FACULTY OF SCIENCE & TECHNOLOGY

BSc (Hons) Software Engineering
2019

Single-Axis Solar Tracking Control & Observation System

by

Mark A. MacKinnon

Faculty of Science & Technology

Department of Computing and Informatics

Final Year Project

Abstract

This report documents the steps I took to create a single-axis, solar tracking and data collection system.

The data collected is information about the Photovoltaic solar panel moving, the sun's position in the sky and the light above the system. A voltage sensor reads the solar panels voltage. Software orientates the solar panel.

It features the following:

- Its portable but currently requires an external power source for the motor.
- Controlling and observing is done via a program on a laptop, connection through USB.
- Java is the language used for the software.
- The current solar panel is of monocrystalline design and has an active surface area of 0.037267m2.

Dissertation Declaration

I agree that, should the University wish to retain it for reference purposes, a copy of my dissertation may be held by Bournemouth University normally for a period of 3 academic years. I understand that once the retention period has expired my dissertation will be destroyed.

Confidentiality

I confirm that this dissertation does not contain information of a commercial or confidential nature or include personal information other than that which would normally be in the public domain unless the relevant permissions have been obtained. In particular any information which identifies a particular individual's religious or political beliefs, information relating to their health, ethnicity, criminal history or sex life has been anonymised unless permission has been granted for its publication from the person to whom it relates.

Copyright

The copyright for this dissertation remains with me.

Requests for Information

I agree that this dissertation may be made available as the result of a request for information under the Freedom of Information Act.

Signed: _____.

Name: Mark MacKinnon

Date:

Programme: Software Engineering

Original Work Declaration

This dissertation and the project that it is based on are my own work, except where stated, in accordance with University regulations.

Signed: _____.

Name: Mark MacKinnon

Date:

Acknowledgments

Mr Ian MacKinnon for helping me assemble the hardware.

Miss Rebecca Schaller, UI/Graphic Designer, IBM, for reviewing this report and noting grammatical errors.

Mr Joe Winchester, Senior Software Engineer, IBM, for reviewing this report and suggesting the use of the system as a solar power assessment tool.

Mr Prakash Parmar, Director, Devria Limited, for reviewing this report.

Locus Energy for their Solar Calculations program which was most valuable in discerning the position of the sun.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 BACKGROUND AND CONTEXT

The migration to solar power and other renewable energy sources is essential to the preservation of our environment. If the population continues using fossil fuels as it currently does, we'll be joining the dodo bird.

To keep up with our demand for electricity, a sustainable should be looked, to reduce our impact on the environment. Solar power is a good example of sustainable energy and has proven to be successful in harnessing the sun's energy throughout the world. Solar panels collecting energy can be static or dynamic. Dynamic panels have the capacity to move. Static panels are fixed in place.

This project is intended to demonstrate a portable single axis solar panel system that can be used to track the sun and collect data about the solar activity in the location. With this information, a conclusion on whether or not to install a more permanent system, can be drawn.

1.2 PROPOSED SOLUTION

Using a combination of hardware and software tools, to write software to control the motion of a solar panel affixed to a single axis and log the following data:

a) Date/Time
b) Solar panel voltage
c) Sun's position
d) Solar panels position
e) Ambient light
f) User input

1.3 AIMS AND OBJECTIVES

My objective is to write software to control the movement of the solar panel and logs data coming from sensors.

1.4 RISK ANALYSIS

ID	Risk	Likelihood	Impact	Mitigation
1	I may be electrocuted working on the small embedded electronics system.	Low	Medium	Experience. Low voltage system. Depower the electronics before working with them.
2	Lack of suitable testing location.	Low	High	Build a portable system.
3	Drive belt breaking	Low	High	Dis-engage the stepper gear before rotating the panel manually. Only operate the stepper motor when the friction on the turntable, induced by wires, is known to be minimised.
4	The solar panel might behave differently to a commercial solar panel	Medium	Medium	Carry out research and align commercial solar panel attributes with my budget.
5	Wiring could be dislocated if rotation with enough force is applied to the turntable while the solar panel is connected to the voltage sensor.	Low	High	Restrict solar panel with software. User guide detailing set up procedure. Integrate one emergency kill switch.

2 BACKGROUND STUDY

Controlling solar panels on one or two axis with software is not a new field and has been done since the 1960's (Nimbus 1. Wikipedia, 2019).

Rotating on a horizontal axel, two solar panels went into space attached to a NASA spacecraft (Nimbus 1. Wikipedia, 2019). To align the solar panels, two sensors were used. One coarse sensor monitored large changes in the light level and the other observed the panels perpendicular offset from the sun.

Commercial use of solar panels on Earth started in the 80's. Dynamic tracking of the sun and strongest light source was done to create more electricity. The exact

A method that roused my interest was Teolan Tomson. He investigated the impact of a dual position solar tracker. By moving a solar panel to where the sun would be between sunrise and midday and then again between midday and sunset, he reported an increase yield of 10-20% over static equivalent (Tomson, 2007).

Using sensors to identify the strongest light source and align a solar panel with it like the Nimbus did, is a tactic taken by many solar tracking system designers. Electrical engineers at the Politehnica University of Bucharest in another system, monitored the light levels at two sensors divided by a small vertical shade. The system was configured to recognise a difference in values from the sensors and adjust the sensors and solar panels position till the sensors returned a indifferent signal, level of light, indicating they were both aligned with the same light source (Tiberiu Tudorache, 2010). Because sensors also collect reflected light, they are liable, if controlling a system, to redirect a solar panel ineffectively.

To conclude this section:

- solar tracking does produce a gain.
- the tilt and angle followed vary for solar trackers.
- multiple sensors are good for defining the direction to move a solar panel.
- solar technology is evolving.

3 METHODOLOGY

3.1 PLANNING

Research of solar panels and current industry/applications. I then thought about ways we can use solar panels more efficiently. It wouldn't have been possible to build a full-sized solar tracker so I built a small working prototype.

3.2 HARDWARE BUILD

I decided to build the hardware first to make sure it was going to work because I knew the hardware would drive the software's design.

3.3 SOFTWARE DESIGN

Whilst the parts arrived I acclimatised to the hardware's API.

3.4 SOFTWARE BUILD AND INTEGRATION

With the system build I started writing more bespoke code to integrate the components, testing as I went.

3.5 DATA COLLECTION

Collecting data with the system working and at intermediate stages as the program was developed. The data was viewed in Microsoft Excel.

4 REQUIREMENTS AND ANALYSIS

The systems requirements are for the;

4.1 SOLAR PANEL

- To produce a voltage
- To be of current commercial standard

4.2 TURNING PLATFORM

- To sustain the weight of the solar panel
- Enable wires to pass down through from the solar panel
- And rotate to a compass degree

4.3 ROTATORY DRIVE

- To push weight of solar panel and turning platform

4.4 SYSTEM CONTAINER

- To be portable
- Protective
- And hold the electrical components of solar tracker

4.5 VOLTAGE SENSOR

- To be able to handle the voltage created by solar panel

4.6 LIGHT SENSOR

- To point vertically up

4.7 LOG

- Have unique titles
- Be of .CSV file format so data can be graphed, analysed and exploited by Microsoft Office products, Excel, Word and Power Point.

5 DESIGN AND IMPLEMENTATION

5.1 CONSTRAINTS

5.1.1 Earth Location Sensing

Needed to calculate the solar azimuth. The system would require an integrated Global Positioning System to autonomously identify its location. I cannot afford to purchase one of these so the location of the system be included in the configuration parameters.

Example format: Latitude (50.1233), Longitude (3.9123)

5.1.2 Orientation Sensing

The system would require an electronic compass to autonomously identify its orientation relative to magnetic north. I can't afford one of these so the orientation of the system will be performed manually by the user aligning the system with South on the white dot attached to the base. A hand held or phone compass may be used for this.

5.2 HARDWARE DESIGN

5.2.1 Industry Standard

To provide a realistic platform for the experiment I used a light weight 260x180x18mm, monocrystalline solar panel.

5.2.2 Portability

The system needed to be small enough to be carried in a back pack so I could transport it between my university and lodgings. To achieve this, I purchased a cheap, portable Vinyl Turntable off eBay which came in a sturdy and small case with plenty of room for the embedded sensors, actuators and wires in the bottom.

5.2.3 Interoperability

The hardware components were chosen to integrate simply with each other and my Macbook. I found a supplier called Phidget which specialises in sensors and other hardware components for small projects. They are simple to set up. All the devices I'll need are available through their website right now including the stepper motor.

5.2.4 Connectivity

The Phidgets, name for the devices, all connect to one central hub. Connection to this hub is through a USB from the program carrier – a Laptop or PC. Power for the sensors comes through the 5V USB and the stepper motor requires a 12V power supply meaning the system will be limiting on the portability aspect of the design.

5.2.5 Costs

Solar Panel: £20

Vinyl Turntable: £10

Phidget Devices: £106

1x. Phidget Hub (PHIDGET HUB0000_0) + £40

1x. Phidget Voltage Sensor (PHIDGET 1135_0) + £17

1x. Phidget Magnetic Sensor (PHIDGET 1108_0) + £7

1x. Phidget Light sensor (PHIDGET LUX1108_0) + £12

1x. Phidget 2.5 Amp Stepper Motor and its Phidget Controller (PHIDGET STC1001_0) + £30

Extras:

Screws to fix the stepper motor and magnetic sensor to the turntables underside. £2.99

Magnet to trigger change in magnetic sensor. £3.19

TOTAL: £142.18

5.2.6 Wiring Diagram

The systems wiring is as follows:

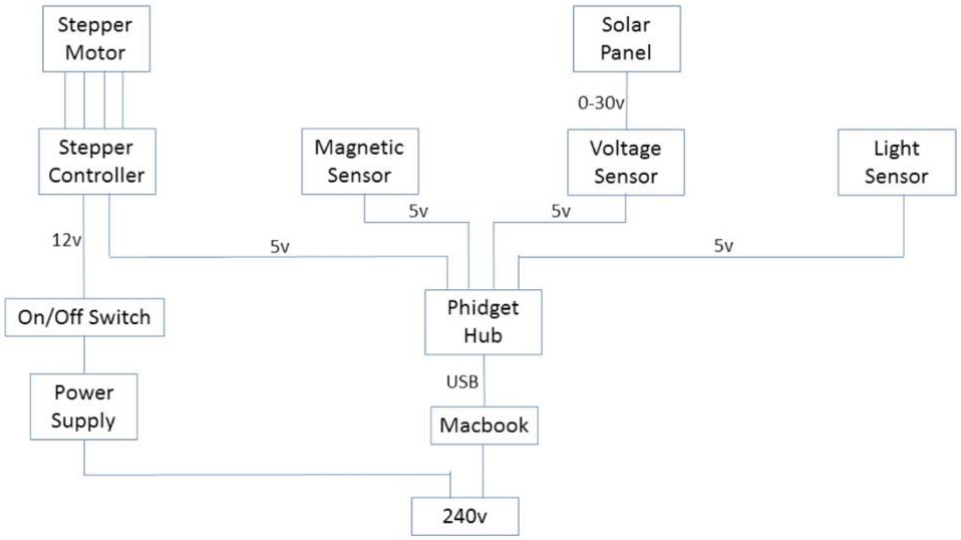


Figure 1 Wiring Diagram

5.3 HARDWARE BUILD

5.3.1 Components

1x. Phidget 2.5 Amp Stepper Motor and its Phidget Controller (PHIDGET STC1001_0) – rotating the turntables platter

1x. Monocrystalline solar panel (260x180x18mm) – mounted to the turntables platter

1x. Phidget Voltage Sensor (PHIDGET 1135_0) – measuring the output of the solar panel

1x. Phidget Magnetic Sensor (PHIDGET 1108_0) – for orientation

1x. 10x4mm Neodymium disk magnet – to induct the magnetic sensor

1x. Phidget Light sensor (PHIDGET LUX1108_0) – measuring the ambient light

1x. Phidget Hub (PHIDGET HUB0000_0) – terminal for wires to computer and Phidget devices

The rest of this section describes the design, building and testing of the system.

Figure 2 Hardware System Overview

5.3.2 Turntable

Small, portable and in a carrying case. Arriving, seemingly fully functional – I removed all electrical components beneath the unit's black upper base. Keeping the turntable, its mount and its rubber drive belt. Replacing the original motor with the Phidget 2.5Amps and running the wires for it out through the speaker hole.

5.3.3 Phidget Hub

Is the device which my laptop connects to and the stepper controller and sensors are also.

5.3.4 Stepper Motor

A stepper motor is designed to turn to a specific position and then hold that position. The main challenge during the hardware build was to attach it firmly to the turntable housing and align it exactly with the drive belt. To do this I used a small mounting bracket, drilled some holes in the turntable housing and bolted it in place.

5.3.5 Stepper Motor Controller

Taped to the inside wall of the turntable, the stepper motor controller is connected to the motor using four wires and to the Phidget Hub.

5.3.6 Stepper Motor Power Supply

The stepper motor controller requires 8-30V DC for power so an old laptop charger with a step-down transformer from the mains to 12V was integrated.

5.3.7 Voltage Sensor

The voltage sensor has 2 connectors onto which the solar panel is connected. These have to be the right way around otherwise the sensor conveys a negative voltage. It outputs a value between 0 and 1 which is a ratio of its maximum reading which is 30V.

5.3.8 Solar Panel

The maximum open circuit voltage of the panel is 21.24V.

It sits on the turntable resting at an angle of 42° which was the suggested tilt angle for 50° latitude by (Landau., 2017). Resting on a wooden block and held by two rubber plugs on the turntable. It doesn't slide off but can be removed for transport. Connected to the voltage sensor, two wires run up through the turntable and plug into an electrical plug/slip connection.

The connective wires are liable to cause damage to the Phidget Voltage Sensor should over tensioning occur. Correct set up is demanded for this to not occur and not allowing the stepper motor to over-rotate anti-clockwise or clockwise.

5.3.9 Magnetic Sensor

At start-up the system has to define what bearing the solar panel is pointing at.
To do this a magnetic switch in the form of the magnetic sensor and magnet is used.
As the magnet passes over the magnetic sensor, it alters the voltage in the sensor which when rises above 0.8 (ratio) indicates the two are aligned and the solar panel is pointing at 45°.

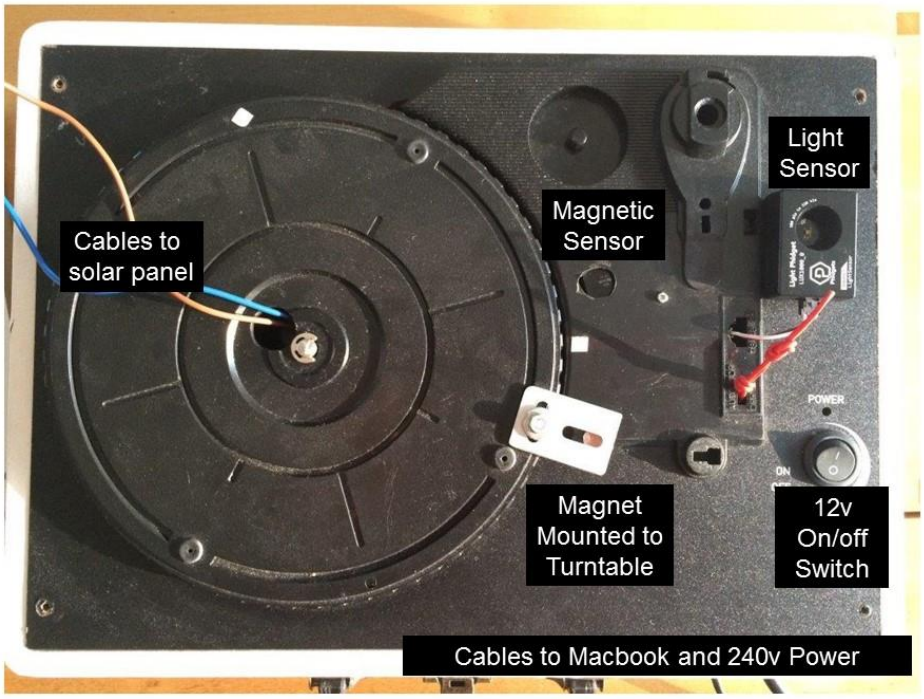


Figure 3 Turntable Showing Magnet and Magnetic Sensor

5.3.10 Light Sensor

The light sensor is fitted to the ‘South’ side of the base unit so it doesn’t get in the way of the rotating solar panel or have its signal obstructed by the sun.

It senses the number of lumens the sensor is exposed to. It has a conical shape so is less accurate when the sun is closer to the horizon.

5.3.11 MacBook

Used as the device on which software controlling the system executes and interfacing with the Phidget Hub and its peripheral devices.

5.3.12 Hardware System Hardware Internals

The hardware internals are arranged in the case as shown below in **Figure 4**.

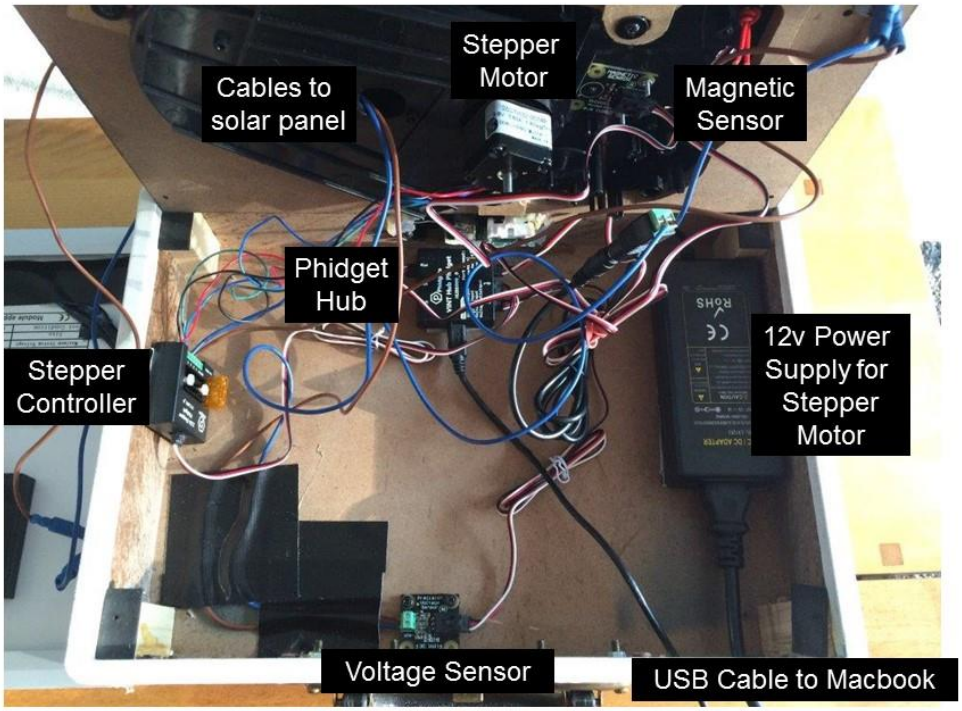


Figure 4 Hardware System Internals

5.3.13 Hardware Installation Procedure

1. Connect 12-volt power supply in the case to 240 power supply using a 13Amp plug.
2. Connect Phidget Hub to Windows or MacOS device where the software can run.

5.3.14 Hardware Start-up Procedure

Due to the design constraints mentioned above, the user has to follow the following checklist before running the program:

1. Set the power switch on the base unit to off. (Mandatory when not under software control)
2. Check the holes in the turntable and the base unit are aligned **and** the solar panel wires go straight through them and are not twisted around the turntable spindle.
3. Position the solar panel on the turntable with a long side to the base and prop it on the wooden block. The solar panel's base has to be on the side with the white dot and resting against the rubber studs either side of it.
4. Rotate the turntable clockwise to align the white dot on the turntable with the white dot referencing South for the system on the base of the unit.
5. Set the power switch of the base unit to on.

The system is now ready for automatic control. Execute the Java program when ready.

5.4 SOFTWARE DESIGN

5.4.1 Language and IDE

Phidgets are programmable in a variety of IDEs and languages but my experience is most with JAVA and Eclipse. This is compatible with Phidgets and there are code examples to become familiar with the API.

5.4.2 Downloaded Static Library

Phidgets Inc. provides a library for use with all its Phidget devices. The library is free to download and use and is called *phidget22.jar*. It's been referenced into the project Eclipse.

The library is available to download at this URL:
https://www.phidgets.com/docs/Language_-_Java_macOS_Eclipse
Date Downloaded: 25th of December 2018

Structure Detail
Root: `Phidget`
Classes are extended from this.
Example: `Phidget.Stepper` class contains functions for the stepper motor. Other devices have their own class. Methods in the root are common. Unique control requires the use of extended classes. My own use of the Phidgets will reflect and utilise this.

5.4.3 Third Party Solar Calculation Code

Locus Energy, a solar energy company, has published JAVA software to GitHub capable of calculating the Solar Azimuth, Zenith and the Sunrise/Sunset time. As requested by Locus Energy the original licence will be stored with the code.

It's available to download at this URL:
https://github.com/LocusEnergy/solar-calculations/tree/master/src/main/java/com/locusenergy/solarcalculations
Date Code Edited: 27th of March 2017.
Date Downloaded: 11th of January 2019.

5.4.4 Package Overview

There are three packages, the default/main, phidgets and solarCalculator.

5.4.4.1 main

This package contains the main class, the logging class and the configuration file (ConfigFile.txt).

5.4.4.2 phidgets

Contains classes relating to the structure and behaviour the embedded phidget system. Experimentation with phidget code examples from Phidgets Inc. showed that phidgets share some common functionality.

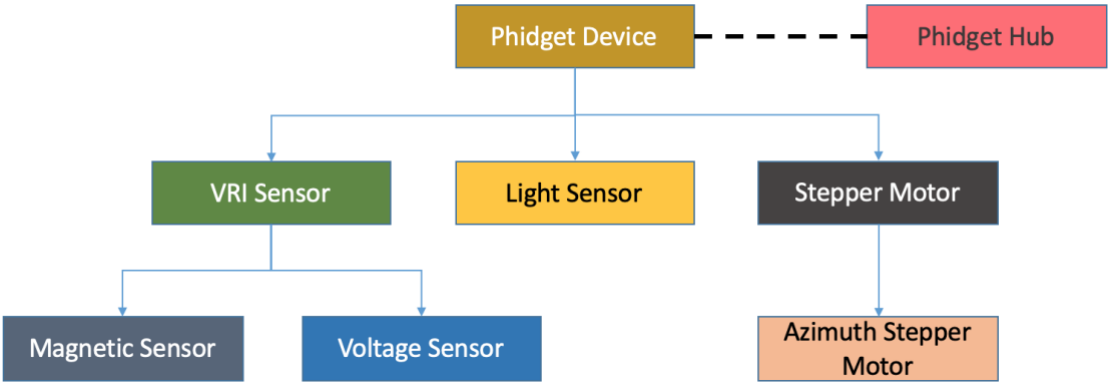


Figure 5 Phidgets Package Class Diagram

The dashed line between the Phidget Device and Hub represent the physical link that is also implemented in software to speed up trouble shooting and improve the scalability of the software by enabling multiple Phidget Hubs to be created.

Headed arrows moving down indicate extension of the Phidget Device. The Phidget Device class will hold the common methods as an abstract class, similarly, VRI Sensor and Stepper Motor will be abstracts.

5.4.4.3 solarCalculator

Holds the Locus Energy files downloaded from GitHub. The package is intended as a calculator for Solar equations. It is unmodified for future work. It contains the two classes, SolarCalculations and TrigCalc, the latter being used solely by SolarCalculations.

5.5 Software Build

5.5.1 Classes

5.5.1.1 SolarTrackerMain

5.5.1.1.1 Description:

This class holds the main method. Methods and variables are static. Because no instance of the system is made. An array of trackers would require instantiating a main class.

SolarTrackerMain					
static	private	-	configFile	:	File
static	private	-	longitude	:	double
static	private	-	latitude	:	double

static	private	-	DST	:	boolean
static	private	-	solarCalculations()	:	SolarCalculations
static	private	-	logDuration	:	double
static	private	-	logInterval	:	double
static	private	-	tracking	:	boolean
static	private	-	ph01	:	PhidgetHub
static	private	-	phDSN	:	integer
static	private	-	ASMtr	:	AzimuthStepperMotor
static	private	-	magSnsr	:	MagneticSensor
static	private	-	luxSnsr	:	AmbientLightSensor
static	private	-	vltgSnsr	:	VoltageSensor
static	private	-	ttsMtrHP	:	integer
static	private	-	magSnsrHP	:	integer
static	private	-	luxSnsrHP	:	integer
static	private	-	vltgSnsrHP	:	integer
static	private	-	separator	:	String
static	private	-	demo	:	boolean
static	public	+	main(String[] args)	:	void
static	private	-	getSolarAzimuth()	:	double
static	private	-	getSolarZenith()	:	double
static	private	-	startLogging()	:	void
static	private	-	buildLogEntry()	:	String
static	private	-	getWeatherCondition()	:	String
static	private	-	config()	:	boolean
static	private	-	configException()	:	Class extends **Exception**

5.5.1.1.2 Algorithm

1. Check the configuration file.
   a) If the configuration file is bad.
      i) Display proper configuration file format.
      ii) Terminate program - unsuccessful execution.
2. Build phidgets to be used in system.
3. Open phidget communication channels.
4. Build SolarCalculations class to exact solar position.
5. Check phidgets are attached.
   a) If not all the phidgets are attached.
      i) Close phidget communication channels.
      ii) Terminate program - unsuccessful execution.
6. Orientate the solar tracking system.

7. Align the solar panel with the solar azimuth.
8. Ask for a description of the weather to title a log file.
9. Start logging.
    a) Check all the phidgets are attached before each entry.
        i) If not all the phidgets are attached.
            1) Close phidget communication channels.
            2) Terminate program - unsuccessful execution.
        ii) If all the phidgets are attached.
            1) Check if tracking is on. Move the solar panel before every log if it is.
            2) Log data about the systems situation
10. Finish logging.
11. Close the phidget channels
12. Terminate program – successful execution.

5.5.1.2 Configuration Text File (ConfigFile.txt)

It must be next to SolarTrackerMain class (in the same directory) and be written with this format:

```
//Longitude
-1.90262
//Latitude
50.7506208
//Daylight Savings Time (boolean)
true
//Log duration (minutes)
60
//Log interval (seconds)
60
//track? (boolean)
true
```

Values entered below the lines starting with // are for the field descripted above. The headings with "//" at the start must not be altered or rearranged.

`//track?` Asks the user if the solar panel should be re-aligned with the sun for every log.

5.5.1.3 Log

5.5.1.3.1 Description:

On instantiation this.fileTitle is set with the fileTitle passed. The File is then created in the Logs directory having the fileTitle affixed with the current time and fileExtension. Successful creation opens the writer classes to be used with the logs File.

Log (*String* fileTitle)				
private	-	fileTitle	:	String
private	-	fileExtension	:	String
private	-	logFile	:	File
private	-	pen	:	BufferedWriter

private	-	fos	:	FileOutputStream
private	-	osw	:	OutputStreamWriter
private	-	getDate()	:	String
private	-	createLogFileName()	:	String
public	+	writeLineToFile(*String*)	:	String
private	-	openFileWriters(*File*)	:	void
public	+	closeWriter()	:	void

5.5.1.4 PhidgetHub

5.5.1.4.1 Description:

Every phidget has to be connected to a phidget hub which acts as the gateway through which the software talks to the phidgets.

PhidgetHub (*int* serialNumber, *boolean* demo, *String* separator)				
private	-	serialNumber	:	integer
protected	=	phidgetDevices	:	ArrayList<PhidgetDevice>
public	+	separator	:	String
public	+	demo	:	boolean
public	+	allPhidgetsAttached()	:	boolean
public	+	closePhidgetChannels()	:	void

5.5.1.4.2 Construction

The serial number of the phidget hub is unique. It was deduced by a phidget22 method and is variable in the solar tracker main passed to this constructor where it sets the phidget hubs. Variables `demo` and `separator` are used by this class and others in the phidgets package to manage the information displayed in the console

`allPhidgetsAttached()` Assesses each phidget device in the Array List to see if they're all attached. Each assessment uses a phidget22 method. Returning a Boolean value of the phidgets attachment status. After all the phidgets have been assessed. If one is not attached, three thousand milliseconds is waited before trying again. On the 5th attempt, if one phidget is still found to be not attached the method returns FALSE after displaying the names of the Phidgets not attached.

TRUE = All the phidgets are attached.

FALSE= NOT all the phidgets are attached.

`closePhidgetChannels()` Before the program terminates. This method is called which in turn calls the methods inside each `PhidgetDevice` in the Array List to close the communication channel between the Phidget hub and the Phidget devices.

5.5.1.5 PhidgetDevice

5.5.1.5.1 Description:

Phidgets have basic methods that they use and this class contains those methods. All of the methods observe and manage the connection between a Phidget (`phi`) and the Phidget hub. Because of its basic capacity, it is an abstract class which phidgets extend with more functionality, unique to their purpose. The name of every one is stored here after being parsed down. Ensuring every child is given a name.

abstract: PhidgetDevice (*PhidgetHub* ph, *String* deviceName)				
protected	=	phi	:	Phidget
protected	=	deviceName	:	String
public	+	openSetChannel(*int* channel)	:	void
protected	=	addAttachListener()	:	void
protected	=	addDetachListener()	:	void
protected	=	addErrorListener()	:	void
public	+	closeChannel()	:	void

5.5.1.5.2 Construction

On creation, deviceName is set. And because each phidget needs a hub, the Phidget hub this phidget is attached to is passed. Identifying the hub enables the PhidgetDevice to add itself to the list of devices connected to the hub. Routines in the PhidgetHub class used this list to act on all the phidgets without calling each individually for repetitive tasks.

Giving each device a name makes the phidgets easier to identify during the running of the program, when data is displayed on the console.

5.5.1.5.3 Variable: Phidget *phi*

The variable *phi*, is an abbreviation of its type, Phidget. The Phidget object is inherited by every PhidgetDevice and as the root class of the phidget22 library, it has a good polymorphic capacity which is utilised by this, the phidgets, package. For example, when the Magnetic Sensor is asked for its ratio-metric value. Its parsed to inherit the functions of the magnetic sensor.

In this class however it doesn't need to be parsed and is only accessible to phidgets classes.

5.5.1.5.4 Method: addAttachListener()

Creating a Phidget and communicating with a phidget are two different things. It requires 3 steps.

Step 1. Assign a Phidget with a hub port.

A hub port can be 0 – 5 and represents the socket the phidget is plugged into.

Step 2. Is to open a communication a channel at the port and tell the Phidget to listen there.

Step 3. Requires a phidget to be plugged in. When a plugged in and powered up an AttachEvent will occur at the port and channel and if a Phidget is listening, it can be notified.

Notification starts with the AttachEvent, onAttachEvent is called by the addAttachListener of the Phidget watching that particular port and channel. AttachListener is an interface created as an anonymous class with the addAttachListener in operation with the Phidget. The onAttachEvent method inside the interface is overridden to output custom information about the AttachEvent; what the Phidget device is called (deviceName) and the port number and channel of the attachment.

5.5.1.5.5 Extension

VRISensor, StepperMotor and LightSensor are direct children of PhidgetDevice.
MagneticSensor and VoltageSensor are VRISensor's children.

Figure 5 depicts the model well.

5.5.1.6 VRISensor

VRI is short for Voltage Ratio Input, a type of phidgets whose data is analysed as ratio. The voltage ratio input output is a double between 0 and 1 with 0.5 representing a neutral state within the sensor.

abstract: VRISensor extends *PhidgetDevice* (*PhidgetHub* ph, *int* portNumber, *string* deviceName)
public + getValue() : double

5.5.1.6.1 Construction

PhidgetDevice is passed deviceName and PhidgetHub, ph. The portNumber sets the hub port to be listened too by the PhidgetDevice's Phidget, phi and the listeners are added to the Phidget as well.

5.5.1.6.2 Method: getValue()

To get the value of the sensor, getValue() parses the super Phidget, phi to a VRI type. Then calls method, getVoltageRatio() in that class to return it to the getValue() caller.

5.5.1.7 VoltageSensor

Calling getValue(), it subtracts the base 0.5 and multiplies the difference by the maximum voltage, 30, for the sensor.

VoltageSensor extends *VRISensor* (*PhidgetHub* ph, *int* portNumber, *string* deviceName)
public + getVoltage() : double

5.5.1.8 MagneticSensor

Testing with the solar panel mounted to the turntable showed that 0.8 was the value needed to recognise when the magnet was above the sensor during the orientation phase.

MagneticSwitch extends *VRISensor* (*PhidgetHub* ph, *int* portNumber, *string* deviceName)				
public	+	getMagAligned()	:	boolean

5.5.1.9 Stepper Motor

This class carries foundational methods for a stepper phidget.

abstract: StepperMotor extends *PhidgetDevice* (*PhidgetHub* ph, *int* portNumber, *string* deviceName)				
public	+	position()	:	double
protected	=	disengageStepperMotor()	:	void
public	+	closeChannel()	:	void

Disengaging the stepper motor cuts power to it and is called from its `closeChannel()` method which overrides the method `PhidgetDevice`.

5.5.1.10 AzimuthStepperMotor

This class inherits the methods in the `StepperMotor` but because its function is unique to the project. It has methods which enable it to move the solar panel to a compass bearing and find the magnet sensor with the magnet affixed to it.

AzimuthStepperMotor extends *StepperMotor* (*PhidgetHub* ph, *int* portNumber, *string* deviceName)				
private	-	azimuthPositionalOffset	:	double
private	-	moveCounter	:	int
private	-	minBearing	:	double
private	-	maxBearing	:	double
protected	=	addAttachListener()	:	void
public	+	findMagnet(*MagneticSensor*)	:	void
public	+	moveASMToPosition(double	:	void

5.5.1.10.1 Construction

Super, StepperMotor, is passed deviceName and PhidgetHub, ph - similarly to other phidgets, addAttachListener is also called.

5.5.1.10.2 Method: addAttachListener()

Configures the Stepper phidget to operate with the load of the Solar Panel and minor frictions caused by the wires wrapped around the spindle. It also configures the motor controller to move in degrees. The ratio between the original stepping units and one degree was calculated by moving the turntable under motor, without wires running through it. 12 times around. The number of steps this took was divided by 12 and that divided by 360. A rescaling factor in the Phidget library for a

stepper is set to move by that many steps as increment of one. Once it is set, it continues to operate with the configuration for the programs duration.

5.5.1.10.3 Method: findMagnet(MagneticSensor)

To orientate the system, the magnet sensor is used because when it is aligned with magnet, the solar panel is pointing at 45 degrees from north. This constant, when assured to be current, adjusts the stepping system in the motor so future position requests will be the same as those on a compass. Holding the position the magnet is found at, is variable azimuthPositionalOffset.

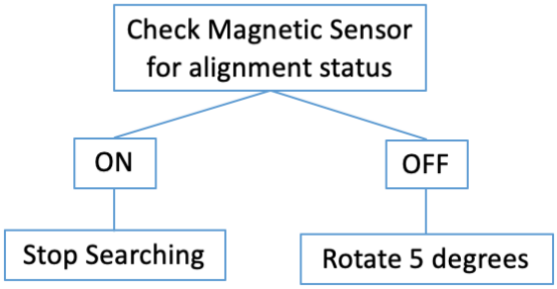


Figure 6 Magnet Alignment Procedure

5.5.1.10.4 Risk Mitigation

It is assumed that the white dots were aligned before the scan begins and the magnet/sensor are less than 180 degrees apart in the anticlockwise direction. Not searching the clockwise sector was decided because the wires cannot be guaranteed to not cause excess friction or damage the terminals when rotating the platter due to their position around the spindle cannot be garunteed.

Once the magnet is found. The method moveASMToPosition(*double* target) checks the target is not less than 40 and more than 360 degrees which can strain the system when moved beyond.

5.5.1.11 AmbientLightSensor

This light sensor data is formatted as Lux and returned when the `getIlluminance()` method is called. The data is inserted into the log file.

AmbientLightSensor extends *PhidgetDevice* (*PhidgetHub* ph, *int* portNumber, *string* deviceName)			
public	+	getIlluminance()	: double

5.5.1.12 SolarCalculations (Third-Party)

This class has the methods needed to discern the suns position.

SolarCalculations (Author: Locus Energy Inc.) (*double* longitude, *double* latitude, *boolean* DST)			
public	+	getSolarAzimuth()	: double
public	+	getSolarZenith()	: double

5.5.1.12.1 Construction

Instantiation of the class requires the longitude, latitude and Daylight savings state of the area which are extracted and passed up from the ConfigFile and through main on instantiation.

6 TESTING

To test the software, I used several methods:

- Unit testing
- Error testing
- Path testing
- Condition testing
- Clear + black box testing.

One example of unit testing is when I instantiated classes and tested their methods using stub and drive style testing. For the solar calculations I used this method to test the solar azimuth values it returned. I compared its results with online calculator results and found they were the same.

To test how the program would handle errors in the configuration file, I used error guessing and path testing. I used condition testing frequently for functional aspects of the program that used counters to return Boolean values.

Reading in the configuration file with a switch statement posed a lot of problems and for this I used clear box testing to ascertain where the problems were. Through this testing I came to the conclusion I would deny user errors and output the expected files format to them.

6.1 SENSOR TESTING

The first sensor I tested was the magnetic sensor. Sampling and displaying the Voltage Ratio Input, VRI, every 200ms I discerned what value the sensor was when the magnet was aligned. Assuring the voltage sensor was giving the right value, I used a voltmeter to test the actual signal of the solar panel and compare the reading the Phidgets reading straight after. It was conclusive that the voltage sensor worked. The light sensor was harder to prove as working because I don't have a comparable reading but it fluctuates as expected so the assumption is made it works.

6.2 ACTUATOR TESTING

When the Stepper motor arrived. I plugged it into the Phidget Control Panel which is Phidgets Inc. desktop application for interacting with Phidgets. It doesn't make code and can only be used to instantly adjust the phidgets control configuration. I used this interface to calculate how many steps were involved in rotating the turntable platter 12 times. The number returned the scale factor which I applied in the code so that every step equates to 1 degree.

7 DATA LOGS

4 Full log files are on the memory stick.

Time	Panel Voltage	Light LUX	Bearing	Solar Azimuth	Offset	Zenith
10:15:52	8.1	2924	125	124.76	-0.24	34.26
10:16:37	8.1	3042	125	124.96	-0.04	34.36
10:17:22	8.1	3043	125	125.15	0.15	34.46
10:18:07	8.1	3128	125	125.34	0.34	34.55
10:18:52	8.1	3020	125	125.53	0.53	34.65
10:19:37	7.8	2615	125	125.73	0.73	34.75
10:20:22	8.1	2918	125	125.92	0.92	34.84

Figure 7 Log Sample

7.1 LOG HEADINGS

7.1.1 Time

The time and date are logged for crosschecking values.

7.1.2 Panel Voltage

Whilst logging, a check is made of every Phidget on the system to make sure its attached so its data can be trusted.

The panel voltage is the electrical voltage being generated by the solar panel at this Time.

7.1.3 Light LUX

Light LUX is the number of lumens currently being observed by the ambient light sensor.

7.1.4 Bearing *of solar panel*

Titled “Bearing” in the log files, this represents the solar panels current compass bearing.

7.1.5 Solar Azimuth

This is calculated by the sun’s current azimuthal angle returned from Locus Energy solar calculations class.

7.1.6 Offset

This is the difference between the Bearing of the solar panel and Solar Azimuth at the time of recording.

7.1.7 Zenith

The zenith angle represents the suns elevation, 0 implies the sun is directly overhead. And 90 when it is on the horizon.

8 CONCLUSION

Figure 8 Completed System, Logging

8.1 SUMMARY

I have written the software to control a single axis solar tracking system and log its sensory data. The hardware is a portable machine that is capable of autonomous alignment with the sun.

Figure 8 depicts the system in operation, logging the solar activity in Winchester.

8.2 EVALUATION

The objective of this project was achieved by writing the software to align a solar panel with the sun and log information from sensors.

The success of this project was primarily based on the result of the systems capability to align the solar panel with the sun and log the data.

The objective to orientate a solar panel autonomously using a combination of hardware and software was achieved using phidget hardware and Java software.

8.3 FUTURE WORK

8.3.1 Commercialization

For this system to be used as a tool to calculate the solar power potential of location, I would like to add information to the log; such as the light's energy level reaching the solar panel (photonic energy) – which could be used to tailor a solar panels semi-conductive material to the location. The other factor I would like to calculate is the average solar radiation reaching the panel. With this information and the expected energy required, an estimation of the solar panel size needed could be determined.

8.3.2 Lunar Tracking

I never did capture data on the night sky ability to create electricity with my monocrystalline solar panel. Waterproofing the system would be required for this.

I would like to get some information about it. I expect a static system would be most efficient but would be happy to be proven wrong.

8.3.3 Dual Axis Tracking

Currently the hardware is not set up to take the weight of another Stepper motor. It might do but with the additional cables and mount needed, the prospect was too complex for this edition of the project. The software however would be extensible to include this extra motor. In the tropics dual axis tracking is used because the sun passes over the equator twice a year. This event is called an equinox.

In England however, far away from the tropics the sun stays in the Southern hemisphere. If we were in the Southern Hemisphere the system would still operate but I would change the compass offset to 225 and suggest starting with a clockwise scan.

8.3.4 Machine Intelligence / Learning

Using data collected I'd like to design an algorithm which moves the solar panel according to a calculation using real time data. The real time data would consist of the ambient light above the system, the position of the sun and the angle of the solar panel vertically and horizontally. The latter being a consideration of what the solar panels capacity to generate power is at offsets from the sun.

Word count (main body of the report): 6951

Word count (artefact): 5062

REFERENCES

Anon., 2012. Solar Panel Comparison. [Online]
Available at: https://energyinformative.org/best-solar-panel-monocrystalline-polycrystalline-thin-film/
[Accessed 28 August 2018].

Anon., 2019. Nimbus 1. [Online]
Available at: https://en.wikipedia.org/wiki/Nimbus_1
[Accessed 31 March 2019].

Energy Informative, 2014, Solar trackers significantly improve performance. [Online]
Available at: https://energyinformative.org/solar-panel-tracking-systems
[Accessed 29 September 2018].

GreatScottLab, 2018. DIY Miniature Solar Tracker. [Online]
Available at: https://www.instructables.com/id/DIY-Miniature-Solar-Tracker/
[Accessed 28 July 2018].

Landau, Charles R., 2017. Optimum Tilt of Solar Panels. [Online]
Available at: http://solarpaneltilt.com/
[Accessed 01 September 2018].

William Spyre, 2018. Renewable electricity capacity and generation. [Report], s.l.: s.n.

Tudorache L. K. Tiberiu, 2010. Design of a Solar Tracker System for PV Power Plants. [Report], Bucharest: s.n.

Teolan Tomson, 2007. Discrete two-positional tracking of solar collectors. [Report], Estonia: Elsevier, Department of Materials Science.

NASA, 2019. The Causes of Climate Change. [Online]
Available at: https://climate.nasa.gov/causes/
[Accessed 13 January 2019].

APPENDIX A – PROJECT PROPOSAL

Undergraduate Project Proposal Form

Degree Title: Software Engineering	**Student's Name:** Mark MacKinnon
	Supervisor's Name: Andrew Main
	Project Title/Area: Robotics/Data Acquisition

Section 1: Project Overview

1.1 Problem definition - use one sentence to summarise the problem:
Estimating the voltage that will produced from a photovoltaic cell/panel can be difficult. Using sensory information from a light sensor and a learning algorithm. The characteristics of a solar tracking system can be taught to a photovoltaic generation mechatronic unit which orientates a solar panel to generate the electricity required to sustain a load on the system.

1.2 Project description - briefly explain your project:
Rotate a fixed photovoltaic solar panel on the horizontal axis across the azimuth to collect data about when best to move the solar panel and where to move it to.

1.3 Background - please provide brief background information, e.g., client:
Solar Energy is expected to consume more of the energy market. Static and Dynamic solar arrays are both common however it is known that the motors wear out after a while and have to be replaced. To reduce downtime, efficient algorithms present themselves as obvious tools as they are flexible, adjustable and maintainable.

1.4 Aims and objectives – what are the aims and objectives of your project?
To collect data about the characteristics of solar power generation and tracking using an object orientated programming language.
To move a solar panel autonomously.
To better understand the physical properties of a photovoltaic solar panel.

Section 2: Artefact

2.1 What is the artefact that you intend to produce?
A Java program that interfaces with a stepper motor and sensors moving a solar panel so it is in line with the azimuth.

2.2 How is your artefact actionable (i.e., routes to exploitation in the technology domain)?
The methodology behind determining the rate of photovoltaic panel and solar alignment application may be of use in the solar energy generation field.

Using an ambient light sensor to assist in the estimation of the expected voltage to be produced when the sun is aligned with panel.

Other factors are expected to be useful for generating electricity efficiently with a solar tracking system so they will be recorded also.

Section 3: Evaluation

3.1 How are you going to evaluate your work?
By the demonstrating the motion of a single axis solar tracking system.
3.2 Why is this project honours worthy?
It integrates the methods taught at university and the current concern about the worlds environment.
3.3 How does this project relate to your degree title outcomes?
Software controls an actuator and sensors. Requiring planning of the systems flow as well as its contained units and their behaviours. This will require topics of conversation taught at Bournemouth University.
Software Engineering is the process of acute definition of a systems behaviour. With multiple components, all of which external objects. Control of the objects with JAVA will demonstrate my ability to systematically control a small robot.
3.4 How does your project meet the BCS Undergraduate Project Requirements?
Solar Tracking is a well understood field but writing software for a machine that does so is beyond that which I have tried to achieve before. Especially in Java. Added complexity comes from multiple sensors. This project is both investigatory and useful to the Solar Power field that is expanding to meet demands for renewable energy.
3.5 What are the risks in this project and how are you going to manage them?
Hardware design features prone to being damaged are the target for risk analysis and a report shall be included in the projects literature. Focusing on the actuator, a stepper motor which control's the direction of the solar panel connected to a voltage sensor. Mitigating this risk will employ software and a user interaction guide.
Electrocution from the wiring is another possibility but can be avoided by depowering the electrical components before work with them.

Section 4: References

4.1 Please provide references if you have used any.

Section 5: Ethics (please delete as appropriate)

5.1 Have you submitted the ethics checklist to your supervisor? **Yes**
5.2 Has the checklist been approved by your supervisor? **Yes**

Section 6: Proposed Plan (please attach your Gantt chart below)

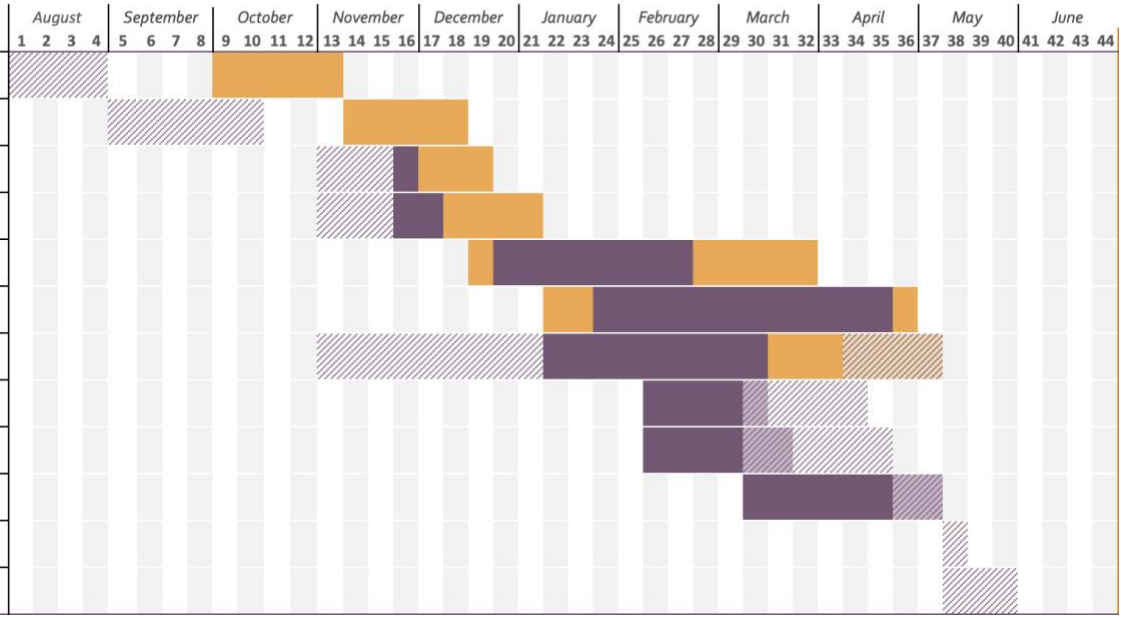
Single Axis Solar Tracking Control & Logging Plan

Select a period to highlight to the right. A legend describing the chart follows.

Period Highlight: 0

Plan Duration | Actual Duration | % Complete | Actual (beyond plan) | % Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Hardware Research	1	4	9	5	100%
Hardware Build	5	6	14	5	100%
Soft. Unit Build	13	4	16	4	100%
Soft. Unit Test	13	5	16	6	100%
Integrated Design	20	8	19	14	100%
Integration Build	24	12	22	15	100%
Evaluate	13	18	22	16	80%
Collect Data	26	9	26	5	95%
Analyse Data	26	10	26	6	80%
Remark/Conclude	30	8	30	8	80%
Submit	38	1			0%
Defence	38	3			0%



APPENDIX B – BU RESEARCH ETHICS CHECKLIST

1. Student Details

Name	Mark MacKinnon
School	Faculty of Science & Technology
Course	Software Engineering
Have you received external funding to support this research project?	No
Please list any persons or institutions that you will be conducting joint research with, both internal to BU as well as external collaborators.	

2. Project Details

Title	Single Axis Solar Tracking & Logging
Proposed Start Date	20-August-2018
Proposed End Date	05-May-2019
Supervisor	Andrew Main
Summary (including detail on background methodology, sample, outcomes, etc.)	
Writing a JAVA program in Eclipses IDE to control the motion of a Solar Panel mounted to a turntable on the horizontal plane. A logging process creating a .CSV file will output the findings.	

3. External Ethics Review (Answer “Yes” go to 4, “No” go to 5)

Does your research require external review through the NHS National Research Ethics Service (NRES) or through another external Ethics Committee?	No

4. External Ethics Review Continued

Answered "Yes" to question 3 will conclude the BU Ethics Review so you do not need to answer the following questions. Note you will need to obtain external ethical approval before commencing your research.

5. Research Literature (Answer "Yes" go to 6, "No" go to 7)

Is your research solely literature based?	No

6. Research Literature Continued (Either answer will conclude the review)

Will you have access to personal data that allows you to identify individuals OR access to confidential corporate or company data (that is not covered by confidentiality terms within an agreement or by a separate confidentiality agreement)?	No
Describe how you will collect, manage and store the personal data (taking into consideration the Data Protection Act 2018, General Data Protection Regulation (GDPR) and the Data Protection Principles).	

7. Human Participants Part 1 (Answer "Yes" go to 8, "No" go to 12)

Will your research project involve interaction with human participants as primary sources of data (e.g. interview, observation, original survey)?	No

8. Human Participants Part 2 (Answer **any** "Yes" go to 9)

Does your research specifically involve participants who are considered vulnerable (i.e. children, those with cognitive impairment, those in unequal relationships—such as your own students, prison inmates, etc.)?	No
Does the study involve participants age 16 or over who are unable to give informed consent (i.e. people with learning disabilities)? NOTE: All research that falls under the auspices of the Mental Capacity Act 2005 must be reviewed by NHS NRES.	No
Will the study require the co-operation of a gatekeeper for initial access to the groups or individuals to be recruited? (i.e. students at school, members of self-help group, residents of Nursing home?)	No
Will it be necessary for participants to take part in your study without their knowledge and consent at the time (i.e. covert observation of people in non-public places)?	No
Will the study involve discussion of sensitive topics (i.e. sexual activity, drug use, criminal activity)?	No

9. Human Participants Part 2 Continued

Describe how you will deal with the ethical issues with human participants?
N/a

10. Human Participants Part 3 (Answer **any** "Yes" go to 11, **all** "No" go to 12)

Could your research induce psychological stress or anxiety, cause harm or have negative consequences for the	No

participant or researcher (beyond the risks encountered in normal life)?	
Will your research involve prolonged or repetitive testing?	No
Will the research involve the collection of audio materials?	No
Will your research involve the collection of photographic or video materials?	No
Will financial or other inducements (other than reasonable expenses and compensation for time) be offered to participants?	No

11. Human Participants Part 3 Continued

Please explain below why your research project involves the above mentioned criteria (be sure to explain why the sensitive criterion is essential to your project's success). Give a summary of the ethical issues and any action that will be taken to address these. Explain how you will obtain informed consent (and from whom) and how you will inform the participant(s) about the research project (i.e. participant information sheet). A sample consent form and participant information sheet can be found on the Research Ethics website.
It doesn't. Although I personally may feel stress close to the deadline. I intend to cope with this by going to bed early.

12. Final Review

Will you have access to personal data that allows you to identify individuals OR access to confidential corporate or company data (that is not covered by confidentiality terms	No

within an agreement or by a separate confidentiality agreement)?	
Will your research take place outside the UK (including any and all stages of research: collection, storage, analysis, etc.)?	No
Please use the below text box to highlight any other ethical concerns or risks that may arise during your research that have not been covered in this form.	

Review Completion Date: 01-October-2018 – Double click to change it!

Supervisor's Review: Choose an item.

Please leave your comments:

Mark's Project involves only writing programs to control harmless hardware. I approve this ethics review.

APPENDIX C – CONTENTS OF USB STICK

The USB stick contains the following files:

Final Report:

Mark MacKinnon - Dissertation - Soft. Eng. 2019.pdf

Code Files:

/Code/ConfigFile.txt

/Code/SolarTrackerMain.java

/Code/Log.java

/Code/phidgets/PhidgetHub.java

/Code/phidgets/PhidgetDevice.java

/Code/phidgets/VRISensor.java

/Code/phidgets/MagneticSensor.java

/Code/phidgets/VoltageSensor.java

/Code/phidgets/StepperMotor.java

/Code/phidgets/AzimuthStepperMotor.java

/Code/phidgets/AmbientLightSensor.java

/Code/solarCalculator/SolarCalculations.java

/Code/solarCalculator/TrigCalc.java

/Code/solarCalculator/Solar Calculator MIT Licence.txt

Phidget Library File:

/Code/Library/phidget22.jar

Log Files:

/Logs/Hazed, Thin Ovc. (06-04-2019) 071501.csv

/Logs/Intermittent Clouds (05-04-2019) 082352.csv

/Logs/Patchy Thin Clouds (19-04-2019) 124510.csv

/Logs/Overcast w/ Rain (05-04-2019) 114404.csv