ON-SITE CONDITION MONITORING OF PV MODULES ASSISTED BY AERIAL THERMOGRAPHIC IMAGE ANALYSIS

FINAL REPORT

RDU1703128

UNIVERSITI MALAYSIA PAHANG

ACKNOWLEDGEMENTS

Special acknowledgment the Research & Innovation Department, Universiti Malaysia Pahang, for funding through an internal research grant scheme (Vot: RDU1703128) that has supported this research. Thank Faculty of Electrical and Electronic Engineering Technology for providing the facilities and equipment needed to produce and complete the project.

ABSTRACT

The aim of this project is to analyze the most major problem in PV module, which is the solar hotspot. Many causes can lead to this kind of defects towards a PV module such as shading effect, impurities present on the module surface, and many more. In this research, the system will capture infrared images of the PV module using an infrared thermal camera and display the images on the LCD display. Region of module which having abnormal temperature can be detected. This region needed to be analysed in order to know if it is a hotspot or not. This system will analyze or check that particular region to automatically determine where is the hotspot had occurred.

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 Project Background

There has been an increasing global awareness of solar PV energy use [1]. The reasons behind this are not only to address climate change, but also to create new financial opportunities and, most importantly, to give access to energy for countless people still deprived of modern energy facilities. There is another alternative energy that had been heavily developed nowadays, such as hydrogen, biomass, geothermal, wind, wave, tidal, hydro, and nuclear. However, solar energy can be said as the most ‘green’ energy due to its non-pollutant process to the environment. Solar energy is inexhaustible and clean. Hence, solar energy from the Sun is the ideal energy.

Solar energy is the process of changing solar irradiance into electrical energy. Earth's solar energy irradiation is extensive, approximately 40 minutes of solar radiation on earth, sufficient for a year's global consumption of human energy [2]. This solar irradiance is converted to electrical using a PV module efficiently. However, there is a major problem that cannot be overcome, but only can be minimized using a bypass diode which is the solar hotspot. This problem is a common case that will or happen when operating a PV module. Overcoming solar hotspot problems is important to keep the sustainability of an installed PV system by reducing any source of the fault, structural defect, or malfunction during either manufacturing or operating stage.

Hotspot phenomenon is a degrading performance occasion for the PV module. It is caused due to shading effect cell degradation or any impurities present on the surface of the solar module that prevents any light from entering the cell. When there is one cell is blocked from getting light, it will operate in a reverse-biased while others will do the other hand. Thus, damaged cells will dissipate power and consequently resulting in an abnormal rise in temperature. Temperature, solar irradiance, and spectral effects [3], as well as the degradation of PV cells and modules [4], often appears after a few years of operation and continues to increase after that, are the main factors which attribute to the discrepancy observed. The extent of cell and module natural degradation after 20–25 years of performance in silicon module productions can nowadays be observed [5]. Due to the complex manner in which these defects appear and interrelate, a deeper understanding of the nature of these defects and the degree to which they correlate with a reduction in PV performance and efficiency is of prime importance both for the early and accurate defect detection in existing technologies and the offering of highly improved PV systems.

This project focus on identifying the hotspot phenomenon that happen on the surface of the PV module. Thermal images of the overall cell temperature are taken using an infrared camera. This study developed a non-invasive technique that can detect localized heating and quantify the area of the hotspots, a potential cause of degradation in PV systems. This is done by the use of infrared thermography, a well-accepted nondestructive evaluation technique that allows contactless, real-time inspection. The images will be quantized according to their region and the temperature difference between the region with ambient temperature is calculated. The system developed allows users to visually determine solar hotspot using infrared thermography. This method is proved to be invasive towards the PV module as it does not have physical contact with the PV process.

1.2 Problem Statement

This hotspot analyzer for the PV module can detect solar hotspots with minimal procedures that need to be done to detect solar hotspot. It can save more time and give the user an easier way to instantaneously know which solar cell on the PV module have hotspot. The previous method has numerous steps that need to conduct such as data collection, calculation of temperature, measuring output power, and image processing. Solar hotspot occasion sometimes is misjudged by analyzing using the visual method because high temperature in solar cell cannot be proven that particular cell has a hotspot.

1.3 Objective

The main objective is to develop a system that can allow the user to detect hotspot on a solar module instantly.

(i) To design an automatic hotspot detection of a solar module using infrared image analysis.
(ii) To determine the severity level of the hotspot of the PV module.
(iii) To developed a non-invasive technique that can detect and localize heating and quantify the area of the hotspots.

1.4 Scope of Project

The scope of the research consists of data collection, data analysis, system modeling and development. The system design is explained in the research design section. In summary, the scopes of the research are listed below:

(i) Data collection of cell temperature from the number of solar modules.

(ii) Effectiveness of the system design algorithm to calculate temperature difference and image processing.

(iii) Analyzing the current problem of another method which is similar objective with this project.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This research focuses more on quick identification of hotspot on the solar panel. However, a clear background of the hotspot must be discussed before entering the next part. The general causes of the hotspot in solar panels divided into two parts, what is manufacture defect factor, and the other one is an external factor such as shading, water corrosion and mores. Hotspot heating occurs when they are shading on cells, the reduced short-circuit current of affected cells becomes lower than the operating current of the module. This will force affected cells to change into a reverse bias condition. The reverse bias condition makes the affected behave like an internal load, and started dissipating the power in the form of heat, which is further causing deformation of the diode inside the solar panel.

In any place like industry or university lab, they used module analyzer to perform the analysis in evaluating solar module. This is because the module analyzer able to provide information such as the I-V curve with the cursor , short circuit current,open-circuit voltage and etc. The information provided is reliable and accurate for determining the fault level that happened inside the solar cell. However, to be able to perform a full check with a short time, a thermal camera appears as a solution to the problem of tracing hotspot in a big solar farm. Thermography was developed to improve the visibility of objects in a dark environment by detecting the object’s infrared radiation. The hotspot on the solar panel emits infrared energy , the energy is known as its heat signature so that the thermal camera can differentiate between the difference in temperature due to the heat signature.

2.2 Heat detection

Heat detection is an essential part of this project. For this project, the primary heat sensor used is the OMRON D6T MEMS sensor. For comparing the accuracy on detecting hotspot, the device such as thermal camera and pyrometer.

2.2.1 Thermal Camera

Thermal cameras are functioning like a normal camera. It has the field of view (FOV) vary between 6 ◦for a telescopic optic and 48◦ for a wide angle optic. For a normal thermal camera, the radiation density does not affect by the object distance to the camera because the thermal camera is increasing every single pixel when the part of the image is increasing. Therefore, temperature measurements are not influenced by the distance between object and camera [6].

Inside every infrared camera, the part whose responsible for thermal imaging is none other than focal plane array (FPA) .Focal plane array is a sensor with 2 dimensional detector pixel matrix specially for certain light spectrum like infrared .The size of this image sensor is ranged between 20 to 1 million pixels [7]. The thermal camera and the diagram of FPA is shown in Figure 2-1 and Figure 2-2 .The Figure 2-3 shows the thermal imaging system by focal plane array .

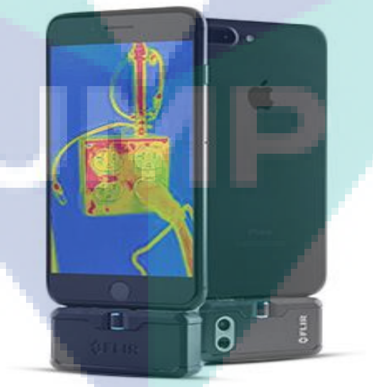


Figure 2-1 FLIR ONE PRO thermal camera

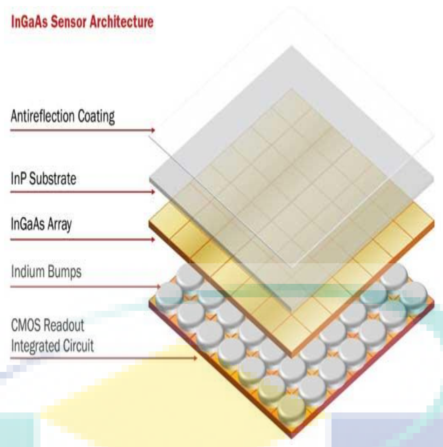


Figure 2-2 The schematic of focal plane array

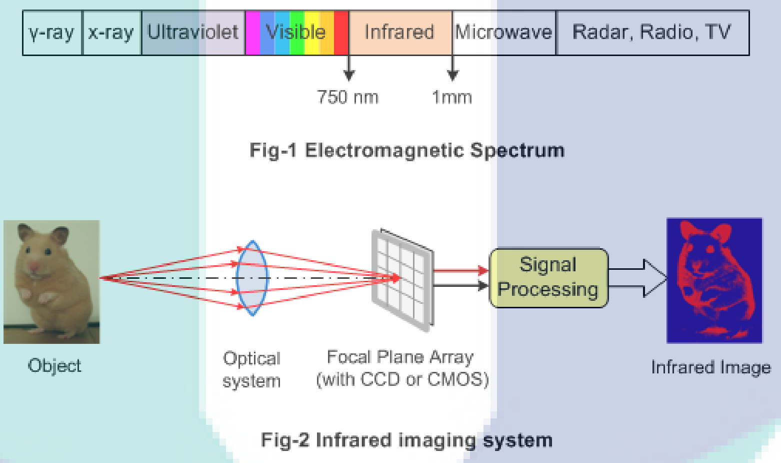


Figure 2-3 Infrared imaging system

2.2.2 Thermal mem sensor

Omron thermal MEM sensor is a sensor to detect infrared radiation and suitable to detect the surrounding temperature . The function of the Omron sensor is to measure the surface temperature of an object where the sensor detects the intensity of the infrared radiation . [8]The sensor use custom designed sensor ASIC and signal processing microprocessor and algorithm into tiny package .[8]

The thermal sensor detects the temperature through a convex silicon lens .The lens has a view angle of 44.2° for x axis and 45.7° for y axis .The MEM sensor consist of thermopile array and the amount of thermopile array is vary according which sensor you are using .For this D6T-44L thermal sensor ,it consists of 16 thermopile array .Inside the sensor ,it has ASIC embedded inside the silicon lens and a microcontroller in PCB to process the temperature data .There is connector from the PCB to allow user connect the external microcontroller to itself and communicate through I^2C .The Figure 2.4 shows the characteristic and the field of view of MEM sensor which has 4x4 array that attribute to 16 pixels and the right figure is the FOV of 1 pixel . Figure 2-5 shows the product structure of MEM sensor .

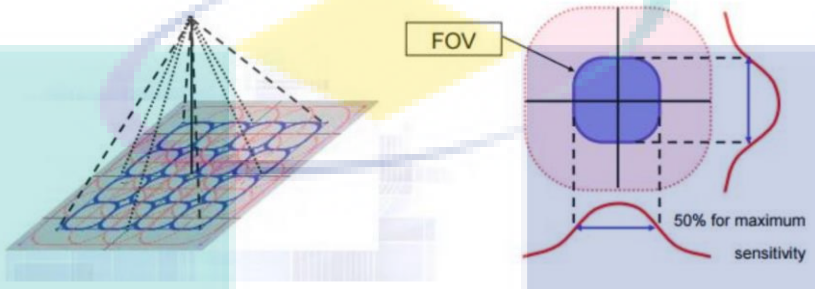


Figure 2-4 Sensitivity Characteristic

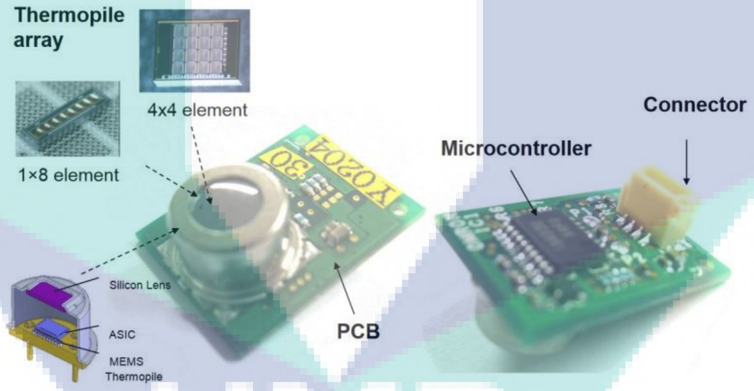


Figure 2-5 Product structure of MEM sensor

The silicon lens consist of thermopile array to gather radiated infrared and direct on itself .The radiated infrared is then transduced into electrical signal and send to the ASIC to convert the electrical signal from sensor signal to digital temperature output .The process of the working principle is shown in Figure 2-6.

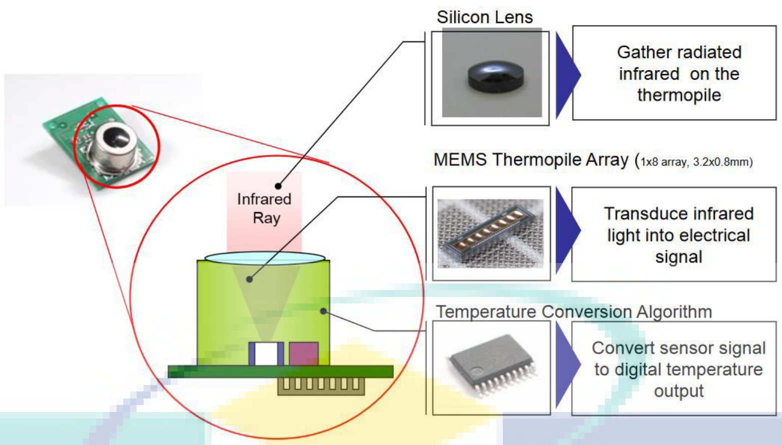

Silicon Lens
Gather radiated infrared on the thermopile
MEMS Thermopile Array (1x8 array, 3.2x0.8mm)
Transduce infrared light into electrical signal
Temperature Conversion Algorithm
Convert sensor signal to digital temperature output
Infrared Ray

Figure 2-6 Working principle of MEM sensor

2.3 Solar cells (Solar panel)

A solar cell is a device to convert sunlight to electrical energy . Sunlight which shines on the solar panel produces both current and voltage .To optimize the optimal energy conversion ,solar panel must be made from a material which absorption of light able to raise the energy level of electron .The electron moves from solar cell to external circuit to dissipate energy and return back to solar cell .

A solar panel is usually made of silicon .They are interconnected silicon cells which form a circuit .The majority of solar panel used wafer-based crystalline cells (c-Si)or thin-film .By theoretically ,crystalline silicon is chosen for manufacturing solar panel due to certain reasons .One of the reason is related to its energy level .The p-n junction of the silicon is responsible for the exchanging of ions inside the solar panel . A solar cells is made of p-type silicon (with extra “holes”)and n-type silicon (with extra electron) .The internal electric field was created by the presence of oppositely charged ions in depletion zone .when the electron from n-side layer to fill up the “hole” of p-side layer .The schematic of p-n junction is shown in Figure 2-7.

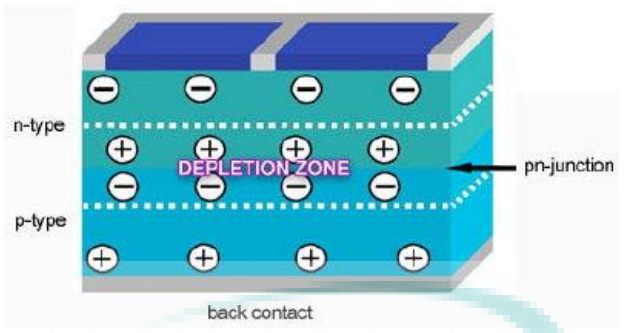


Figure 2-7 The schematic of P-N junction in solar cell

When the sunlight shine on the solar panel ,the electron are escaped and leave vacancies behind .The electric field will move electron to n-type layer and holes to p-type layer .If there is a conductor such as metallic wire between n type and p type layer ,there must be as exchange of ion and creating the flow of electricity .

Crystalline silicon is popular for its existence in PV industry .The reason is it is a semiconducting material which can control its conductivity through doping .Generally ,the crystalline silicon divided into several categories which is monocrystalline silicon ,multi-crystalline silicon ,and several other species of silicon which is not popular to be used in PV industry .In the country which is rich of sunlight like Malaysia ,the rate of photon generation is actually increase rapidly due to the temperature heat up the solar panel and gradually reduces the band gap in crystalline silicon structure [9].Therefore ,the electron is able to move from valence band to conduction band to allow to allow the flow of electron in p-n junction .The Figure 2-8 shows the band gap of conductor (metal) ,semiconductor ,and insulator [10].

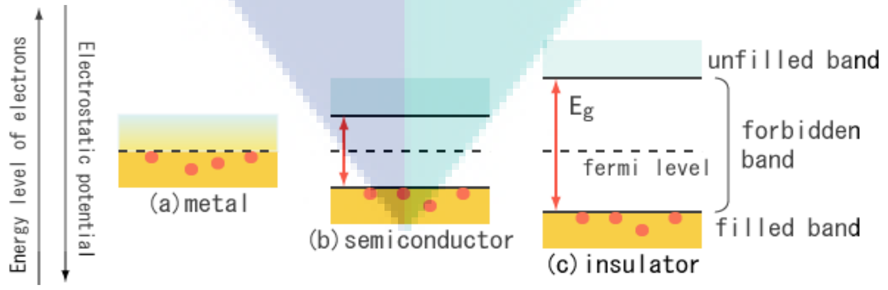


Figure 2-8 The band gap of conductor ,semi conductor and insulator

2.4 Hotspot in solar panel

Hotspot are areas of increasing temperature on the part of solar panel .They are the outcome of a confined decrease in efficiency which causing lower power output and degradation of material .They are several causes for hotspot to occur .One of them is cell mismatch where cells of varying current are connected in series .The other reasons is cell damage which is caused by defects in manufacture process .Other external factor such as shading ,dirt and water corrosion can be the causes of hotspot [11]. Reversed bias in solar cell is generally caused by shading or partial shading on solar cell .Shading is a problem for solar cell ,since it can reduce the power output to zero ,and probably caused microscopic damage to solar cell [12] .The reverse bias model of the solar cell is shown in Figure 2-9.

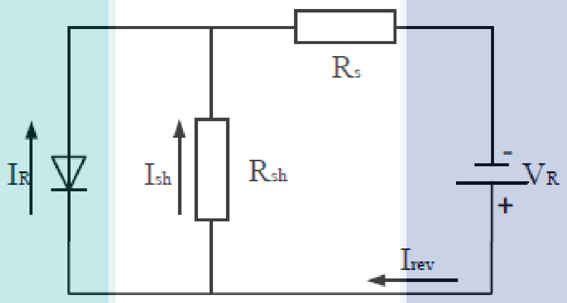


Figure 2-9 Reverse bias model of solar cell

Shading is a condition where there is a blockage between solar panel and sunlight .Shading could caused the mismatch of current and further lead to loss in power .The shade impact depends on the severity and area of the shade [13] .Shading has become a major challenge to deal with ,as shading is significant design factor affecting the performance of PV system .Shading a cell with a very high opacity causes it bypass diode to begin conducting when any current passes through it and keep supply the load ,but if the bypass diode is absent ,the cell will be reverse biased and possibly produce a damaging reverse breakdown voltage to causes hotspot failure .[14]

2.5 Infrared Thermography

Infrared thermography is a nondestructive testing technique without affecting the future usefulness of the equipment .By comparing to the other NDT techniques such as X-Ray and ultrasonic .Infrared thermography allows the detection of relatively shallow subsurface defects which ranged few millimetre in depth .Infrared thermography can monitor thermal behaviour of most of the electrical equipment .Although heat is not conclusive indication of all electrical problem .However ,heat produced by abnormally high resistance often precedes electrical failures[15] .The advantages of infrared thermography are providence of fast inspection rate ,safe from harmful radiation and their results is relatively easy to interpret .On the other hands ,the difficulties in infrared thermography are their capabilities to inspect only subsurface defects ,and only able to inspect a limited thickness of material under the surface [16]. In the application of Infrared thermography, the Infrared thermography can be done in two ways which are active infrared thermography and passive infrared thermography .The two approaches in infrared thermography is shown in Figure 2-10.

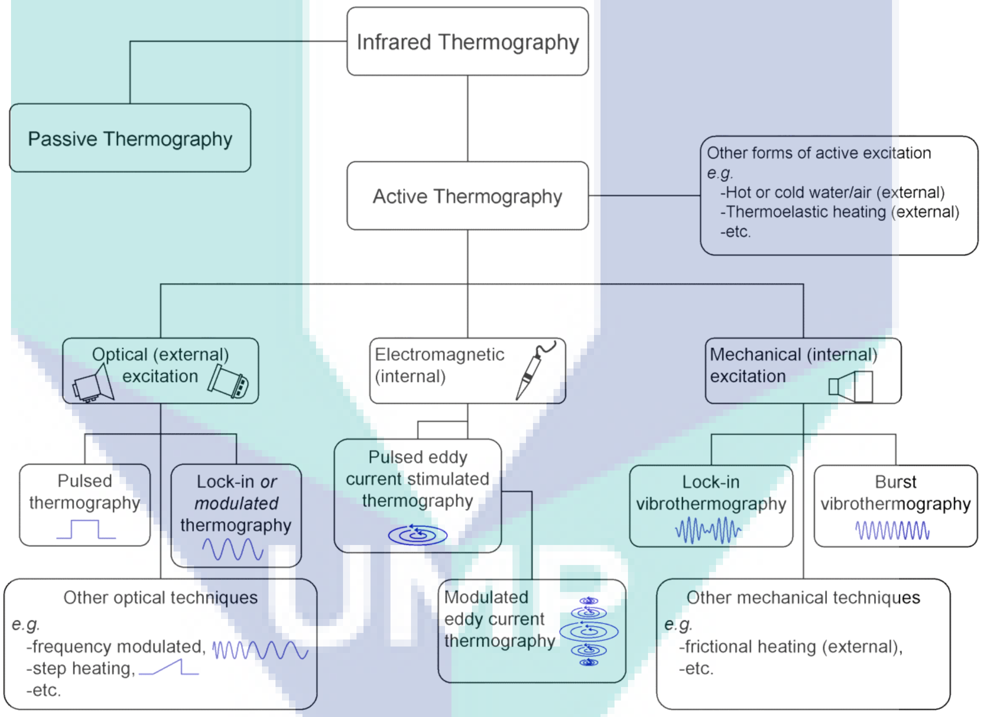

Infrared Thermography
Passive Thermography
Active Thermography
Other forms of active excitation
e.g.
-Hot or cold water/air (external)
-Thermoelastic heating (external)
-etc.
Optical (external) excitation
Electromagnetic (internal)
Mechanical (internal) excitation
Pulsed thermography
Lock-in or modulated thermography
Pulsed eddy current stimulated thermography
Lock-in vibrothermography
Burst vibrothermography
Other optical techniques
e.g.
-frequency modulated,
-step heating,
-etc.
Modulated eddy current thermography
Other mechanical techniques
e.g.
-frictional heating (external),
-etc.

Figure 2-10 Infrared thermography approaches

Overall ,active infrared thermography is approach in which energy source is required to produce a thermal contrast between the feature of interest and the background .(e.g : specimens with internal flaw .) .Meanwhile ,passive infrared thermography in the approach which the features of the interest is naturally higher or lower temperature than the background .(e.g. :surveillance of people on a scene).

Active infrared thermography are used to inspect the solar panel in term of the degradation in material in crystalline .For performing active infrared thermography on solar panel ,step heating by light source is essential and it is one kind of excitation used .The heating was done in two configuration which is backside and front side heating to check the overall delamination in solar panel .If the solar panel experienced delamination in certain part ,the thermal image will show a high temperature because of low thermal conductivity [3] Meanwhile there is no evidence that passive infrared thermography being used in solar panel inspection ,because passive thermography inspection is used to inspect on building structure on moisture evaluation and heat insulation .

Amongst existing condition monitoring techniques and nondestructive testing methods today, infrared thermography is considered a promising tool for fast and reliable fault detection. Infrared thermography is a method which detects infrared energy from the solar module, convert it into temperature and displays image of temperature distribution. It is a reliable method to observe the performance of solar cell whether it is operating in a normal condition or not. Infrared thermography uses mid wave (MWIR, from ~3 to 5μm) or long-wave (LWIR, from ~7 to 14μm) infrared sensors to obtain thermal images or thermogram of objects under inspection. Based on Planck’s black body radiation law, all objects emit infrared radiation proportional to their temperatures.

Solar hotspot is a major problem for this alternative energy as it can give a significant effect on the solar module performance. The cells exposed to the shading condition adversely affects the performance of the PV module due to the increase in the power loss according to Pragyanshree Samantaray (2016) [6]. Until now, there is no a complete and effective method to prevent this significant problem toward solar energy. Hence, this problem should be monitored and replaced quickly in order to prevent the module performance from getting even worse. Thus, methods to effectively detect solar hotspot is important to keep the solar module performance at its best.

Numerous study had been attempted to explain about how to detect or cluster solar hotspot problem on solar module through infrared thermography. K-means colour quantization is a useful method to quantize certain area of the solar module. It is easy to identify part of solar module if the module is classified into region based on their temperature. April M. Salazar (2016) research use K-means colour quantization and CIE L*a*b* colour space to quantize local heating spot. Through infrared thermography and k-means clustering, local heating areas were isolated [7]. Moreover, according to Erees Queen B. Macabebe (2016), the K-Means clustering produced the quantized image represented by the contours while DBSCAN resulted to the segmented image isolating the hotspot area as one of its clusters. Ultimately, the area of the hotspot can be determined and, with more data sets, may be correlated to the drop in the efficiency of the solar PV module [8].

Other visual method to determine solar hotspot is through in-line thermography has been conducted by Stefan Schenk (2014), using uncool bolometric camera and take continuous series of image, first image as reference. According to him, this method is fast because it analyses the solar module in the interval of 70ms. The images during that interval is observe and the parameter coefficient of the module is analyzed for references [9].

Visual inspection method also can be done to observe the overall performance of solar module such as checking physical defects on the surface of solar module. Projects by E. Lorenzo (2012) had done three different methods altogether, visual inspection, infrared inspection and electrical inspection. He checks physical defects and followed by infrared thermography by comparing temperature difference. Then, electrical inspection is done by connecting ‘T’ connectors at the output wires. Voltage losses is compared due to the hotspot. It is proposed to reject any module exhibiting hotspots whose corresponding voltage losses (in relation to a non-defective module being part of the same string), within the PV system in normal operation, exceeds the allowable peak power losses fixed at standard warranties. This is also applicable to PV modules with defective bypass diodes, regardless the derived hotspot temperature [10].

In conclusion, all of the method are related on how to detect hotspot through visual images and analysing parameters such as voltages. However, there are still no method exist in order to prevent or distinguish solar hotspot permanently but only a way to minimize the effect. Hence, this research is performed for getting to identify solar hotspot automatically. Thus, user can easily know where is the hotspot occur on solar panel and finding a solution to overcome that particular cell from being affected by the problem.

Table 1: Summary of hotspot detection methods

Method	Detail	Results	Comments
K-means clustering and CIE L*a*b color space (April M. Salazar, and Erees Queen B. Macabebe,2016)	Use infrared camera to capture and display temperature with reflective temperature of 10°C. Several images taken at several time intervals to eliminate effect of glare or temporary shading factors. Measure temperature at four points of module. The image is processed with K-means to cluster certain area that is abnormal to each other.	Defect(crack) caused that region to dissipate more power. Hence, produced colors corresponding to higher temperatures. Under reverse-bias, external current is fed to the PV module and the solar cells begin to emit light in the infrared range	Through infrared thermography and k means clustering, local heating areas were isolated. Relative percent area affected by the hotspot as well as the temperature were obtained using the Hotspot Detection algorithm developed.
Image Segmentation Using K-Means Color Quantization and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Genevieve C. Ngo and Erees Queen B. Macabebe, 2016)	Infrared camera and infrared sensor is used. Images is processed with K-means color quantization to quantize the captured images into discreet number of colors. Then, the images are send to DBSCAN algorithm to segment the images.	Pre-processing includes the K-Means algorithm. In this algorithm, the number of clusters was set to k = 15. Data normalization is applied to image converting the 0 - 255 pixel values into 0 – 1.0	The K-Means clustering produced the quantized image represented by the contours.
In-line thermography	Uncool bolometric camera takes continuous series of	Temperature of solar cell is both calculated and	This method is fast because analyzing the solar module in

(Klaus Ramspeck, Stefan Schenk, Denny Duphorn, Axel Metz, Michael Meixner,2014)	image as references (before apply user defined voltage bias). IV testing is done with interval of 70ms. Voltage is set and image is analyzed during reversing biasing. Reverse current and voltage, temperature changes during biasing is observed.	measured. The expected value from theoretical calculation is compared with physical measurements. Difference more than 5°C counted as hotspot.	the interval of 70ms. The images during that interval is observe and the parameter coefficient of the module is analyzed for references.
IR inspection on hotspots and derived acceptance/rejection criteria (R. Moretón , E. Lorenzo, L. Narvarte,2012)	Visual inspection of PV module and check physical defects if visible. Infrared inspection using infrared camera and calculate temperature difference using formula based on standardize irradiance. At night, electroluminescence is used. 'T' connectors are insert at module output wires to compare the operating voltage losses due to hotspot.	Micro-cracks cause current drift and a corresponding heat that leads to the burning of the metallization fingers and bubbles at the rear of the modules. Infrared inspection did not reflect the total hotspot, but only the hotspots observed some months after the substitution. $\Delta T^*HS > 30$ °C.	Electroluminescence method can misjudge hotspot during night time.
Analysis of hotspot formation in solar cells	Solar cell is divided into three regions hotspot center, outside hotspot and non-hotspot areas. Infrared images are	Developed localized increase in surface temperature after 1200s, at which point the	Results presented in this work reveal a direct correlation between areas of high impurity contaminants and

	taken at 300s interval. AES analysis is done. Auger spectrum using ion gun to sputter the surface. Three AES spectrum is obtained according to the three regions mentioned. SEM image of specimen (scanning electron microscopy). EDX (Energy-dispersive X-ray spectroscopy) .	maximum temperature profile reached 80 °C. SEM image at region temperature 80°C clearly indicates focal point of heating at the hotspot. It is observed that from regions away from the hotspot, the impurity concentration starts to decrease as can be seen by the reduction in oxygen concentration and other transition elements	hotspot heating in different samples of mc-Si. Transition elements and oxygen must be minimized so as to increase the life expectancy of these devices and overall improve systems reliability
Thermal Image Analysis and a Simulation Model	Thermal camera (portable thermal imager) employed for the performed measurements featured a 320 x 240 uncooled focal plane array. Microbolometer detector. Temperature range of 233 to 1273 K(–40 to 1000 °C), temperature resolution of 0.1 K (0.1 °C)image update rate of 8.5 fps, spectral range of 8 to 14 µm long.	Physical defects cause current drift and a corresponding heat that leads to the burning of the metallization fingers and bubbles at the rear of the modules	Black-and-white reproduction, rainbow color maps can be confusing because of their lack of perceptual ordering and can be misleadingly interpreted through the introduction of non-data-dependent gradients

CHAPTER 3

METHODOLOGY

3.1 Introduction

The thermal images of the solar panel will be captured by using Adafruit AMG8833 IR Thermal Camera Breakout. Then, infrared images will be displayed on the SPFD5408 2.4 inch TFT LCD display. Thermal Camera Breakout will be connected through the microcontroller which is the Arduino Mega 2560 to undergo algorithm that detect hotspot automatically based on temperature difference based on the environment. An extra sensor which is the LM35 Temperature Sensor will also be connected alongside the thermal camera breakout to display the ambient temperature of the surrounding area.

3.2 Hardware Platform

Figure 3-1 Block diagram of overall process of the system design shows the system design that consists of the thermal camera breakout and temperature sensor as the main input. Both of this hardware played an important part in this project as it is needed for capturing and measurement. The type of sensor used for visual imaging is an Adafruit AMG8833 IR Thermal Camera Breakout which display 8x8 array of infrared thermal sensors. When connected to microcontroller, it will return an array of 64 individual infrared temperature readings over I2C. It's like those other thermal cameras, but compact and simple enough for easy integration. User can interface with the system design by capture thermal images of solar panel in order to perform the hotspot detector algorithm. Then, the thermal images are send to the microcontroller to be analysed and k-means colour quantization is performed on the images. This process will quantize the captured image into discreet number of colours. Hence, local heating area on the solar module can be localized because of the dominant colour from the thermal images.

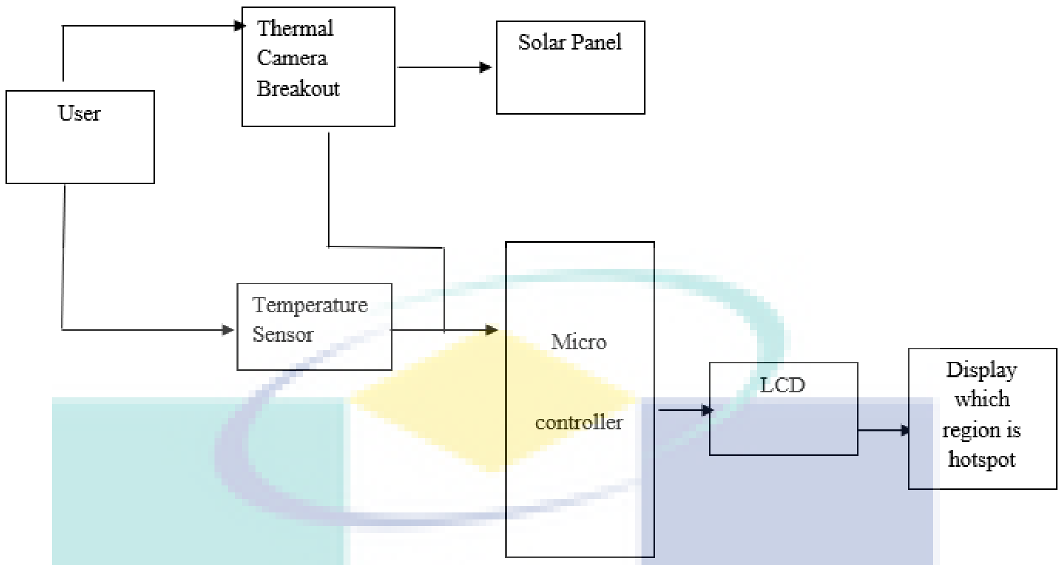


Figure 3-1 Block diagram of overall process of the system design

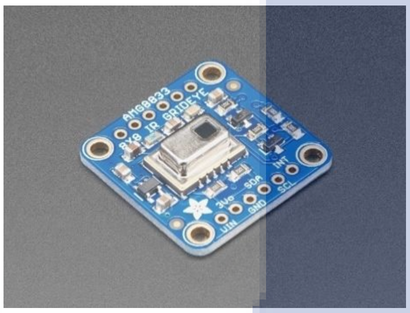


Figure 3-2 Adafruit AMG8833 IR Thermal Camera Breakout that is used as thermal visualize.

Other major parts that are important in this system design is temperature sensor. This sensor is responsible to measure the temperature of the surrounding area. The DS18B20-PAR is a 1 wire parasite-power digital thermometer in 3 pin TO-92 package as shown in Figure 3-2 Adafruit AMG8833 IR Thermal Camera Breakout that is used as thermal visualize.below. It provides 9 to 12bit centigrade temperature measurements and has an alarm function with non-volatile user programmable upper and lower trigger points. The DS18B20-PAR does not need an external power supply because it derives power directly from the data line (parasite power). It has an operating temperature range of -55°C to +100°C and is accurate to ±0.5°C. Both of the components mentioned above will function simultaneously when the user turns on the system design. Thermal images from the camera and ambient temperature from the sensor will be send to microcontroller to be analysed and display on the Liquid Crystal Display (LCD).

Figure 3-3 LM35 Temperature Sensor IC used to measure ambient temperature.

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 microcontroller. This microcontroller board come along with 54 digital input or output pins. 14 of them can be used as PWM outputs. 16 of them can be used as analog inputs. 4 of them are UARTs (hardware serial ports). It also come with a 16 MHz crystal oscillator, a power jack, USB connection, reset button and an ICSP header. To support the microcontroller, it contains everything. It can be simply connected with a USB cable or it can be powered with a AC-to-DC adapter or battery to make it alive. This microcontroller or microcontroller board is compatible with many shields that are designed for Arduino. The operating voltage of this microcontroller board is 5 volts. The input voltage that is recommended is 7-12V. The input voltage limit is 6-20V. The DC Current per I/O pins is 40 mA. The DC current for 3.3V pin is 50 mA. The flash memory is about 256 kB of which 8 kB used by the bootloader. The clock speed of this microcontroller is 16 MHz which is quite high for this affordable board This can be counted as the brain of this system design as it is important for the solar hotspot automatic detector. All of the data collected will be received by this microcontroller to be analysed and calculated.

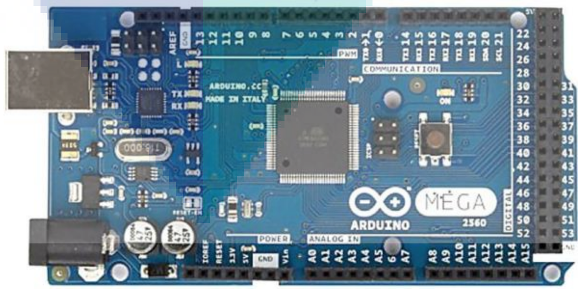


Figure 3-4 Arduino Mega 2560 that is used as the microcontroller to carry out visual analyzing and system design algorithm

3.3 System Design

The system design flow starting with capturing thermal images of the solar panel to be analysed. Firstly, K-means colour quantization will be held to clustered certain region that is abnormal due to its colour (red indicating high temperature). It is easy to identify which cell is affected when the area of solar module is clustered into certain region. Then, the value of temperature of each array from the thermal camera is taken into account. Each array temperature is compared with the ambient temperature in order to get the temperature difference of the solar cell. Solar panels are tested at **25 °C** and thus solar panel temperature will generally range between **15 °C** and **35 °C** during which solar cells will produce at maximum efficiency. However, solar panels can get as hot as **65 °C** at which point solar cell efficiency will be hindered. Install factors like how close the panels are installed to the roof can impact the typical heat of your solar system. This algorithm will analyse temperature using two different condition which is comparing it with average temperature of solar cell and comparing with the range of normal operating range of temperature during normal condition.

$$\Delta T^{*}HS1 = Tcell(max) - Taverage \tag{3.1}$$

$$\Delta T^{*}HS2 = Tcell(max) - Tambient(max) \tag{3.2}$$

$$Tambient(max) = Tambient + 30\ °C,\ \text{(range between 25 to 60°C on normal condition)} \tag{3.3}$$

where *Tcell(max)* is the highest temperature of solar cell captured by the thermal array sensor and *Taverage* is the average temperature of the solar cell. Temperature difference, *ΔT*HS1* that is more than 20 °C can be counted as solar hotspot and will display 'FAULT' as the condition. If the *ΔT*HS1* is within the range of 5 °C and 20 °C, the display will set the condition as 'WARNING'. If the *ΔT*HS1* exceed 20 °C, 'FAULT' will be displayed as the condition on the lcd display. However, the algorithm will set as 'FAULT' when the second condition is a positive value. This is because the highest temperature exceed the normal range of solar operating temperature. Highest temperature will compared with the highest range from the ambient temperature This algorithm will calculate and identify which region can be classified as hotspot. The process or flowchart of this system design is shown in Figure 3-9. Next, that identified region will be display through the LCD alongside with the thermal images. User can view the thermal images of solar panel including with label that said it is the region of hotspot.

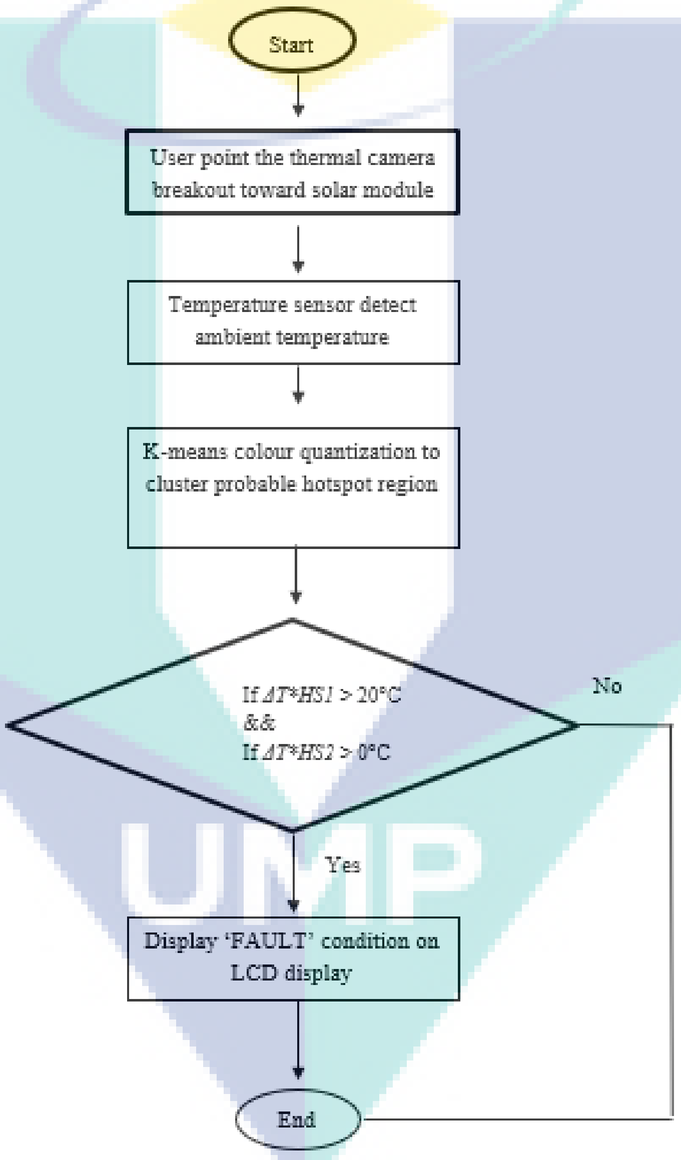

Start
User point the thermal camera breakout toward solar module
Temperature sensor detect ambient temperature
K-means colour quantization to cluster probable hotspot region
If $\Delta T^*HS1 > 20°C$
&&
If $\Delta T^*HS2 > 0°C$
No
Yes
Display 'FAULT' condition on LCD display
End

Figure 3-5 Flowchart of the process to detect hotspot phenomenon on the solar module

3.4 System Hardware

The system is composed into a few hardware part ,while the embedded microcontroller that used in this project is Arduino UNO ,and the sensor that used to collect temperature data is OMRON D6T-44L thermal sensor .The input and output of the system is shown in Figure 3-6.

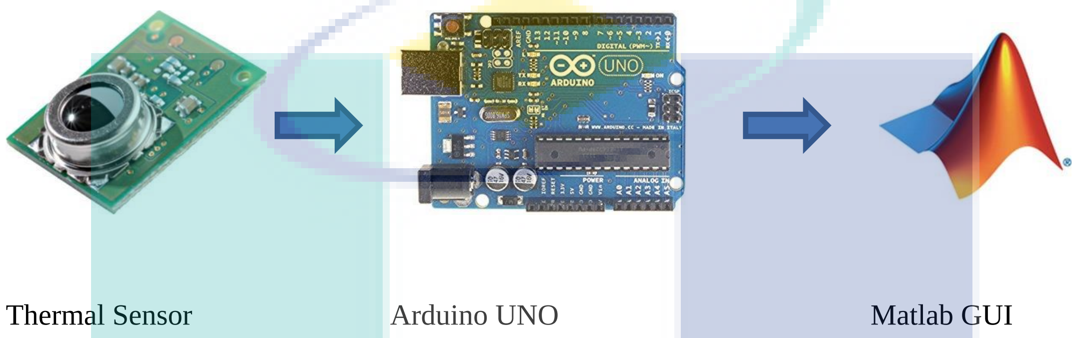


Figure 3-6 Block Diagram

The thermal sensor is powered by DC source of 5V and connected to Arduino UNO .It communicate through I2C protocol .The Arduino UNO will receiving data from thermal sensor and the received data will be analyse in Matlab .The hotspot will be detected if any of the array temperature is higher than the calculated solar cell temperature .The solar cell temperature calculation is shown in the equation below .

$$T_{cell} = T_{air} + \frac{NOCT - 20}{800} S \qquad (3.4)$$

where ,

T_{air} = ambient temperature in °C

T_{cell} = cell temperature in °C

$NOCT$ = normal operating cell temperature in °C

S = insolation level in W/m2

In this part ,all the components will put together by putting the pin of sensor to the specific pin of Arduino Uno .The overall concept of the set up is shown in Figure 3-7 below .

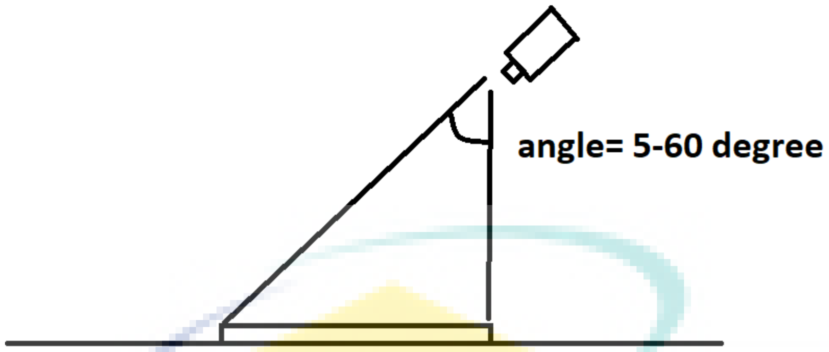


Figure 3-7 The setup of the device to solar panel

3.4.1 Microcontroller

Arduino UNO is a microcontroller board based on the ATmega328 .It has 14 digital input/output pins (of which 5 can be used for PWM outputs) ,6 analog inputs, a 16 MHz ceramic resonator ,a USB connection , a power jack ,and ISCP header ,and a reset button .It contains everything to support the microcontroller ; simply connect it to a computer with a USB cable or power it with a AC-DC adapter or battery to get start .

The UNO differs from all preceding boards in that it does not use the FTDI USB-to-serial converter .Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode .Meanwhile , Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

The summary of the Arduino UNO technical datasheet is given in Table 3.2. Arduino UNO can be powered up by external supply and it is automatically selected depend on what power source connected . The power pins are given as below :

- VIN - The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V - This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board.
- 3V3-A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND - Ground pins.

The ATmega328 has 32 KB with 0.5 KB used for the bootloader. It also has 2 KB of SRAM and 1 KB of EEPROM which can be read and written with the EEPROM library.

Table 3.2 Techinical Specification of Arduino UNO.

Microcontroller	ATmega328
Operating Voltage	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limit)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

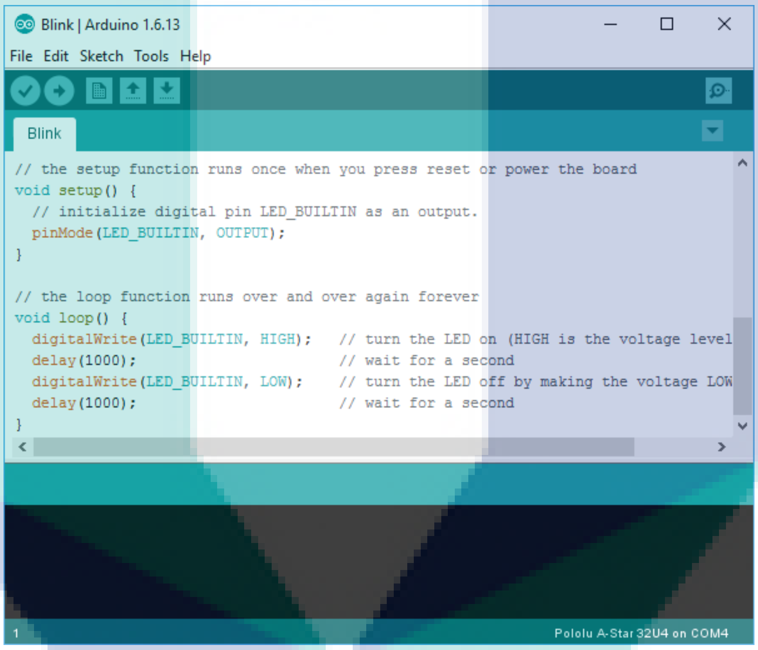
Arduino UNO has 14 pins as its input and output ,using pinMode(),digitalWrite and digitalRead() functions .The board operate at 5 V and can provide and receive a maximum of 40 mA .It also has a internal pull-up resistor of 20 – 50 kΩ which is disconnected by default .Other than that ,Arduino UNO consists of the specialised function :

- Serial: 0 (RX) and serial :1 (TX) .Used to receive (RX) and transmit (TX) TTL serial data .These pins are connected to the specific pins of the ATmega8U2 USB to TTL serial chip .
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the function of analogWrite() .
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED (pin 13) : Built in LED device which will switch on when the pin 13 is in HIGH value ,switch off when the value is LOW .
- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.
- AREF. Reference voltage for the analog inputs which is used with analogReference().
- Reset. Used to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Arduino UNO has several facilities for communicating with a computer ,another Arduino ,or other microcontroller .The ATmega328 basically provide UART TTL (5V) serial communication through RX pin and TX pin .While ATmega16U2 on the board channelling the serial communication through USB and appear a virtual com port in computer software .16U2 is using a standard USB COM driver so no external driver is used .In Arduino software ,they provide serial monitor where user can read the data that being sent by Arduino board ,TX and RX will flash alternatively to indicate the data is being transmitted and receive via the USB-to-serial chip and USB connection to the computer . A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I^{2}C (TWI) and SPI communication and also includes a Wire library to simplify usage of the I^{2}C bus .

Arduino UNO can be programmed by using Arduino IDE .The user are required to choose the correct com port and board specification .ATmega328 in Arduino UNO comes with a preburned bootloader that allows user to upload the sketch to arduino without any external programmer as it communicates by using the original STK500 protocol .ISCP is also another option for programming the microcontroller .

Arduino UNO is designed that allows it to reset by software running on a connected computer .A 100 nanofarad capacitor is used to connect the hardware flow control lines (DTR) of the ATmega8U2/16U2 to the reset line ATmega328 .When the reset line is pulled low ,the Arduino chip is reset and Arduino software have a shorter timeout for uploading the code by simply pressing upload button .The figure of Arduino software is shown in Figure 3-8.



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Figure 3-8 Arduino IDE

3.4.2 D6T-44l-06 Omron Thermal Sensor

The thermal sensor is non contact sensor which is used for sensing surface temperature of an object .The thermal used for this project is OMRON D6T-44L-06 thermal sensor which sized 14mm*18mm .The main component of this thermal sensor are silicon lens ,thermopile sensor chips ,analog circuit and logic circuit for converting to digital value which shows us the temperature data .The particular thermal sensor consists of 16 arrays of data .It is a supersensitive sensor that is useful for detecting the presence of hotspot .

The sensor works at 5V DC which coordinated with the power of the Arduino UNO, to interface with Arduino UNO we have to know the I^2C address of the D6T-44L sensor which is 0x0A and the command to get the temperature data which is 0x4C. The Arduino UNO is then request for the temperature of the captured temperature from the address 0x0A and by sending the 0x4C the data will return into an array. The array size is 16bit-width, signed, 10 times the estimation of degree centigrade. Which intend to get the real estimation of the temperature, the value need to divided by 10, the explanation behind it is to transmit float information simpler through convert it to integer and convert back back to float after data transmission .

The WireExt.h library was utilized instead of Wire.h to let the Arduino UNO to communicate with the sensor. The reason is the Wire.h only support until 32 bytes of data as appeared in Figure 3-9.Therefore ,it is not enough for D6T-44L-06 thermal sensor to sends 35 bytes of data through Wire.h .

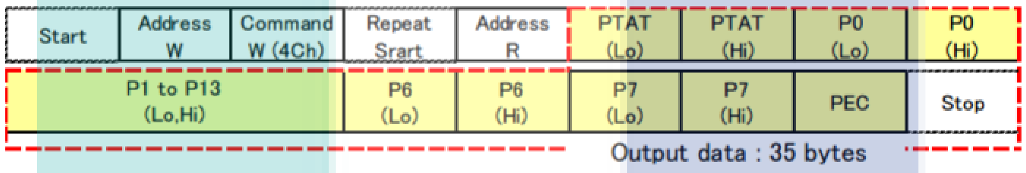


Figure 3-9 Number of Bytes of Output Data from D6T-44L-06 Thermal Sensor .

The equation for calculating the temperature for MEM sensor is summing up the background and object temperature and divided by the area as shown in the formula below .

Calculation for near object:

$$Tx(near) = \frac{(\mathrm{Tb1} + \mathrm{To1})}{\mathrm{Area\ 1}} \tag{3.5}$$

Calculation for far object

$$Tx(far) = \frac{(\mathrm{Tb2} + \mathrm{To2})}{\mathrm{Area\ 2}} \tag{3.6}$$

The main communication that MEM sensor used with Arduino is through I^2C protocols as it consist 4 of the pins which are Vcc,GND ,SDA and SCL .The SDA and SCL are responsible controlling the data line and clock line of I^2C data .Therefore the connection will be set up as below :

- Vcc :5V
- GND: GND
- SDA:Pin 4
- SCL:Pin 5

The pinout of MEM sensor is shown in Figure 3-10.

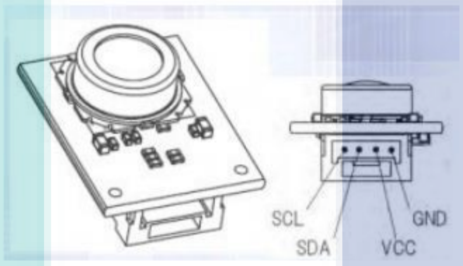


Figure 3-10 Pinout of MEM sensor

3.5 System software

The software used for this project consists of 2 software which are Arduino IDE and Matlab .The Arduino is mainly act as a platform for building an electronic project which can include a big variety of electronic component .While ,Matlab is most likely a software that used to analyse data ,develop algorithm and create model and application .

3.5.1 Arduino IDE

Arduino is an open-source stage utilized for building electronics projects. Arduino comprises of both a physical programmable circuit board (regularly alluded to as a microcontroller) and a bit of programming, or IDE (Integrated Development Environment) that keeps running on your PC, used to compose and transfer computer code to the physical board. The Arduino stage has turned out to be very well known with individuals simply beginning with hardware, and all things considered. Not at all like most past programmable circuit sheets, the Arduino does not require a different bit of equipment (called a developer) so as to stack new code onto the board – you can essentially utilize a USB link. Furthermore, the Arduino IDE utilizes a streamlined variant of C++, making it simpler to figure out how to program. At long last, Arduino gives a standard frame factor that breaks out the elements of the small scale controller into a more available bundle.

3.5.2 Matlab

MATLAB is an elite language for specialized registering. It coordinates calculation, representation, and programming in a simple to-utilize condition where issues and arrangements are communicated in natural scientific documentation. Run of the mill utilizes include:

- Math and calculation
- Calculation advancement
- Demonstrating, recreation, and prototyping
- Information examination, investigation, and representation
- Logical and designing illustrations
- Application advancement, including Graphical User Interface building

MATLAB is an intelligent framework whose fundamental information component is a array that does not require dimensioning. This enables you to tackle numerous specialized figuring issues, particularly those with grid and vector details, in a small amount of the time it would take to compose a program in a scalar noninteractive dialect, for example, C or Fortran. The name MATLAB stands for matrix laboratory. MATLAB was initially composed to give simple access to framework programming created by the LINPACK and EISPACK ventures, which together speak to the best in class in programming for network calculation.

MATLAB has developed over a time of years with contribution from numerous clients. In college conditions, it is the standard instructional device for basic and propelled courses in arithmetic, building, and science. In industry, MATLAB is the instrument of decision for high-efficiency research, advancement, and examination.

MATLAB highlights a group of use particular arrangements called tool compartments. Important to most clients of MATLAB, tool stash enable you to learn and apply particular innovation. Tool compartments are far reaching accumulations of MATLAB capacities (M-documents) that stretch out the MATLAB condition to take care of specific classes of issues. Regions in which tool kits are accessible incorporate flag handling, control frameworks, neural systems, fluffy rationale, wavelets, reproduction, and numerous others.

3.6 Proposed System

Our project proposed work is to integrate the thermal sensor with microcontroller to perform the data collection from the sensor , following by integrate the Arduino component with Matlab GUI to perform serial reading and perform analysis with suitable function such as solar cell temperature calculation .The overall flow chart of hotspot detection system is shown in Figure 3-11.

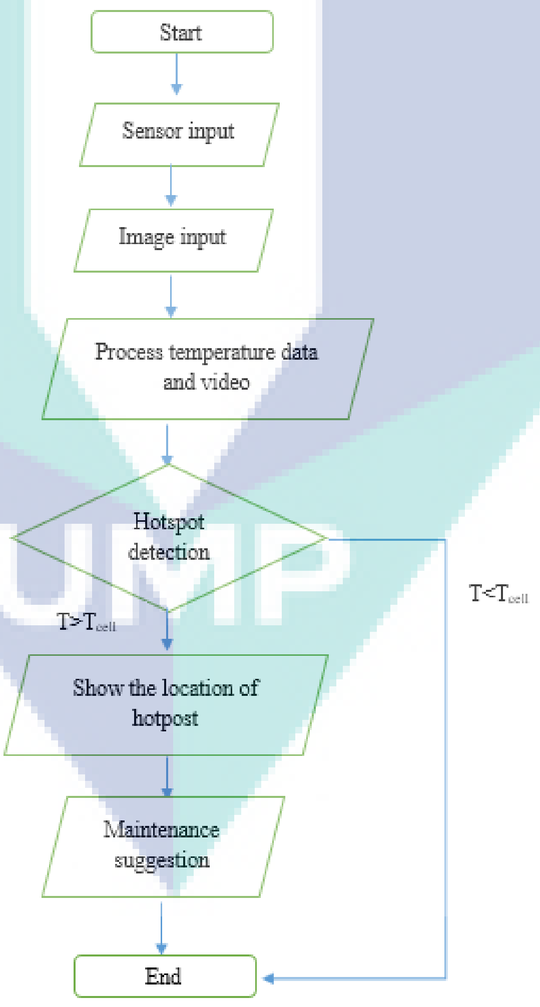


Figure 3-11 Flow chart of hotspot detection system

3.7 Irradiance Measurement

This part discusses two aspects, the software development part and the hardware development part. The hardware development of this project discussed about assembling of solar reference cell, sensors, microcontroller and the display unit. Software development discussed about the measurement of solar irradiance by interpreting a series of data collected on solar cell and developing the equations and programming to calculate the irradiance. Figure 3.12 shows the flowchart of the system.

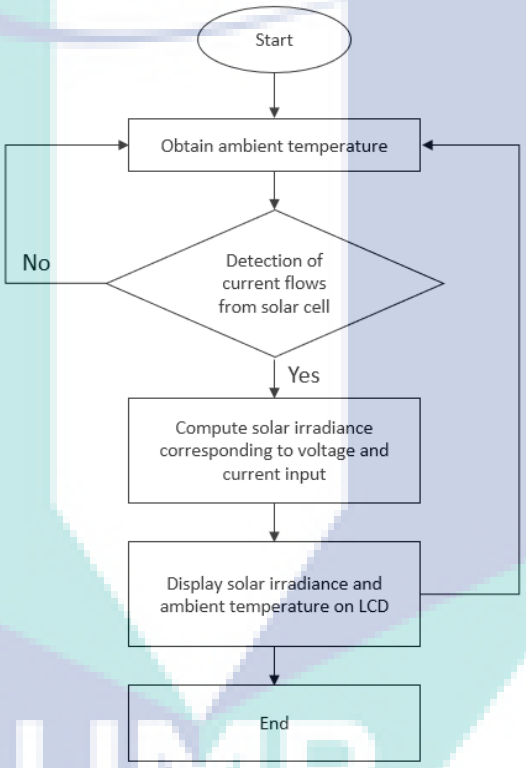

Start
Obtain ambient temperature
Detection of current flows from solar cell
No
Yes
Compute solar irradiance corresponding to voltage and current input
Display solar irradiance and ambient temperature on LCD
End

Figure 3.12: Flowchart of the System

The solar radiation monitoring device starts with the detection of ambient temperature. The ambient temperature will be detected by the temperature sensor LM35 and the data collected will be sent to the microcontroller Arduino UNO. Next, the microcontroller will detect if there is any current flow in the solar cell. There will only be current flows in the solar cell if there is solar radiation reaches the solar cell. If there is not current flows, indicating there is no solar radiation to be measured, or the solar radiation is too weak, leading to extremely low current flow in the solar cell to be detected. The microcontroller will continue detect current flow until there is a valid current value to be obtained. When there is current flow detected in the solar cell, the microcontroller works to interpret the current readings, and compute the corresponding solar radiation. This computation for solar radiation values corresponding to current flows in solar cell requires initial data collection for the solar cell with simple circuit which will be further explained in result and discussion. The solar radiation and ambient temperature readings will be displayed by the microcontroller on the LCD on the handheld measuring device.

Figure 3.13 shows the block diagram of the system. The solar cell works as the solar reference cell will be connected a current sensor module ACS712 to detect the current flows in the solar cell whenever there is solar radiation reaching on the solar cell surface. The current sensor module will be connected to the microcontroller Arduino UNO to interpret the current readings on solar cell. At the same time, a temperature sensor LM35 will collect the ambient temperature data and send the readings to microcontroller. Arduino UNO which works as the microcontroller of the system will interpret and compute solar radiation readings corresponding to the current flows in solar cell. The solar radiation values and ambient temperature will be displayed on the LCD of the handheld device. The data can be sent to the developed app on smartphone for user to have a clearer monitoring purpose.

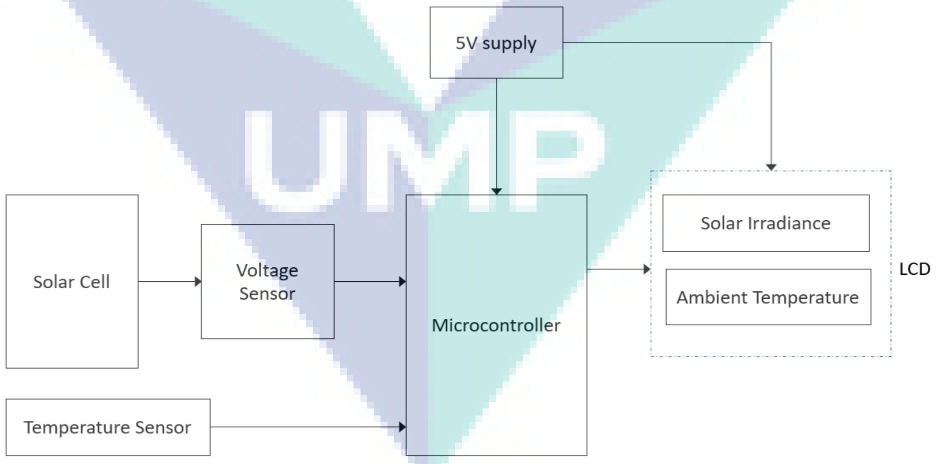


Figure 3.13: System Block Diagram

The microcontroller, Arduino Uno is connected to a temperature sensor, LM35, voltage sensor module B25 which is connected to a mini solar cell and a 30 Ω resistor, and finally connected to a 2x16 yellow backlight LCD for displaying purpose. The connection of each pin of the temperature sensor LM35 to the Arduino Uno is showed in the table below.

Table 3.3: Connection of Each Pin between Temperature Sensor LM35 and Arduino Uno

Temperature Sensor LM35	Arduino Uno
V_{out}	A0
+V5	5V
GND	GND

Temperature sensor LM35 is an affordable and easy-to-use temperature sensor which is suitable for Arduino programming to detect ambient temperature. In this system, LM35 is chose to act as the temperature sensor to detect the ambient temperature.

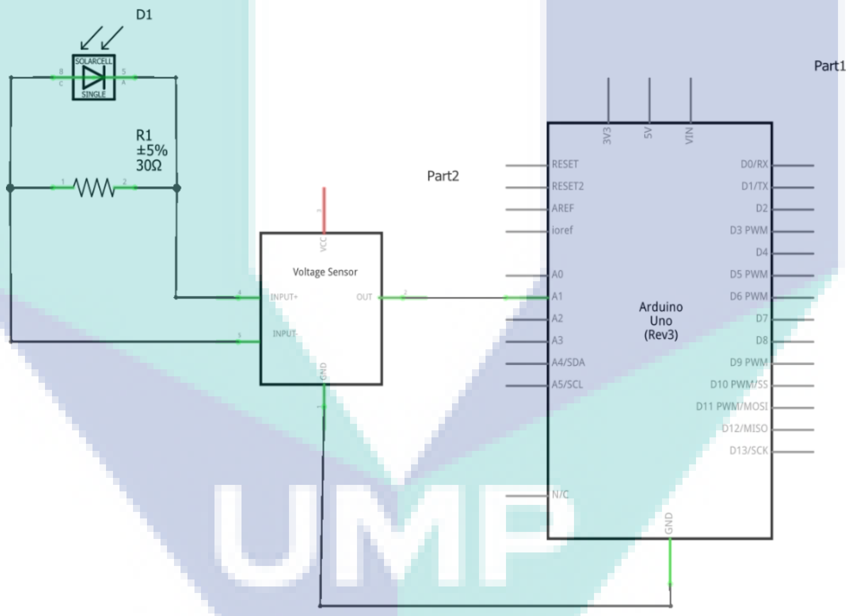


Figure 3.14: The Schematic Diagram of Connection between Solar Cell, 30 Ω Resistor, Voltage Sensor and Arduino Uno

The connection of each pin of the solar cell, 30 Ω Resistor and the Arduino Uno is showed in the Table 3.3 below.

Table 3.4: Connection of Each Pin between Solar Cell, 30 Ω Resistor, Voltage Sensor and Arduino Uno

Solar Cell	**30 Ω Resistor**
Positive pin (DC)	Either pin
Negative pin (DA)	Either pin
Voltage Sensor	**Solar Cell**
Input (+)	Positive pin (DC)
Input (-)	Negative pin (DA)
Voltage Sensor	**Arduino Uno**
Out	A1
GND	GND
VCC	-

Solar cell is connected to a small value resistor, a 30 Ω resistor in a series circuit. The voltage sensor module is used to measure the voltage drop across the 30 Ω resistor by connecting to each pin of the resistor or solar cell. The output of the voltage sensor module is measured and calculated by Arduino Uno by connecting to the A1 pin of Arduino Uno. Figure 3.15 shows the schematic diagram and connection between LCD and Arduino Uno. Table 3.4 shows the connection of each pin of the LCD and Potentiometer to the Arduino Uno.

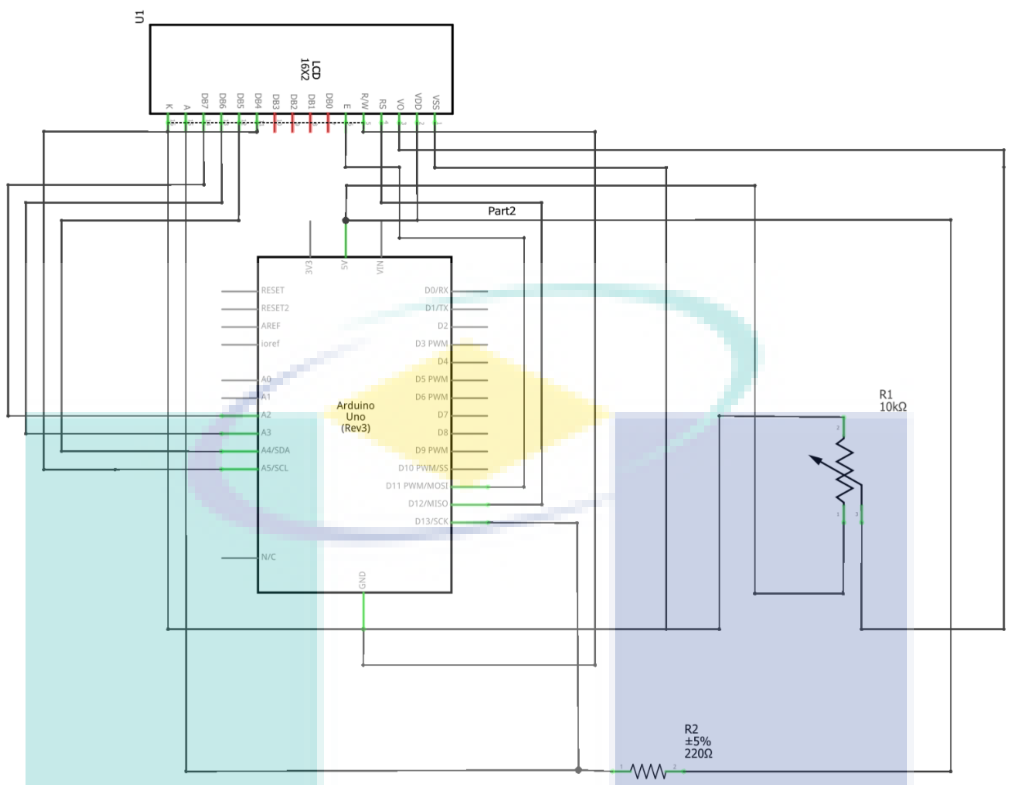


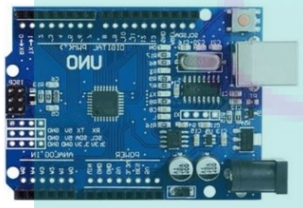
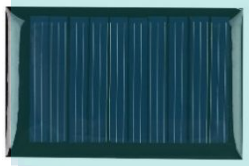
Figure 3.15: The Schematic Diagram of Connection between LCD and Arduino Uno

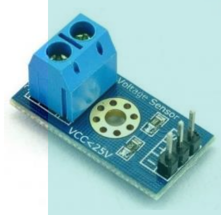
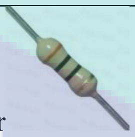
Table 3.5: Connection of Each Pin between LCD, Potentiometer and Arduino Uno

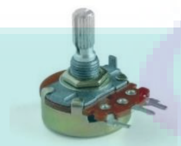
2x16 Yellow Backlight CD	**Arduino Uno**
K	GND
A	D13/SCK
DB7	A2
DB6	A3
DB5	A4/SDA
E	D11 PWM/ MOSI
R/W	GND
RS	D12/MISO
2x16 Yellow Backlight CD	**Potentiometer**
VO	Signal
2x16 Yellow Backlight CD	**Arduino Uno**
VDD	5V
VSS	GND
Potentiometer	**Arduino Uno**
+5V	5V
GND	GND

A 2x16 Yellow Backlight LCD is used to display the solar irradiance and ambient temperature of the system. A 10k Ω potentiometer and 220 Ω resistor is also used for the LCD circuit. Table 3.5 below shows the descriptions of components that will be used for the solar radiation monitoring system.

Table 3.6: List of Components

Components	**Descriptions**	**Functions**
Arduino Uno 	A microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.	Process data obtained from temperature sensor and current sensor. Display solar irradiance and ambient temperature on LCD. Transmit data to app.
Solar Cell 	A mini silicon solar cell with the size of 5.2cm x 2.9cm x 0.3cm. Able to produce output of 5V, 0.15W and 30mA.	Convert solar energy to electrical energy through photoelectric effect.
LCD (16x2) Yellow Backlight Module	This is a basic text-based 16 characters by 2 lines LCD with white character on blue backlight. It utilizes the extremely common HD44780 parallel interface chipset. The LCD has a size of 8.03 x 3.61 x 1.0 cm.	Display the solar irradiance and ambient temperature.

LM35 LM35DZ NS TO-92 Temperature Sensor	A low-cost temperature sensor that able to read ambient temperature with the operating condition of 4 to 30V. Rated for full −55˚ to +150˚C range.	Detect the ambient temperature.
B25 Voltage Sensor Module 	This module is based on principle of resistive voltage divider design. Arduino analog input voltages up to 5V, the voltage detection module input voltage not greater than 5Vx5=25V or 3.3Vx5=16.5V. Arduino AVR chips have 10-bit AD, so this module simulates a resolution of 0.00489V (5V/1023), so the minimum voltage of input voltage detection module is 0.00489V x 5 = 0.02445V.	Detect voltage drop across the resistor connected in series to the solar cell. The voltage drop is used to calculate the current flows in the solar cell circuit.
30 Ω Resistor 	A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. Carbon film resistors are a fixed form type resistor. • Resistance: 30 Ω • Film: Carbon Film • Tolerance: 5% • Power rating: 1/4Watt	Connect in series to the solar cell to obtain the voltage drop across it.
220 Ω Resistor	A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. Carbon film	Used in LCD connection to prevent burn.

	resistors are a fixed form type resistor. • Resistance: 220 Ω • Film: Carbon Film • Tolerance: 5% Power rating: 1/4Watt	
10kΩ Potentiometer 	A three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat. • Value: 10K ohm • Type: Linear potential meter • Power: 1/4 Watts	Used in LCD connection to control the contrast of wording.

The system proposed should include Arduino programming in determining the relationship between solar irradiance, current and voltage in the circuit. Figure 3.16 illustrates the series circuit of a solar cell and a 30Ω resistor.

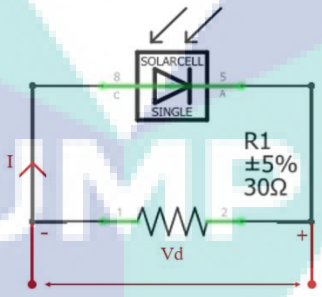


Figure 3.16: Series Circuit of Solar Cell and 30 Ω Resistor

Arduino programming with C or C++ language will be used to interpret data obtained from temperature sensor and voltage sensor module connected to solar cell. The voltage sensor module is used to determine the voltage drop, Vd across the resistor connected in series to the solar cell. By applying the Ohm's law, V = IR, the current flows in the circuit can be determined. The current, I flows in the circuit as showed in Figure 3.10 at any instant can be calculated, which

$$I = \frac{Vd}{30}. \quad (3.7)$$

The current data is used to calculate the real time solar irradiance based on the relationship developed between solar irradiance, voltage and current output from solar cell in the data collection phase. At data collection phase, the current and voltage readings from the voltage sensor module will be compared to a multimeter reading to evaluate the accuracy of the voltage sensor module. Solar irradiance meter, multimeter and voltage sensor module are the equipment and components that used to collect the parameters which are the solar irradiance, current and voltage. Table 3.6 shows the list of equipment and components used with the parameters to be measured.

Table 3.7: List of Equipment/Components and the Parameters to be Measured

Equipment/ Component	**Parameters to be measured**
Seaward Solar Irradiance Meter	Solar Irradiance [W/m^2]
Multimeter	Current [mA]
Voltage Sensor Module	Voltage [V]

Figure 3.17 shows the data collection by using voltage sensor module, multimeter and solar irradiance meter. The recorded parameters will be used to plot graph to obtain the relationship between the solar irradiance, current and voltage in Microsoft Excel. A suitable line of equation is drawn, and the equation is obtained for the programming purpose in Arduino. Table 3.7 shows the recorded readings for solar irradiance, current and voltage.

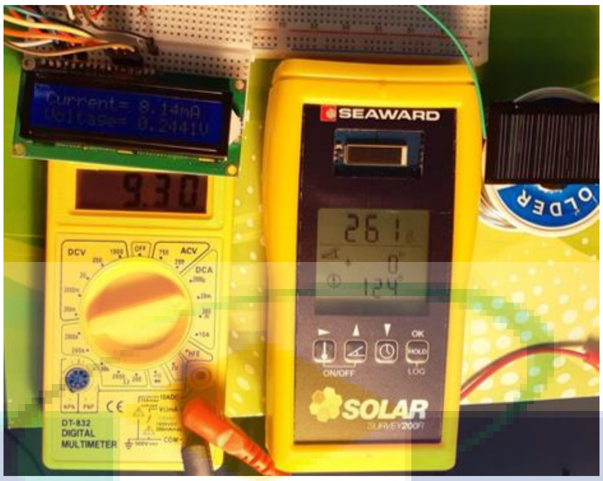


Figure 3.17: Data Collection by Using Voltage Sensor Module, Multimeter and Solar Irradiance Meter

Table 3.8: Recorded Solar Irradiance, Current and Voltage Readings

No	**Solar Irradiance Meter**	**Multimeter**	**Voltage Sensor**		$\lvert Approx - Exact \rvert$ **(mA)**	**Error (%)**
	Solar Irradiance [W/m^2]	**Current [mA]**	**Current [mA]**	**Voltage [V]**		
1	239	7.79	7.32	0.2197	0.47	6.03
2	237	7.79	6.51	0.1953	1.28	16.43
3	239	8.18	7.32	0.2197	0.86	10.51
4	244	8.19	6.51	0.1953	1.68	20.51
5	245	8.22	7.32	0.2197	0.9	10.95
6	257	8.64	8.14	0.2441	0.5	5.79
7	242	8.63	8.14	0.2441	0.49	5.68
8	252	8.65	8.14	0.2441	0.51	5.90
9	267	8.97	8.14	0.2441	0.83	9.25
10	255	8.98	8.14	0.2441	0.84	9.35
11	261	9.3	8.14	0.2441	1.16	12.47
12	275	9.3	8.14	0.2441	1.16	12.47
13	271	9.3	8.14	0.2441	1.16	12.47
14	264	9.38	8.14	0.2441	1.24	13.22
15	274	9.38	8.14	0.2441	1.24	13.22
16	265	9.35	8.95	0.2686	0.4	4.28
17	271	9.74	8.95	0.2686	0.79	8.11
18	299	10.06	8.95	0.2686	1.11	11.03
19	292	10.08	8.95	0.2686	1.13	11.21
20	313	10.55	9.77	0.293	0.78	7.39
21	314	10.54	9.77	0.293	0.77	7.31
22	312	10.66	9.77	0.293	0.89	8.35

23	318	11.22	9.77	0.293	1.45	12.92
24	321	11.22	9.77	0.293	1.45	12.92
25	326	11.21	9.77	0.293	1.44	12.85
26	325	11.19	10.58	0.3174	0.61	5.45
27	336	11.47	10.58	0.3174	0.89	7.76
28	336	11.51	10.58	0.3174	0.93	8.08
29	340	11.51	10.58	0.3174	0.93	8.08
30	321	11.49	10.58	0.3174	0.91	7.92

Table 3.7 above shows part of the recorded data. The readings are recorded in such a table in Microsoft Excel and then a graph is plot. The error between the actual reading (multimeter reading) and the approximate reading (sensor reading) is calculated based on the following equation:

$$Error\ (\%) = \frac{|\ Approx - Actual\ |}{Actual}\ x\ 100\% \qquad (3.8)$$

Equation (3.8) is crucial for knowing how accurate the sensor is reading compared to the actual reading. Multiple calibration process should be done to achieve better accuracy of system. Ideal accuracy of system is having a percentage of error less than 10%. Figure 3.18 illustrates the graph of solar irradiance vs current plotted based on Table 3.6

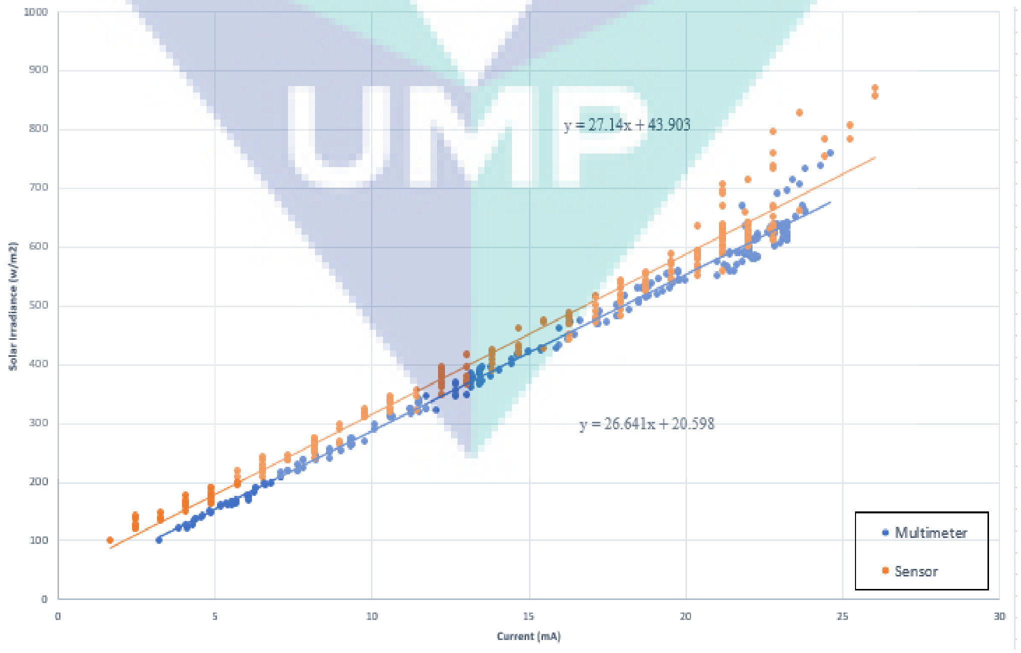


Figure 3.18: Graph of Solar Irradiance vs Current

The relationship between the solar irradiance and the current can be interpreted by an equation based on the collected data. The equation will be used in Arduino programming to obtain the respective solar irradiance at any instant where the current or voltage in the circuit is measured by using the voltage sensor. The calculated solar irradiance by Arduino Uno will be displayed on LCD along with the ambient temperature data obtained from temperature sensor.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

Results gathered from data testing is analysed and compared with a more advance and expensive infrared camera. This is to get the efficiency of the design infrared camera in order for it to captured accurate temperature and perform the developed algorithm efficiently.

4.2 Results

All of these preliminary results is gathered to test the simple algorithm which is compared the cell temperature with ambient temperature. This chapter is discussing about the experiment that has been done on using the thermal camera and the data acquired is analysed. Thermal data is recorded based on the experimental setup which is 3 meter above the solar panel and 60 ° setup angle . The data is compared between the thermal image temperature and the temperature calculation in android application . The analysed result will then use to determine the location of hotspot ,and verifying whether the elevated temperature spot is a hotspot .

Thermal images of solar several solar module is taken under direct sunlight as show in Figure 4-6 and the temperature is displayed through the infrared camera. Moreover, the short circuit current and open circuit voltage is also measured using a multi-meter in order to analyzed the PV module performance under normal operating conditions.

$$\begin{aligned} \Delta T^*HS &= Tcell - Tamb \\ &= 41.3°C - 25°C \\ &= 16.3°C \end{aligned}$$

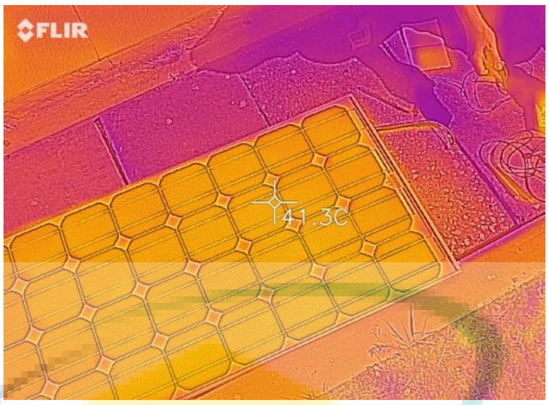


Figure 4-1 Temperature of solar cell of module 1.

It can be assumed that there no hotspot phenomenon on that solar module as it temperature difference does not exceed 30°C. Visual analyzing of the figure above can be seen that there is no significant difference in the solar cell temperature.

Table 4.1: Short circuit current and open circuit voltage reading

Short circuit current, Isc	3.45A
Open circuit voltage, Voc	21.0V

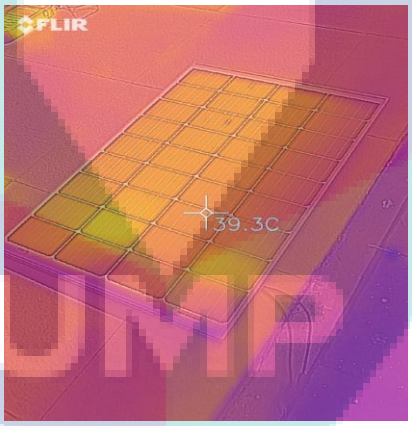


Figure 4-2 Data measured from the output of solar module

$$\begin{aligned} \Delta T^{*}HS &= T_{cell} - T_{amb} \\ &= 39.3°C - 25°C \\ &= 14.3°C \end{aligned}$$

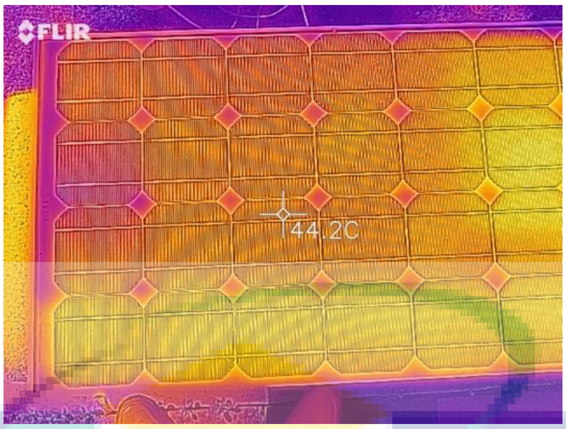


Figure 4-3 Temperature of solar cell of module 3.

$$\Delta T^{*}HS = T_{cell} - T_{amb}$$
$$= 44.2°C - 25°C$$
$$= 19.2°C$$

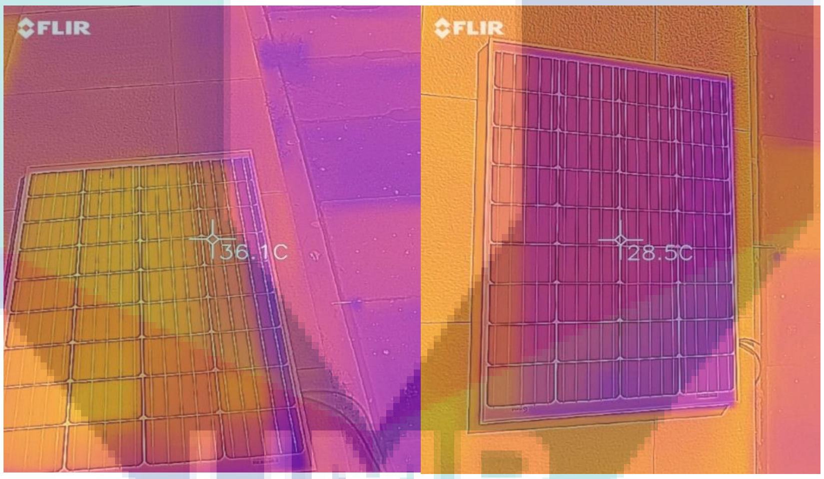


Figure 4-4 Temperature of other solar cell from the solar module.

$$\Delta T^{*}HS = T_{cell} - T_{amb}$$
$$= 36.1°C - 25°C$$
$$= 11.1°C$$

$$\Delta T^{*}HS = T_{cell} - T_{amb}$$
$$= 28.5°C - 25°C$$
$$= 3.5°C$$

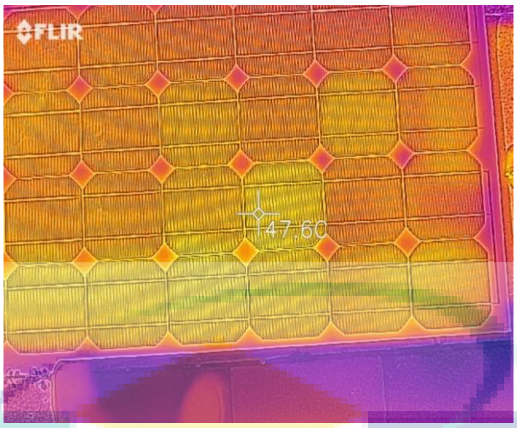


Figure 4-5 Solar cell with the highest temperature from solar module 1.

$\Delta T^*HS = T_{cell} - T_{amb}$
$= 47.6°C - 25°C$
$= 22.6°C$

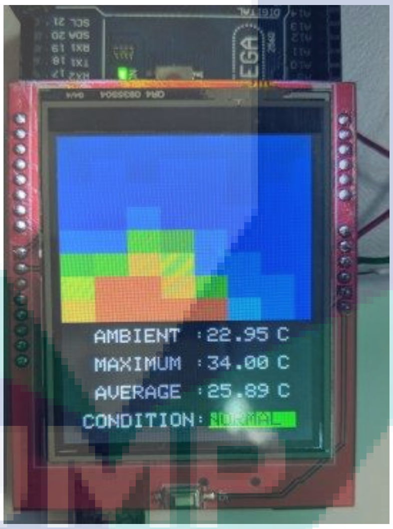


Fig 11. Module analyzer

4.3 System Integration

At the stage of system integration ,the hardware and software components of this system will be integrating together .Therefore ,the system integration framework should be used to illustrate the system description and operation as well as contributing a clear set of operations that are associated with hotspot detection system .This framework is aimed to reduce the complexity and time usage on detecting the hotspot on solar panel .The system integration is shown as Figure 4-6.

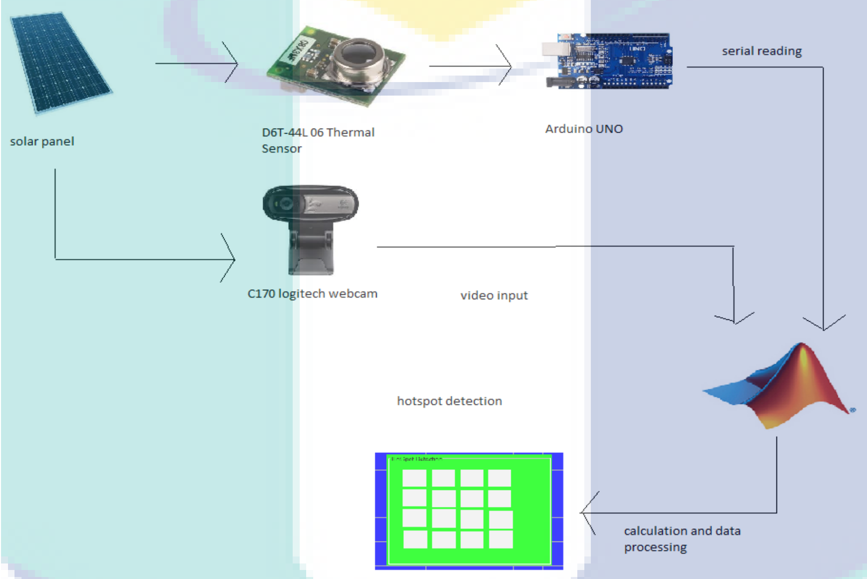


Figure 4-6 System integration

4.4 Hotspot detection

The experiment was carried out under 2 different times which is 11 am and 12 pm .The date of experiment was 10 May 2018 .The thermal data was recorded by using different equipment such as IR thermometer for detecting the surface temperature of solar panel and digital thermometer for detecting the ambient temperature .The set up of the experiment is shown in Figure 4-7.

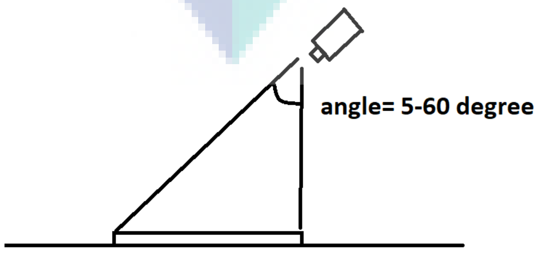
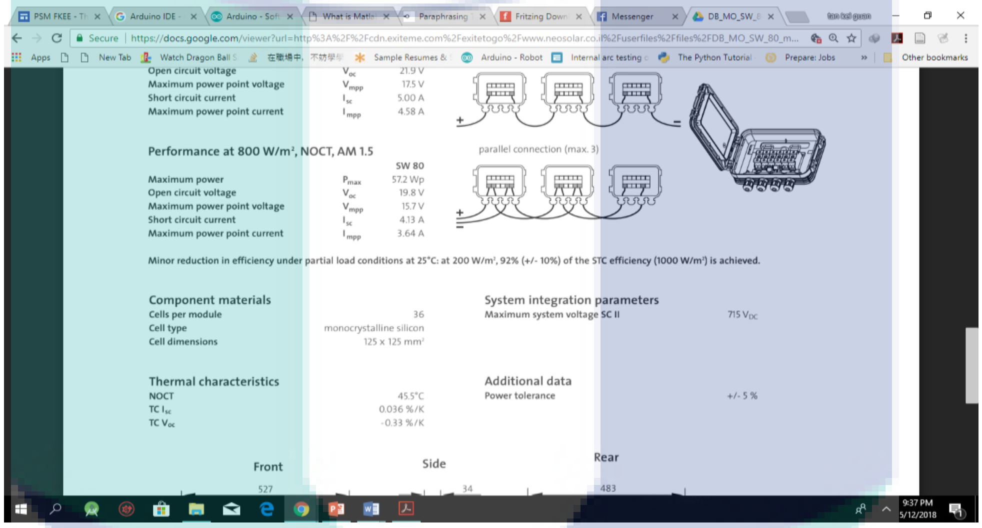


Figure 4-7 setup of the experiment

The insolation level from 11 am to 12 pm vary from 900 W/m2 to 950 W/m^2 and the data was recorded by solar power meter to use as a parameter for Matlab GUI .The second parameter which is important for cell temperature calculation is NOCT of solar panel .The NOCT of solar panel can be obtained from the datasheet behind the solar panel or in online document .In this project ,the model of solar panel is SolarWorld SW80 mono R5E ,the NOCT this solar panel is 45.5°C .as shown in Figure 4-8.



Open circuit voltage	V_{oc}	21.9 V
Maximum power point voltage	V_{mpp}	17.5 V
Short circuit current	I_{sc}	5.00 A
Maximum power point current	I_{mpp}	4.58 A

Performance at 800 W/m², NOCT, AM 1.5

		SW 80
Maximum power	P_{max}	57.2 Wp
Open circuit voltage	V_{oc}	19.8 V
Maximum power point voltage	V_{mpp}	15.7 V
Short circuit current	I_{sc}	4.13 A
Maximum power point current	I_{mpp}	3.64 A

Minor reduction in efficiency under partial load conditions at 25°C: at 200 W/m², 92% (+/- 10%) of the STC efficiency (1000 W/m²) is achieved.

Component materials

Cells per module	36
Cell type	monocrystalline silicon
Cell dimensions	125 x 125 mm²

System integration parameters

Maximum system voltage SC II	715 V_{DC}

Thermal characteristics

NOCT	45.5°C
TC I_{sc}	0.036 %/K
TC V_{oc}	-0.33 %/K

Additional data

Power tolerance	+/- 5 %

Figure 4-8 Datasheet of SW80 mono R5E

The experiment is start by identify the insolation level and the NOCT and input it into calculation panel ,then connect the Arduino UNO along with the thermal sensor to the computer to let them identify the comport that is available to connect .After everything is connected ,the GUI will run and updating the temperature array provided the device must be positioned to face the solar panel .The below section show the result of experiment on 2 different time which are 11 am and 12 pm .

<u>CASE 1 (11am)</u>

Ambient temperature (system) :38°C

Ambient temperature (digital thermometer):36.4°C

Insolation (solar power meter):927.5 W/m^2

Calculated temperature (system):67.56°C

Table 9 Temperature data (IR thermometer)

52.3	57.7	59.2	58.8
52.9	59.2	58.7	59.8
62.5	63.7	63.6	62.5
60	62.7	62.6	62.5
60.8	62.1	61.8	60
58.3	60.3	59.2	57.8
58.3	59.8	59.8	58.3
57	58.6	59	58.6
55.7	56.1	57.1	58.3

Table 10 Temperature data (system)

55.9	61.1	62.6	60.4
54.6	60.3	59.8	60.3
63.6	64.6	64.6	63
62.8	64	64.3	65
63.3	63.3	62.9	61.8
60.1	61.2	60.3	59.8
60.2	61	61	60.1
59.3	59.8	60.2	59.8
57.3	57.4	58.5	59.6

CASES2(12PM)

Ambient temperature (system) :38.3°C

Ambient temperature (digital thermometer):36.9°C

Insolation (solar power meter):940.7 W/m^2

Calculated temperature (system):68.28°C

Table 11 Temperature data (IR thermometer) 12PM

55.2	55.1	59.7	56.5
55.2	58	56.5	53.6
55.4	57.5	56.5	55
57.1	59.6	59.2	58.2
57.1	59.6	59.7	58.1
56.6	57.6	57.7	55.3
55.2	59.5	59.1	58.3
59	60.1	60.1	58.8
57.9	58.1	58.6	58.2

Table 12 Temperature data (system) 12PM

56.4	56.2	60.2	56.7
56.8	60	57.7	54
57	59.8	57.7	56
58.3	61.1	59.8	60.2
58.4	61.1	60.1	60.1
57.5	59.8	59.6	56.4
56.4	60.1	59.8	59.4
59.7	60.5	60.4	59.3
58.9	59.1	59.6	59.2

From the result above ,it showed that there is some difference in the data from both system and IR thermometer ,the system developed sensing and measuring the temperature at a higher reading than the IR thermometer ,although the distance of measurement for both instrument are the same .The difference ranged from 1°C to 2°C .The factor could be possibly caused by their manufactured sensitivity ,although they work with the same mechanism which is sensing the infrared radiation .The IR thermometer is used to measure the surface temperature by the assist of laser pointer .As for D6T-44L ,it already consist of 4x4 array ,so the temperature that measure by this particular thermal sensor is actually an area temperature of the surface .Therefore ,the reading are different from one another .

4.5 MATLAB GUI Interface

A graphical user interface is constructed by using Matlab for monitoring the temperature and show it in 4x4 arrays from solar panel .It has all feature which show the real time value ,such as ambient temperature , calculated cell temperature and a second 4x4 array which shows the location of hotspot if any array temperature is higher than the calculated cell temperature .The Matlab receive the data through serial reading and the data is process and divided into pixel .The pixel is able to show the colour based on the temperature level .The GUI of the system is shown in Figure 4.9 .

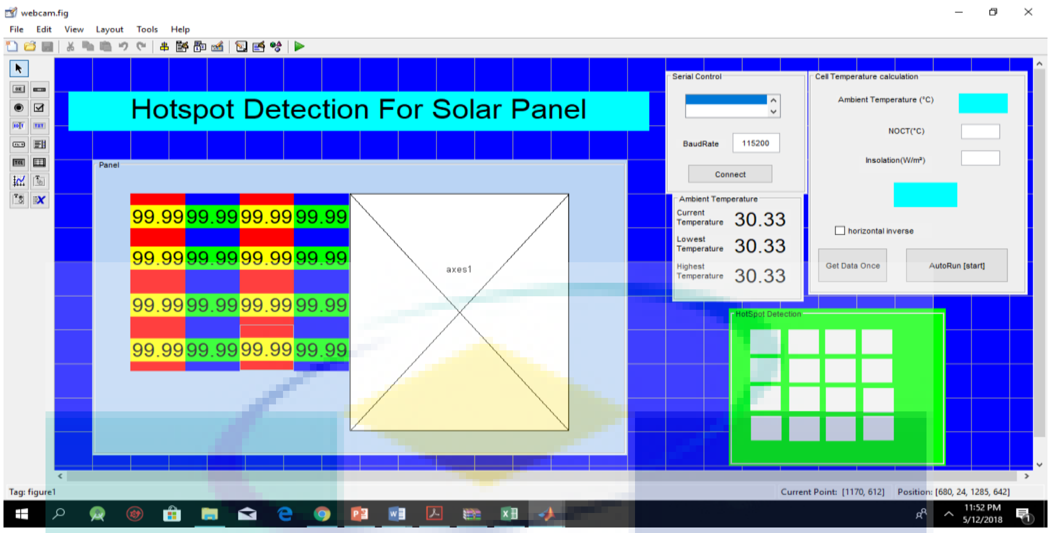


Figure 4-9 Graphical User Interface of Hotspot Detection System

4.6 Irradiance Measurement

This part describes the entire project progress outcome, from the data collection procedure at the very beginning to the analysing data and then design of system. Several times of data collection have been done to determine the relationship between current flows from solar cell and the voltage and current readings by recording these parameters corresponding to each solar irradiance level. These parameters have been recorded and the error was determined to determine the accuracy of sensor and for calibration purposes during the system test. The parameters recorded are plotted into graphs to illustrate the relationship between current, voltage and solar irradiance and an equation of relation is determined. This equation of relation is used for Arduino programming by input current to determine the respective solar irradiance. The input current is determined by using the Ohm's law, where the voltage reading from voltage sensor module is used to calculate the current. The system developed is always compared to the Seaward Solar Irradiance Meter to ensure the accuracy of the system is improved through calibration process.

4.6.1 Data Collection

At the initial state of data collection, an experiment is conducted to determine if there is any relationship between current flows from solar cell and the solar irradiance. The circuit connection and setup are as shown in Figure 4.10. The devices and components used are solar cell, 30Ω resistor, multimeter with crocodile clips and a

Seaward Solar Survey 200R. The current reading measured by multimeter and the solar irradiance reading measured by Seaward Solar Survey 200R is recorded in Table 4.5.

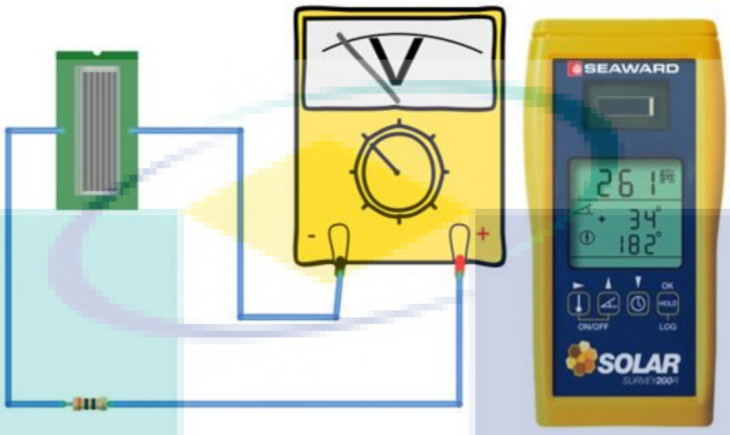


Figure 4.10: Measurement of Current and Solar Irradiance

Table 4.13: Table of Current Readings from Multimeter and Solar Irradiance Readings

Current, I(mA)	Seaward Solar Survey 200R Irradiance, Irr(W/m^2)
3.76	122
2.83	123
3.95	127
4.16	135
4.31	141
4.48	147
5.08	161
5.36	175
5.54	181
5.61	184
5.65	184
5.76	189
5.87	191
5.97	196
6.04	200
6.37	210
6.67	220
9.6	320
13.26	442
13.58	451
13.81	458
13.92	463
13.69	456
13.68	457
13.7	458
13.81	461
14.06	465
14.3	476
14.27	463
17.43	502
17.49	503
17.52	504
17.5	535
17.65	538
17.55	535
17.62	554
17.54	550
17.98	549
17.49	550
17.42	549
5.5	183
5.49	180
5.52	180
5.31	177
5.31	174
4.9	159
4.93	160
4.62	149
4.28	140
4.28	147
4.05	131
3.61	124
3.24	110
3.26	114
3.11	107
3.11	105

A graph of solar irradiance vs current output from solar cell has been plotted in Figure 4.11. From the graph, the relationship between solar irradiance and current output from solar cell is said to be a linear relationship. As the solar irradiance increases, the output current from solar cell also increases.

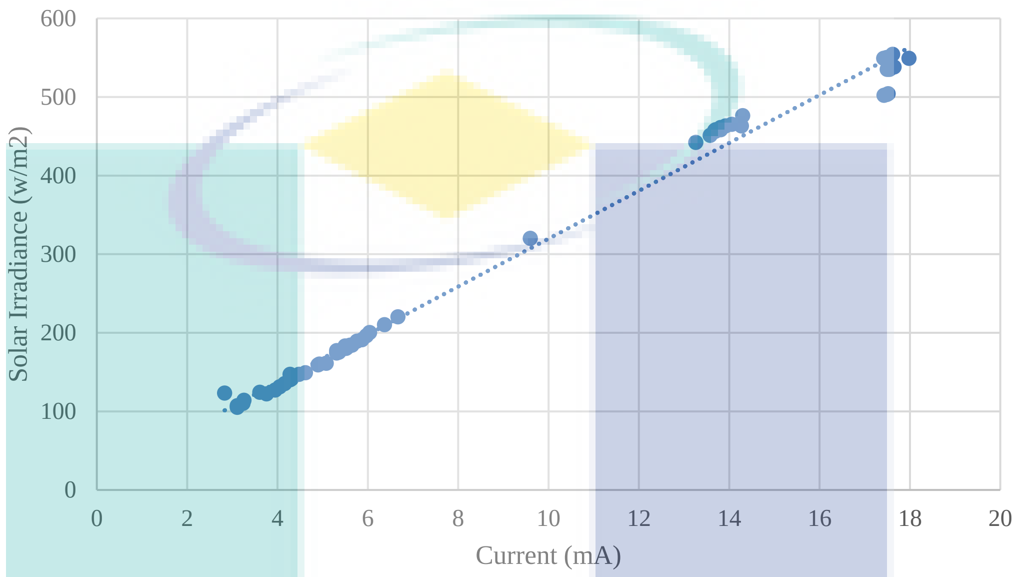


Figure 4.11: Graph of Solar Irradiance vs Current Output of Solar Cell

The data collection process also involves determining a suitable value for the resistor which must be connected in series to the solar cell. A 100kΩ is used to repeat the experiment and the same parameters are recorded in Table 4.6.

Table 4.14: Table of Current Readings and Actual Solar Irradiance Readings

Current, I(uA)	**Seaward Solar Survey 200R Irradiance, Irr(W/m^2)**
53.9551	0
53.9551	0
54.1992	0
54.4434	100
54.4434	104
54.6875	107
54.1992	113
54.9316	126

Current, I(uA)	**Seaward Solar Survey 200R Irradiance, Irr(W/m^2)**
60.7910	486
60.7910	501
61.0352	503
60.7910	505
60.5469	514
60.5469	514
60.7910	517
60.5469	519

54.9316	128	61.0352	521
55.6641	128	60.5469	530
55.6641	139	60.5469	531
55.9082	141	60.7910	534
56.3965	154	60.5469	537
56.6406	155	61.0352	538
56.1523	163	60.5468	547
57.1289	170	60.7910	549
56.1523	170	60.5469	550
57.1289	171	60.7910	556
56.8848	172	60.7910	556
55.9082	178	60.7910	563
56.8848	180	60.5469	565
56.6406	193	60.7910	566
56.8848	200	60.7910	572
56.6406	200	60.5469	572
56.8848	200	60.5469	578
57.1289	208	60.7910	583
57.6172	212	60.7910	586
57.3730	215	60.5469	594
57.3730	221	60.5469	594
57.3730	227	60.7910	594
57.8613	229	60.7910	601
57.8613	238	60.7910	606
57.8613	241	60.5469	613
58.1055	243	61.0352	619
57.6172	249	60.7910	624
58.1055	250	60.5469	632
58.1055	262	60.7910	632
57.8613	266	60.7910	632
58.5937	266	60.5469	635
58.5937	272	60.7910	644

58.8379	293
58.8379	306
59.3262	318
58.3262	319
59.5703	321
59.5703	345
59.8415	356
59.8145	362
60.0586	368
60.0586	369
59.8145	375
59.8145	378
59.8145	381
60.0586	391
60.0586	397
60.3027	400
59.8145	401
60.0586	417
60.0586	422
60.3027	443
60.5469	443
60.3027	454
60.3027	454
60.5469	464
60.7910	482
61.0352	486
61.0352	648
60.7910	667
60.7910	671
59.5703	678
60.7910	679
61.0352	688
61.0352	694
61.0352	694
61.0352	700
61.2793	705
61.7910	710
61.0352	714
61.0352	715
61.0352	718
61.0352	718
61.0352	719
61.0352	720
60.5468	724
60.5469	724
60.5690	724
60.7910	725
60.3027	725
60.0586	725
60.7910	727
60.7910	728

In Figure 4.12, the graph illustrated shows inconsistent current readings at different irradiance level, unlike for the 30Ω resistor circuit. Before trying 30Ω resistor, a 1Ω and 5Ω resistors are tried, but the voltage sensor is not detecting any voltage drop across the small value resistors. The voltage sensor only showing output when the solar irradiance is high, about 500W/m^2 and above. Thus, the system design will be using the 30Ω resistor. The following experiments will be using 30Ω resistor as well.

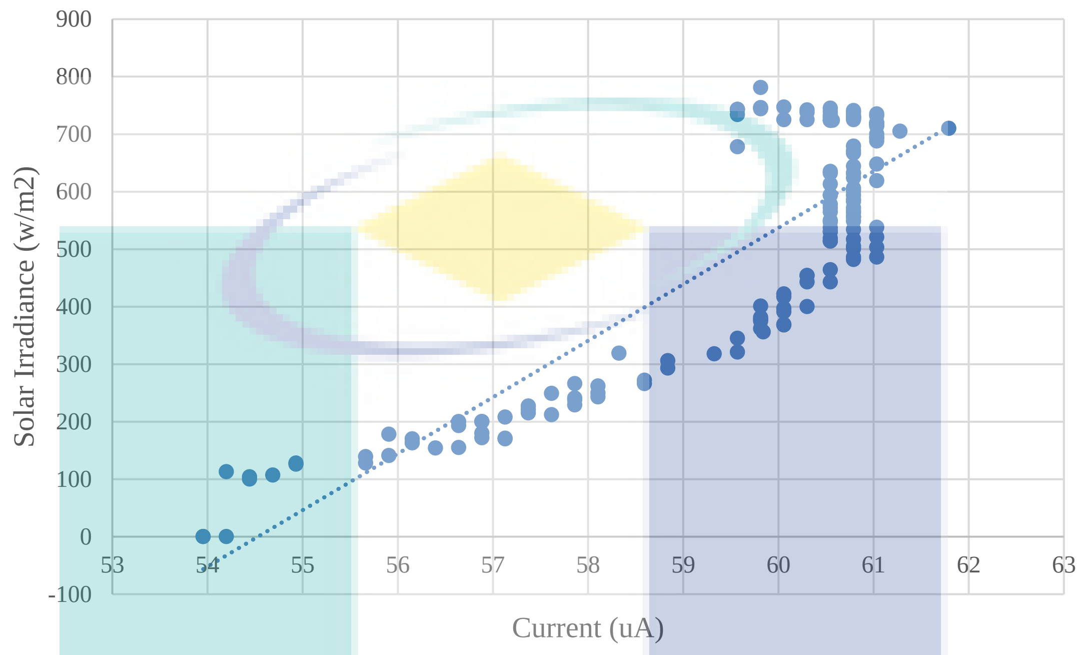


Figure 4.12: Graph of Solar Irradiance vs Current Output of Solar Cell

A more complete, first data collection is done and recorded in table to obtain the equation of relation and calculating the percentage of error between the sensor reading and the multimeter reading to attain a better accuracy of system. The accuracy of system is to be improved through calibration process.

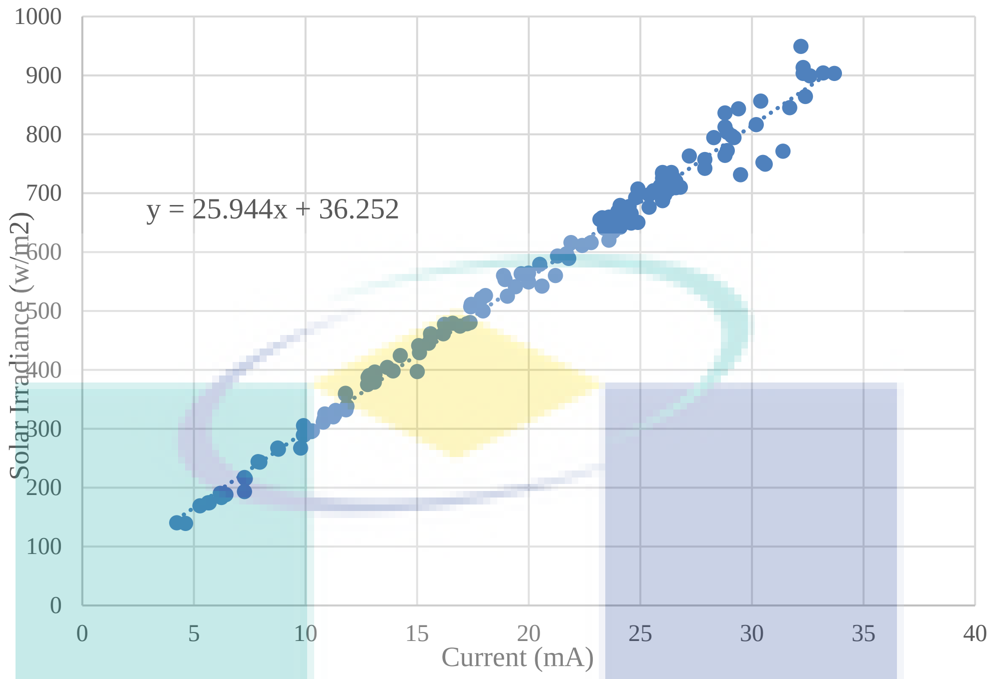


Figure 4.13: Graph of Solar Irradiance vs Current Output of Solar Cell

Figure 4.4 illustrates the relationship of solar irradiance and current output from solar cell in a consistent linear line of graph, with minimal outliers. The relation can be representation by the equation (4.1):

$$y = 25.944x + 36.252 \tag{4.1}$$

which y is the solar irradiance [w/m^2] and x is the current output from solar cell [mA]. The experiment above is repeated as second data collection on another day to ensure the results are consistent, with the same components and devices. The results are recorded in table. Figure 4.14 illustrates a graph plotted by using data in table.

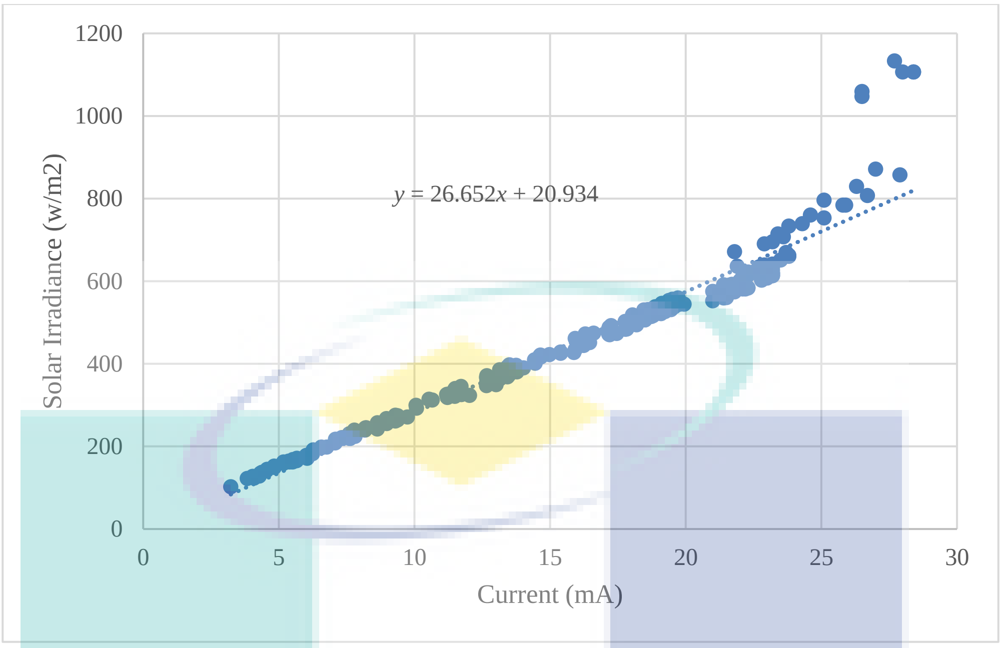


Figure 4.14: Graph of Solar Irradiance vs Current Output of Solar Cell

The repeated experiment verified that the relation between solar irradiance and current output from solar cell is linear. Thus, the system will be using the equation (4.2) to compute the solar irradiance.

$$y = 26.652x + 20.934 \tag{4.2}$$

4.6.2 Calibration and Accuracy of System

Equation (4.2) is applied to the system designed and a lot of data is taken again to calculate the percentage of error between sensor reading and multimeter reading, followed by calibration process. The data before calibration is recorded in table (refer to Appendix C). Figure 4.15 and Figure 4.16 shows a photo of measurement taken from two systems, the system developed and Seaward Solar Survey 200R device.

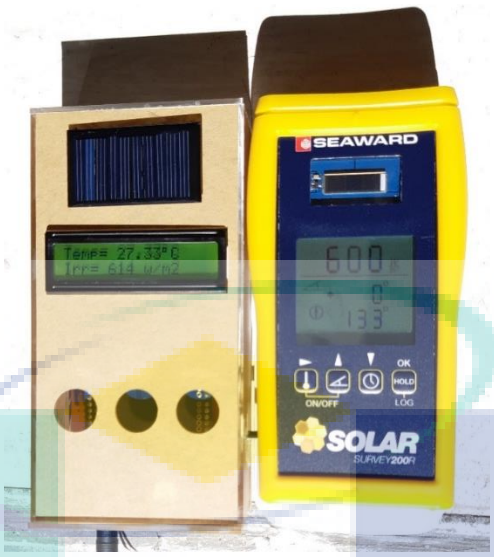


Figure 4.15: Comparing System to Actual Solar Irradiance Reading

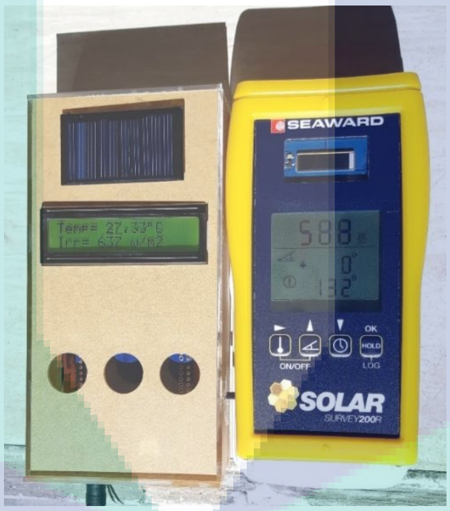


Figure 4.16: Comparing System to Actual Solar Irradiance Reading

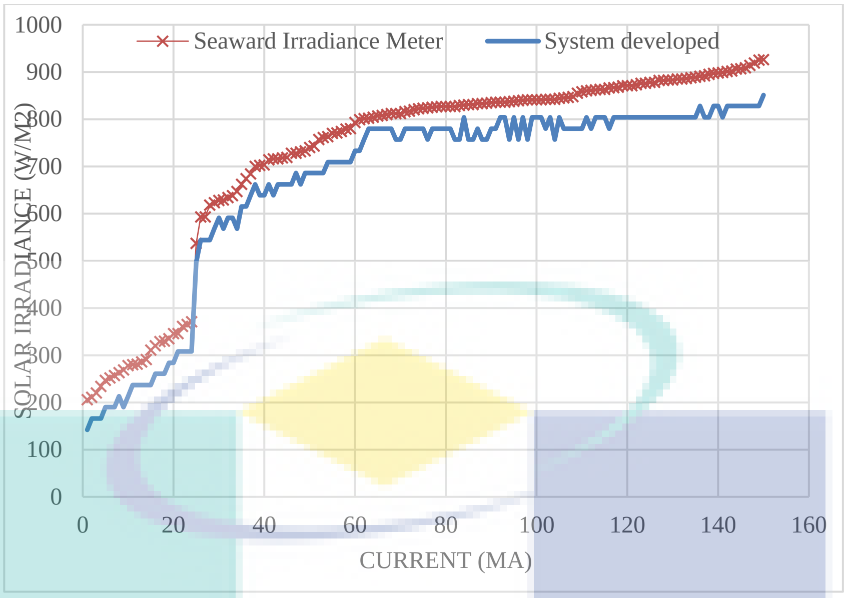


Figure 4.17: Solar Irradiance graph of Seaward Irradiance Meter and System Developed before Calibration

From table (refer to Appendix C) and Figure 4.8, there is some error and difference of solar irradiance readings between the system developed and the Seaward Solar Survey 200R device. Thus, the data was divided into several clusters and the average difference for that specific cluster is calculated. The average difference will be used for the first calibration as shown in the Table 4.7 as shown below.

Table 4.15: First Calibration

Cluster	**Calibration**
Irradiance <= 400 w/m^2	+ 58.125 w/m^2
400 w/m^2 < Irradiance <= 600 w/m^2	+ 46 w/m^2
600 w/m^2 < Irradiance <= 700 w/m^2	+ 54.7 w/m^2
700 w/m^2 < Irradiance <= 800 w/m^2	+ 60.4583 w/m^2
800 w/m^2 < Irradiance <= 900 w/m^2	+ 60.075 w/m^2
Irradiance > 900 w/m^2	+ 81.5556 w/m^2

After the first calibration, the data collection process is repeated. The new percentage of errors are calculated and tabulated as shown in table (refer to Appendix D) and a graph is plotted in Figure 4.18.

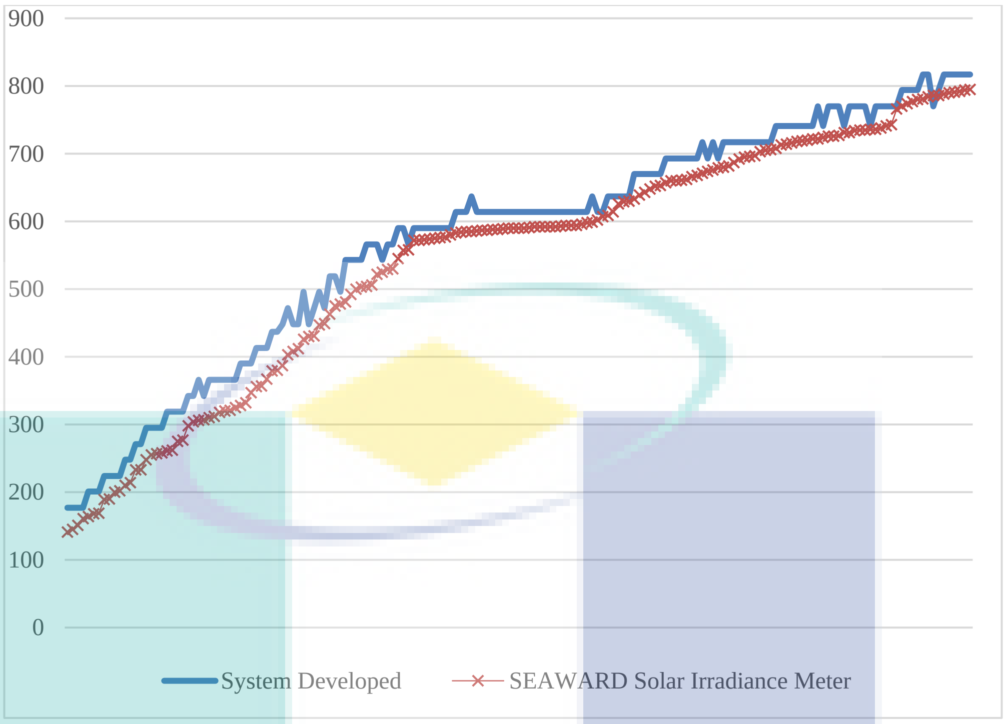


Figure 4.18: Solar Irradiance graph of Seaward Irradiance Meter and System Developed after First Calibration

Since the accuracy for the system developed for solar irradiance at 590 w/m^2 is high enough (less than 10% error), the second calibration will be done for the clusters with solar irradiance lower than 590 w/m^2. The second calibration is as shown in Table 4.8.

Table 4.16: Second Calibration

Cluster	**Calibration**
Irradiance <= 260 w/m^2	-33.5263 w/m^2
26 w/m^2 < Irradiance <= 300 w/m^2	-49 w/m^2
300 w/m^2 < Irradiance <= 350 w/m^2	- 48.8883 w/m^2
350 w/m^2 < Irradiance <= 405 w/m^2	-57.7143 w/m^2
405 w/m^2 < Irradiance <= 510 w/m^2	-44.5625 w/m^2

After the second calibration, the data is collected again and recorded in table, to check if the error is reduced to desired level. Figure 4.19 illustrates a graph plotted by using data in table.

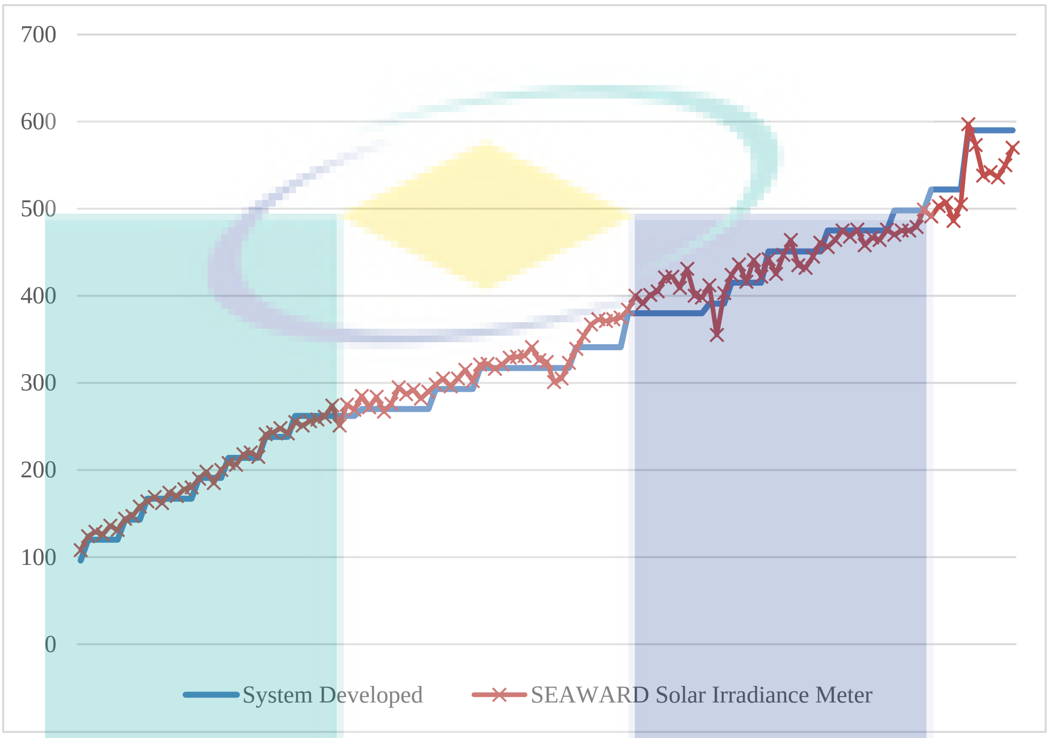


Figure 4.19: Solar Irradiance graph of Seaward Irradiance Meter and System Developed after Second Calibration

From Figure 4.20, we can see the percentage of error of solar irradiance readings between the system developed and Seaward Solar Survey 200R device has been reduced. The percentage of error is successfully been reduced to around 10% or lesser. Figure 4.21 shows a photo of prototype testing. A prototype of this system is developed and compared the solar radiation measurement with a Seaward Solar Survey 200R device. The comparison shows slight difference in the solar irradiance measurement. A mini solar cell is placed on top on the surface to collect solar radiation and converts into electricity. A temperature sensor LM35 is placed on the right surface, located close to the bottom surface. The solar irradiance is computed and displayed on the yellow backlight LCD. The prototype is powered by a 9V battery.

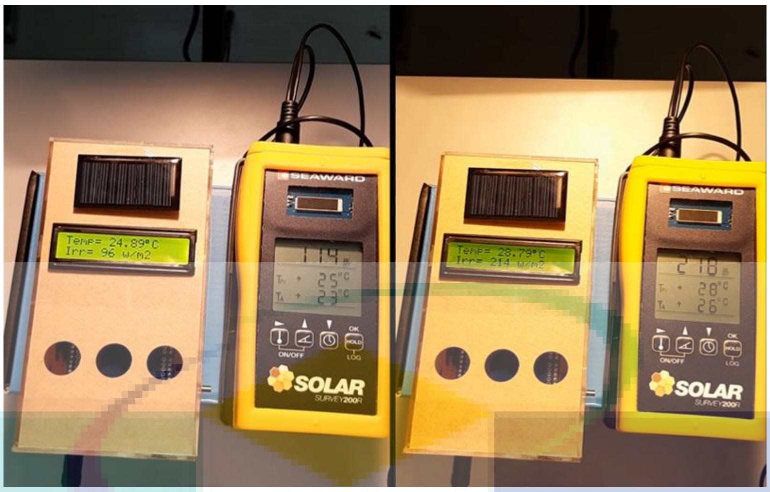


Figure 4.20: Comparing Solar Irradiance Reading from the System Developed and Seaward Solar Survey 200R Device

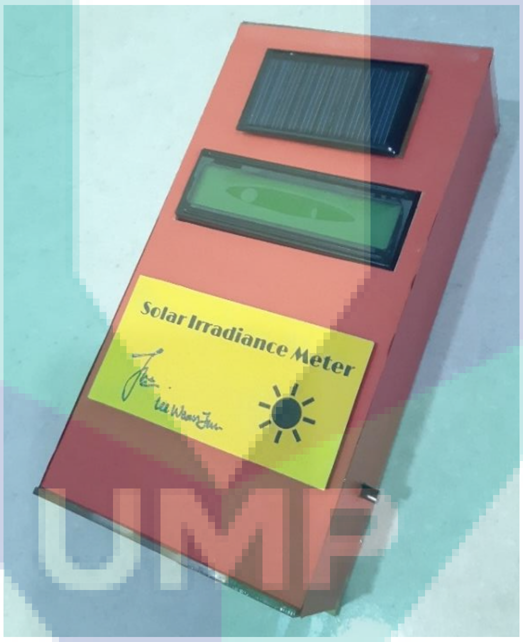


Figure 4.21: Final Prototype of the System Developed

CHAPTER 5

CONCLUSION

5.1 Introduction

In this chapter ,the overall result will be conclude and discussed .In conclusion , the system developed will be able to detect the hotspot on solar panel .The hotspot detection system is developed for solar panel by integrating the source code from Arduino with Matlab GUI . A system that can analyse and indicate the location of hotspot is built through implementation of the equation in Matlab GUI .

5.2 Limitation

Since the algorithm of the hotspot detection system is limited to detect hotspot through the magnitude of temperature .Therefore ,the system is not able to further verify whether the hotspot is caused by electrical fault through I-V characteristic .Therefore ,the only thing that could detect the hotspot is through the comparison of focused cell temperature with the built device such as IR thermometer or IR camera and cell temperature calculated from application.

Even though the device able to measure solar radiation and ambient temperature, the accuracy can be further improved. The system developed used a voltage sensor module with ±0.02V is having not enough resolution for the voltage drop across the small value resistor connected to the solar cell. The solar cell with 30mA maximum current output is too small to be detected. Minimal voltage drops across the 30 Ω resistor is not detectable due to the low resolution of voltage sensor module. As the voltage drop increases to a certain level which is detectable by the voltage sensor module, the new solar irradiance only able to be computed.

5.3 Future Recommendation

This device can set to nearby the solar panel all the time and can act as a surveillance system , so that any user can access to system through implementation of IoT. It will reduce the time taken for hotspot detection in solar panel. To improve the device, a voltage sensor or current sensor with better resolution and accuracy which can detect extremely small current flows from the solar cell should be implemented. A voltage amplifier circuit can also be inserted in the circuit to amplify the voltage drop. In the other way round, a solar cell with greater output current can be used. However, the size of the solar cell should be small enough to be placed in a handheld device. More features such as data logging, angle measurement and compass can be added to the system too.

REFERENCES

1. Vivek Khambalkar, Sandip Nage, Renewable energy: An assessment of public awareness (July, 2010)
2. Solar Energy Australia, Commercial solar energy, solar panels, industry news, technology (Oct 28, 2014)
3. S. Nann and K. Emery, "Spectral effects on PV-device rating," Solar Energy Materials and Solar Cells, vol. 27, no. 3, pp. 189–216, 1992.
4. A. Parretta, M. Bombace, G. Graditi, and R. Schioppo, "Optical degradation of long-term, field-aged c-Si PV modules," Solar Energy Materials and Solar Cells, vol. 86, no. 3, pp. 349–364, 2005.
5. S. Kaplanis and E. Kaplani, "Energy performance and degradation over 20 years performance of bp c-Si PV modules," Simulation Modelling Practice and Theory, vol. 19, no. 4, pp. 1201–1211, 2011.
6. Pragyanshree Samantaray, Sushree Sasmita, Performance of Solar PV Module under partial shading conditions (2016)
7. April M. Salazar, Erees Queen B. M acabebe1, Hotspots Detection in PV Modules Using Infrared Thermography (2015)
8. Genevieve C. Ngo, Erees Queen B. Macabebe, Image Segmentation Using K-Means Color Quantization and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) for Hotspot Detection in PV Modules (2016)
9. Klaus Ramspeck, Stefan Schenk, Denny Duphorn, Axel Metz, Michael Meixner, In-line thermography for reliable hot spot detection and process control (2014)
10. R. Moretón,, E. Lorenzo, L. Narvarte, Experimental observations on hotspots and derived acceptance/rejection criteria.
11. Modules Using Infrared Thermography," MATEC Web of Conferences, vol. 70, p. 10015, 2016.
12. E. Kaplani, "Detection of Degradation Effects in Field-Aged c-Si Solar Cells through IR Thermography and Digital Image Processing," International Journal of Photoenergy, vol. 2012, pp. 1–11, 2012.
13. A. Sinha, O. S. Sastry, and R. Gupta, "Detection and characterisation of delamination in PV modules by active infrared thermography," Nondestructive Testing and Evaluation, vol. 31, no. 1, pp. 1–16, Jan. 2016.
14. C. Dechthummarong, B. Wiengmoon, D. Chenvidhya, C. Jivacate, and K. Kirtikara, "Physical deterioration of encapsulation and electrical insulation properties of PV modules after long-term operation in Thailand," Solar Energy Materials and Solar Cells, vol. 94, no. 9, pp. 1437–1440, Sep. 2010.
15. M. C. A. García, W. Herrmann, W. Böhmer, and B. Proisy, "Thermal and electrical effects caused by outdoor hotspot testing in associations of PV cells: OUTDOOR HOT-SPOT TESTING," Progress in PVs: Research and Applications, vol. 11, no. 5, pp. 293–307, Aug. 2003.
16. "Technical article on infrared cameras." [Online]. Available: http://www.optris.com/technical-article-infrared-cameras. [Accessed: 14-Nov-2017].

17. "focal plane array - Google Search." [Online]. Available: https://www.google.com/search?q=focal+plane+array&rlz=1C1CHBF_enMY721MY721&oq=focal+plane+array+&aqs=chrome..69i57.3932j0j7&sourceid=chrome&ie=UTF-8. [Accessed: 14-Nov-2017].
18. Omron, "Infrared MEMS Thermal Sensor D6T Product Series."
19. N. I. Ahmad, M. Z. A. Kadir, M. Izadi, N. H. Zaini, M. A. . Radzi, and N. Azis, "Effect of temperature on a poly-crystalline solar panel in large scale solar plants in Malaysia," 2015, pp. 244–248.
20. "P/N Junctions and Band Gaps." [Online]. Available: http://solarcellcentral.com/junction_page.html. [Accessed: 15-Nov-2017].
21. "hot-spot-mitigation.pdf." [Online]. Available: https://docs.google.com/viewer?url=http%3A%2F%2Fwww.dupont.com%2Fcontent%2Fdam%2Fdupont%2Fproducts-and-services%2Fsolar-PV-materials%2Fsolar-PV-materials-landing%2Fdocuments%2Fhot-spot-mitigation.pdf. [Accessed: 15-Nov-2017].
22. "Shading PVEducation." [Online]. Available: http://www.pveducation.org/pvcdrom/modules/shading. [Accessed: 17-Nov-2017].
23. G. Shobana, P. Sornadeepika, and R. Ramaprabha, "Global Maximum Power Point Tracking of PV Array under Partial Shaded Conditions," International Journal of Engineering Research, vol. 2, pp. 219–223, Jul. 2013.
24. Guide To Interpreting I-V Curve Measurements of PV Arrays. Solmetric Application Note PVA-600-1, 2011.
25. C. Hellier, Handbook of Nondestructive Evaluation ,first edition. McGraw-Hill, 2001.
26. X. Maldague, Application of Infrared Thermography In Nondestructive Evaluation. Université Laval Quebec City, Québec.

APPENDIX A
HOTSPOT DETECTION ARDUINO CODE

```
/*
****************************************
** Declaration for Mem Thermal Sensor **
****************************************
*/
#include <WireExt.h>
#include <Wire.h>
#define D6T_addr 0x0A
#define D6T_cmd 0x4C
#define ByteOfData 35
byte rbuf[ByteOfData];
char ReadSerialCmd;
String Temperature;
float Temp_float[17];

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

}
void loop()
{
Temperature = GetTemperatureData();

if (Serial.available() > 0)
{
ReadSerialCmd = Serial.read();
if (ReadSerialCmd == 'R')
{
Serial.println(Temperature);
}

}
}



String GetTemperatureData()
{
int i;
float Temp[17];
String tTemperature;
Wire.beginTransmission(D6T_addr);
Wire.write(D6T_cmd);
Wire.endTransmission();
delay(70);
i = 0;
if (WireExt.beginReception (D6T_addr) >= 0) {
while (i < (ByteOfData)) {
rbuf[i] = WireExt.get_byte ();
//Serial.println(rbuf[i]);
i++;
}
WireExt.endReception ();
}

for (i = 0; i < 17; i++)
{
Temp[i] = ((rbuf[(i * 2)] + (rbuf[(i * 2 + 1)] << 8)) * 0.1);
}
// The Camera is Fixed Vertically Inversed So the sensor value need to be inversed
Temp_float[0] = Temp[0]; //1st Array is Ambient Temperature
for (i = 1 ; i < 17 ; i++)
{
Temp_float[i] = Temp[17 - i];
}
for (i = 0 ; i < 17 ; i ++)
{
tTemperature = tTemperature + Temp_float[i];
if (i < 16)
{
tTemperature = tTemperature + ",";
}
}
return tTemperature;
}
```

APPENDIX B
MATLAB CODE

```
function varargout = webcam(varargin)
% TEMPERATURESENSORGUI MATLAB code for TemperatureSensorGUI.fig
%      TEMPERATURESENSORGUI, by itself, creates a new
TEMPERATURESENSORGUI or raises the existing
%      singleton*.
%
%      H = TEMPERATURESENSORGUI returns the handle to a new
TEMPERATURESENSORGUI or the handle to
%      the existing singleton*.
%
%      TEMPERATURESENSORGUI('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in TEMPERATURESENSORGUI.M with the
given input arguments.
%
%      TEMPERATURESENSORGUI('Property','Value',...) creates a new
TEMPERATURESENSORGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before TemperatureSensorGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to TemperatureSensorGUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help TemperatureSensorGUI

% Last Modified by GUIDE v2.5 02-May-2018 22:18:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @webcam_OpeningFcn, ...
                   'gui_OutputFcn',  @webcam_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before TemperatureSensorGUI is made visible.
function webcam_OpeningFcn(hObject, eventdata, handles, varargin)
global isLooping
global delayReceive
global toggle
global LowestTemp;
global HighestTemp;
global Temperature;
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to TemperatureSensorGUI (see VARARGIN)

serialPorts = instrhwinfo('serial');
nPorts = length(serialPorts.SerialPorts);
set(handles.lstPort, 'String', ...
    [{'Select a port'} ; serialPorts.SerialPorts ]);
set(handles.lstPort, 'Value', 2);

axes(handles.axes1);
vid= videoinput('winvideo',1);
hImage=image(zeros(1024,768,3),'Parent',handles.axes1);
preview(vid,hImage);

%automatic loop purpose
isLooping = 0;
delayReceive = 0.05;
toggle = 0;
LowestTemp = 0;
HighestTemp = 0;
% Choose default command line output for TemperatureSensorGUI
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);
Main_Loop(hObject, handles);
% UIWAIT makes TemperatureSensorGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = webcam_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on selection change in lstPort.
function lstPort_Callback(hObject, eventdata, handles)
% hObject    handle to lstPort (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns lstPort contents as cell array
%        contents{get(hObject,'Value')} returns selected item from lstPort


% --- Executes during object creation, after setting all properties.
function lstPort_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lstPort (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in cmdConnect.
function cmdConnect_Callback(hObject, eventdata, handles)
% hObject    handle to cmdConnect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if strcmp(get(hObject,'String'),'Connect') % currently disconnected
    serPortn = get(handles.lstPort, 'Value');
    if serPortn == 1
        errordlg('Select valid COM port');
    else
        serList = get(handles.lstPort,'String');
        serPort = serList{serPortn};
        serConn = serial(serPort, 'TimeOut', 1, ...
            'BaudRate', str2num(get(handles.txtBaudrate, 'String')));

        try
            fopen(serConn);
            handles.serConn = serConn;

         % enable Tx text field and Rx button
         %   set(handles.Tx_send, 'Enable', 'On');
         %   set(handles.rxButton, 'Enable', 'On');

            set(hObject, 'String','Disconnect')
        catch e
            errordlg(e.message);
        end

    end
else
  %  set(handles.Tx_send, 'Enable', 'Off');
  %  set(handles.rxButton, 'Enable', 'Off');

    set(hObject, 'String','Connect')
    fclose(handles.serConn);
end
Main_Loop(hObject, handles);
guidata(hObject, handles);



function txtBaudrate_Callback(hObject, eventdata, handles)
% hObject    handle to txtbaudrate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txtbaudrate as text
%        str2double(get(hObject,'String')) returns contents of txtbaudrate as a double


% --- Executes during object creation, after setting all properties.
function txtBaudrate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtbaudrate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function txtCommand_Callback(hObject, eventdata, handles)
% hObject    handle to txtCommand (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txtCommand as text
%        str2double(get(hObject,'String')) returns contents of txtCommand as a double
TxText = get(handles.txtCommand, 'String');
fprintf(handles.serConn, TxText);

%currList = get(handles.history_box, 'String');

%set(handles.history_box, 'String', ...
%    [currList ; ['Sent @ ' datestr(now) ': ' TxText] ]);
%set(handles.history_box, 'Value', length(currList) + 1 );

%set(hObject, 'String', '');

% --- Executes during object creation, after setting all properties.
function txtCommand_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtCommand (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Send_Serial(handles, Tx)
%TxText = get(handles.txtCommand, 'String');
fprintf(handles.serConn, Tx);

function Receive_Serial(hObject,handles,d)
% d
% disp('d s value received')
% pause(2)
global EnableSaveFile;
global delayReceive
try
    RxText = fscanf(handles.serConn);
    TempArray = strsplit(RxText,',');
    determineTempLevel(TempArray);

    Update_Array(hObject,handles,TempArray,d);
%     disp(TempArray(1))
    Update_Label(handles,TempArray(1));
    pause(delayReceive);
    if EnableSaveFile == 1
    Append_File(RxText);
    end
%     currList = get(handles.history_box, 'String');
    if length(RxText) < 1
%         RxText = 'Timeout @ ';
%         set(handles.history_box, 'String', ...
%             [currList ; [RxText datestr(now)] ]);
    else
%         set(handles.history_box, 'String', ...
%             [currList ; ['Received @ ' datestr(now) ': ' RxText ] ]);

    end
```

```
%     set(handles.history_box, 'Value', length(currList) + 1 );
catch e
%     disp(e)
end


function Update_Label(handles, Temperature)
global LowestTemp;
global HighestTemp;
global TempDouble;
TempDouble = str2double(Temperature);
disp('Update label loop')
% disp(TempDouble)
if (LowestTemp == 0 )||(LowestTemp > TempDouble)
    LowestTemp= TempDouble;
end
if (HighestTemp == 0 ) ||(HighestTemp < TempDouble)
    HighestTemp=TempDouble;
end
disp('Update label inner loop')
set(handles.lblCurrentTemp,'String',(TempDouble));
set(handles.lblLowestTemp,'String',(LowestTemp));
set(handles.lblHighestTemp,'String',(HighestTemp));

   %                  Dont hv cam invert so no need TemperatureArray1
function Update_Array(hObject, handles, TemperatureArray1,d)
% d
% disp('Update_Array')
% pause(2)
global delayDraw;
global counterDraw;
counterDraw = 0;
delayDraw = 0;
%TemperatureArray=zeros(1,17);
counterDraw=counterDraw+1;
disp('in the outside loop')

TemperatureArray = TemperatureArray1;

 if (get(handles.chkHorizontalInverse,'Value')==1)
 TemperatureArray(2)=TemperatureArray1(5);
  TemperatureArray(3)=TemperatureArray1(4);
  TemperatureArray(4)=TemperatureArray1(3);
  TemperatureArray(5)=TemperatureArray1(2);
  TemperatureArray(6)=TemperatureArray1(9);
  TemperatureArray(7)=TemperatureArray1(8);
  TemperatureArray(8)=TemperatureArray1(7);
  TemperatureArray(9)=TemperatureArray1(6);
  TemperatureArray(10)=TemperatureArray1(13);
  TemperatureArray(11)=TemperatureArray1(12);
  TemperatureArray(12)=TemperatureArray1(11);
  TemperatureArray(13)=TemperatureArray1(10);
  TemperatureArray(14)=TemperatureArray1(17);
  TemperatureArray(15)=TemperatureArray1(16);
  TemperatureArray(16)=TemperatureArray1(15);
  TemperatureArray(17)=TemperatureArray1(14);
 end

if(counterDraw >= delayDraw)
disp('in the loop')

set(handles.Array11,'String',(TemperatureArray(2)));
set(handles.Array12,'String',(TemperatureArray(3)));
set(handles.Array13,'String',(TemperatureArray(4)));
set(handles.Array14,'String',(TemperatureArray(5)));
set(handles.Array21,'String',(TemperatureArray(6)));
set(handles.Array22,'String',(TemperatureArray(7)));
set(handles.Array23,'String',(TemperatureArray(8)));
set(handles.Array24,'String',(TemperatureArray(9)));
set(handles.Array31,'String',(TemperatureArray(10)));
set(handles.Array32,'String',(TemperatureArray(11)));
set(handles.Array33,'String',(TemperatureArray(12)));
set(handles.Array34,'String',(TemperatureArray(13)));
set(handles.Array41,'String',(TemperatureArray(14)));
set(handles.Array42,'String',(TemperatureArray(15)));
set(handles.Array43,'String',(TemperatureArray(16)));
set(handles.Array44,'String',(TemperatureArray(17)));


set(handles.Array11,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(2)));
set(handles.Array12,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(3)));
set(handles.Array13,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(4)));
set(handles.Array14,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(5)));
```

```
set(handles.Array21,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(6)));
set(handles.Array22,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(7)));
set(handles.Array23,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(8)));
set(handles.Array24,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(9)));
set(handles.Array31,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(10)));
set(handles.Array32,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(11)));
set(handles.Array33,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(12)));
set(handles.Array34,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(13)));
set(handles.Array41,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(14)));
set(handles.Array42,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(15)));
set(handles.Array43,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(16)));
set(handles.Array44,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(17)));

set(handles.BArray11,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(2)));
set(handles.BArray12,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(3)));
set(handles.BArray13,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(4)));
set(handles.BArray14,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(5)));
set(handles.BArray21,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(6)));
set(handles.BArray22,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(7)));
set(handles.BArray23,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(8)));
set(handles.BArray24,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(9)));
set(handles.BArray31,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(10)));
```

```
set(handles.BArray32,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(11)));
set(handles.BArray33,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(12)));
set(handles.BArray34,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(13)));
set(handles.BArray41,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(14)));
set(handles.BArray42,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(15)));
set(handles.BArray43,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(16)));
set(handles.BArray44,'backgroundcolor',ColorOnTemperature(handles,
TemperatureArray(17)));

% d
% disp('in update array')
% pause(2)
     c1=[1 0 0 ];
     c2=[0 1 0];
     if str2double(TemperatureArray(2)) > d
     set(handles.b1,'backgroundcolor',c1);
 else
     set(handles.b1,'backgroundcolor',c2);
 end
 if str2double(TemperatureArray(3)) > d
     set(handles.b2,'backgroundcolor',c1);
 else
     set(handles.b2,'backgroundcolor',c2);
 end
 if str2double(TemperatureArray(4)) > d
     set(handles.b3,'backgroundcolor',c1);
 else
     set(handles.b3,'backgroundcolor',c2);
 end
 if str2double(TemperatureArray(5)) > d
     set(handles.b4,'backgroundcolor',c1);
 else
     set(handles.b4,'backgroundcolor',c2);
 end
  if str2double(TemperatureArray(6)) > d
     set(handles.b5,'backgroundcolor',c1);
 else
     set(handles.b5,'backgroundcolor',c2);
```

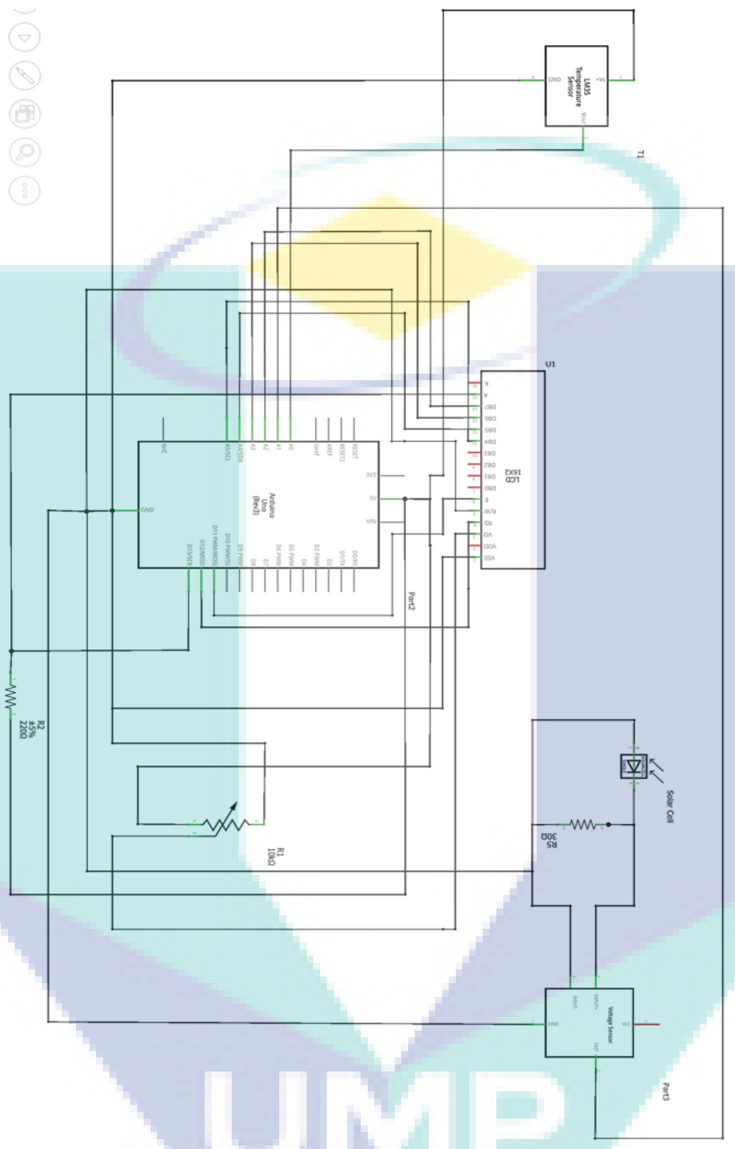
```
end
if str2double(TemperatureArray(7)) > d
   set(handles.b6,'backgroundcolor',c1);
else
   set(handles.b6,'backgroundcolor',c2);
end
if str2double(TemperatureArray(8)) > d
   set(handles.b7,'backgroundcolor',c1);
else
   set(handles.b7,'backgroundcolor',c2);
end
if str2double(TemperatureArray(9)) > d
   set(handles.b8,'backgroundcolor',c1);
else
   set(handles.b8,'backgroundcolor',c2);
end
if str2double(TemperatureArray(10)) > d
   set(handles.b9,'backgroundcolor',c1);
else
   set(handles.b9,'backgroundcolor',c2);
end
if str2double(TemperatureArray(11)) > d
   set(handles.b10,'backgroundcolor',c1);
else
   set(handles.b10,'backgroundcolor',c2);
end
if str2double(TemperatureArray(12)) > d
   set(handles.b11,'backgroundcolor',c1);
else
   set(handles.b11,'backgroundcolor',c2);
end
if str2double(TemperatureArray(13)) > d
   set(handles.b12,'backgroundcolor',c1);
else
   set(handles.b12,'backgroundcolor',c2);
end
 if str2double(TemperatureArray(14)) > d
   set(handles.b13,'backgroundcolor',c1);
else
   set(handles.b13,'backgroundcolor',c2);
end
if str2double(TemperatureArray(15)) > d
   set(handles.b14,'backgroundcolor',c1);
else
```

```
    set(handles.b14,'backgroundcolor',c2);
 end
 if str2double(TemperatureArray(16)) > d
    set(handles.b15,'backgroundcolor',c1);
 else
    set(handles.b15,'backgroundcolor',c2);
 end
 if str2double(TemperatureArray(17)) > d
    set(handles.b16,'backgroundcolor',c1);
 else
    set(handles.b16,'backgroundcolor',c2);
 end
counterDraw=0;
end


guidata(hObject, handles);

function Color=ColorOnTemperature(handles, Temperature)


global tempLevel_1;
global tempLevel_2;
global tempLevel_3;
global tempLevel_4;


cmptempLevel_1=tempLevel_1;
cmptempLevel_2=tempLevel_2;
cmptempLevel_3=tempLevel_3;
cmptempLevel_4=tempLevel_4;


%tempLevel_1=24;
%tempLevel_2=25;
%tempLevel_3=28;
%tempLevel_4=30;

%tempLevel_1=26;
%tempLevel_2=27;
%tempLevel_3=28;
%tempLevel_4=30;

% [1 1 0] y yellow
% [1 0 1] m magenta
% [0 1 1] c cyan
% [1 0 0] r red
% [0 1 0] g green
% [0 0 1] b blue
% [1 1 1] w white
% [0 0 0] k black

tTemperature = str2double(Temperature);
  if (tTemperature < cmptempLevel_1)
    Color = [0 1 1];
  end

  if (tTemperature >= cmptempLevel_1) && (tTemperature <= cmptempLevel_2 )
  Color = [0 0 1];
  end

  if (tTemperature >= cmptempLevel_2) && (tTemperature <= cmptempLevel_3)
  Color = [1 1 0];
  end

  if (tTemperature >= cmptempLevel_3) && (tTemperature <= cmptempLevel_4)
  Color = [1 0 1];
  end

  if (tTemperature > cmptempLevel_4)
  Color = [1 0 0];
  end


function determineTempLevel(TemperatureArray)

global tempLevel_1;
global tempLevel_2;
global tempLevel_3;
global tempLevel_4;

tempLevel_1=str2double(TemperatureArray(2));
tempLevel_4=str2double(TemperatureArray(2));

for i=2:17
    if (tempLevel_4 < str2double(TemperatureArray(i)))
        tempLevel_4 = str2double(TemperatureArray(i)); %Highest Temperature
    end
    if (tempLevel_1 > str2double(TemperatureArray(i)))
        tempLevel_1 = str2double(TemperatureArray(i)); %Lowest  Temperature
    end
end


levelGap = (tempLevel_4 - tempLevel_1)/5 ;
tempLevel_1 = tempLevel_1 + levelGap;
tempLevel_2 = tempLevel_1 + levelGap ;
tempLevel_3 = tempLevel_2 + levelGap ;
tempLevel_4 = tempLevel_4 - levelGap;

function Main_Loop(hObject,handles)
global Run;
global isLooping;
global delayReceive;
global toggle;
global TempDouble;

disp('MainLoop b4 Run')
while true
    if (isLooping==1)
        disp(isLooping)
        Send_Serial(handles,'R\n');
        pause(delayReceive);
        a=TempDouble;
        b=str2double(get(handles.y,'string'));
        c=str2double(get(handles.z,'string'));
        d=a+((b-20)/800)*c ;
        ee=d;

        set(handles.u,'string',d);
        pause(delayReceive);
%           d
%           disp('d s value in main loop')
%           pause(2)
        Receive_Serial(hObject,handles,ee);


    end


    if (toggle == 1)
        Send_Serial(handles,'R\n');
        a=TempDouble;
        b=str2double(get(handles.y,'string'));
        c=str2double(get(handles.z,'string'));
        d=a+((b-20)/800)*c;
        ee=d;
        set(handles.u,'string',d);
        pause(delayReceive);
%         d
%         disp('d s value in main loop')
%         pause(2)
        Receive_Serial(hObject,handles,ee);
        set(handles.u,'string',d);
        pause(delayReceive);
        toggle=0;

    end
    pause(delayReceive)
end
guidata(hObject,handles)



function y_Callback(hObject, eventdata, handles)
% hObject    handle to y (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of y as text
%        str2double(get(hObject,'String')) returns contents of y as a double


% --- Executes during object creation, after setting all properties.
function y_CreateFcn(hObject, eventdata, handles)
% hObject    handle to y (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function z_Callback(hObject, eventdata, handles)
% hObject    handle to z (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of z as text
%        str2double(get(hObject,'String')) returns contents of z as a double


% --- Executes during object creation, after setting all properties.
function z_CreateFcn(hObject, eventdata, handles)
% hObject    handle to z (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in optAverage.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to optAverage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of optAverage
% --- Executes on button press in cmdSendR.

function cmdSendR_Callback(hObject, eventdata, handles)
global delayReceive;
global toggle
% hObject    handle to cmdSendR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% TxText = get(handles.txtCommand, 'String');
% fprintf(handles.serConn, TxText);
```

```
disp('toggle = 1')
toggle = 1;
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function cmdConnect_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cmdConnect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on button press in AutoRuncmd.
function AutoRuncmd_Callback(hObject, eventdata, handles)
% hObject    handle to AutoRuncmd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Run;
global isLooping;
global delayReceive;
global TempDouble;
disp('autoruncmd')
Run = 1;
% hObject    handle to cmdAutoRun (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if strcmp(get(hObject,'String'),'AutoRun [start]') % currently disconnected
  isLooping=1;
  disp('Looping = 1')
  set(hObject, 'String','AutoRun[Stop]')

else
    set(hObject, 'String','AutoRun [start]')
    disp('looping = 0')
    isLooping=0;
end


guidata(hObject, handles);


% --- Executes on button press in chkHorizontalInverse.
function chkHorizontalInverse_Callback(hObject, eventdata, handles)
% hObject    handle to chkHorizontalInverse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chkHorizontalInverse
```

APPENDIX B
SCHEMATIC DIAGRAM OF IRRADIANCE METER



LM35 Temperature Sensor
T1
U1
LCD 16X2
Arduino Uno (Rev3)
Part2
R2 ±5% 220Ω
R1 10kΩ
R5 30Ω
Solar Cell
Voltage Sensor
Part3