful in performing hardware operation such as solar tracking mechanism. However, Arduino IDE software is not flexible in performing software operation due to limitations and restrictions in the open-source libraries. In this project, Arduino IDE software was unable to perform operation and send data to two clients from different network protocol when tried to have Arduino to upload data to Thingsboard under MQTT protocol and Google Spreadsheet under HTTP protocol. Python and Java programming language will be a better option for software operation.

ESP32 is recommended to be replaced with Raspberry Pi. Raspberry Pi uses python language which is better than Arduino IDE software in software operation and development, and it is more powerful in running complex software operation.

Furthermore, it can perform data transmission via cellular 3G or 4G network with the addition of GSM module.

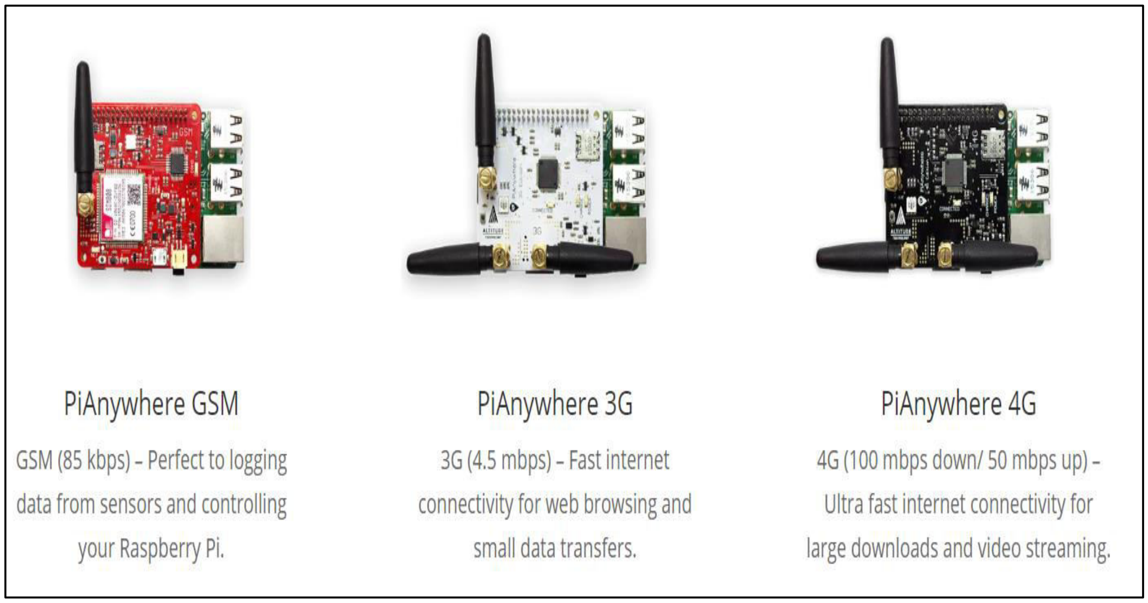

PiAnywhere GSM
GSM (85 kbps) – Perfect to logging data from sensors and controlling your Raspberry Pi.
PiAnywhere 3G
3G (4.5 mbps) – Fast internet connectivity for web browsing and small data transfers.
PiAnywhere 4G
4G (100 mbps down/ 50 mbps up) – Ultra fast internet connectivity for large downloads and video streaming.

Figure 5.1: Raspberry Pi with Addition of GSM Module

One major drawback of real-time IoT data collection system is that the data will be lost if the IoT device is disconnected from the Internet or if the Internet is down. To solve this problem, Raspberry Pi can help to store the backup data in its operating system and upload the data once it is connected to the Internet. The storage of the operating system depends on the size of the memory card inserted to Raspberry Pi. The method proposed is possible to be done, but it will require strong programming skills.

5.2.3 Waterproof Housing for the Circuitry and Power Supply of Solar Tracker

A tragic happened when it suddenly rained heavily while performing solar irradiance data collection on 21st August 2019. The solar tracker was wet as shown in the picture below. The circuitry and power supply of the project is not waterproof, but fortunately, both circuitry and power supply still functioned well. Therefore, it is recommended to include waterproof housing to increase the durability of the project.

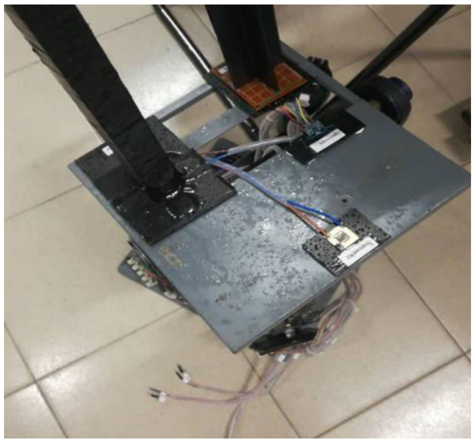


Figure 5.2: The Solar Tracker Was Wet Due To Heavy Rain

REFERENCES

Arduino.cc., 2017. *Getting Stared with Arduino and Genuino UNO.* [online] Available at: <https://www.arduino.cc/en/Guide/ArduinoUno> [Accessed 12 April 2019]

AmritaVirtual Lab, 2019. *Solar Energy Measurements - Pyrheliometer.* [online] Available at:<https://vlab.amrita.edu/?sub=77&brch=298&sim=1745&cnt=1> [Accessed 12 August 2019]

Aziz, et al., 2016. Evaluation of Solar Energy Potential in Malaysia. *Trends in Bioinformatics.* [online] Available at: <http://docsdrive.com/pdfs/ansinet/tb/2016/35-43.pdf> [Accessed 12 August 2019]

Blynk Inc, 2019. *Build your first IoT app in five minutes.* [online] Available at: <https://blynk.io/en/getting-started> [Accessed 12 July 2019]

BurkayKirnik, 2019. *How to Measure Angle With MPU-6050(GY-521).*[online] Available at: <https://www.instructables.com/id/How-to-Measure-Angle-With-MPU-6050GY-521/> [Accessed 12 June 2019]

Cope, S., 2011. *Beginners Guide To The MQTT Protocol.* [online] Available at: <http://www.steves-internet-guide.com/mqtt/> [Accessed 1 June 2019]

EKO Instruments, 2017. *DNI Measurements with a pyrheliometer.* [online] Available at: <https://eko-eu.com/applications/dni-measurements-with-a-pyrheliometer> [Accessed 12 August 2019]

Electrical4U, 2019. *Light Dependent Resistor| LDR and Working Principle of LDR.* [online] Available at: <https://www.electrical4u.com/light-dependent-resistor-ldr-working-principle-of-ldr/> [Accessed 11 August 2019]

Elprocus, 2013. *Darlington Transistor Working along with Applications.* [online] Available at: <https://www.elprocus.com/darlington-transistor-working-with-applications/> [Accessed 12 August 2019]

Fedkin, M., 2018. *5.1. What are concentrating photovoltaics?* [online] Available at: <https://www.e-education.psu.edu/eme812/node/537> [Accessed 12 August 2019]

Google, 2019. *Files you can store in Google Drive.* [online] Available at: <https://support.google.com/drive/answer/37603?hl=en> [Accessed 12 July 2019]

Gupta, A, 2018. *4 Layers Of The Internet Of Things.* [online] Available at: <https://analyticstraining.com/4-layers-of-the-internet-of-things/> [Accessed 13 May 2019]

Gupta, S., 2019. *Transimpedance Amplifier - Current to Voltage Converter.* [online] Available at: <https://circuitdigest.com/tutorial/transimpedance-amplifier-design-working-and-applications> [Accessed 13 August 2019]

Guillemette, P., 2019. *Battle of the IoT networks: Cellular versus Wi-Fi.* [blog] 25 April 2019. Available at: <https://internetofthingsagenda.techtarget.com/blog/IoT-Agenda/Battle-of-the-IoT-networks-Cellular-versus-Wi-Fi> [Accessed 16 August 2019]

harshi1990, 2013. *Solar Tracker.* [online] Available at: <https://www.slideshare.net/harshi1990/solar-tracker> [Accessed 29 April 2019]

Hossain, et al., 2015. Azimuth-Altitude Dual Axis Soalr Tracker. *International Conference on Mechanical Engineering and Renewable Energy,* [e-journal], pp. 1-6. Available through: ResearchGate website <https://www.researchgate.net/publication/286195219_Azimuth-Altitude_Dual_Axis_Solar_Tracker> [Accessed 11 August 2019]

IRENA, 2012. Solar Photovoltaics. *Renewable Energy Technologies: Cost Analysis Series.* [online] Available at: <https://www.irena.org/documentdownloads/publications/re_technologies_cost_analysis-solar_pv.pdf> [Accessed 12 August 2019]

Koutroulis, E. and Kalaitzakis, K., 2003. Development of an integrated data-acquisition system for renewable energy sources systems monitoring. *Renewable Energy 28.* [online] Available at: <https://www.tuc.gr/fileadmin/users_data/elci/Koutroulis/J.03.pdf> [Accessed 12 August 2019]

Landau, C.R., 2017. *Optimum Tilt of Solar Panels.* [online] Available at: <https://www.solarpaneltilt.com/> [Accessed 12 August 2019]

Lee, J.F. and Rahim, N.A., 2013. Performance Comparison of Dual-Axis Solar Tracker VS Static Solar System in Malaysia. *IEEE Conference on Clean Energy and Technology (CEAT),* [e-journal], pp. 102-107. Available through: Universiti Tunku Abdul Rahman Library website < https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/6775608> [Accessed 29 April 2019]

Maas, M., 2012. *Multi-Junction Solar Cells with Concentrators.* [online] Available at: <http://large.stanford.edu/courses/2012/ph240/maas2/> [Accessed 12 August 2019]

MediaFire, 2019. *wifi repeater file. zip.* [online] Available at: <https://www.mediafire.com/file/bf96acaje8gd7jd/wifi_repeater_files.zip/file> [Accessed 12 July 2019]

Microcontroller Tutorials, 2019. *Log Data with NodeMCU and Google Sheets.* [online] Available at: <https://www.teachmemicro.com/log-data-nodemcu-google-sheets/> [Accessed 12 July 2019]

Naked Solar Ltd, 2019. *Solar Panel Mounting.* [online] Available at: <https://nakedsolar.co.uk/solar-pv/solar-panel-mounting/> [Accessed 12 August 2019]

Omni Instruments, 2009. *CMP6 Pyranometer.* [online] Available at: <https://www.omniinstruments.co.uk/weather-stations-and-instruments/pyranometers-solar-irradiance/cmp6-pyranometer.html> [Accessed 12 August 2019]

Pandian, J.D., 2019. *How does the location of sunrise and sunset change throughout the year? (Advanced).* Available at: <http://curious.astro.cornell.edu/people-and-astronomy/161-our-solar-system/the-earth/day-night-cycle/184-how-does-the-location-of-sunrise-and-sunset-change-throughout-the-year-advanced> [Accessed 12 August 2019]

Pons, R., 2019. *Understanding Azimuth and Elevation.* [online] Available at: <https://www.photopills.com/articles/understanding-azimuth-and-elevation> [Accessed 11 August 2019]

Racharla, S. and Rajan, K., 2016. Solar Tracking System - A review. *International Journal of Sustainable Engineering,* [e-journal], pp. 1-56. Available through: ResearchGate website <https://www.researchgate.net/publication/312067334_SOLAR_TRACKING_SYSTEM-_A_REVIEW> [Accessed 11 August 2019]

RandomNerdTutorials.com, 2013. *ESP32 ADC - Read Analog Values with Arduino IDE.* [online] Available at: <https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/> [Accessed 1 July 2019]

RandomNerdTutorials.com, 2013. *Installing the ESP32 Board in Arduino IDE (Windows, MAC OS X, Linux).* [online] Available at: <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/> [Accessed 11 June 2019]

Ray, S. and Tripathi, A.K., 2016. Design and Development of Tilted Single Axis and Azimuth-Altitude Dual Axis Solar Tracking Systems. *IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems,* [e-journal], pp. 1-6. Available through: Universiti Tunku Abdul Rahman Library website <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/7853190> [Accessed 11 August 2019]

RF Wireless World, 2012. *Advantages of Data Loggers | disadvantages of Data Loggers.* [online] Available at: < http://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Data-Loggers.html > [Accessed 13 April 2019]

Solar Power World, 2019. *How does a solar tracker work?* [online] Available at: <https://www.solarpowerworldonline.com/2013/04/how-does-a-solar-tracker-work/> [Accessed 12 August 2019]

SolarSystemQuick.com, 2010. *Polaris Star.* [online] Available at: <https://www.solarsystemquick.com/universe/polaris-star.htm> [Accessed 12 August 2019]

Soniak, M., 2017. *What's the Difference Between "Mostly Sunny" and "Partly Cloudy"?* [online] Available at: <http://mentalfloss.com/article/56820/whats-difference-between-mostly-sunny-and-partly-cloudy> [Accessed 16 August 2019]

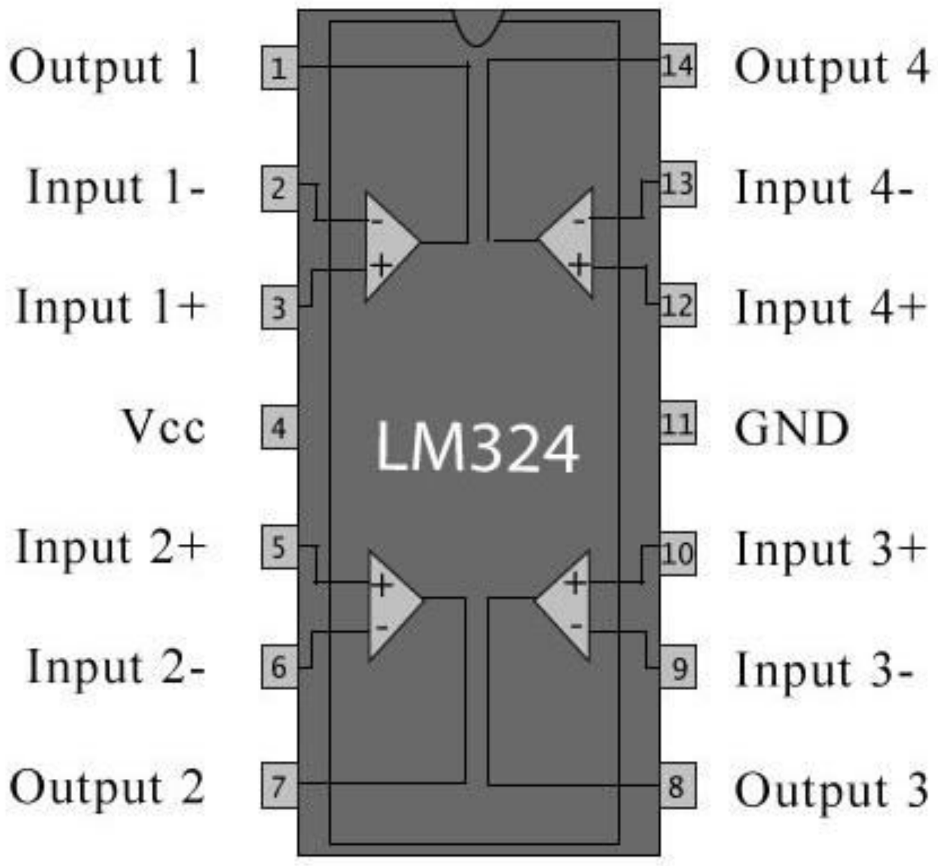
SunPower Corporation, 2019. *Commercial solar panel degradation: What should you know and keep in mind.* [online] Available at: <https://businessfeed.sunpower.com/articles/what-to-know-about-commercial-solar-panel-degradation> [Accessed 12 August 2019]

Thonti, V., 2017. *Relay: Construction, Working and Types.* [online] Available at: <https://circuitdigest.com/article/relay-working-types-operation-applications> [Accessed 12 August 2019]

Whitlock, R., 2016. *Tracking the sun: trackers for solar power systems.* [online] Available at: <https://interestingengineering.com/tracking-the-sun-trackers-for-solar-power-systems> [Accessed 11 August 2019]

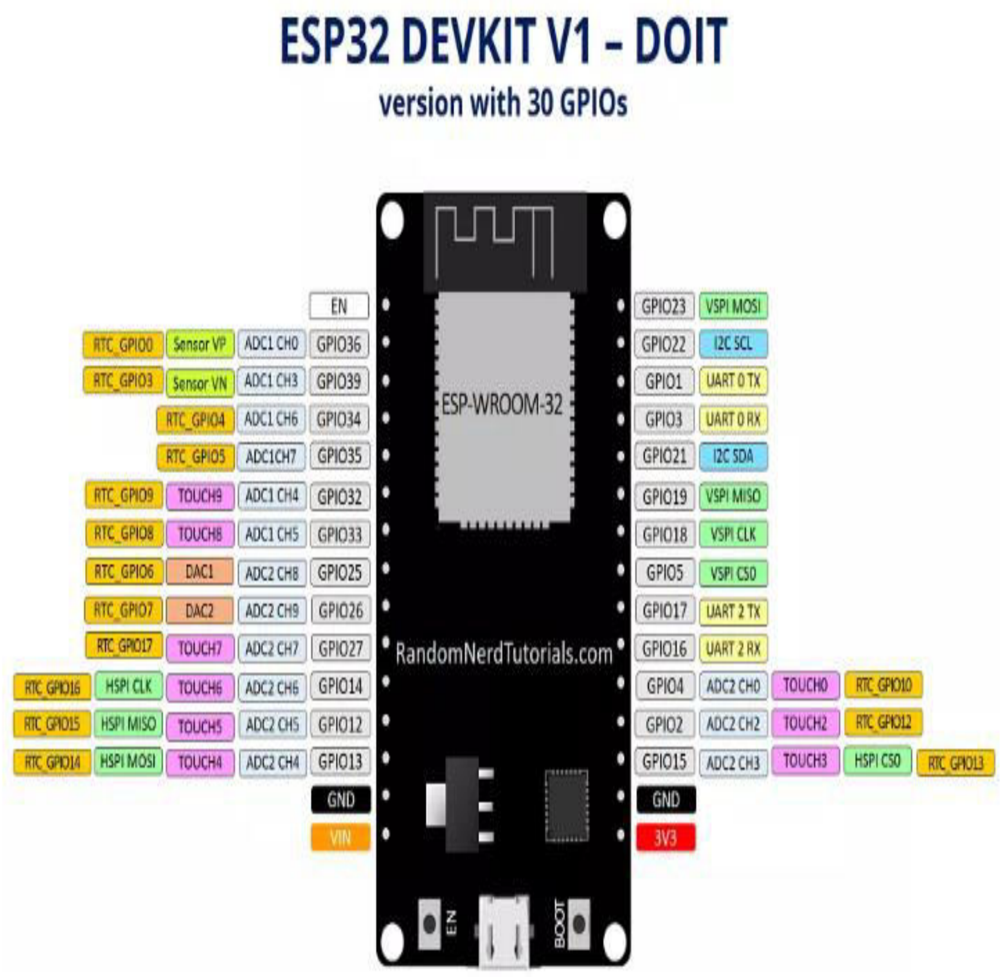
APPENDICES

APPENDIX A: LM324N Pinout

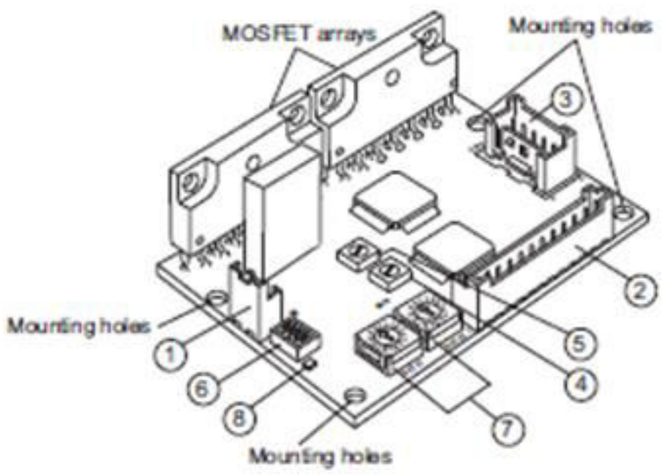


Pinout Lm324n

APPENDIX B: ESP32 Devkit V1 Pinout


ESP32 DEVKIT V1 - DOIT
version with 30 GPIOs
EN
RTC_GPIO0 Sensor VP ADC1 CH0 GPIO36
RTC_GPIO3 Sensor VN ADC1 CH3 GPIO39
RTC_GPIO4 ADC1 CH6 GPIO34
RTC_GPIO5 ADC1CH7 GPIO35
RTC_GPIO9 TOUCH9 ADC1 CH4 GPIO32
RTC_GPIO8 TOUCH8 ADC1 CH5 GPIO33
RTC_GPIO6 DAC1 ADC2 CH8 GPIO25
RTC_GPIO7 DAC2 ADC2 CH9 GPIO26
RTC_GPIO17 TOUCH7 ADC2 CH7 GPIO27
RTC_GPIO16 HSPI CLK TOUCH6 ADC2 CH6 GPIO14
RTC_GPIO15 HSPI MISO TOUCH5 ADC2 CH5 GPIO12
RTC_GPIO14 HSPI MOSI TOUCH4 ADC2 CH4 GPIO13
GND
VIN
GPIO23 VSPI MOSI
GPIO22 I2C SCL
GPIO1 UART 0 TX
GPIO3 UART 0 RX
GPIO21 I2C SDA
GPIO19 VSPI MISO
GPIO18 VSPI CLK
GPIO5 VSPI CS0
GPIO17 UART 2 TX
GPIO16 UART 2 RX
GPIO4 ADC2 CH0 TOUCH0 RTC_GPIO10
GPIO2 ADC2 CH2 TOUCH2 RTC_GPIO12
GPIO15 ADC2 CH3 TOUCH3 HSPI CS0 RTC_GPIO13
GND
3V3
ESP-WROOM-32
RandomNerdTutorials.com
EN
BOOT

APPENDIX C: CRD5107P Motor Driver Datasheet



①Power supply connector (CN1)
Connect to a 24 VDC power supply.

②Input/output signal connector (CN2)
Connect to I/O signals.

③Motor connector (CN3)
Connect to motor leads.

④Motor operating current potentiometer (RUN)
Set the operating current of the motor.
If there is sufficient thrust, the current setting can be reduced to suppress increases in motor/driver temperatures.
The potentiometer is factory set to the rated current.

⑤Motor standstill current potentiometer (STOP)
Set the current when the motor is at a standstill (in the current cutback state).
The potentiometer is factory set to 50% of the rated current.

⑥Function select switches (1P/2P, OFF/SD, R2/R1)
- Pulse input mode select switch (1P/2P): Switch the pulse input mode between 1-pulse input mode and 2-pulse input mode.
- Smooth drive function select switch (OFF/SD): Set or cancel the smooth drive function.
- Resolution select switch (R2/R1): Switch the reference resolution between R1 and R2.

⑦Resolution setting switch (DATA1, DATA2)
You can set a desired resolution by selecting it from among the 16 resolutions.

⑧Power supply input indicator (LED)
This LED remains lit while the power supply is input.

Connector No.	Pin No.	Type	Signal		Description
CN1	1	Input	POWER	+	+24 VDC
	2	Input		–	GND
CN2	1	Input	PLS (CW)	+	Pulse input (CW pulse)
	2	Input		–	
	3	Input	DIR. (CCW)	+	Rotation direction input (CCW pulse)
	4	Input		–	
	5	Input	A.W.OFF	+	All windings off input
	6	Input		–	
	7	Input	C/S	+	Resolution select input
	8	Input		–	
	9	Input	C.D.INH	+	Current cutback release input
	10	Input		–	
	11	Output	TIMING	+	Excitation timing output
	12	Output		–	
CN3	1	Output	MOTOR		Blue motor lead
	2	Output			Red motor lead
	3	Output			Orange motor lead
	4	Output			Green motor lead
	5	Output			Black motor lead

- When this switch is set to 1-pulse input mode, the inputs are the pulse input and the rotation direction input.
- When this switch is set to 2-pulse input mode, the inputs are CW and CCW.

APPENDIX D: AZURSPACE Concentrator Tripe Junction Solar Cell

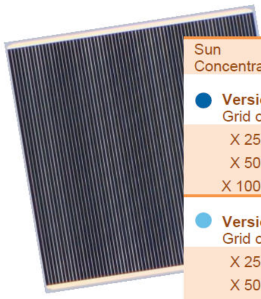
AZURSPACE
SOLAR POWER GMBH

Concentrator Triple Junction Solar Cell

Cell Type: 3C44 – 10 x 10mm²
Application: Concentrating Photovoltaic (CPV) Modules

3C44

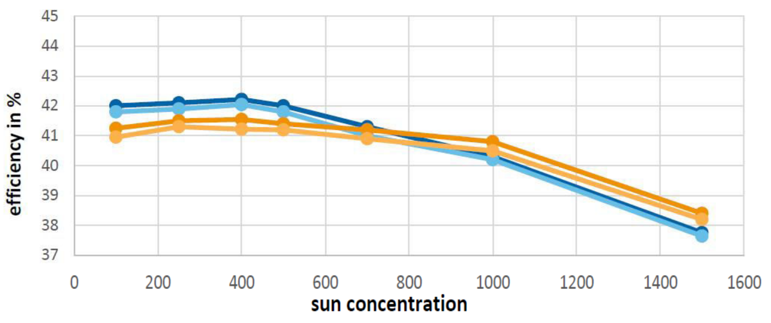
Typical Average Electrical Data



Sun Concentration	I_{sc} [A]	V_{oc} [V]	I_{MPP} [A]	V_{MPP} [V]	P_{MPP} [W_{MPP}]	FF [%]	η [%]
● **Version MC/Air** Grid optimized for medium concentration + Antireflective Coating adapted to air							
X 250	3,85	3,06	3,77	2,80	10,59	89,9%	42,1
X 500	7,66	3,11	7,54	2,81	21,20	88,9%	42,0
X 1000	15,35	3,15	15,07	2,69	40,56	83,8%	40,3
● **Version MC/Glass** Grid optimized for medium concentration + Antireflective Coating adapted to glass							
X 250	3,82	3,07	3,76	2,80	10,55	89,9%	41,9
X 500	7,61	3,11	7,50	2,81	21,04	88,8%	41,8
X 1000	15,36	3,15	14,98	2,70	40,46	83,8%	40,2

Sun Concentration	I_{sc} [A]	V_{oc} [V]	I_{MPP} [A]	V_{MPP} [V]	P_{MPP} [W_{MPP}]	FF [%]	η [%]
● **Version HC/Air** Grid optimized for high concentration + Antireflective Coating adapted to air							
X 250	3,76	3,06	3,68	2,83	10,44	90,7%	41,5
X 500	7,49	3,11	7,38	2,83	20,88	89,6%	41,4
X 1000	15,17	3,14	14,82	2,77	41,08	86,4%	40,8
● **Version HC/Glass** Grid optimized for high concentration + Antireflective Coating adapted to glass							
X 250	3,75	3,06	3,68	2,82	10,39	90,5%	41,3
X 500	7,47	3,10	7,35	2,82	20,73	89,5%	41,2
X 1000	14,97	3,14	14,71	2,77	40,77	86,9%	40,5

Efficiency versus Sun Concentration



Measurement conditions: 1.5 AMd – 1000 W/m² (ASTM G 173-03), T = 25 °C, designated measurement area = 100,51 mm²

Issue date: 2015-04-02
Released for publication (level 1 of 5)

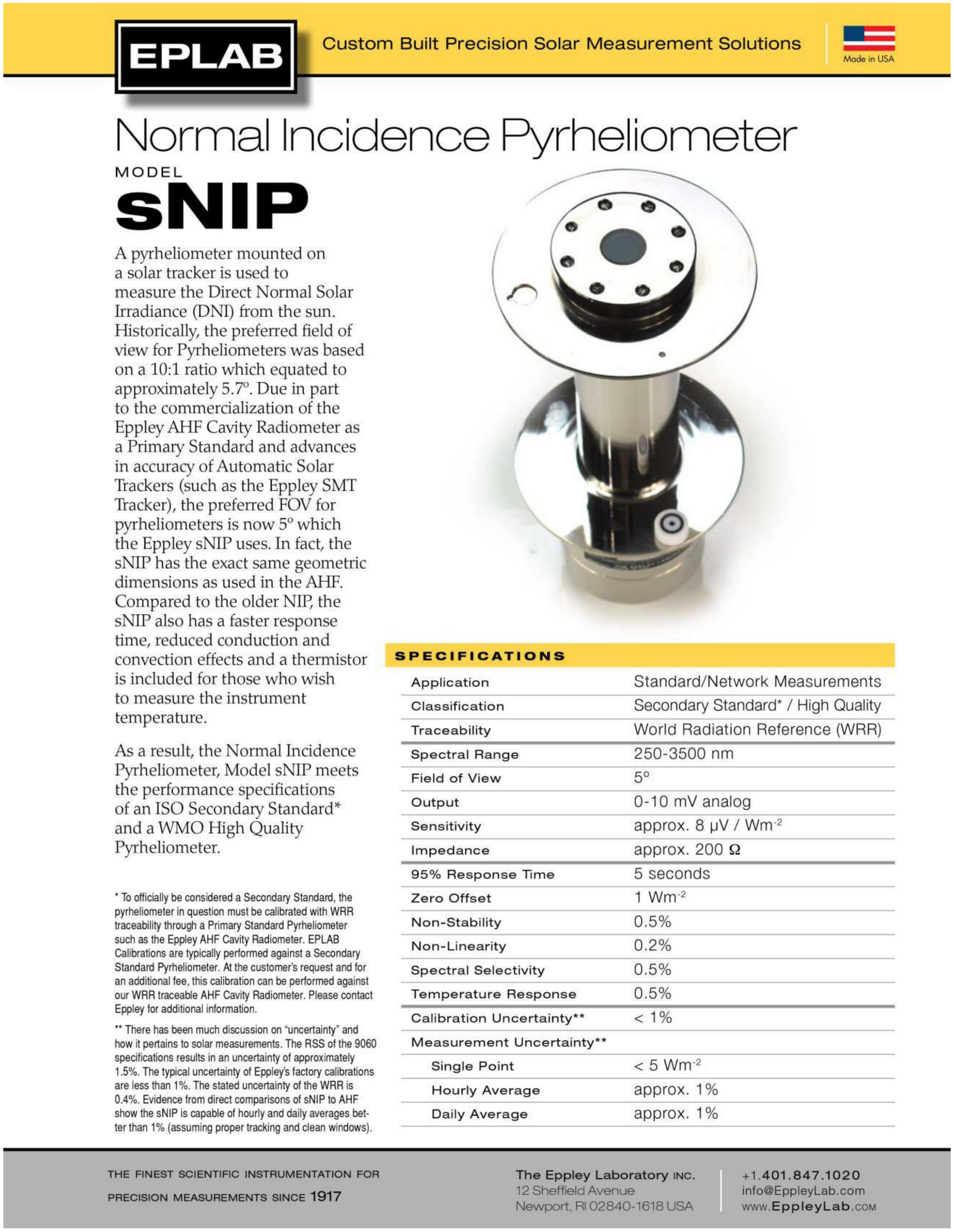
HNR 0004355-00-00
Page **1** of **4**

AZUR SPACE Solar Power GmbH
Theresienstr. 2
74072 Heilbronn
phone: +49 7131 67 2603
telefax: +49 7131 67 2727
e-mail: info@azurspace.com
website: www.azurspace.com

MANAGEMENTSYSTEM
DIN EN ISO 9001:2000
DIN EN ISO14001:2005
OHSAS 18001:1999
DQS certificated, Reg. 062403

CPV

APPENDIX E: Normal Incidence Pyrheliometer Datasheet



EPLAB Custom Built Precision Solar Measurement Solutions

Made in USA

Normal Incidence Pyrheliometer

MODEL

sNIP

A pyrheliometer mounted on a solar tracker is used to measure the Direct Normal Solar Irradiance (DNI) from the sun. Historically, the preferred field of view for Pyrheliometers was based on a 10:1 ratio which equated to approximately 5.7°. Due in part to the commercialization of the Eppley AHF Cavity Radiometer as a Primary Standard and advances in accuracy of Automatic Solar Trackers (such as the Eppley SMT Tracker), the preferred FOV for pyrheliometers is now 5° which the Eppley sNIP uses. In fact, the sNIP has the exact same geometric dimensions as used in the AHF. Compared to the older NIP, the sNIP also has a faster response time, reduced conduction and convection effects and a thermistor is included for those who wish to measure the instrument temperature.

As a result, the Normal Incidence Pyrheliometer, Model sNIP meets the performance specifications of an ISO Secondary Standard* and a WMO High Quality Pyrheliometer.

* To officially be considered a Secondary Standard, the pyrheliometer in question must be calibrated with WRR traceability through a Primary Standard Pyrheliometer such as the Eppley AHF Cavity Radiometer. EPLAB Calibrations are typically performed against a Secondary Standard Pyrheliometer. At the customer's request and for an additional fee, this calibration can be performed against our WRR traceable AHF Cavity Radiometer. Please contact Eppley for additional information.

** There has been much discussion on "uncertainty" and how it pertains to solar measurements. The RSS of the 9060 specifications results in an uncertainty of approximately 1.5%. The typical uncertainty of Eppley's factory calibrations are less than 1%. The stated uncertainty of the WRR is 0.4%. Evidence from direct comparisons of sNIP to AHF show the sNIP is capable of hourly and daily averages better than 1% (assuming proper tracking and clean windows).

SPECIFICATIONS

Application	Standard/Network Measurements
Classification	Secondary Standard* / High Quality
Traceability	World Radiation Reference (WRR)
Spectral Range	250-3500 nm
Field of View	5°
Output	0-10 mV analog
Sensitivity	approx. 8 µV / Wm^{-2}
Impedance	approx. 200 Ω
95% Response Time	5 seconds
Zero Offset	1 Wm^{-2}
Non-Stability	0.5%
Non-Linearity	0.2%
Spectral Selectivity	0.5%
Temperature Response	0.5%
Calibration Uncertainty**	< 1%
Measurement Uncertainty**	
Single Point	< 5 Wm^{-2}
Hourly Average	approx. 1%
Daily Average	approx. 1%

THE FINEST SCIENTIFIC INSTRUMENTATION FOR PRECISION MEASUREMENTS SINCE 1917

The Eppley Laboratory INC.
12 Sheffield Avenue
Newport, RI 02840-1618 USA

+1.401.847.1020
info@EppleyLab.com
www.EppleyLab.com

APPENDIX F: Arduino Test Code (Tilts upwards)

```
/* This is the test code for solar tracker to tilt upwards */

void setup()
{
  pinMode(9,OUTPUT);   // D9 ---> Clockwise Direction
  pinMode(10,OUTPUT);   // D10 ----> Counterclockwise Direction
  pinMode(7,OUTPUT);   // D7 --->  Relay Control

}

void loop()
{
  digitalWrite(7,HIGH);   // Enable the relay control for movement in vertical axis
  digitalWrite (9,LOW);   // Disable clockwise pulse for upward movement

  //Give counterclockwise for upward movement
  digitalWrite(10,HIGH);
  delay(10);
  digitalWrite(10,LOW);
  delay(10);
}
```

APPENDIX G: Arduino Test Code (Tilts downwards)

```
/* This is the test code for solar tracker to tilt downwards */

void setup()
{
  pinMode(9,OUTPUT);   // D9 ---> Clockwise Direction
  pinMode(10,OUTPUT);   // D10 ----> Counterclockwise Direction
  pinMode(7,OUTPUT);   // D7 --->  Relay Control

}

void loop()
{
  digitalWrite(7,HIGH);    // Enable the relay control for movement in vertical axis
  digitalWrite (10,LOW);   // Disable counterclockwise pulse for upward movement

  //Give clockwise pulse for downward movement
  digitalWrite(9,HIGH);
  delay(10);
  digitalWrite(9,LOW);
  delay(10);
}
```

APPENDIX H: Arduino Test Code (Moving Left)

```
/* This is the test code for solar tracker to go left */

void setup()
{
  pinMode(9,OUTPUT);   // D9 ---> Clockwise Direction
  pinMode(10,OUTPUT);  // D10 ----> Counterclockwise Direction
  pinMode(7,OUTPUT);  // D7 --->  Relay Control

}

void loop()
{
  digitalWrite(7,LOW);    // Disable the relay control for movement in horizontal axis
  digitalWrite (10,LOW);  // Disable counterclockwise pulse to move left

  //Give clockwise pulse to move left
  digitalWrite(9,HIGH);
  delay(10);
  digitalWrite(9,LOW);
  delay(10);
}
```

APPENDIX I: Arduino Test Code (Moving Right)

```
/* This is the test code for solar tracker to go right */

void setup()
{
  pinMode(9,OUTPUT); // D9 ---> Clockwise Direction
  pinMode(10,OUTPUT); // D10 ----> Counterclockwise Direction
  pinMode(7,OUTPUT); // D7 --->  Relay Control

}

void loop()
{
  digitalWrite(7,LOW); // Disable the relay control for movement in horizontal axis
  digitalWrite (9,LOW); // Disable clockwise pulse to move right

  //Give counterclockwise pulse to move right
  digitalWrite(10,HIGH);
  delay(10);
  digitalWrite(10,LOW);
  delay(10);
}
```

APPENDIX J: Arduino Test Code (LDR Test)

```
/* This is the test code for LDR reading (without calibration) */

void setup()
{
  Serial.begin(9600); // Set serial monitor for checking LDR readings
}

void loop()
{
  int topLeft = analogRead(A0);   // A0 ---> Top Left LDR
  int topRight = analogRead(A1);  //A1 ---> Top Right LDR
  int bottomLeft = analogRead(A2);  //A2 ---> Bottom Left LDR
  int bottomRight = analogRead(A3);   //A3 ---> Bottom Right LDR

  int avgTop = (topLeft + topRight) / 2;  // Light intensity at upper region
  int avgBottom = (bottomLeft + bottomRight) / 2;  // Light intensity at lower region
  int avgLeft = (topLeft + bottomLeft) / 2;  // Light intensity at left region
  int avgRight = (topRight + bottomRight) / 2;  // Light intensity at right region

  //Serial monitor to display light intensities ...
  Serial.print("Top Left: ");
  Serial.println(topLeft);
  Serial.print("Top Right: ");
  Serial.println(topRight);
  Serial.print("BottomLeft: ");
  Serial.println(bottomLeft);
  Serial.print("BottomRight: ");
  Serial.println(bottomRight);
  Serial.print("Average Top: ");
  Serial.println(avgTop);
  Serial.print("Average Bottom: ");
  Serial.println(avgBottom);
  Serial.print("Average Left: ");
  Serial.println(avgLeft);
  Serial.print("Average Right: ");
  Serial.println(avgRight);
  Serial.println();

  delay(5000);
}
```

APPENDIX K: ESP32 Arduino Test Code (Voltage Reading- Using ESP32)

```
/*This is the test code for ESP32 to read voltage values*/

#include "BigNumber.h" // Library for more decimal points

const int READ = 15;  //Pin D15 of ESP32 to perform ADC reading

/* Global declaration (all set to zero by default) */
BigNumber ADC = 0;   //ADC reading from ESP32
BigNumber VOLTS = 0;   //Voltage converted from Analog input pin D15

void setup()
{
  Serial.begin(115200);

  /*Setting for ADC reading resolution and step size */
  analogReadResolution(12); //12 bits
  analogSetAttenuation(ADC_11db);

  BigNumber::begin(4); // 4 decimal places
}

void loop()
{
  ADC = analogRead(READ);

  VOLTS = ADC * BigNumber ("3.3000") / BigNumber ("4095.0000");

  /* Serial Monitor Checking Purpose */
  Serial.print("VOLTAGE: ");
  Serial.println(VOLTS);
  delay(1000);
}
```

APPENDIX L: ESP32 Arduino Test Code
(Voltage Reading- Using ESP32 with ADS1015 Module)

```
/* This is the test code for ESP32 with the aid of ADS1015 module to read voltage values */
/* Take Note: VDD (ADS1015) to 3V3 (ESP32) & GND (ADS1015) to GND (ESP32) */
/* Take Note (for I2C purpose): pin SCL (ADS1015) to pin D22 (ESP32) , pin SCA (ADS1015) to pin D21 (ESP32) */

#include "BigNumber.h" // Library for more decimal places

//Libraries for ADS1015 module
#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1015 ads; // Global declaration for ADS1015 module

void setup()
{
  Serial.begin(115200);

  /* Setup for ADS1015 module */
  ads.begin();
  ads.setGain(GAIN_ONE); // 1 bit = 2mV, Range: +/- 4.096V

  BigNumber::begin(4); // 4 decimal places

}

void loop()
{
  BigNumber VOLTS; // Voltage value from ADS1015

  VOLTS = BigNumber (ads.readADC_SingleEnded(0)) * BigNumber ("0.0020");  //0.002V = 2mV
  //ads.readADC_SingleEnded(0) means ADC reading at pin A0 of ADS1015 module

  /* Serial Monitor Checking Purpose */
  Serial.print("Voltage: ");
  Serial.println(VOLTS);

  delay(1000);

}
```

APPENDIX M: ESP32 Arduino Test Code
(Voltage-to-Current Convertion - ESP32 with ADC module)

```
/* Take Note: VDD (ADS1015) to 3V3 (ESP32) & GND (ADS1015) to GND (ESP32) */
/* Take Note (for I2C purpose): pin SCL (ADS1015) to pin D22 (ESP32) , pin SCA (ADS1015) to pin D21 (ESP32) */

#include "BigNumber.h" // Library for more decimal places

//Libraries for ADS1015 module
#include <Wire.h>
#include <Adafruit_ADS1015.h>

Adafruit_ADS1015 ads; // Global declaration for ADS1015 module

void setup()
{
  Serial.begin(115200);

  /* Setup for ADS1015 module */
  ads.begin();
  ads.setGain(GAIN_ONE); // 1 bit = 2mV, Range: +/- 4.096V

  BigNumber::begin(4); // 4 decimal places

}

void loop()
{
  BigNumber VOLTS; // Voltage value from ADS1015 (from converter 1)
  BigNumber VOLT1; // from converter 2
  BigNumber mA; // calculated current of converter 1
  BigNumber MA; // calculated current of converter 2

  // Voltage reading formula for ADC module
  VOLTS = BigNumber (ads.readADC_Differential_0_1()) * BigNumber ("0.0020");  //0.0020V = 2mV / bit
  // A0 connected to output pin of converter 1 and A1 to ground (Voltage Differential)

  VOLT1 = BigNumber (ads.readADC_Differential_2_3()) * BigNumber ("0.0020"); //0.0020V = 2mV / bit
  // A2 connected to output pin of converter 1 and A3 to ground (Voltage Differential)

  if (VOLTS > 5)
  {
    mA = 0;
  }

  else
  {
   if (VOLTS < 0)
   {
    mA = 0;
   }

   else
   {
    mA = BigNumber (VOLTS) * BigNumber("1000.00007") / BigNumber ("148.0500"); // 148.05 ohm (between 148 ohm and 148.1
ohm)
   }
  }

  if (VOLT1 > 5)
  {
   MA = 0;
  }

  else
  {
   if (VOLT1 < 0)
   {
    MA = 0;
   }

   else
   {
    MA = BigNumber (VOLT1) * BigNumber("1000.0000") / BigNumber("148.5500"); // 148.55 ohm (between 148.5 ohm and 148.6
ohm)
   }
  }

  /* Serial Monitor Checking Purpose */
  Serial.print("Voltage: ");
  Serial.print(VOLTS);
  Serial.print(" ");
  Serial.println(VOLT1);
  Serial.print("mA: ");
  Serial.print(mA);
  Serial.print(" ");
  Serial.println(MA);
  Serial.println(" ");
}
```

APPENDIX N: Coding for Solar Tracking Mechanism (Arduino UNO)

```
/* Library and Global Declarations for Gyroscope*/
#include<Wire.h>

/* Library for Serial Communication to ESP32 */
#include <SoftwareSerial.h>

SoftwareSerial ESP32 (3,4); //RX, TX

/* Global Declaration for Gyroscope */
const int MPU_addr=0x68;
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
int minVal=265;
int maxVal=402;
double x;
double y;
double z;
double Y_Angle;

/* Global Declaration for Solar Tracking */
int CHECK = 0;
int topLeft = 0;
int topRight = 0;
int Right = 0;
int bottomLeft = 0;
int bottomRight = 0;
int avgTop = 0;
int avgBottom = 0;
int avgLeft = 0;
int avgRight = 0;
int topLEFT = 0;
int topRIGHT = 0;
int rIGHT = 0;
int bottomLEFT = 0;
int bottomRIGHT = 0;
int avgAll = 0;
int avgAll_Previous = 0;
int START;


void setup()
{
  Serial.begin(115200);
  ESP32.begin(115200); // Start Serial Communication with ESP32

  /* Output Declaration for Solar Tracking */
  pinMode(7,OUTPUT); // D7 ---> Relay Control
  pinMode(9,OUTPUT); // D9 ---> Clockwise Control
  pinMode(10,OUTPUT); // D10 ---> Counterclockwise Control
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT); // D12 ---> STOP indicator

  /* Setup for Gyroscope */
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);

  START = 1;
}

void Initialize() //Initialize Y-axis before start to track
{
  ESP32.flush();
  ESP32.write("INITIALIZING...");
  digitalWrite(11,HIGH); // Initialization indicator ON (Start to initialize)

  Y_Angle = Y_ANGLE(); // Get Y-angle

  delay(10);
  while (Y_Angle < 359)
  {
    Y_Angle = Y_ANGLE();

    if (Y_Angle < 200) // if the LDR pointing forward
    {
      /* Tilt upwards until the ldr point perpendicular to the ground */
      UP();
     }

    else if (Y_Angle > 200) // if the LDR pointing backwards
    {
      /* Tilt downwards until the ldr point perpendicular to the ground */
      DOWN();
    }
    delay(1);

    Y_Angle = Y_ANGLE();

    if (Y_Angle > 359)
    {
      break;
    }

    delay(1);
  }
  START = 0;
  delay(1000);
  digitalWrite(11,LOW);
  delay(1000);
}



double Y_ANGLE() // Y-angle calculation
{
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);

  AcX=Wire.read()<<8|Wire.read();
  AcY=Wire.read()<<8|Wire.read();
  AcZ=Wire.read()<<8|Wire.read();

  int xAng = map(AcX,minVal,maxVal,-90,90);
  int yAng = map(AcY,minVal,maxVal,-90,90);
  int zAng = map(AcZ,minVal,maxVal,-90,90);

   x= RAD_TO_DEG * (atan2(-yAng, -zAng)+PI);
   y= RAD_TO_DEG * (atan2(-xAng, -zAng)+PI);
   z= RAD_TO_DEG * (atan2(-yAng, -xAng)+PI);

   return y;
}

void LDR_VALUES() //Read LDR values and average
{
```

```
topLeft = analogRead(A0);
Right = analogRead(A1);
bottomLeft = analogRead(A2);
bottomRight = analogRead(A3);

//Calibration for top right LDR, because it will always get higher reading regardless light intensities
if (Right < 30)
{
  topRight = Right - 27;


  if (topRight < 0 || topRight == 0)
  {
    topRight = 1;
  }
}


else if (Right > 29 && Right < 40)
{
  topRight = Right - 36 ;


  if (topRight < 0 || topRight == 0)
  {
    topRight = 1;
  }
}

else if (Right > 39 && Right < 50 )
{
  topRight = Right - 42;


  if (topRight < 0 || topRight == 0)
  {
    topRight = 1;
  }
}

else if (Right > 49 && Right < 60)
{
  topRight = Right - 52;
```

```
   if (topRight < 0 || topRight == 0)
   {
     topRight = 1;
   }
  }

  else
  {
   topRight = Right;
  }

  avgTop =(topLeft + topRight) / 2;
  avgBottom = (bottomLeft + bottomRight) / 2;
  avgLeft =  (topLeft + bottomLeft) / 2;
  avgRight =  (topRight + bottomRight) / 2;
 }

void LDR_CHECK() // Checing and comparing LDR values when stop
{
  topLEFT = analogRead(A0);
  rIGHT = analogRead(A1);
  bottomRIGHT = analogRead(A2);
  bottomRIGHT = analogRead(A3);

  //Calibration for top right LDR, because it will always get higher reading regardless light intensities
  if (rIGHT < 30)
  {
   topRIGHT = rIGHT - 27;


   if (topRIGHT < 0 || topRIGHT == 0)
   {
     topRIGHT = 1;
   }
  }


  else if (rIGHT > 29 && rIGHT < 40)
  {
   topRIGHT = rIGHT - 36;


   if (topRIGHT < 0 || topRight == 0)
```

```
    {
      topRIGHT = 1;
    }
  }

  else if (rIGHT > 39 && rIGHT < 50 )
  {
    topRIGHT = rIGHT - 42;



    if (topRIGHT < 0 || topRight == 0)
    {
      topRIGHT = 1;
    }
  }

  else if (rIGHT > 49 && rIGHT < 60)
  {
    topRIGHT = rIGHT - 52;
    if (topRIGHT < 0 || topRIGHT == 0)
    {
      topRIGHT = 1;
    }
  }

  else
  {
    topRIGHT = rIGHT;
  }
}

void STOP() // STOP
{
  digitalWrite(12,HIGH); // STOP indicator HIGH
  digitalWrite(7,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
}

void BREAK() // BREAK
{
  digitalWrite(12,LOW);
  digitalWrite(7,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
}

void UP() // GO UP
{
  digitalWrite(7,HIGH);
  digitalWrite(9,LOW);
  digitalWrite(10,HIGH);
  delay(0.5);
  digitalWrite(10,LOW);
  delay(0.5);
 }

void DOWN() // GO DOWN
{
  digitalWrite(7,HIGH);
  digitalWrite(10,LOW);
  digitalWrite(9,HIGH);
  delay(0.5);
  digitalWrite(9,LOW);
  delay(0.5);
}

void LEFT() // GO LEFT
{
  digitalWrite(7,LOW);
  digitalWrite(10,LOW);
  digitalWrite(9,HIGH);
  delay(0.5);
  digitalWrite(9,LOW);
  delay(0.5);
}

void RIGHT() // GO RIGHT
{
  digitalWrite(7,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,HIGH);
  delay(0.5);
  digitalWrite(10,LOW);
  delay(0.5);
}
```

```
void loop() // Main Program
{
 if (START == 1)
 {
  Initialize();
 }
 /* Set all outputs to LOW (Initialization) */
 digitalWrite(7,LOW);
 digitalWrite(9,LOW);
 digitalWrite(10,LOW);
 digitalWrite(12,LOW);

 /* Read LDR values before decideing to stop or to track */
 LDR_VALUES();

 /* IF TOO DARK, STOP THE TRACKER */
 if ((topLeft > 8) && (topRight > 8) && (bottomLeft > 8) && (bottomRight > 8))
 {
  ESP32.flush();
  ESP32.write("CLOUDY(STOP)");
  while ((topLeft > 8) && (topRight > 8) && (bottomLeft > 8) && (bottomRight > 8))
  {
   STOP(); // STOP the tracker
   digitalWrite(11,HIGH);
   LDR_VALUES(); // Get LDR values

   if ((topLeft < 6) || (topRight < 6) || (bottomLeft < 6) || (bottomRight < 6)){
    BREAK();  // If not so dark, terminate the stop loop
    delay(5000);
    break;
    }

    delay(1000); // If it is still too dark, wait for 1s before checking the new LDR values in this stop loop.
  }
 }

 /* IF NOT SO DARK, TRACK! */
 else
 {
  digitalWrite(12,LOW); // Turn off STOP indicator because the solar tracker does not stop


   ESP32.flush();
   ESP32.write("SUNNY(TRACKING)");
   delay(2000);
```

```
LDR_VALUES(); // Read LDR values before proceed to compare the light intensity of both upper and lower regions

/* CHECK UP/DOWN */
//IF UPPER  REGION BRIGHTER
if (avgTop < avgBottom)
{
 //Serial.println("UP");
 while(avgTop < avgBottom) //While looping for going UP
 {
  Y_Angle = Y_ANGLE(); //read Y-angle

  if ((Y_Angle < 305) && (Y_Angle > 300))
  {
   BREAK(); // Terminate if the solar tracker tilts to its upward limit
   break;
  }

  else // If the solar tracker does not reach the limit to move upwards
  {
   if ((abs(avgTop - avgBottom)< 2) || (abs (topLeft - bottomLeft) < 1) || (abs (topRight - bottomRight) < 1))
   {
    BREAK(); // Terminate if the light intensity of upper and lower region are the same
    break;
   }

   else if (avgTop > avgBottom)
   {
    BREAK(); // Terminate if the lower region is brighter than the upper region
    break;
   }

    else
    {
     UP(); // Go UP if the solar tracker does not reach the upward limits, the light intensity of both upper and lower region are
different, and upper region is still brighter than lower region
    }
  }

  LDR_VALUES(); //Get LDR values after going UP for 1 pulse, then repeat the upward movement until it is terminated
 }
}

//IF LOWER REGION BRIGHTER
```

```
 else if (avgTop > avgBottom)
 {
  //Serial.println("DOWN");
  while(avgTop > avgBottom) //While looping for going DOWN
  {
   Y_Angle = Y_ANGLE(); // read Y-angle

   if ((Y_Angle > 40) && (Y_Angle < 50))
   {
    BREAK(); //Terminate if the solar tracker tilts to its downwards limit
    break;
   }

   else // If the solar tracker does not reach to its downwards limit
   {
    if ((abs(avgTop - avgBottom)< 2) || (abs (topLeft - bottomLeft) < 1) || (abs (topRight - bottomRight) < 1))
    {
     BREAK(); //Terminate if the light intensity of upper and lower region are the same
     break;
    }

    else if (avgTop < avgBottom)
    {
      BREAK(); //Terminate if the upper region is brighter than the lower region
      break;
      }

     else
     {
      DOWN(); // Go DOWN if the solar tracker does not reach to its downwards limit, the intensity of upper and lower region
are different, and lower region is still brighter than upper region
     }
   }

   LDR_VALUES(); //Get LDR values after going DOWN for 1 pulse, then repeat the downward movement until it is terminated
  }
 }

 delay(1000); // Delay for 1s before proceed to check LEFT/RIGHT...

 /* CHECK LEFT/RIGHT */
 //IF RIGHT REGION BRIGHTER
 if (avgRight < avgLeft)
 {
```

```
//Serial.println("RIGHT");
while (avgRight < avgLeft) //While looping for operation when right region is brighter than left region
{
 Y_Angle = Y_ANGLE(); // read Y-angle

 //IF THE LDR POINTING UPWARDS PERPENDICULAR TO THE GROUND OR POINTING BACKWARDS
 if ((Y_Angle > 300))
 {
  if ((abs(avgRight - avgLeft)< 2) || (abs (topLeft - topRight) < 1) || (abs (bottomLeft - bottomRight) < 1))
  {
   BREAK(); //Terminate if the light intensity of right and left region are the same
   break;
  }

  else if (avgRight > avgLeft)
  {
   BREAK(); // Terminate if left region is brighter than right region
   break;
   }

   else
   {
    if (avgTop < avgBottom)
    {
     LEFT(); //If the 2 conditions above are not fulfilled, go LEFT
    }

    else
    {
     RIGHT();
    }
   }
  }

 //IF THE LDR POINTING FORWARD
 else
 {
  if ((abs(avgRight - avgLeft)< 2) || (abs (topLeft - topRight) < 1) || (abs (bottomLeft - bottomRight) < 1))
  {
   BREAK(); //Terminate if the light intensity of right and left region are the same
   break;
  }

  else if (avgRight > avgLeft)
```

```
        {
         BREAK(); //Terminate if left region is brighter than right region
         break;
        }

        else
        {
         RIGHT(); //If the 2 conditions above are not fulfilled, go RIGHT
        }
      }

      LDR_VALUES(); //Get LDR values after 1 pulse of movement (either LEFT or RIGHT) is done, then repeat the cycle until it is
terminated
    }
  }

  //IF LEFT REGION BRIGHTER
  else if (avgRight > avgLeft)
  {
    //Serial.println("LEFT");
    while (avgRight > avgLeft) //While looping for operation when left region is brighter than right region
    {
     Y_Angle = Y_ANGLE(); //read Y-angle

     //IF THE LDR POINTING UPWARDS PERPENDICULAR TO THE GROUND OR POINTING BACKWARDS
     if ((Y_Angle > 300))
     {
      if ((abs(avgRight - avgLeft)< 2) || (abs (topLeft - topRight) < 1) || (abs (bottomLeft - bottomRight) < 1))
      {
        BREAK(); //Terminate if the light intensity of right and left region is the same
        break;
      }

      else if (avgRight < avgLeft)
      {
        BREAK(); //Terminate if right region is brighter than left region
        break;
      }

      else
      {
        if(avgTop < avgBottom)
        {
          RIGHT(); //If the 2 conditions above are not fulfilled, go RIGHT
          }

          else
          {
            LEFT();
          }
         }
        }

      //IF THE LDR POINTING FORWARD
      else
      {
       if ((abs(avgRight - avgLeft)< 2) || (abs (topLeft - topRight) < 1) || (abs (bottomLeft - bottomRight) < 1))
       {
         BREAK();//Terminate if the light intensity of right and left region are the same
         break;
       }

       else if (avgRight < avgLeft)
       {
         BREAK(); //Terminate if right region is brighter than left region
         break;
        }

        else
        {
          LEFT(); // If the 2 conditions above are not fulfilled, go LEFT
        }
      }

       LDR_VALUES(); //Get LDR values after 1 pulse of movement (either LEFT or RIGHT) is done, then repeat the cycle until it is
terminated
    }
   }

   delay (1000); // Give a delay for 1s before checking condition to enter STOP loop

   avgAll = (topLeft + topRight + bottomLeft + bottomRight) / 4; // First, calculate recent overall average

   if ((abs(avgAll - avgAll_Previous) < 2) && (avgAll_Previous > 0))
   {
     CHECK = CHECK + 1; //Increment the iteration of counter to enter STOP loop if the recent overall average has only slight
difference with the previous overall average
   }
```

```
  else
  {
    CHECK = CHECK - 1; //Decrement the iteration of counter to enter STOP loop if the recent overall average differs a lot from
the previous overall average

    if (CHECK < 0)
    {
      CHECK = 0; // Set number of iteration to be zero if the number of iteration is reduced to be less than zero
    }
  }

  avgAll_Previous = avgAll; //Update the previous overall average with the recent one

  delay(1000); // delay another 1s

  if (CHECK == 5)
  {
    ESP32.flush();
    ESP32.write("SUNNY(STOP)");
    //STOP if the number of iteration is equal to 5
    //Assuming the solar tracker is facing the SUN
    while(CHECK == 5) //While loopig when the number of iteration is equal to 5...
    {
      STOP(); // STOP the solar tracker
      delay(5000);
      LDR_CHECK(); //Get the current LDR values at STOP position, and compare with the LDR values when entering STOP looop
      if ((abs(topLeft-topLEFT)> 2) || (abs(topRight-topRIGHT)> 2)|| (abs(bottomLeft-bottomLEFT)> 2) || (abs(bottomRight-
bottomRIGHT)> 2) )
      {
        delay(1000);
        BREAK(); //Terminate if there is difference of light intensity at any one of the LDRs or when the solar tracker is not
pointing the SUN
        CHECK=0; //Then set the number of iteration to zero
        break;
      }
    }
  }

  delay(1000); //delay another 1s before proceed for solar tracking...
 }
}
```

APPENDIX O: Coding for IoT Data Collection System (ESP32 Arduino)

```
#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>
#include "BigNumber.h"

/* Declaration of Indicators */
int LED_BUILTIN = 2; // Pin D2 (ESP32 built-in LED) ---> WiFi Connection Indicator
int PUSH = 15; // Pin D15 ---> Data Transfer Indicator (Google Sheet)
int Data = 4; // Pin D4 ---> Data Transfer Indicator (Blynk)

String STATS = " ";

int INITIALIZE;
unsigned long lastSend;
unsigned long pushingbox;
unsigned long current;

Adafruit_ADS1015 ads; // Global declaration for ADS1015 module

double Pyrheliometer;
double Pyranometer;

const char* ssid = "Solar_Tracker"; //Wi-Fi network you want to connect
const char* password = "UTAR_1234";

const char host[] = "api.pushingbox.com";
const int httpPort = 80;

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
const char auth[] = "951c69db77154adaa6c754fcbb90ed8d";

BLYNK_READ(V1)
{
  Blynk.virtualWrite(V1, Pyrheliometer);
}

BLYNK_READ(V2)
```

```
{
  Blynk.virtualWrite(V2, Pyranometer);
}

WidgetLCD lcd (V0);

void setup()
{
  Serial.begin(115200);

  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(Data, OUTPUT);
  pinMode(PUSH, OUTPUT);

  /* Setup for ADS1015 module */
  ads.begin();
  ads.setGain(GAIN_ONE); // 1 bit = 2mV, Range: +/- 4.096V

  BigNumber::begin(4); // 4 decimal places

  WiFi.begin(ssid, password);

  Blynk.begin(auth, ssid, password);

  lcd.clear();

  INITIALIZE = 1;
  lastSend = 0;
  pushingbox = 0;

}

void Wifi()
{
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  {
    digitalWrite(LED_BUILTIN, LOW); //ESP32 LED LOW if connected to WiFi
    delay(500);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(500);
    Serial.println("Connecting to WiFi ...");
  }
```

```
 digitalWrite(LED_BUILTIN, HIGH);
 Serial.print("Connected to the WiFi network! "); //ESP32 LED HIGH if connected to WiFi
}

void Solar_Irradiance()
{
 BigNumber VOLTS; // Voltage value from ADS1015
 BigNumber VOLT1;
 BigNumber mA;
 BigNumber MA;

 VOLTS = BigNumber (ads.readADC_Differential_0_1()) * BigNumber ("0.0020");
 VOLT1 = BigNumber (ads.readADC_Differential_2_3()) * BigNumber ("0.0020");

 if (VOLTS > 5)
 {
  mA = 0;
 }

 else
 {
  if (VOLTS < 0)
  {
   mA = 0;
  }

  else
  {
   mA = BigNumber (VOLTS) * BigNumber("1000.00007") / BigNumber ("148.0500"); // 148.05 ohm (between 148 ohm and
148.1 ohm)
  }
 }

 if (VOLT1 > 5)
 {
  MA = 0;
 }

 else
 {
  if (VOLT1 < 0)
  {
   MA = 0;
  }

  else
  {
   MA = BigNumber (VOLT1) * BigNumber("1000.0000") / BigNumber("148.5500"); // 148.55 ohm (between 148.5 ohm and 
148.6 ohm)
  }
 }

 /* Serial Monitor Checking Purpose */
 Serial.print("Voltage: ");
 Serial.print(VOLTS);
 Serial.print(" ");
 Serial.println(VOLT1);
 Serial.print("mA: ");
 Serial.print(mA);
 Serial.print(" ");
 Serial.println(MA);
 Serial.println(" ");

 BigNumber Pyrh = mA/ BigNumber ("15.3000") * BigNumber("1000.0000");
 BigNumber Pyra = MA/ BigNumber ("15.3000") * BigNumber("1000.0000");

 Pyrheliometer = round(Pyrh * BigNumber ("1000.0000")) / double(1000); // round up to 2 decimal places
 Pyranometer = round(Pyra * BigNumber("1000.0000")) / double(1000); // round up to 2 decimal places

 Serial.print("Pyrheliometer:");
 Serial.println(Pyrheliometer);
 Serial.print("Pyranometer:");
 Serial.println(Pyranometer);

 if (Serial.available())
 {
  STATS = Serial.readString();
 }

 lcd.clear();
 lcd.print(0, 0, "STATUS :");
 lcd.print(0, 1, STATS);

 Serial.print("STATS: ");
 Serial.println(STATS);

}
```

```
void loop()
{
 digitalWrite(LED_BUILTIN, HIGH);

 digitalWrite(Data, LOW);
 digitalWrite(PUSH, LOW);

 while ( WiFi.status() != WL_CONNECTED)
 {
  digitalWrite(LED_BUILTIN, LOW);
  digitalWrite(Data, LOW);
  digitalWrite(PUSH, LOW);
  Wifi();
 }


 if ( millis() - lastSend > 1000 ) // Update and send only after 1 seconds
 {
  Serial.println(millis());

  while (!Blynk.connected())
  {
   Serial.println("Connecting to Blynk...");
   digitalWrite(LED_BUILTIN, LOW);
   digitalWrite(Data, LOW);
   digitalWrite(PUSH, LOW);

   if ( WiFi.status() != WL_CONNECTED)
   {
    digitalWrite(LED_BUILTIN, LOW);
    digitalWrite(Data, LOW);
    digitalWrite(PUSH, LOW);
    Wifi();
   }

   Blynk.begin(auth, ssid, password);

   if (Blynk.connected())
   {
    Serial.println("[DONE]");
    break;
   }
  }
  digitalWrite(Data, HIGH);
  Solar_Irradiance();

  Blynk.run();
  Serial.println("SENT TO BLYNK !");

  lastSend = millis();
  Serial.println(" ");
}

 if ( millis() - pushingbox > 300000) //Update and send after 5 minutes
  {
   WiFiClient client;

   Serial.println(millis());

   while (!client.connected())
   {
    Serial.println("Connecting to Pushingbox API...");
    digitalWrite(LED_BUILTIN, LOW);
    digitalWrite(Data, LOW);
    digitalWrite(PUSH, LOW);

    if ( WiFi.status() != WL_CONNECTED)
    {
     digitalWrite(LED_BUILTIN, LOW);
     digitalWrite(Data, LOW);
     digitalWrite(PUSH, LOW);
     Wifi();
    }

    if (client.connect(host, httpPort))
    {
     Serial.println( "[DONE]" );
     digitalWrite(LED_BUILTIN, HIGH);
    }
   }
   digitalWrite(PUSH, HIGH);

   // We now create a URL for the request
   String url = "/pushingbox?";
   url += "devid=";
   url += "v99B1CB5254AB3D5";
```

```
    url += "&Pyrheliometer(W/m2)=" + String(Pyrheliometer);
    url += "&Pyranometer(W/m2)=" + String(Pyranometer);

    // This will send the request to the server
    client.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection: close\r\n\r\n");
    Serial.println("SENT TO GOOGLE FORM!");
    client.flush();
    client.stop();
    pushingbox = millis();



    return;
  }
}
```