PV Data Logger Report

Supervisor: Gareth Lee
Customer: David Parlevliet

Nathan Froese
11/18/2013
Abstract
Photovoltaic monitoring is a vital part for the implementation and optimization of solar energy as an electricity source. Today’s PV monitoring systems are constructed from costly tools and complex designs. This thesis project discusses the design of a basic and cost efficient PV weather monitoring system. The design consists of hardware processed via a microcontroller, a low powered computer and a software designed database program, Microsoft SQL Server Management Studio Express. The stored measurements can be viewed on LabVIEW, MS Access and MS Excel.
Acknowledgements

I would like to thank Dr Gareth Lee my project advisor who guided and helped me with various insights along the way. Dr David Parlevliet I would also thank for assisting in the initial design requirements.

The author would also like to thank the assistance from certain individuals of knowledge and dedication in this project that was greatly valued and appreciated:

- Mr Lafeta lava – For assisting and always giving time to provide help with electrical equipment required during this project.
- Mr Will Stirling – In assisting in understanding MS SQL Server Database software

Thanks to all of my friends and colleagues who have assisted me with their thoughts and ideas though this project and degree. And most of all my family, for supporting me all the way during the course of my engineering degree.
# Table of Contents

Abstract .......................................................................................................................... i  
Acknowledgements ......................................................................................................... ii  
List of Figures .................................................................................................................. vii  
List of Tables ................................................................................................................... ix  
Terminology and Acronyms ........................................................................................... x  

1 Introduction .................................................................................................................. 1  
1.1 Project Structure ...................................................................................................... 1  
1.2 Project Objectives ................................................................................................. 1  
1.3 Weather system monitoring .................................................................................. 2  
1.4 Literature Review .................................................................................................... 3  

2 Hardware .................................................................................................................... 4  
2.1 Overview ................................................................................................................. 4  
2.2 Design ...................................................................................................................... 5  
2.2.1 Raspberry Pi ...................................................................................................... 5  
2.2.2 Arduino Uno ........................................................................................................ 7  
2.2.3 Arduino Uno Shields ............................................................................................ 8  
2.2.4 Polymer Li-Ion Battery ..................................................................................... 10  
2.2.5 2.5W Solar Panel ............................................................................................... 11  
2.2.6 GPS Bee kit ....................................................................................................... 12  
2.2.7 DS18B20 Temperature Sensors .......................................................................... 14  
2.2.8 DHT22 Ambient Temperature and Humidity Sensor ....................................... 16  
2.2.9 Anemometer Wind Speed Sensor ....................................................................... 17  
2.2.10 Wind Vane Sensor ............................................................................................ 17  

3 Communications ....................................................................................................... 19  
3.1 Overview ................................................................................................................. 19  
3.2 GPS UART Communications .................................................................................. 19  
3.2.1 Software Serial .................................................................................................. 19  
3.2.2 NEMA .............................................................................................................. 19  
3.3 DS18B20 1-Wire Communications ....................................................................... 21  
3.3.1 1-Wire Line Initialization .................................................................................. 24  
3.3.2 ROM Command Sequence .............................................................................. 25  
3.3.3 Memory Function Command Sequence .......................................................... 28  
3.4 DHT22 PWM Communications ............................................................................ 31
11.1 Arduino Programs ............................................................. 54
11.2 Python Programs.............................................................. 54
11.3 LabVIEW Programs.......................................................... 54
11.4 Configuring Raspberry Pi ..................................................... 54
11.5 Parts List ........................................................................... 54
11.6 Gantt chart ........................................................................ 54
11.7 OpenSqlServerPort ............................................................ 54
List of Figures

Figure 1 SPI connection layout Colin M.L. Burnett. (Wikimedia n.d.) ........................................... xi
Figure 2 overall PV Weather Monitoring System ................................................................. 4
Figure 3 Raspberry Pi top and general board layout (Raspberry Pi n.d.) ........................................ 5
Figure 4 Raspberry Pi bottom and SD card insert port ............................................................. 6
Figure 5 Raspberry Pi GPIO Usable pins initial state on WiringPi ............................................. 6
Figure 6 Raspberry Pi GPIO layout (Adafruit n.d.) ................................................................. 7
Figure 7 Arduino Uno Board top (littlebirdelectronics n.d.) ..................................................... 7
Figure 8 Arduino Uno Board Layout (Arduino 2013) ............................................................... 8
Figure 9 Solar Charger Shield V2.0 (littlebirdelectronics n.d.) ................................................. 9
Figure 10 XBee Shield V2.0 (littlebirdelectronics n.d.) ............................................................. 9
Figure 11 Prototyping Shield for Arduino (littlebirdelectronics n.d.) .......................................... 10
Figure 12 Polymer Lithium Ion Battery 2000mAh (littlebirdelectronics n.d.) ............................. 11
Figure 13 2.5W Solar Panel 116X160 (littlebirdelectronics n.d.) ............................................... 11
Figure 14 GPS Bee Kit with mini embedded antenna (seeedstudio 2013) ................................. 12
Figure 15 GPS Bee schematic (seeedstudio 2013) ................................................................. 13
Figure 16 Powering the DS18B20 with External Supply (Tushev 2013) ..................................... 15
Figure 17 DS18B20 BLOCK DIAGRAM PARASITE POWER (maxim integrated n.d.) .............. 15
Figure 18 DS18B20 1-Wire Hardware Configuration (maxim integrated n.d.) ............................ 15
Figure 19 DHT22 electrical connection diagram (MaxDetect Technology Co., Ltd. n.d.) ............. 16
Figure 20 Grove DHT22 sensor (seeedstudio 2013) ................................................................. 16
Figure 21 RJ45 connection between anemometer and wind vane (Argent Data Systems n.d.) ....... 17
Figure 22 Anemometer circuit with pull-up resistor and capacitor to stop debouncing ............... 17
Figure 23 anemometer and wind vane RJ45 pin layout (Argent Data Systems n.d.) ..................... 18
Figure 24 1-Wire daisy chain connection layout (maxim integrated n.d.) ................................. 21
Figure 25 1-Wire device data flow diagram (maxim integrated n.d.) .......................................... 22
Figure 26 1-Wire 3 phase transaction sequence flowchart summary ........................................ 22
Figure 27 1-Wire timeline of the 3 phase transaction (maxim integrated n.d.) ............................ 23
Figure 28 1-Wire Reset/Presence Waveforms (maxim integrated n.d.) ..................................... 24
Figure 29 1-Wire signalling Read/Write bit Waveforms (maxim integrated n.d.) ......................... 24
Figure 30 1-Wire Layout and internal characteristics of the ROM ID (maxim integrated n.d.) ...... 25
Figure 31 Summarisation of the 1-Wirer Initialisation and ROM Command Sequence transaction process (maxim integrated n.d.) ................................................................. 26
Figure 32 Summarisation of the 1-Wirer Memory Function Command Sequence transaction process (maxim integrated n.d.) ................................................................. 28
Figure 33 1-Wire Memory Functions Flow Chart Summary ...................................................... 29
Figure 34 DH22 overall communication process (MaxDetect Technology Co., Ltd. n.d.) .......... 31
Figure 35 Anemometer Program code sequence ....................................................................... 32
Figure 36 Wind Vane program code sequence .......................................................................... 32
Figure 37 Raspberry Pi Remote Connection start up screen ..................................................... 34
Figure 38 PyODBC connection string code .............................................................................. 35
Figure 39 Check to see if table exists then insert the CSV flat file content ............................... 35
Figure 40 Activity Monitor ........................................................................................................ 36
Figure 41 Microsoft SQL Server Management Studio initial layout ........................................ 37
Figure 42 Frequency (Hz) versus Time (5 seconds) Light Sensor not calibrated graph ............... 39
Figure 43 Temperature (*C) versus Time (5 seconds) for DHT22 ambient temperature sensor ........ 39
Figure 44 the MLX90614-AAA (littlebirdelectronics n.d.) .......................................................... 41
Figure 45 SMBus I2C connection to MLX90614 IR sensor (bildr.blog 2011) ............................... 42
Figure 46 Typical response of a silicon solar cell pyranometer (courtesy of Apogee Instruments, Inc.) ........................................................................................................................................... 43
Figure 47 Parts to construct the Pyranometer (May 2011) ............................................................ 44
Figure 48 Sample data of Handmade Pyranometer versus Apogee Pyranometer without amplifier (May 2011) ................................................................................................................................. 45
Figure 49 APC220 to microcontroller connection (APPCON Technologies March) ..................... 46
Figure 50 6.4.2 2.4G NRF24L01 + PA + LNA Wireless Module with and without antenna (NORDIC Semiconductor 2008) .................................................................................................................. 47
Figure 51 Primary receiver using MultiCeiver (NORDIC Semiconductor 2008) ......................... 47
Figure 52 VBAT connected to analog pin 0 for battery voltage measurement .............................. 48
List of Tables
Table 1 Raspberry Pi Model B Specifications Summary (Raspberry Pi n.d.) ........................................... 5
Table 2 Polymer Li-Ion Battery Summary (littlebirdelectronics n.d.) ...................................................... 10
Table 3 GPS Bee kit Specification (seeedstudio 2013) ............................................................................ 13
Table 4 GPS Bee Pin definition and Rating (seeedstudio 2013) ................................................................. 14
Table 5 DS18B20 Specification Summary (maxim integrated n.d.) ........................................................... 14
Table 6 DHT22 Specification Summary ...................................................................................................... 16
Table 7 wind vane expected results (Argent Data Systems n.d.) ............................................................... 18
Table 8 1-Wire Specifications summary (maxim integrated n.d.) ............................................................. 21
Table 9 1-Wire ROM Function Commands Summary (maxim integrated n.d.) ........................................ 26
Table 10 Search ROM Algorithm Lookup Table (maxim integrated n.d.) .................................................. 27
Table 11 1-Wire Memory Function Commands Summary (maxim integrated n.d.) ................................. 28
Table 12 Temperature/Data Relationships (maxim integrated n.d.) ............................................................ 30

List of Equations
Equation 1 ADC conversion equation ......................................................................................................... 32
Terminology and Acronyms

CSV – Comma Separated Values (a type of flat text file)

DMA – Direct Memory Access allows other operations to execute while the CPU performs a specific task. This is very useful at any moment the CPU can’t keep up the rate of data transfer. Various hardware system such as disk drive controllers graphics cards and memory cards use DMA. Overall this reduces the CPU overhead compared with other processing systems that do not have it.

Database Mirroring – Allows continuous stream of transaction logs from a source server to a single destination. In an event of primary system failure applications can reconnect to the database on the secondary server.

Database Snapshots- Creates instant read-only views of a database that can be used to quickly recover accidental changes to a database re-applying the original pages.

DSP – Digital Signal Processor is a specialized microprocessor optimized for fast operational digital signalling needs.

FTP – File Transfer Protocol is a standard protocol used to transfer files from one host to another host via TCP-based networks such as the internet. FTP is created on a client-server configuration and uses separate control and data connections amongst the client and the server.

GPS – Global Positioning System is a space-based satellite navigation system that provides location and time in various weather conditions, anywhere on or close to the earth where there is an unobstructed line of sight to four or more satellites. Applied to various applications such as military, civil and commercial users across the world.

GUI – Graphical User Interface is a type of interface that allows users to virtually interact with electronic devices via graphical icons and visual indicators.

I2C – Inter-Integrated Circuit is a multimaster serial single-ended computer bus created by Phillips applied for connecting low-speed peripherals to various devices such as motherboard, embedded system or cell phone. I²C uses only two bidirectional open drain lines.

SCK – Serial Clock line used in I2C

SDA – Serial Data line used in I2C

SMBus – System Management Bus is a single-ended two-wire bus based on I²C though there are several differences between them in the areas of electrical, timing, protocols and operating modes. These differences are specific to input voltage, sink current, frequency minimum and maximum, timing, protocols- ACK and NACK usage, Address Resolution, Time-out feature and more.

LabVIEW - Laboratory Virtual Instrument Engineering Workbench is a system-design platform and development environment for visual programming language from National Instruments. This graphical language is called “G”.

x
MARS – Multiple Active Results Sets removes the limit of one pending request on a given session.


ODBC – Open Database Connection.

PV – Photovoltaic.

PWM – Pulse Width Modulated

PY – Python a programming language code.

PyODBC – Python open database connection library.

RDP – Remote Desktop Protocol used to allow remote desktop connection from one computer to another

RPI – Raspberry Pi a credit card size computer.

SPI – Serial Peripheral Interface is a synchronous serial data link that operates in full duplex mode. Multiple slave devices are allowed with individual slave select lines. SPI is known as a four-wire serial bus, which is also often referred to as synchronous serial interface (SSI).

SCLK – Serial Clock which the master outputs

MISO – Master In Slave Out is where the master receives data the slave transmit.

MOSI – Master Out Slave In is where the master transmits and the slave receives data.

SS – Slave Select output from the master. Default sets itself low in order to activate the device set line high.

SQL – Structured Query Language
SSH – Secure Shell
TMSR – Read Sample Time
TTL – Transistor-transistor logic
TWI/I2C – Two Wire Interface
UART – Universal Asynchronous Receiver and transmitter
USB – Universal Serial Bus
XML – Extensible Markup Language
TxD – Transmit
RxD – Receive
LSB – Least Significant Bit used for explaining RS232 Serial Communication
MSB – Most Significant Bit used for explaining RS232 Serial Communication
1 Introduction

Data logging of sensor measurement of weather conditions affecting photovoltaic solar panels informs the user of suitable locations of certain solar panel types to use for larger PV installations. Weather conditions measured are solar panel temperature, ambient temperature, humidity, irradiance, wind speed and direction. Along with weather conditions the location is found using GPS. These measurements have significance in the solar panel industry. With greater amount of houses installing PV solar panel systems it is of great importance in identifying the best type suited for in that location with maximum power output over the long term. From measuring and data logging, trends can become applied diminishing the potential for solar panel inefficiencies due to wrong solar panel type installed.

1.1 Project Structure

An elaborate introduction following the project structure covers background information, a summarized review, and a summary of existing solutions, scope of the project and objectives are distinguished. After this, the project report is separated into specific design phases that encompass each stage of the project report:

- **Hardware** covers the select specifications of each device used to design the PV weather monitoring system;
- **Communications** details the process each device talks to and from the Arduino Uno microcontroller;
- **Software** encompasses the way in which each software program relates to the overall design towards the overall system and structure of the programming code;
- **Testing** shifts the project’s focus to analysis of results and errors occurred within this project;
- **Future Improvements** details in depth all the possible feasible upgrades to further enhance the PV monitoring systems accuracy and performance;
- **Recommendations** provides overall cost and conclusion of the PV Weather Monitoring System.

1.2 Project Objectives

The Overall aim of this thesis project is to develop and design a weather data logger measurement system to monitor and collect into a Microsoft SQL Server Database, weather information of the different solar panel types. In designing the database system, selection of devices, including global positioning system, temperature sensors, humidity sensor, light intensity, wind speed and direction. For each device the following goals were as follows:

**Microcontroller** –
- Energy efficient for remote powering via solar power and battery power
- Within the range 12 to 16 bit resolution for analog and digital conversion capability
- I2C, SPI, UART serial communication

**SQL Database Server** –
Able to set security permissions for certain users to configure
• A table for a certain microcontroller recording data out in the field
• Multiple user to access PV Data Logger information efficiently

Raspberry Pi –
• Efficient power usage
• Use of remote desktop connection for configuring program and software setting
• The expandability to allow for addition devices to become connected via the system

GPS –
• For location in decimal GPS format to be provided in documenting where data was recorded

Temperature Sensors –
• To collect the panel temperature of the solar panel when in operation
• High resolution and accuracy is of utmost importance

Humidity –
• Not of great importance to the system therefore no requirement were given from the customer

Light intensity –
• Important device for this project that requires high resolution
• No operation commercial standard pyranometer were operational during this project
• Research and provide in detail of how to construct a pyranometer for next thesis student undertaking this project

Wind Speed –
• Simple use of a switch and a timer interrupt needed

Wind Direction –
• Varying wind directions therefore no high resolution is needed

1.3 Weather system monitoring
A weather monitoring system consists of a data logger which is an electronic device, mostly based on a digital processor, that records data over time within a certain location. Data loggers are generally designed to be small, battery operated and portable to collect data in various locations. To record various locations the data is first sent via wireless radio frequency then, once collected from the client, sends the data to the server via Structured Query Language (SQL) transactions. From the transactions sent to the server, various users can access the server data via their own credentials.
Most advanced accurate weather monitoring systems costs in the thousands, therefore this project explores the possible designs that can be used to lower cost and allow future students undertaking this project to advance the system, create additional PV Data logger locations and collect larger amount of data information.

1.4 Literature Review
There are various papers relating to the use a PV data loggers around the world. Another student Mael Riou (Riou 2012) has done a project previously similar to this project but on a larger system that is not made for relocating into remote areas. As for further research into other various reports that were discovered to have similar problems, the technique used in this project was not found documented.
2 Hardware

2.1 Overview

This project is based on an Arduino Uno microcontroller and a Raspberry Pi computer. The Arduino microcontroller was used for collecting data information from weather sensor instruments as a central hub connected via USB serial to a Raspberry Pi computer for processing data into the server. Weather instruments connected to Arduino Uno are GPS Bee, four DS1820B, DHT22, TSL235R, and anemometer and wind vane. Figure 2 shows the overall hardware components used in setting the requirements of the system which this report will explore.

![Diagram of the overall PV Weather Monitoring System](image)

*Figure 2 overall PV Weather Monitoring System*
2.2 Design
This design was made by integrating various components from multiple vendors. Part 2 discusses the crucial hardware components and explains in depth the important specifications of each component used. These components include

2.2.1 Raspberry Pi
The Raspberry Pi Model B uses a Broadcom BCM2835 system on a chip (SoC) that includes an ARM1176JFZ 700MHz with 512MB RAM and VideoCore IV GPU as shown below in figure 3. The Debian Linux ARM operating system is used and is root partitioned onto a SD card of 4GB or more is recommended though for this project 16GB was used. The Raspberry Pi has significant graphics capability which is useful especially for remote desktop connection such as XRDP (this is covered further in Part 4.1 of this report). The Raspberry Pi also has very efficient processing power that appeared suitable use for SQL inserts into MS SQL server that is covered later on in section 4.3. For Wi-Fi network connection a USB Wi-Fi dongle is connected in one of the two USB ports. The important specifications of the Raspberry Pi Model B are shown below in table 1.

Table 1 Raspberry Pi Model B Specifications Summary (Raspberry Pi n.d.)

<table>
<thead>
<tr>
<th>Type</th>
<th>Single-board computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Linux (Debian GNU/Linux)</td>
</tr>
<tr>
<td>Power ratings</td>
<td>3.5W (700mA)</td>
</tr>
<tr>
<td>Power Source</td>
<td>5V via MicroUSB or GPIO header</td>
</tr>
<tr>
<td>SoC</td>
<td>Broadcom BCM2835 (with CPU, GPU, DSP, SDRAM)</td>
</tr>
<tr>
<td>CPU</td>
<td>700 MHz ARM1176JZF-S core (ARM11 family, ARMv6 instruction set)</td>
</tr>
<tr>
<td>GPU</td>
<td>Broadcom VideoCore IV</td>
</tr>
<tr>
<td>Memory (SDRAM)</td>
<td>512MB (shared with GPU)</td>
</tr>
<tr>
<td>Onboard Storage</td>
<td>SD Card slot</td>
</tr>
<tr>
<td>Low-level peripherals</td>
<td>8 GPIO, UART, I2C bus, SPI bus with 2 chip selects, I2S audio, +3.3V, +5.5V, GND</td>
</tr>
<tr>
<td>USB 2.0 ports</td>
<td>2</td>
</tr>
<tr>
<td>Video Input</td>
<td>CSI input connector for RPF camera module</td>
</tr>
<tr>
<td>Video Outputs</td>
<td>Composite RCA, HDMI</td>
</tr>
<tr>
<td>Audio Outputs</td>
<td>3.5mm jack, HDMI, I2S audio</td>
</tr>
</tbody>
</table>

Figure 3 Raspberry Pi top and general board layout (Raspberry Pi n.d.)
None of the GPIO pins are used for this project. This is mentioned later as a future improvement in this report of using TxD pin14 and RxD pin15 UART pins with an APC220 RF module. Figure 4 shows the initial state of the GPIO pins and Figure 4 shows the layout of the pins on the Raspberry Pi for Figure 5 and 6 GPIO pins shown should have been used for this project in regards to linking the RF module but was not completed.
2.2.2 Arduino Uno

The Uno is a single board microcontroller shown in figure 7 based on the ATmega328 chip as shown in figure 8. The board consists of an Atmel 8 bit AVR with a clock speed of 16MHz. The board operates at 5V with a recommended input voltage range 7-12V. Powering the Uno can be done via its USB connection or non-USB power from AC to DC adapter or battery. At the current stage of this project the Uno is powered via USB connection though for future implementation of separating the Uno from Raspberry Pi the Solar Charger Shield V2.0 will power the Uno using VIN pin via battery power charged by solar panel explained further in section 2.2.3 of the report. The ATmega328 chip contains 32KB of flash memory with 0.5KB used for the bootloader, 2KB of SRAM and 1KB of EEPROM. Six analog inputs and 14 Digital I/O pins is the total amount available to the Uno of which 1 analog and 6 digital pins are used. The analog pins provide 10 bit resolution (8 bit resolution if using digital PWM output). Below is a table summary of the Arduino Uno specifications and pin layout listing the possible functions on the Arduino Uno board and the ATmega328 chip.
2.2.3 Arduino Uno Shields

In order to allow use of various parts acquired for the prototype project three stackable Arduino Uno shields were ordered. These include connected in order to the Uno:

- Solar Charger Shield V2;
- XBee Shield V2.0;
- Prototyping Shield for Arduino.

The solar charger as shown in figure 9 allows adaptive battery power and also acts as energy harvester for remote charging. A 3.3V battery is used within the required voltage range of 2.7V-4.2V that is shifted up to 5V output to the Arduino Uno VIN pin allowing operation. This shield contains the following features:

- Short circuit protection;
- Led battery status indication (Red for charging, Green for fully charged);
- 5V via mini USB port for powering small devices like Raspberry Pi (in low power state);
- VBAT pin to measure battery voltage on any Arduino Uno analog input pin;
- On/Off power switch.
The maximum allowed current provided from the solar charger is 700mA. The Raspberry Pi and Arduino Uno could be charged by the Solar Charger Shield if the battery bank and solar charging power is increased. Though the Raspberry Pi connected to the solar charger would not be able to reach its maximum current draw of 700mA. To achieve 100mA max from Arduino Uno and 700mA from Raspberry Pi Model B the LiPo Rider Pro (Little Bird Company Pty Ltd n.d.) another type of charger similar function to the solar charger shield except it allows maximum current draw of 1A. The LiPo Rider Pro is discussed further in section 6.3 of future improvements. As mentioned in section 2.2.2 the Solar Charger shield was not able to fully become incorporated due to the USB power connection from the Arduino Uno powering the prototype system. The issues of incorporating the solar charger fully is discussed further in section 5.1 Issues Faced.

Figure 9 Solar Charger Shield V2.0 (littlebirdelectronics n.d.)

The second shield as shown in figure 10 incorporated stacked on top of the Solar Charger Shield is an XBee Shield V2.0 created by Seeeduino (Little Bird Company Pty Ltd n.d.). XBee Shield is used to allow Tx/Rx UART serial communication connection for the GPS Bee explained further in section 2.2.6. The Xbee connection uses 3.3V supply and has jumper connection points along the digital pins to select two digital pins for GPS Bee transmit or receive UART communication. UART Communications is explained later on in section 3.2.

Figure 10 XBee Shield V2.0 (littlebirdelectronics n.d.)
The third shield stacked above the XBee Shield, is the Prototyping Shield in figure 11 created by DFRobot Company as shown below (DFRobot n.d.). Benefit of incorporating this type of shield with mini breadboard allows adapting, expanding or removing parts without significant issues when only in experimental phase not production phase. Having 5V rail and ground rail creates simple connection to weather sensor instruments.

![Figure 11 Prototyping Shield for Arduino](littlebirdelectronics n.d.)

### 2.2.4 Polymer Li-Ion Battery

To allow future inclusion of remote power to the Arduino Uno, a polymer Li-Ion battery of 2000mAh 3.7V shown in figure 12 is incorporated with its specification in table 2. This type of battery was a suitable alternative to disposable alkaline batteries. The disadvantage of this type of battery is the dimensions do not allow placement inside the Solar Charger stackable shield therefore was placed below the Arduino Uno. Due to the Arduino Uno USB power this is not fully remotely powered.

*Table 2 Polymer Li-Ion Battery Summary (littlebirdelectronics n.d.)*

<table>
<thead>
<tr>
<th>Nominal Capacity</th>
<th>2000mAh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal Voltage</td>
<td>3.7V</td>
</tr>
<tr>
<td>Input connector</td>
<td>2mm JST connector</td>
</tr>
<tr>
<td>Cell Voltage</td>
<td>3.7-3.9 V</td>
</tr>
<tr>
<td>Standard Discharge Current</td>
<td>0.2C</td>
</tr>
<tr>
<td>Max Discharge Current</td>
<td>2.0C</td>
</tr>
<tr>
<td>Dimensions</td>
<td>5.8x54x54mm</td>
</tr>
<tr>
<td>Weight</td>
<td>36g</td>
</tr>
</tbody>
</table>
2.2.5 2.5W Solar Panel
This typical type of solar panel is constructed of monocrystaline silicon that performs high solar energy conversion efficiency at approximately 15%-17%. The clear epoxy coating with hard-board backing used on the solar panel gives robust sealing for outside use as shown in figure 13. The maximum voltage produced by the solar panel is 10V as shown in table 3 which matches the recommended Arduino Uno Input range as shown in table 1 Arduino Uno summary. Although if Raspberry Pi was included the solar panel would not be able to produce enough power to charge the battery and provide power to the system. Solar panel was an implementation of a future improvement creating remote system.

Table 3 Solar Panel Specifications

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>160x116x1.5mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical voltage</td>
<td>5.5V</td>
</tr>
<tr>
<td>Typical current</td>
<td>450mA</td>
</tr>
<tr>
<td>Open-circuit voltage</td>
<td>8.2V</td>
</tr>
<tr>
<td>Peak Open-circuit voltage</td>
<td>10V</td>
</tr>
<tr>
<td>Maximum load voltage</td>
<td>6.4V</td>
</tr>
<tr>
<td>Connector</td>
<td>2mm JST connector</td>
</tr>
</tbody>
</table>
2.2.6 GPS Bee kit

Global Positioning System (GPS) Bee Kit is compatible with any XBee Shield that uses the same pin connection and packaging format as the XBee wireless module. It uses a 50 channel u-blox 5 engine with 4 Hz position update rate, high immunity to signal jamming and uses UART for this system. The embedded antenna saves cost buying the GPS Bee module. The purpose for adding GPS onto the Arduino system was to allow the PV Data Logging testing module to become moved and record to various locations from the data collected for analysis. From analysis the most effective solar panel type gives the customers advantage for maximum power over time.

*Figure 14 GPS Bee Kit with mini embedded antenna (seeedstudio 2013)*
Table 4 GPS Bee kit Specification (seeedstudio 2013)

<table>
<thead>
<tr>
<th>Item</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Supply</td>
<td>2.7V</td>
<td>3.0V</td>
<td>3.6V</td>
</tr>
<tr>
<td>Peak Supply Current (Vcc=3.6V)</td>
<td></td>
<td></td>
<td>150mA</td>
</tr>
<tr>
<td>Acquisition</td>
<td></td>
<td>102mA</td>
<td></td>
</tr>
<tr>
<td>Tracking</td>
<td></td>
<td>44mA</td>
<td></td>
</tr>
<tr>
<td>Antenna Gain</td>
<td></td>
<td>30dB</td>
<td></td>
</tr>
<tr>
<td>Operation temperature</td>
<td>-40°C</td>
<td></td>
<td>85°C</td>
</tr>
</tbody>
</table>

Figure below shows the pin configuration

Figure 15 GPS Bee schematic (seeedstudio 2013)
### Table 5 GPS Bee Pin definition and Rating (seeedstudio 2013)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vcc</td>
<td>Power supply 2.7-3.6VDC</td>
</tr>
<tr>
<td>2</td>
<td>TX</td>
<td>Serial Port 1</td>
</tr>
<tr>
<td>3</td>
<td>RX</td>
<td>Serial Port 2</td>
</tr>
<tr>
<td>4</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>EX</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>11</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>TP</td>
<td>Configurable Pulse output (default 1Hz, max 4Hz)</td>
</tr>
<tr>
<td>16</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>17</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>18</td>
<td>NC</td>
<td>Reserved</td>
</tr>
<tr>
<td>19</td>
<td>SCL</td>
<td>I2C clock pin (configuration only)</td>
</tr>
<tr>
<td>20</td>
<td>SDA</td>
<td>I2C data pin (configuration only)</td>
</tr>
</tbody>
</table>

#### 2.2.7 DS18B20 Temperature Sensors

MAXIM’s DS18B20 digital temperature sensor shown in figure 16 provides temperature readings, when connected to Uno digital pin 6, from the back of the four solar panels. The resolution used for the DS18B20 connected to the Arduino Uno is 10 bit. Up to 127 devices can be connect to a single digital pin that uses 1 wire setup connecting between the 5V and signal to a 4.7k pull-up resistor explained further in section 3.3.1. There are two power methods: one is parastic power mode figure 17, when data bus is high energy is drained to charge the sensors integrated capacitors that will provide use as a power source. The second that is implemented for this project is called the normal power scheme using external supply as shown below in figure 14 and figure 18.

### Table 6 DS18B20 Specification Summary (maxim integrated n.d.)

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>3V-5V power/data</td>
</tr>
<tr>
<td>Accuracy (from -10°C to +85°C)</td>
<td>±0.5°C</td>
</tr>
<tr>
<td>Resolution</td>
<td>9 to 12 bit selectable</td>
</tr>
<tr>
<td>1 wire interface</td>
<td>1 digital pin for communication</td>
</tr>
<tr>
<td>Address</td>
<td>Unique 64 bit ID burned into chip</td>
</tr>
<tr>
<td>Query time</td>
<td>Less than 750ms</td>
</tr>
<tr>
<td>Wire setup</td>
<td>3 wires (VCC, GND, DATA)</td>
</tr>
</tbody>
</table>
Figure 16 Powering the DS18B20 with External Supply (Tushev 2013)

Figure 17 DS18B20 BLOCK DIAGRAM PARASITIC POWER (maxim integrated n.d.)

Figure 18 DS18B20 1-Wire Hardware Configuration (maxim integrated n.d.)
2.2.8 DHT22 Ambient Temperature and Humidity Sensor

Digital humidity and temperature sensor in figure 19 and figure 20 is a 3 wire sensor connected to Arduino Uno digital pin 7. Between the 5V power and signal there is a 1k pull-up resistor used to allow the PWM pulse train signal.

![DHT22 electrical connection diagram](image1)

*Figure 19 DHT22 electrical connection diagram (MaxDetect Technology Co., Ltd. n.d.)*

Main advantage using this device was its low cost of around $18 AUS and high accuracy, although collecting data takes longer, it is still within the required sampling interval of 5 seconds. Connector to the Grove DHT22 was removed and wires were soldered to each pin.

![Grove DHT22 sensor](image2)

*Figure 20 Grove DHT22 sensor (seeedstudio 2013)*

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>3-5V</td>
</tr>
<tr>
<td>Max Current</td>
<td>2.5mA</td>
</tr>
<tr>
<td>Humidity Accuracy (range 0-100%)</td>
<td>2-5%</td>
</tr>
<tr>
<td>Temperature Accuracy (range -40 to 125 °C)</td>
<td>±0.5°C</td>
</tr>
<tr>
<td>Sampling Rate</td>
<td>2 seconds</td>
</tr>
<tr>
<td>Dimension</td>
<td>15.1x25x7.7mm</td>
</tr>
<tr>
<td>Input connection</td>
<td>3 pins VCC, DATA and GND</td>
</tr>
</tbody>
</table>

*Table 7 DHT22 Specification Summary*
2.2.9 Anemometer Wind Speed Sensor
The Anemometer has three cup-type shaped paddles that move a magnet in a circular motion. Each revolution cause a reed switch to close when the magnetic field is present. As a constant wind speed of 2.4km/h it causes the switch on the anemometer to close once per second. The reed switch is connected to an interrupt on digital pin 2 with a 12k pull-up resistor between the power and signal. To stop debouncing of the reed switch a 100nF capacitor is connected across the reed switch as shown in figure 22. The two wires are connected to the wind vane via an RJ25 connection as shown in figure 21.

![Figure 21 RJ45 connection between anemometer and wind vane (Argent Data Systems n.d.)](image)

![Figure 22 Anemometer circuit with pull-up resistor and capacitor to stop debouncing](image)

2.2.10 Wind Vane Sensor
The wind vane contains 8 reed switches as shown in figure 23, each connected to a different resistor. The vane’s magnet may close 2 switches at once allowing up to 16 different positions displayed. Each separate external resistor is used to create a voltage divider, producing voltage output that is measured at analog pin A1. The resistance values for all 16 positions possible are provided in table 8 below.
### Table 8 wind vane expected results (Argent Data Systems n.d.)

<table>
<thead>
<tr>
<th>Direction (Degrees)</th>
<th>Resistance (Ohms)</th>
<th>Voltage (V=5v, R=10k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.5</td>
<td>6.57k</td>
<td>3.84v</td>
</tr>
<tr>
<td>45</td>
<td>8.2k</td>
<td>1.98v</td>
</tr>
<tr>
<td>67.5</td>
<td>891</td>
<td>2.25v</td>
</tr>
<tr>
<td>90</td>
<td>1k</td>
<td>0.41v</td>
</tr>
<tr>
<td>112.5</td>
<td>688</td>
<td>0.45v</td>
</tr>
<tr>
<td>135</td>
<td>2.2k</td>
<td>0.90v</td>
</tr>
<tr>
<td>157.5</td>
<td>1.41k</td>
<td>0.62v</td>
</tr>
<tr>
<td>180</td>
<td>3.9k</td>
<td>1.40v</td>
</tr>
<tr>
<td>202.5</td>
<td>3.14k</td>
<td>1.19v</td>
</tr>
<tr>
<td>225</td>
<td>16k</td>
<td>3.08v</td>
</tr>
<tr>
<td>247.5</td>
<td>14.12k</td>
<td>2.93v</td>
</tr>
<tr>
<td>270</td>
<td>120k</td>
<td>4.62v</td>
</tr>
<tr>
<td>292.5</td>
<td>42.12k</td>
<td>4.04v</td>
</tr>
<tr>
<td>315</td>
<td>64.9k</td>
<td>4.78v</td>
</tr>
<tr>
<td>337.5</td>
<td>21.88k</td>
<td>3.43v</td>
</tr>
</tbody>
</table>

The anemometer and wind vane are connected from the RJ45 male connector station to the Arduino Uno using RJ45 6 pin out female connector.

![Figure 23 anemometer and wind vane RJ45 pin layout (Argent Data Systems n.d.)](image-url)
3 Communications

3.1 Overview
Implementation of various communication was of utmost importance to this research. TTL serial communication was the most commonly utilized method for the Arduino. The Arduino Uno microcontroller baud rate can range from minimum of 300 to maximum of 115200 bps though the selected baud rate used was 9600 bps for this project. For the Arduino Uno to send or receive various types of data certain methods of communication were used, such as UART for the GPS module, 1-Wire for the DS18B20 temperature sensor, PWM for DHT22, frequency counting for TSL235R and the timer interrupt for anemometer. TTL serial communication and methods of communication between sensors to Arduino Uno and Arduino Uno to Raspberry Pi is discussed further below in section 3.

3.2 GPS UART Communications

3.2.1 Software Serial
The Arduino Uno has one Universal Asynchronous Receiver/Transmitter (UART) used for both operator communications and GPS communications. The purpose of UARTs is to convert data between parallel and serial forms. To provide more than one serial communication channel Software Serial was used. To use the GPS Bee acquiring location the Arduino GPS code was used and edited. The next section will explain further the type of GPS massages received and how the NEMA code was removed.

3.2.2 NMEA
The National Marine Electronics Association (known as NMEA) created the first GPS receiver (Baddeley 2001). The data structure used was GPRMC that begins with ‘$’ at the start of each sentence with comma separated values. In particular the status A for active or V for void, latitude, longitude are provided followed by with ‘*’ then checksum value in hex to finish the line. To remove unwanted values the cases linked to the unwanted values like speed were removed from the Arduino code. The GPRMC only provides the minimum amount of data that made it suitable to use for acquiring the location.

The data logger will print the following characters as feedback:

- # - String received, checksummed and written
- * - String received but there was no checksum
- ~ - string received with checksum, but checksum didn’t match
- ! – string received but the data was too large for our buffer
Once the NMEA string is collected, it is then converted into decimal GPS coordinates so the data can be displayed on a graphical map. Below in figure 24 explains the sequence taken to covert NMEA string to decimal GPS data.

- SGPRMC,220516,A,3204.00103,S,11550.15038,E,173.8,231.8,130694.004.2,W*70
- String to Float number “atof(tmplongitude.c_str())”
- Negative if South or West
- Truncate
- -32.0667,115.836 (MDU PS court yard)

*Figure 24GPS NMEA Conversion to Decimal GPS data*
3.3 DS18B20 1-Wire Communications

DALLAS SEMICONDUCTOR CORP (maxim integrated n.d.) have developed a one-wire communication bus supported by many of their products. It consists of a signal data pin connected via a pull-up resistor to the voltage supply. Multiple devices can be connected to this data line via the daisy chain setup as shown in figure 25. Advantages of 1-Wire communication provides low-speed data, signalling, and power over a single signal pin. The concept of 1-Wire is similar to I2C but with lower data rates as shown in table 9 and a longer range. The 1-Wire communication protocol has data, clock, and power all in one wire whereas I2C has one wire for data and the other wire for the clock. Typically used to communicate with small inexpensive instruments, for this project the DS18B20 1-Wire digital thermometer was connected.

![Figure 25 1-Wire daisy chain connection layout (maxim integrated n.d.)](image)

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Communication Speed</td>
<td>15.4kbps</td>
</tr>
<tr>
<td>Maximum Communication Speed</td>
<td>125kbps</td>
</tr>
<tr>
<td>Wire communication</td>
<td>Bidirectional Half Duplex</td>
</tr>
<tr>
<td>Device Configuration</td>
<td>1 Master, multiple Slaves</td>
</tr>
<tr>
<td>Time Slot (T)</td>
<td>Standard: 60µs Overdrive: 8µs</td>
</tr>
</tbody>
</table>
The DS18B20 Digital Thermometer has four main components as shown below in figure 25:
1. 64 bit lasered ROM;
2. Temperature sensor;
3. Non-volatile temperature alarm triggers TH and TL (not used and not discussed for this project);

Figure 26 1-Wire device data flow diagram (maxim integrated n.d.)

From these main components a three phase transaction process protocol for accessing the DS18B20 Digital Thermometer via 1-Wire data line is a three phase transaction as follows in figure 26 and figure 27:

Figure 27 1-Wire 3 phase transaction sequence flowchart summary
Figure 28 1-Wire timeline of the 3 phase transaction (maxim integrated n.d.)
3.3.1 1-Wire Line Initialization

Initializing communication with other 1-Wire slave devices requires a reset pulse and presents a pulse as shown below figure 29 in detail. The master being the Arduino Uno pulls the line down for a length of time to make sure all slave devices are reset. Then after reset the master pulls the data line up waiting for a response from the four DS18B20 slave devices. If the reset pulse has been collected via the slave devices the slave devices sends a presence pulse to the master.

![Figure 29 1-Wire Reset/Presence Waveforms (maxim integrated n.d.)](image)

Once the four DS18B20 device slaves are synchronised to the Arduino Uno master the data for the DS18B20 is read and written through the use of time slots to manipulate bits and a command word to specify the transaction as shown in figure 30.

![Figure 30 1-Wire signalling Read/Write bit Waveforms (maxim integrated n.d.)](image)
3.3.2 ROM Command Sequence

Each DS18B20 has a unique ROM code that is 64-bits long as shown in figure 31. The starting 8 bits consist of the 1-Wire family code for the DS18B20 is 28 (in hexadecimal) as shown below in figure 31 at the start of the 4 thermometer IDs. Following from the family code the next 48 bits are the unique serial number to identify the devices connected along the bus. For this project the DS18B20 with position and address are:

- Thermometer 1 ID 0x28, 0x05, 0x1F, 0xBB, 0x04, 0x00, 0x00, 0x7D
- Thermometer 2 ID 0x28, 0x3A, 0x17, 0xBB, 0x04, 0x00, 0x00, 0x8A
- Thermometer 3 ID 0x28, 0x60, 0x0F, 0xBB, 0x04, 0x00, 0x00, 0x29
- Thermometer 4 ID 0x28, 0x62, 0x5B, 0xA8, 0x04, 0x00, 0x00, 0x7D

The last 8 bits provide a Cyclic Redundancy Checksum (CRC) used for error checking, applied in digital networks and storage devices to detect accidental alterations to raw data. Creating a CRC requires a certain polynomial length, the longer the less probability of error detection failing, although it increases the chance of message collisions. Most commonly used CRC polynomial lengths are 9 bits (CRC-8), 17 bits (CRC16), 33 bits (CRC32), 65 bits (CRC64). DS18B20 CRC transmitted compares the CRC received from that calculated and if matched the master continues otherwise the read operation repeats.

![Figure 31 1-Wire Layout and internal characteristics of the ROM ID (maxim integrated n.d.)](image-url)
Below in table 9 is a selection of ROM-Level Commands. For this project the ROM ID is known, therefore the sequence used is to read ROM, match ROM command is used though initially to find the device address search ROM command was required. The next section as shown summarized in figure 32 will explain the search ROM process in detail.

---

**Figure 32 Summarisation of the 1-Wire Initialisation and ROM Command Sequence transaction process (maxim integrated n.d.)**

---

**Table 10 1-Wire ROM Function Commands Summary (maxim integrated n.d.)**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip ROM [CCh]</td>
<td>Skips device selection for single device on bus</td>
</tr>
<tr>
<td>Read ROM [33h]</td>
<td>Read 64-Bit ROM ID of single device on bus</td>
</tr>
<tr>
<td>Match ROM [55h]</td>
<td>Identify device on Bus with known ROM ID</td>
</tr>
<tr>
<td>Resume</td>
<td>Restart Communication with selected device</td>
</tr>
<tr>
<td>Overdrive-Skip ROM</td>
<td>Skip device selection and put device into overdrive</td>
</tr>
<tr>
<td>Search ROM [F0h]</td>
<td>Find devices on the 1-Wire bus</td>
</tr>
<tr>
<td>Alarm Search [ECh]</td>
<td>Finds if device has exceeded outside the limits of temperature high or low</td>
</tr>
</tbody>
</table>
The Search ROM algorithm requires the following sequence of events:

- Master Initiates Search ROM command;
- Master reads the 1\textsuperscript{st} bit location value (1\textsuperscript{st} read) for all device on the bus;
- Master reads the compliment of the first bit location value (2\textsuperscript{nd} read);
- Master compares 1\textsuperscript{st} read and 2\textsuperscript{nd} read;
- Master writes a bit back to participation devices (write direction bit);
- Participating devices compare the bit sent by the aster to the value they have;
  - If slaves has the correct bit value it continues to participate;
  - Otherwise if the slave does not have the correct bit value the opposite will occur.
- Master repeats the above sequence 63 more times to identify the whole ROM code of one slave device;
- Master repeats this process until all four DS18B20 devices are found.

Table 11 below explains each condition when Read 1 and Read 2 is executed on the data bus.

*Table 11 Search ROM Algorithm Lookup Table (maxim integrated n.d.)*

<table>
<thead>
<tr>
<th>Read 1 =Bit (True)</th>
<th>Read 2 =Bit (Complement)</th>
<th>Information known from Read 1 &amp; 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Condition 1: Some devices on bus have logic 1 , others have logic 0 at bit location being read</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Condition 2: All devices on bus have 0 at the bit location being read</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Condition 3: All devices on bus have 1 at the bit location being read</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Condition 4: No devices present</td>
</tr>
</tbody>
</table>
3.3.3 Memory Function Command Sequence

After completion of the ROM Command Sequence the Memory Function Command Sequence executes the Scratchpad protocol sequence for reading the scratchpad data and converting the data to a temperature reading as shown in summarized figure 31 and overall flowchart in figure 32. Table 12 summarizes what each memory command sequence performs.

Table 12 1-Wire Memory Function Commands Summary (maxim integrated n.d.)

<table>
<thead>
<tr>
<th>INSTRUCTION</th>
<th>DESCRIPTION</th>
<th>PROTOCOL</th>
<th>1-WIRE BUS AFTER ISSUING PROTOCOL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TEMPERATURE CONVERSION COMMANDS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convert T</td>
<td>Initiates temperature conversion.</td>
<td>44h</td>
<td>&lt;read temperature busy status&gt;</td>
</tr>
<tr>
<td><strong>MEMORY COMMANDS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Scratchpad</td>
<td>Writes bytes into scratchpad at addresses 2 through 4 (TH and TL temperature triggers and configuration).</td>
<td>4Eh</td>
<td>&lt;write data into 3 bytes at address 2 through 4&gt;</td>
</tr>
<tr>
<td>Read Scratchpad</td>
<td>Reads bytes from scratchpad and reads CRC byte.</td>
<td>BEh</td>
<td>&lt;read data up to 9 bytes&gt;</td>
</tr>
<tr>
<td>Copy Scratchpad</td>
<td>Copies scratchpad into non-volatile memory (addresses 2 through 4 only).</td>
<td>48h</td>
<td>&lt;read copy status&gt;</td>
</tr>
<tr>
<td>Recall E^2</td>
<td>Recalls values stored in non-volatile memory into scratchpad (temperature triggers).</td>
<td>B8h</td>
<td>&lt;read temperature busy status&gt;</td>
</tr>
<tr>
<td>Read Power Supply</td>
<td>Signals the mode of DS18B20 power supply to the master.</td>
<td>B4h</td>
<td>&lt;read supply status&gt;</td>
</tr>
</tbody>
</table>

Figure 33 Summarisation of the 1-Wire Memory Function Command Sequence transaction process (maxim integrated n.d.)
Figure 34 1-Wire Memory Functions Flow Chart Summary
The Read Scratchpad transfers the data in the 1-Wire bus least significant bit (LSB) first. The most significant bit (MSB) of the temperature register contains the “sign” bit to tell whether or not the temperature is positive or negative. In table 13 below assumes 12 bit resolution. The Arduino Uno can adjust its resolution to be configured from 9 to 12 bit resolution depending on program code configuration setup (maxim integrated n.d.).

Table 13 Temperature/Data Relationships (maxim integrated n.d.)

<table>
<thead>
<tr>
<th>TEMPERATURE</th>
<th>DIGITAL OUTPUT (Binary)</th>
<th>DIGITAL OUTPUT (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>+125°C</td>
<td>0000 0111 1101 0000</td>
<td>07D0h</td>
</tr>
<tr>
<td>+85°C</td>
<td>0000 0101 0101 0000</td>
<td>0550h*</td>
</tr>
<tr>
<td>+25.0625°C</td>
<td>0000 0001 1001 0001</td>
<td>0191h</td>
</tr>
<tr>
<td>+10.125°C</td>
<td>0000 0000 1010 0010</td>
<td>00A2h</td>
</tr>
<tr>
<td>+0.5°C</td>
<td>0000 0000 0000 1000</td>
<td>0008h</td>
</tr>
<tr>
<td>0°C</td>
<td>0000 0000 0000 0000</td>
<td>0000h</td>
</tr>
<tr>
<td>-0.5°C</td>
<td>1111 1111 1111 1100</td>
<td>FFF8h</td>
</tr>
<tr>
<td>-10.125°C</td>
<td>1111 1111 0101 1110</td>
<td>FF5Eh</td>
</tr>
<tr>
<td>-25.0625°C</td>
<td>1111 1110 0110 1111</td>
<td>FF6Fh</td>
</tr>
<tr>
<td>-55°C</td>
<td>1111 1100 1001 0000</td>
<td>FC90h</td>
</tr>
</tbody>
</table>

*The power on reset register value is +85°C.
3.4 DHT22 PWM Communications
The DHT22 uses the same 1-Wire communication except it sends 40 bits of data for each measurement to the Arduino Uno MCU. Of those 40 bits data sent from DHT22 16 bits of humidity data, 16 bits of temperature data and 8 bits of checksum are sent to the Arduino Uno. Below in figure 35 shows the whole communication process.

![Figure 35 DH22 overall communication process](image)

3.5 TSL235R Light Sensor Light to frequency
When testing the TSL235R light sensor the frequency range is from 53 Hz to 814 KHz. From the frequency counter and the change in temperature using DS18B20 was proposed to measure irradiance in W/m² for accurate measurement although full testing has not been completed. Further testing and research came to the conclusion that TSL235R simply could not provide the data resolution and rate of change in irradiance for further use.

3.5.1 Frequency Counter
In order to count the number of cycles the frequency count (PRJC n.d.) (not the frequency measure (PRJC n.d.)) Arduino library was used that could measure the required range up to 5MHz. Frequency counter measures the number of cycles that occur during a fixed gate time interval. This allows the Arduino Uno less processing, required though at lower frequencies the smaller cycles count results in limited resolution, which occurs at frequencies ranging from 53Hz to 1KHz. Due to the project requirement measuring light in the daytime low frequencies are not an issue of concern.

3.6 Anemometer Interrupt Timer
Timer interrupts are used to measure a given time interval that execute independently from the project main code. Therefore rather than running a loop repeatedly you can allow the timer to complete that task for you while other code is executed. A simple example to demonstrate this would be a LED heartbeat blinker. For this program there are two timers (out of the possible three timers) timer 1 and timer 2 on Arduino Uno digital pins 2 and 3. The Anemometer is connected to the first interrupt at digital pin 2. These timers function by incrementing a counter variable, known as a counter register. The counter size samples at 5 second intervals and when it overflows the reset is active, setting the timer value back to zero. Overflow has a flag bit to notify of the time it was triggered. Like all interrupts you can specify an Interrupt Service Routine (ISR) to execute your code...
when timer overflows. ISR automatically resets the overflow bit flag making a useful option for simplicity and speed to use like the anemometer as shown in figure 36.

![Flowchart for Anemometer Program code sequence](image)

**Figure 36 Anemometer Program code sequence**

### 3.7 Wind Vane Analog to Digital Conversion

Wind vane applies the voltage divider to measure the resistance from the wind vane, which changes direction. Using a 10K pull-up resistor the ADC equation for reading the values is

**Equation 1 ADC conversion equation**

\[ M = 1023 \times \left( \frac{R}{10000} + R \right) \]

From converting the resistance value the data received to check to see which one in the ADR variable array it is matched with to display the wind direction as a text variable. The flowchart in figure 37 summarizes to overall ADC process.

![Flowchart for Wind Vane program code sequence](image)

**Figure 37 Wind Vane program code sequence**
4 Software

4.1 Arduino 1.0.5 C++ Program Editor

The Arduino version 1.0.5 programming language based on C++ was used in order to debug and download sensor libraries for communication. In order to simplify understanding of the code used the following format in the main code was used:

1. Include function - to refer hardware code to the sub file program;
2. Define function – Gives the sub program variable its pin allocation;
3. Declare Variable types – Setting the variable initially a given data type etc. string or number;
4. Setup() function – Activates serial and defined pins when your program starts before running the program loop();
5. Loop() function – Executes code and any other functions called within the loop;
6. Functions – These are custom functions created to reduce clutter within the loop.

4.2 Raspberry Pi

4.2.1 XRDP

For windows remote connection to access the Raspberry Pi desktop securely XDRP an open source remote desktop program, was used as shown in figure 38. The purpose for using XRDP (archlinux n.d.) was for editing the Python code and allowing future software programming configurations if hardware is installed on both Arduino and Raspberry Pi. The RDP port used for the XRDP connection to user is on port 3389 (archlinux n.d.). XRDP was only installed on the Raspberry Pi to access the Linux desktop making it useful for other users who want to access that remote device with more than one user on separate remote desktop screen session window with windows remote desktop on the student computers (archlinux n.d.).

When the window session is closed without logging out, the user can access the same session again next time RDP connection is established. Although if the user accessing the remote Raspberry Pi desktop after exiting windows manager or desktop environment from the session window or the geometry/resolution is changed, the session will close and a new session will be opened. XRDP uses the Xvnc or X11rdp (archlinux n.d.) backend remote desktop protocol to present a GUI to the user (archlinux n.d.).
4.2.2 SPE (Stani’s Python Editor)
For editing python code SPE, a cross-platform integrated development environment (IDE) software was used. Operating Systems that can use SPE are Linux, Mac OS X and Windows. Key features include:

- Program editing- Syntax highlighting, auto completion, auto indentation, call tips, multiple tabs syntax checking;
- Integrated GUI designer with WxGlade program and debugger Winpdb program;
- Lower tabs contain a Shell for command line interpreter testing, identifying other python programs; Output displays printed code or any error messages that have occurred during the program execution. Recent shows python program files recently opened and notes for further commenting on the program.

Figure 38 Raspberry Pi Remote Connection start up screen
4.2.3 FreeTDS ODBC driver
Executing PyODBC software requires an adaptive ODBC driver of which FreeTDS was used for this project since it is open source. FreeTDS has been in use since the early 1997. FreeTDS stands for free tabular data stream protocol. Other programming language software such as Perl or PHP can also use FreeTDS. TDS 7.2 Microsoft version (referred to as 9.0) was used supporting for variable characters, variable binary data and extended marked language (xml) datatypes (Bruns and Lowden 2001). Once Driver was successfully installed testing of the MS SQL was done using tsql.

4.2.4 Python Open Database Connection
Python Open Database Connection (PyODBC) is a Python module that makes it possible to use ODBC allowing connection to various databases such as Windows, Linux, MAC OS X and more. PyODBC implements the Python Database API Specification v2.0 and also various other features. Using PyODBC there are four main steps for a successful SQL transaction (in order) these are connection, cursor, execute and commit. Connection strings establish the Open Database Connection (ODBC) Driver, server IP address, Username, Password and Port number as shown in Figure 39.

```python
#connection for client to MS SQL table variables
collection_string = '{DRIVER=DSN=mssql1;UID=a;PWD=a1;TDS_Version=8.0;Port=1433}'
# this sets the driver, data source name , user id, password, driver version and port need
conn = pyodbc.connect(collection_string)#open database connection
```

Figure 39 PyODBC connection string code

To test whether or not a connection to the database is possible, a function is first created to recall within the main loop program before any inserts to the database are executed. A try statement is used to test the connection of the cursor and continues to run when an ODBC connection error is caught otherwise an unsuccessful connection exits the try method.

The cursor is a pointer to where data is sent or received via the Structured Query Language (SQL). To create a cursor pointer the connection string is linked. The cursor can then be used to execute or fetch functions as shown in Figure 40 below, the use of fetch function “fetchone()” in the project code. The “fetchone()” checks to see if the SQL Server Database has the specific table. If so the Comma Separate Values (CSV) flat file contents within the execution command are inserted, otherwise a table is created and CSV flat file contents are inserted. Each time an SQL transaction is executed it must be committed or auto committed otherwise the execution will not be stored onto the Microsoft (MS) SQL Server used in this project.

```python
if cursor.tables(table='PVDFP13').fetchone(): # Check if a table exists
    print 'Yes, a table exists. Populating table with data'
else:
    # creates table
    print 'No, a table does not exist, but we will create one'
    cursor.execute("CREATE TABLE PVDFP13 (ID NOT NULL IDENTITY(1,1) PRIMARY KEY, [Data_Time] datetime, [Long"
    cursor.commit() # Must commit changes or SQL insert does not occur
rows = csv.reader(file('results.txt', 'r'), delimiters, ) #collects the data from each row to become sent to MS
for row in rows:#inserting each row into MS SQL table
    print row
    cursor.execute("INSERT INTO PVDFP13 (Date_Time, Longitude, North_South, Latitude, West_East, Temperature,
    cursor.commit() # Must commit changes or SQL insert does not occur
```

Figure 40 Check to see if table exists then insert the CSV flat file content
4.3 Microsoft SQL Server Management Studio Express

Microsoft SQL Server was used by the Raspberry Pi to execute SQL insert transactions. Fundamental reasons for using Microsoft SQL Server Management Studio Express 2005 are:

- **Manageability** – Simple use to deploy, manage, enhance organization data and analytical applications. The single management console format allow administrators anywhere within the Murdoch University Organization to monitor, manage and fine tune all the PV databases at different location and associate services across this organization.
- **Availability** – Has additional backup and restore features such as database mirroring, failover clustering, and database snapshot to minimize downtime, helping the system remain stable.
- **Scalability** – The use partitioning of large tables and indexes enhances optimizes query performance versus very large databases.
- **Security** – the security improvements include enforced policies for SQL login passwords, specifying permission through various levels in the authorization space. An example of a security improvement would be the use of Activity Monitor to check the current status of logged in users, database connecting to, status and how many open transactions as shown in figure 41.

![Activity Monitor - NATHAN-PC](image)

*Figure 41 Activity Monitor*
There are four main sections as shown below in figure 42 when first opening Microsoft SQL Server that are:

- **Registered Servers** – Located on the top left lists the available Server Engines and what there service status is currently by default one Sever engine is the local computer and others can be due to other software programs such as LabVIEW program installed on local host will use the built in Citadel Server.
- **Object Explorer** – List the file types used within the Server Engine. The main files used for this project are databases, tables, security and management. Databases identifies all database within that Server Engine. Tables is where the SQL data transactions take place for reading or writing depending on application. Security sets the users privileges and permissions on certain databases and tables.
- **Tabbed window documents** – located in the middle has tabs of file summary within that folder, SQL query editing and table editing.

![Microsoft SQL Server Management Studio initial layout](image)

**Figure 42 Microsoft SQL Server Management Studio initial layout**

Setting the SQL Server required creating a database called “PV_Dataloggin RPi” from which the tables could be created and data inserted into from the Raspberry Pi. At any time the table can be accessed and a copy at that time of the table is pulled from the server below is an example.
5 Testing

5.1 Issues Faced

5.1.1 Arduino Uno

5.1.1.1 Light sensor Interrupt timer

During configuring the TSL235R light sensor, the initial design was to use an interrupt timer to collect several pulses and convert the total amount to frequency, which would then be converted into irradiance. The problem that occurred was when the light sensor reached over 120 KHz frequency the Arduino Uno system clock stopped functioning and appeared not to be able to handle the amount of pulses from a maximum frequency of 820 KHz counted at such a fast gate interval. To resolve this issue the decision was to use a longer gate interval counter by using the frequency counter on digital pin 8 that could measure frequency ranges accurately from 1 KHz to 5MHz.

5.1.2 Light sensor measurement to temperature increase

Initially recording results collected from the light sensor without any calibration scale factor had shown signs of decrease in the number of pulses when the temperature was increased as shown in figure 43. To overcome this error affecting the irradiance results a scale factor in the positive rate of change in temperature over time with the negative rate of change in light intensity over time should give a close to zero light intensity change over time. These results were taken in an enclosed box with a constant 1000W/m^2 light source and started initially at 25°C ending up after a couple of minutes at 80.5°C approximately. The equation used is as follows below in equation 2.

\[
\text{Equation 2 Initial irradiance measurement equation}
\]

\[
\text{irradiance}(W/m^2) = \left( \frac{\text{frequency(Hz)}}{\text{area(m^2)}} \times 100 \right) \times \text{scaled \_ratio} \times (\text{scaled factor} \times \frac{\text{dht22 temperature}}{25})
\]

Scaled T ratio= the scaled ratio to the initial room temperature at 25°C

Scaled factor= the rate at which the temperature affects the frequency count
Figure 43 Frequency (Hz) versus Time (5 seconds) Light Sensor not calibrated graph

Time response to the slight change in light intensity was not evident as shown by the graph above in figure 44. This is one significant downfall that needs to be resolved. A functioning pyranometer was not able to be accessed from the university to test further the poor time response of the light sensor. Improvements to this aspect of the light sensor is discussed further in Future Upgrade Improvements section.
5.2 Raspberry Pi

5.2.1 Handling Exceptions
To test whether the ODBC connection is open or closed from the Raspberry Pi to the Microsoft SQL Server a try statement with an “except pyodbc.Error” exception handler inside is used. It is supposed to catch the error continuously if ODBC connection is closed until the Microsoft SQL Server has an active ODBC connection for the Raspberry Pi to transfer the SQL data.

5.3 Microsoft SQL Server Management Studio Express

5.3.1 Opening the specific port to allow connection access
Identifying the issues towards opening the SQL server ports for SQL transaction to occur was difficult and complicated at first. To resolve this issue a Windows Batch File called “OpenSQLPort” was used which, when executed, opened the port and added a firewall exception to allow connections from other devices through port 1433 specific to Microsoft SQL Server program within “Server and Configuration Manager”. Further details into setting the correct IP addresses, ports and firewall exceptions is provided in the appendix.
6 Future Upgrade Improvements

6.1 Freescale Semiconductor – FRDM-KL46Z Microcontroller

The Freescale Semiconductor – FRDM-KL46Z (freescale semiconductor n.d.) microcontroller would replace Arduino Uno due to the following advancements:

- 12 bit Digital to Analog Converter (DAC) with Direct Memory Access (DMA) support using the full 12 bits the DS18B20 temperature sensor has to offer
- 16 bit SAR ADC with DMA support useful when applying the MLX90614 IR Thermometer explained in detail later
- 2 I2C modules
- 3 UART modules (1 low power and 2 standard power)
- Costs approximately $14.95 over half the price of the Arduino Uno

6.2 MLX90614 IR Thermometer Temperature Sensors

The MLX90614-AAA infrared thermometer shown in figure 45 should be used for measuring temperatures as it does not require physical contact to the object surface as compared to the DS18B20 temperature sensor. The reason for this is due to the initial design of each type of PV panel may be insulated such that the required temperature of the panel is not accurately collected. MLX90614 infrared thermometer is calibrated in various temperature ranges: -40 to 85°C for ambient temperature and -70 to 382.2°C for the object temperature. Average accuracy for this system will range from 0.5-1°C when ambient temperature ranging 0-100°C and object temperature ranging -40-120°C which is well within the maximum temperature limits that were recorded with the MS SQL Server Database. Infrared ambient light and sunlight are apparently not affected due to an optical filter long-wave pass band filter that cuts off visible and near infrared light from 5.5 till 14µm mentioned on page 2 of Melexis MLX90614 datasheet provided in the appendix.

![Figure 45 the MLX90614-AAA (littlebirdelectronics n.d.)](image)

Internal 17 bit ADC and a DSP provide MLX90614’s high accuracy and resolution. The I2C bus (output method SMBus) is used on analog pins 4 as Serial Data Line (SDA) and 5 as Serial Clock Line (SCL) shown in figure 46 to make full use of the 0.02 °C highest resolution at a certain temperature range.
Both data and clock line are connected to 3.3V power via two separate 4.7KΩ pull up resistors also a 0.1µF capacitor is connected between 3.3V and ground to filter any noise.

Another option of data and clock line is the Two Wire Interface (TWI) interface (output method Pulse Width Modulated (PWM)) on digital pins 9 as data line and 8 as clock line although the disadvantage is this configuration uses 0.14°C resolution therefore not the full resolution. The main advantage of the TWI interface is its simple software configuration as compared to the I2C connection method which requires significant program configuration in order to function four MLX90614 infrared thermometer sensors constantly for the solar panels tested.

Figure 46 SMBus I2C connection to MLX90614 IR sensor (bildr.blog 2011)

Serial Management Bus (SMBus) uses communication pseudo code in the following order:

1. Send START bit;
2. Send Slave Address with read or write bit;
3. Send Command;
4. Send Repeated START_BIT;
5. Send Slave Address with read or write bit;
6. Read Data Byte Low (etc. master must send ACK bit);
7. Read Data Byte High (etc. master must send ACK bit);
8. Read PEC (etc. master can send ACK or NACK bit);
9. Send Stop bit.
6.3 Light Sensors

A similar case study in “Measuring Sunlight at Earth’s Surface” by David Brooks (Brooks 2007) describes a cost efficient process to build your own pyranometer. A typical high quality pyranometer uses thermopile detectors (groups of thermocouples embedded into special materials) that closely approach ideal radiation measurement from the sun. However the cost of acquiring such an instrument is several thousand dollars. Therefore to cut cost down the “surrogate” pyranometer consist of miniature silicon solar cell detectors. Commercial pyranometers of this type still cost in the range of $200 to several hundred dollars. The disadvantage of using solar cell-based pyranometer is that their response to solar radiation is greatly peaked in the near infrared and also does not expand along the entire solar spectrum. This is shown by Apogee Instruments, Inc., graph in figure 47 shows the response of their solar cell-based PYR pyranometer as a function of wavelength (Brooks 2007).

![Figure 47 Typical response of a silicon solar cell pyranometer (courtesy of Apogee Instruments, Inc.)](image)

Construction of solar cell-based pyranometer is fairly simple and is possible to produce them at a low cost. However, they require precise design relying on understanding of the foundation hardware principles. Key factors such as stability and calibration are always important no matter the cost. One simple way of calibrating an inexpensive pyranometer is comparing it to a commercial pyranometer.

The solar cell pyranometer is constructed from a silicon photo detector producing a small current proportional to the amount of sunlight incident on the detector. This detector measures 2x2mm housed into a clear epoxy T 1-3/4 case.

Typical photodetectors are exposed to the sunlight and measure the voltage producing approximately 0.5V even in very low light. Measuring the open circuit voltage is not proportional but measuring the flow of electrons is proportional to the amount of sun light falling onto the detector. The Arduino Uno not being able to measure current directly but using an operational amplifier would convert the current to a voltage for the microcontroller to measure. Another way would be using a simple resistor will allow the detector to apply dissipated energy across the load resistor, which therefore produces a usable voltage signal proportional to the current via Ohm’s Law shown in equation 3:
Equation 3 Ohms Law

\[ V = IR \]

One more significant design configuration for the most fundamental pyranometer is ideally the pyranometer should respond to direct sunlight proportional to the cosine proportional to zenith angle (the angle between the sun and vertical). Photodetectors never have perfect cosine response therefore to improve their response some type of sunlight diffuser is placed over the detector (Brooks 2007). The most common type of diffuser used is Teflon consisting of good spectral transmission properties hence it is mostly unaffected via long term sunlight exposure (Brooks 2007).

Figure 48 Parts to construct the Pyranometer (May 2011)

Construction Parts as shown above in figure 48:

- ABS plastic case, 3.15" x 1.57" x 0.8", DigiKey HM380-ND ($2.00/$1.66 1/10 units)
- Silicon photodetector, DigiKey Corporation PDB-C139-ND
- 470-Ohm, 1/4-Watt, 1%, metal film resistor, DigiKey Corporation 470XBK-ND
- T 1-3/4 LED holder, All Electronics HLED-4, 10 for $1.20
- Rubber or soft plastic grommet, 1/8" ID with 3/16" panel hole, McMaster-Carr 9600K17
- 1 mm (0.039") thick x 3/8" diameter Teflon® disk (punch from 6"x6" square sheet, McMaster-Carr 8545K12)

Consumables:

- Small "bullseye" bubble level, McMaster-Carr 2147A61
- Small tube of cyanoacrylate liquid glue ("superglue")
- Soldering iron
- Miscellaneous tools – pliers, wire cutters, etc.
One Example of this cost effective design constructed, calibrated using the Apogee pyranometer and tested was done by Gary Reysa results shown in figure 49.

Figure 49 Sample data of Handmade Pyranometer versus Apogee Pyranometer without amplifier (May 2011)
6.4 Wireless RF Link

6.4.1 APC220 RF module

The first radio frequency module tested was the APC220 RF module (APPCON Technologies March). Major benefits of using this type was as follows:

- Easy connection using UART/TTL interface therefore no added hardware conversion;
- Frequency ranging 418-455MHz (default 434MHz);
- 3.3-5.5V power;
- Low power consumption;
- Used for large data transfers with 512 bytes maximum data buffer;
- Long range communication of 1000 meters;
- More than 100 channels to use;
- Once set up it only requires 5 pins Enable, Tx, Rx, 5V, GND as shown in figure 50;
- Only uses 2 data lines.

Disadvantages the APC220 showed was in its lack of inbuilt error checking. So no Cyclic Radiantly Check (CRC) checksum at the end of each transferred string, lack of documentation specifically configuring the RF Frequency, no way identifying the NET ID or NODE ID through programming code and no token ring program code to prevent collisions. No USB to TTL converter was available during the project to connect and fully setup the APC220 settings with RF MAGIC software program that easily configures the device. Once the following software libraries checksum, identifying node device and token ring are created APC220 will become very useful to use in most long range applications the popular XBee that already comes with its libraries.

6.4.2 2.4G NRF24L01 + PA + LNA Wireless Module

The NRF24L01 (NORDIC Semiconductor 2008) as shown in figure 51 was the most promising wireless module providing all the built in features that the APC220 does not provide such as:

- Increased data rates when closer than 750 meters of 1-2Mbps
- Maximum range 1000m (256Kbps)
Features of Enhanced ShockBurst are:

- 1-32 bytes dynamic packet payload length;
- Automatic packet handling;
- Auto packet transaction handling;
- 6 data pipe MultiCeiver for 1:6 star networks as shown in figure 52.

The disadvantages of using this type of RF module is that it uses Arduino pins D8 as CS, D9 as CSN, D10 as SCK, D11 as MOSI, D12 as MISO and D13 as IQR. With moving the GPS UART Tx and Rx pins to the Arduino pins D3 and D4 all the digital pins are used. This is the most preferred type that would be suitable for this project if a large remote area was needed to be covered without huge software editing downtime.
6.5 Battery voltage logging
An added feature that will need to be connected when the RF modules are successfully installed is the battery status pin VBAT located on the solar charger shield shown in figure 53 must be connected to pin A0 of the Arduino type microcontrollers. This will notify the user when the battery is at low state, not charging the certain microcontroller station and is about to lose power soon so certain corrections can be taken.

![Figure 53 VBAT connected to analog pin 0 for battery voltage measurement](image)

6.6 IP Configuration
Currently the IP addresses for the Microsoft SQL Server and Raspberry Pi are not permanently set. This needs to be addressed by the Murdoch University IT personnel when this project is commissioned to remote areas for reliable SQL Server data transfer. Having dynamic IP addresses prevents the multiple Raspberry Pis to connect via the server situated on Murdoch’s internet network. Also currently, if the PV Remote Data Logger is connected outside Murdoch University network it will not be allowed to connect to the Microsoft SQL Server due to Murdoch University’s IT section not setting their router to allow these specific IP addresses access to their network.
### 7 Cost Analysis

#### Current parts used

<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity</th>
<th>Price per unit</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Uno</td>
<td>1</td>
<td>$30.00</td>
<td>$30.00</td>
</tr>
<tr>
<td>RASPBERRY-PI/8GB-USD</td>
<td>1</td>
<td>$45.00</td>
<td>$45.00</td>
</tr>
<tr>
<td>Micro USB power</td>
<td>1</td>
<td>$5.00</td>
<td>$5.00</td>
</tr>
<tr>
<td>Solar Charger Shield</td>
<td>1</td>
<td>$16.34</td>
<td>$16.34</td>
</tr>
<tr>
<td>Xbee Shield</td>
<td>1</td>
<td>$11.20</td>
<td>$11.20</td>
</tr>
<tr>
<td>Prototyping Shield</td>
<td>1</td>
<td>$5.50</td>
<td>$5.50</td>
</tr>
<tr>
<td>GPS Bee</td>
<td>1</td>
<td>$42.20</td>
<td>$42.20</td>
</tr>
<tr>
<td>DS18B20</td>
<td>1</td>
<td>$10.80</td>
<td>$10.80</td>
</tr>
<tr>
<td>Wind Vane</td>
<td>1</td>
<td>$32.50</td>
<td>$32.50</td>
</tr>
<tr>
<td>Anemometer</td>
<td>1</td>
<td>$32.90</td>
<td>$32.90</td>
</tr>
<tr>
<td>RJ45 8 pin connector</td>
<td>1</td>
<td>$2.11</td>
<td>$2.11</td>
</tr>
<tr>
<td>Jumper Wire 50-Piece Rainbow Assortment M-M 6&quot;</td>
<td>1</td>
<td>$15.50</td>
<td>$15.50</td>
</tr>
<tr>
<td>Jumper Wire 50-Piece Rainbow Assortment M-F 6&quot;</td>
<td>1</td>
<td>$15.50</td>
<td>$15.50</td>
</tr>
</tbody>
</table>

**Total Project Cost**

<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity</th>
<th>Price per unit</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total Project Cost</strong></td>
<td></td>
<td></td>
<td>283.35</td>
</tr>
</tbody>
</table>

#### Future Improvement Parts

<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity</th>
<th>Price per unit</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRF24L01 + PA + LNA Wireless Module w/ Antenna</td>
<td>2</td>
<td>$18.44</td>
<td>$36.88</td>
</tr>
<tr>
<td>Rechargeable Batteries</td>
<td>1</td>
<td>$7.00</td>
<td>$7.00</td>
</tr>
<tr>
<td>Freescale FRDM-KL46Z Microcontroller</td>
<td>1</td>
<td>$14.72</td>
<td>$14.72</td>
</tr>
<tr>
<td>Infrared Thermometer MLX90614</td>
<td>4</td>
<td>$21.95</td>
<td>$87.80</td>
</tr>
<tr>
<td>Pyranometer</td>
<td>1</td>
<td>$20.00</td>
<td>$20.00</td>
</tr>
</tbody>
</table>

**Total Cost of Future Improvements**

<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity</th>
<th>Price per unit</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total Cost of Future Improvements</strong></td>
<td></td>
<td></td>
<td>166</td>
</tr>
</tbody>
</table>

**Overall Project Cost of Parts including Future Improvements**

<table>
<thead>
<tr>
<th>Product</th>
<th>Quantity</th>
<th>Price per unit</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall Project Cost of Parts including Future Improvements</strong></td>
<td></td>
<td></td>
<td>$400.95</td>
</tr>
</tbody>
</table>

The Overall Project Cost including Future improvements factors the removal of the Arduino Uno and the four DS18B20 temperature sensor replaced with Freescale FRDM-KL46Z Microcontroller and four MLX90614 infrared thermometers. From the given budget total of $500 AUS that was allowed for this project the predicted cost of the overall project compared to the allowed budget is well within. The cost would be over budget if more than one microcontroller was to be put out into the field transmitting data to the same Raspberry Pi.

### 8 Recommendations

One future recommendation is to replace the Arduino Uno microcontroller and further increase the accuracy in resolution, especially the temperature sensors and lights sensors. Also adding the NRF24L01+ wireless module onto the Raspberry Pi and Arduino Uno type microcontroller is of the utmost importance to allow wireless communication in remote areas using wireless Radio Frequency (RF) data communication.
9 Conclusion

The PV Data Logger project has come to its completion date. Not all objectives were fully completed although all tasks were attempted. Further research at later stage gave insight and knowledge into the future upgrade improvements that have been discussed in depth.

The Microsoft SQL Server Database collection of data was achieved from the Raspberry Pi with connection of the instrument sensor devices to the Arduino Uno microcontroller as set out in project overview. All of the instrument devices except for the light sensor were not able to be calibrated to give an acceptable accurate data results.

The measured light intensity and temperature readings from the database were collected used in Excel to graph certain times of the day and calculate the difference in the rate change versus time.

The initial goal set out of the project was to produce a low cost, efficient and accurate system of which low cost and efficient was successfully achieved. In continued development of thesis projects that employ these various aspects, future Murdoch University students in the field of industrial computer systems and electrical power engineering obtain these significantly useful skills that are adapted into the workforce.
10 References


   http://home.comcast.net/~saustin98/misc/WeatherStationADC.txt.


11 Appendixes

11.1 Arduino Programs

11.2 Python Programs

11.3 LabVIEW Programs

11.4 Configuring Raspberry Pi

11.5 Parts List

11.6 Gantt chart

11.7 OpenSqlServerPort