

Remote Web-based Monitoring of the Brewing Process

# i Preface

This report describes project work carried out within Engineering Projects at Sheffield Hallam University between September 2018 to April 2019.

The submission of the report is in accordance with the requirements for the award of the degree of insert the full name of the degree here under the auspices of the University.

# ii Acknowledgements

# iii Abstract

**Contents Page**

Contents

[i Preface 4](#_Toc4850678)

[ii Acknowledgements 4](#_Toc4850679)

[iii Abstract 4](#_Toc4850680)

[Nomenclature 6](#_Toc4850681)

[1.0 Introduction 7](#_Toc4850682)

[2.0 Literature Survey/Theory 7](#_Toc4850683)

[2.1 Choosing hardware 7](#_Toc4850684)

[2.2 Project Scope 10](#_Toc4850685)

[2.2.1 Just Add Water Kits 10](#_Toc4850686)

[2.2.2 Hobby Brewing Starter Kits 10](#_Toc4850687)

[2.2.3 Programming Languages 10](#_Toc4850688)

[3.0 Aim and Objectives 11](#_Toc4850689)

[4.0 Approach & methodology 11](#_Toc4850690)

[5.0 Project Development 12](#_Toc4850691)

[5.1 Hardware and wiring 12](#_Toc4850692)

[5.1.1 Arduino Pro Mini 12](#_Toc4850693)

[5.1.2 Raspberry Pi 12](#_Toc4850694)

[5.2 Software and Programming 13](#_Toc4850695)

[5.2.1 Arduino Pro Mini 13](#_Toc4850696)

[6.0 Project Outcomes and Progress 17](#_Toc4850697)

[7.0 Project Management 17](#_Toc4850698)

[8.0 Project Testing 17](#_Toc4850699)

[9.0 Conclusions 17](#_Toc4850700)

[9.1 Taking the Project Further 18](#_Toc4850701)

[10.0 References 18](#_Toc4850702)

[1.0 Appendix Figure 1 showing the pinout and wiring of the whole project. 22](#_Toc4850703)

[2.0 Appendix Fig.2 showing a 2-channel relay board used for switching mains supply 23](#_Toc4850704)

# Nomenclature

This page shows a table with common abbreviations and their meanings.

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| V | Volts |
| UART | Universal Asynchronous Receiver-Transmitter |
| Rx | Receive (when discussing UART connections) |
| Tx | Transmit (when discussing UART connections) |
| GPIO | General Purpose Input Output |
| PCB | Printed Circuit Board |
| LED | Light Emitting Diode |
| IDE | Integrated Development Environment |

# 1.0 Introduction

This project brings together the work that was done from year one, the making of a temperature sensor, and year two, the making of a web hosted video camera, some elements of year three and some external content too. Doing this project builds on my understanding and knowledge of topics that were studied done throughout my university career.

When hobbyists or industrial brewers of alcohol brew, temperature monitoring, especially for hobby brewers, is done manually via dipping a thermometer into the liquid mixture this poses a variety of issues the major issues concern health and safety because during the boiling of the brew mixture the temperature can get up to 100 degrees Celsius which can cause serious burns. Furthermore, if the brewer is a micro-brewery or industrial supplier may have very large vats of liquid mixture and during a measuring session there may be a person stood on a ladder or walkway having to lean over such a boiling vat, this risks serious injury. While this works, it is hoped that this project will replace this manual method with something more up to date by using a microcontroller and a single board computer to make this process automated. Also, to help brewers know what to do and when, the system incorporate a timing system that will deliver email-based alerts so that brewers know when to take the next appropriate actions.

# 2.0 Literature Survey/Theory

The process of brewing alcohol has three main stages the first of these stages being called the mash and the second being called boiling and finally there is fermentation were the brew sits in an insulated environment for an extended period of time what the yeast performs anaerobic respiration, producing alcohol in the process, completing the beverage.

## 2.1 Choosing hardware

This project will be constructed using an embedded system and a single board computer. For the choice of programmable microcontroller for the base of the embedded system a prebuilt solution on a premade printed circuit board (PCB) was chosen due to these being generally similar in price to the micro controller chips on their own and require less manufacturing and tooling costs. Furthermore, this approach cut large chunks of time off the development process and allowed the focus to remain on building the project instead of reinventing things that were already readily available cheap products.

Initially looking at the specifications for two major contenders, the STM32F103C8T6 (Ali Express, 2010) would be a suitable development platform as Table 1 shows it has one of the better clock speeds and better memory capacities. Any of the boards to be chosen could be directly soldered too as to reduce the form factor of the probe. Direct soldering means that the length of cable the microcontroller has from the Raspberry Pi 3 (Raspberry Pi Foundation, 2019) can be determined by the distance from the brew to the Raspberry Pi 3. For this project however, it was decided that the Arduino pro mini a 3.3v (EBay, n.d) was to be used, Table 1 shows it is a low power low cost option and it is a platform that has the greatest familiarity surrounding it also there are a wide variety of libraries available on the Arduino playground (Arduino, 2018). These libraries can be used for a wide variety of projects and some have been used for reading the input from the DS18B20 and can easily be interfaced with the Raspberry Pi 3 (Raspberry Pi Foundation, 2019). The Arduino Pro Mini board was chosen over the Arduino Uno (Arduino, 2018) as it is physically smaller, cheaper and runs on 3.3v whereas the Arduino Uno runs on 5v increasing the power usage over time and meaning that the project would require two power outlets to run instead of one. The downside to using the Arduino Pro Mini is that it has a fixed clock speed of 8MHz, however this is not an issue as I only need it to act as a temperature sensor in my project prototype.

|  |  |  |  |
| --- | --- | --- | --- |
| Specification | Arduino Pro Mini | Arduino Uno | STM32F103C8T6 |
| Size L,W (mm) | 33.02,17.78 | 68.6,53.4 | 22.86,53.34 |
| Memory (kb) | 32 | 32 | 64 |
| Clock Speed (MHz) | 8 | 16 | 72 |
| Price (£) | 1.5 | 17.04 | 1.38 |

Table 1 showing relevant specifications for potential microcontroller boards.

Having chosen the platform for a microcontroller, a decision needed to be made about the single board computer to be used. From research, there are three main competitors on the market. The three main competitors are the Intel Compute Stick (Intel Corporation, 2016), the BeagleBone Black (Beagleboard.org Foundation, 2018), and the Raspberry Pi 3 (Raspberry Pi Foundation, 2019). Having never used the Intel compute stick and the BeagleBone Black and having vast amounts of experience with multiple iterations of the Raspberry Pi experience leans towards what a larger experience and knowledge base. While the Intel compute stick runs on windows and the two other boards run on Linux. The Intel compute stick was eliminated due to the lack of knowledge and user experience around using it and developing projects with it. Furthermore, due to the large price tag of £99.33 the Intel Compute Stick made the cost of this project go way out of sensible range. This left the BeagleBone black and the Raspberry Pi 3 - between these two boards there is a lot of difference as shown in Table 2 such as the available random-access memory (RAM) with the Raspberry Pi having double that of its counterpart the Beaglebone Black. However, the cost of the BeagleBone black for lesser specifications to the Raspberry Pi was not something that could be justified. The Raspberry Pi Foundation also offer a camera module (Raspberry Pi Foundation, 2016) that will be used as it is a plug and play camera that can be used without additional setup on the Raspberry Pi 3. As well as the addition of the camera, the Raspberry Pi also comes with its own bespoke operating system (Raspberry Pi Foundation, n.d) that can be downloaded for free and is tailor made for the Raspberry Pi. Due to the reasons laid out above and the costing of the two remaining single board computers where, the Raspberry Pi retails at £32 (The Pi Hut, 2019) and the BeagleBone Black retails at £68.99 (Premier Farnell Limited, 2018), this led to the choice of the Raspberry Pi 3 being the chosen single board computer for my project.

|  |  |  |  |
| --- | --- | --- | --- |
| Specificaion | Raspberry Pi 3 | Beaglebone Black | Intel Compute Stick |
| Physical Size ( width mm, length mm) | 85.60, 56.5 | 86.40 × 53.3 | 103 × 37 |
| Processor Name, Speed (GHz) | Quad-core ARM Cortex A53, 1.2 | AM335x 1GHz ARM Cortex-A8 | Intel Core M processor, Unknown |
| RAM (Mb) | 1024 | 512 | 2048 |
| Storage Size (Gb) | Expandable as it uses micro SD cards | 4GB | 32 |
| WiFi enabled  (Yes/No) | YES | NO | YES |
| Price (£) | 32 | 68.99 | 99.33 |

Table 2 showing the available specifications for the 3 single board computer options that are to be reviewed

The temperature sensor that will be being used for this project is the DS18B20 (EBay, n.d) a one wire temperature sensor that often comes prebuilt in a waterproof housing with a long cable attached that can be directly soldered onto the microcontroller board even the prebuild waterproofed sensors only cost a few pounds including shipping and handling and are accurate to within half a degree Celsius this is enough for this project. The sensor that was purchased cost £2.45 and was shipped for free with no customs charges. Manufacturing the sensor especially for this project would not be time efficient and would not be very cost effective. Due to the parts for building such a sensor would cost more than that of one that was mass produced and sold online.

## 2.2 Project Scope

### 2.2.1 Just Add Water Kits

The “just add water kits” such as the beer buddy kit (Young, n.d) will not be discussed or explored because these kits do not conform to what my project was originally designed to do. This is because these kits just require the user to add warm water and wait for the fermentation stage to complete. My project is designed to monitor the whole brewing process from before the fermentation stage all the way to the finished product. However, while these kits do require some temperature regulation and my project could be used to monitor or regulate the temperature of these brews while they ferment, there won’t be a discussion just add water kits as they are not the intended end target.

### 2.2.2 Hobby Brewing Starter Kits

Starter kits come with all the needed parts to start brewing alcohol, such as the full equipment style of kits from the Home Brew Shop (The Brew Home Shop, 2019) but do not come with anything other than a manual way for measuring the temperature and automated kind of temperature regulation is absent. This is where the project will come in and replace this manual method of temperature acquisition. The focus of the project will be on hobby and micro brewers due to their need of a more automated and simplified process for information and process monitoring. (Why?)

### 2.2.3 Programming Languages

Having done research about the programming languages that are being used, it was found that Python (Python Software Foundation, 2001) for the temperature update code was a bad choice as it overcomplicates the update system to the webserver and adds an unnecessary language to the project. Instead, JavaScript was used in Python's place - this then extended into using JavaScript to create the webserver and manage the serial communications with the Arduino and handling the webserver call to send emails. Following these findings, a technique called AJAX was used in order to create a timer based updating control for web page elements both automatically and on user input.

# 3.0 Aim and Objectives

The main aim of this project is to produce a system that can be used to monitor the brewing process remotely from a web page via a temperature sensor and a web enabled camera.

Objectives for this project are:

1. Create a working temperature acquisition system
2. Setup Raspberry Pi for data acquisition from the Arduino system
3. Create a completed automatically updating web page
4. Create a completed automatically updating web page with video feed
5. Create a completed web page with a working web server solution (prebuilt or custom) with port forwarding network permissions permitting
6. The completed web page is hosted locally
7. To create an email-based update system to notify the user of impending changes that need to be made or problems that need solving time permitting an SMS system could be implemented also.
8. Time permitting a relay and a heating element could be added to be able to fully automate the temperature regulation of the brew
9. Time permitting create a light source for the camera that will automatically switch on in dark environments.

# 4.0 Approach & methodology

Methodology develop a computing system talk about the raspberry pi as a webserver port forwarding and dns etc

# 5.0 Project Development

## 5.1 Hardware and wiring

The main development style that was followed throughout the building of this project was a step by step process that included elements of work that were known and elements of work that were not known.

### 5.1.1 Arduino Pro Mini

This projects work started with the design and building of the temperature probe the parts required for this section of the build were the DS18B20 the Arduino Pro Mini with a six pin male header and two, two kilo-ohm resistors, or a four kilo-ohm resistor if one was available however this was not the case for me so I used the combo of resistors stated above. This building process was fast and simple and required little skill the first step was to solder all of the components to the Arduino Pro Mini the six-pin header was soldered to the six through hole solder points on the UART connector. Following this the DS18B20 was soldered to the 3.3v power, ground and the analogue A0 pin finally making sure to solder the four kilo-ohm resistor across the power and data lines, this was initially forgotten and cost a few hours in programming time whilst figuring out what was the issue with the hardware. During the programming of the Arduino Pro Mini there were a number of mistakes that were overcome the first of these mistakes was not using the two libraries that were needed to even get data from the sensor in the first place.

### 5.1.2 Raspberry Pi

While the Arduino Pro Mini require some wiring up the Raspberry Pi only needs some basic wiring as shown by Figure 1 in 1.0 Appendix. The first thing completed was the UART connection between the Raspberry Pi and the Arduino Pro Mini. This required four wires one for 3.3v input, one for ground, one that connects the Rx of the Arduino Pro Mini to the Tx, GPIO pin 14, of the Raspberry Pi and a final wire that connects the Tx of the Arduino Pro Mini to the Rx, GPIO pin 15, of the Raspberry Pi. Following this there were two LEDs added to a breadboard with 330-ohm resistors in series with each of the LEDs. These LEDs are there to simulate the activating of the relay block. Finally, the Raspberry Pi Camera () module was connected to the camera serial interface port.

## 5.2 Software and Programming

Following on from the discussion about the hardware side of this development this section details the software side of this development and the actions taken towards this development.

### 5.2.1 Arduino Pro Mini

The Arduino Pro Mini’s code was written in the Arduino IDE (Arduino, 2018)

Having attempted to write the code for the Arduino several times and initially not realising that I needed to use two libraries the OneWire and DallasTemperature libraries. During early iterations of the design of the code I used the Delay() function to control the timings of the program however, as the design became more sophisticated and more features were added this function became difficult to use and was replaced with a new method of timing control. The reason that the Delay() function was replaced was due to the nature of its working, as the delay function counts out its allotted time it holds up the entire microcontroller, making timing for multiple loops of the same program incredibly difficult or impossible. I then created a state machine to separate the tasks the Arduino had to complete and make timing these tasks easier. I began using the millis() function, the millis() function does not use the whole microcontroller like the Delay() function does, as a basis of an if statement to control the intervals at which individual tasks happen. Once the working code was uploaded to the Arduino, I set about setting up the Raspberry pi. This required using 2 pieces of software that can take some time to run up to five to ten minutes. As the Raspberry Pi 3 runs its operating system from a micro SD card I decided to use a 32GB card this is far more space than will ever be required by my project however, it means that I can avoid worrying about running out of space during any of my build stage. Having chosen an SD card, I then wiped it using a program called SD card formatter (SD Association, 2018) fully wiping the card like this removes any data on the card and sets all the space into one partition so that when we write the Raspbian disk image (Raspberry Pi Foundation, 2018) to the SD card there won’t be any problems. Having written the operating system to the SD card I then setup the Raspberry Pi for first time boot and after booting and logging in with the username pi and the password raspberry I set about setting up the operating system for my project the first commands that should always be run when logging into the Raspberry Pi are the sudo apt-get update and the sudo apt-get dist-upgrade -y commands the first of these commands updates the Raspberry Pi’s package lists to the latest version this allows the Raspberry Pi to find the latest packages and their updates if there are any the second of these commands updates the out of date packages present on the system. I then set about uninstalling unnecessary applications such as the office suite that comes with Raspbian, I did these in batches in order to minimise error and keep the processing time per command low. I ran the following uninstall commands to uninstall the following programs and remove their config files as I won’t be using them and they’re cluttering up the operating system the final command that I ran out of these three was to remove and dependency packages that were now on the system and were redundant. The following three bullet points are the commands that I used in the console.

* sudo apt-get purge minecraft-pi wolfram-engine scratch -y
* sudo apt-get scratch2 libreeoffice\*
* sudo apt-get autoremove

The next step was to set the Raspberry Pi up for my project and there were two pieces of software that I wanted to install Apache2 webserver software (The Apache Software Foundation, 1997) and Motion4.0 (Motion Project, 2018) a security camera streaming software. I also set the Raspberry Pi’s local IP to be static with help from a guide on the internet (ModMyPi LTD, 19 April 2016) as this greatly helps during and port forwarding and any testing that I wanted to do. The following two bullet points are the install commands that I used in the console.

* sudo apt-get install apache2
* sudo apt-get install motion

After installing both of these I performed some first time use tests by opening a browser and using the Raspberry Pi’s local IP address in the address bar, in my case 192.168.1.128, this brought up the Apache2 welcome page that I will be changing out later. With Apache2 working I turned my attention to Motion having set up Motion before I knew there were a couple of things I needed to do the first of which is to tell the Raspberry Pi that to use the camera module that I own so I opened the etc/modules file and added this the camera that I was using at the end of that file I then saved my changes and closed the file. I then changed to editing the config file for motion and then changed the daemon setting from off to on this allows motion to start on boot and run in the background. The next setting, I needed to change was the resolution of the camera that I was using from the default to 768 wide by 1024 high next I changed the framerate from 1 to 60 this is so that the viewer of the stream gets a clear and smooth video feed. One of the major features of motion is its ability to capture images of things when they move, as part of its security side of the package, as I don’t want this to happen due to me not wanting the SD card to run out of space, I switched this setting off in the config file. In order to have the stream hosted within the local network I need to have a it hosted on an open port and have the web control setup on an open port too for the stream I selected port 8081 and for the web control I selected 8082 I then changed the local host setting from off to on saved and exited the config file and restarted motion using sudo systemctl restart motion. The following bullet points show the changes made in to the config files talked about above.

* sudo nano /etc/modules
  + adding “bcm2835-v4l2” to the end of the file
* sudo nano /etc/motion/motion.conf
  + changing daemon off to daemon on
  + changing the height and width properties to match the cameras height 768 width 1024
  + changing the framerate from 1 to 60
  + changing output\_pictures on to output\_pictures off
  + changing stream\_port 0 to stream\_port 8081
  + changing stream\_localhost off to stream\_localhost on
  + changing webcontrol\_port 0 to webcontrol\_port 8082

Now that the web services are setup, I opened Geaney Brush Matthew, Hopf Dominic, Lanitz Frank, Treleaven Nick, Tröger Enrico and Wendling Colomban. (2006), one of the raspberry pi’s IDEs, to write the code for the website in two files the index.html file contains the main HTML code for the webpage detailing the main objects on the page and the files that are linked with it. The second file called MainPage.css this file details the details of the content on the webpage such as the position, size and colours. After coding the HTML and CSS files were moved to the /var/www/html folder where the test index.html was replaced with my webpage’s index.html currently the page and stream are accessible on a local network only with the webpage on port 80 hosting the stream in an iframe. Now that the webpage was as set up as it could be I started on writing the code for interfacing the Arduino and the Raspberry Pi through a UART connection using the Raspberry Pi’s GPIO (general purpose input and output) pins and the Arduino’s pre setup connection initially this took a little trial and error. First I connected the raspberry pi’s 3.3v supply (gpio pin 1) ground (gpio pin 6) TxD (gpio pin 8) and RxD (gpio pin 10) to a breadboard for easy prototyping of circuitry initially and incorrectly I connected Tx and Rx directly to the respective pins on the Arduino however at closer inspection the Tx of the Arduino needed to be connected to the Rx of the raspberry pi and the Rx of the Arduino to the Tx of the raspberry pi. Following this I went into the raspi-config menu with the command sudo raspi-config and went to option 5 interfacing options then to option P6 serial and disabled console over serial option but kept the hardware enabled. After setting up the required hardware options I wrote some code that I adapted from (Instructables, n.d) see appendix code listing 5 and tested it, where it failed, I then ran the command ls –l /dev to see what port I was using and the name of that port in this case I was using serial0 which is /dev/ttyS0 instead of /dev/ttyAMA0 which was on the instructables article the code for getting my temperature readings from the Arduino into the Raspberry Pi now worked. After doing some research into JavaScript, to become more informed upon how to make webpages more interactive, I found that having written the serial communications code in Python I made my project overly complex so I set about researching a better and more streamlined solution and came back with writing the whole thing in JavaScript and running it on the webserver. In order to make a start on this changeover I needed to install node.js (), which is a JavaScript webserver package, having done some work on other areas of my project over the recent days I ran the update commands again and then ran the command to install node.js and ran a verification command to make sure that it was installed correctly. The next two bullet points are the commands I used to install and verify node.js.

* Installing node.js for raspberry pi by running the upgrade commands sudo apt-get update && sudo apt-get dist-upgrade -y then running sudo apt-get install -y nodejs
  + running node -v verifies the version and that the install went according to plan

Following this installation, I wrote a proof of concept test code in node.js using the serialport module and a tutorial (w3schools, 1999) the code takes data from the serial port and writes it to the console this code worked first time as I used the ideas and concepts that I had learned while writing the python code.

This proof of concept allowed me to move forward with the building of the new webserver in node.js. I built this webserver initially using code from W3schools () on port 8080 this code then immediately replaced the apache2 webserver solution that was currently in place. To get this working on boot I edited the crontab file and rebooted as I could still access the web page, I knew that my code was working.

Work plan for the second semester

1. Write as I work. Write the Reports and presentations as I go so as not to forget anything
2. Build the new web server and get it to replace the current server (apache)
   1. in order to use the node.js with the pi’s gpio we need to install the onoff module this was done with the npm install onoff command
   2. in order to host a webservice that we can interface our AJAX and serial port javascript to we need to install socket.io this is done with the command nmp install socket.io –save
   3. (follow on with code to create webserver use ajax and node.js to create and updateable webpage that displays the data.)
3. Build the website up so that it includes the JavaScript elements, the updating graphs and incorporate the GPIO code.
4. While keeping motion installed and the primary camera streaming method attempt to make a new camera streaming method.
5. Tidy up the scripts that have been written and get everything to run on boot
6. Run a testing process on the system for local connections.
7. Attempt to find a method of getting the internet as current accommodation doesn’t allow port forwarding. Find out if the university network lab will let me test port forwarding.
8. Find a method of creating or using a ready-made solution for an emailing service and test it.

# 6.0 Project Outcomes and Progress

Project was completed talk about the learning outcomes (bitten off more than could chew however made it through)

# 7.0 Project Management

GANNT CHARTS

# 8.0 Project Testing

The testing procedure for this project is described in the table below each test is numbered.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | Test conditions | What should happen? | What actually happened | Improvements or other comments |
|  |  |  |  |  |

# 9.0 Conclusions

What did you learn and what was the outcome of the project

## 9.1 Taking the Project Further

Having completed this project there are a number of steps that can be taken to improve it, make it more attractive to use and make it more efficient. The first of these steps should be to add a form of enclosure whether or not that is 3D printed, vacuum formed or some other method of creating an enclosure this project needs it to protect it from splashes and spills and other actions that could damage it. Furthermore, given more time adding in the ability for the project to not only monitor the temperature but regulate it would help with ease of use. This upgrade would include using a 2-channel relay (Fig.2) system one of the relay channels would be used for the heater and the other channel would be used for the lighting solution this would allow for safe control of electrical systems on a mains supply. Furthermore, adding more temperature probes, such as adding ones for the top middle and bottom of the vessel for getting the most accurate temperature throughout the brew being able to average this would mean that the brew could have its temperature regulated as accurately as possible. Whilst this project functions well at its major function it would benefit from the user being able to look back over previous data from previous brews this would enable the user to craft better beverages and enable them to advance their hobby or profession. The final suggestion for this project is to improve the overall functionality to include the above suggestions and improve the current new code and to improve the user interface to make it more user friendly and more intuitive to use and read.

# 10.0 References

AliExpress. (2010). *STM32F103C8T6 ARM STM32 Minimum System Development Board Module for Arduino DIY KIT*. Retrieved from <https://www.aliexpress.com/item/STM32F103C8T6-ARM-STM32-Minimum-System-Development-Board-Module-for-arduino-DIY-KIT/32918062954.html> accessed 16/06/18

Amazon. (n.d). *SainSmart 2-Channel Relay Module.* Retrieved from <https://www.amazon.com/SainSmart-101-70-100-2-Channel-Relay-Module/dp/B0057OC6D8> accessed 22/03/19

Arduino. (2018). *Arduino Uno Rev 3* Retrieved from <https://store.arduino.cc/arduino-uno-rev3> accessed 10/06/18

Arduino. (2018). *Arduino – Software.* Retrieved from <https://www.arduino.cc/en/Main/Software> accessed 11/06/18

Arduino. (2018). *Arduino Playground.* Retrieved from <https://playground.arduino.cc/> accessed 11/06/18

The Apache Software Foundation (1997) *Apache HTTP Server Project* Retrieved from <https://httpd.apache.org/> accessed 20/08/18

BeagleBoard.org Foundation. (2018, 28 June). *Beaglebone Black* Retrieved from <https://beagleboard.org/black> accessed 29/06/18

Brush Matthew, Hopf Dominic, Lanitz Frank, Treleaven Nick, Tröger Enrico and Wendling Colomban. (2006) *Geany* Retrieved from <https://www.geany.org/> accessed 21/08/18

Burton, Miles. (2018) *Arduino Temperature Control library* retrieved from <https://github.com/milesburton/Arduino-Temperature-Control-Library> accessed 17/08/18

EBay (n.d) .*Enhancement V2 Pro Mini 8MHz (3.3V/5V adjustable)MEGA328P(Arduino-Compatible)*.Retrieved from <https://www.ebay.co.uk/itm/Enhancement-V2-Pro-Mini-8MHz-3-3V-5V-adjustable-MEGA328P-Arduino-Compatible-/321413432145> accessed 10/06/18

EBay. (n.d). *DS18B20 Waterproof Sensor Thermal Probe Temperature Thermometer RPi ESP8266 etc*. Retrieved from <https://www.ebay.co.uk/itm/DS18B20-Waterproof-Sensor-Thermal-Probe-Temperature-Thermometer-RPi-ESP8266-etc/272667082078?hash=item3f7c397d5e:rk:2:pf:0&var> accessed 21/06/18

Emmeshop (n.d) *Read and Write from serial port with Raspberry Pi*. Retrieved from <https://www.instructables.com/id/Read-and-write-from-serial-port-with-Raspberry-Pi/> accessed 20/08/18

The Home Brew Shop. (2019). Beer *Starter Kit With Barrel – Equipment Only*. Retrieved from <https://www.the-home-brew-shop.co.uk/acatalog/Beer_Starter_Kit_With_Barrel_Equipment_Only.html> accessed on 17/08/18

Intel Corporation. (2016). *Intel Compute Stick STK1A32SC* Retrieved from <https://www.intel.co.uk/content/www/uk/en/products/boards-kits/compute-stick/stk1a32sc.html?_ga=2.30965564.2058863326.1546697690-426372419.1532765125> accessed 25/06/18

Kumar, Shambhoo., & Rajan, Rajeev. (2017, November 16) *How to use DS18B20 Waterproof Temperature Sensor* Retrieved from [https://create.arduino.cc/projecthub/iotboys/how-to-use-ds18b20-water-proof-temperature-sensor-2adecc accessed 15/08/18](https://create.arduino.cc/projecthub/iotboys/how-to-use-ds18b20-water-proof-temperature-sensor-2adecc%20accessed%2015/08/18)

ModMyPi LTD. (2016, 19 April). *How to give your Raspberry Pi a Static IP Address - UPDATE* Retrieved From <https://www.modmypi.com/blog/how-to-give-your-raspberry-pi-a-static-ip-address-update> accessed 18/08/18

Motion Project (2018) *Motion - Project* Retrieved from <https://motion-project.github.io/index.html> accessed 20/08/18

The Pi Hut. (2019). *Raspberry Pi 3 Model B*. Retrieved from <https://thepihut.com/products/raspberry-pi-3-model-b?src=raspberrypi> accessed 19/06/18

Premier Farnell Limited. (2018). *BeagleBone Black Wireless with Wi-Fi & Bluetooth*. Retrieved from <https://cpc.farnell.com/element14/bbone-black-wireless/beaglebone-black-wireless/dp/SC14310> accessed 19/06/18

Python Software Foundation. (2001). *Python* Retrieved from <https://www.python.org/> accessed 15/09/18

Raspberry Pi Foundation. (n.d). *Downloads* Retrieved from <https://www.raspberrypi.org/downloads/> accessed 17/08/18

Raspberry Pi Foundation. (2016, April). *Camera Module V2* Retrieved from <https://www.raspberrypi.org/products/camera-module-v2/> accessed 20/06/18

Raspberry Pi Foundation. (2019) *Raspberry Pi 3 model B* <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/> accessed 19/06/18

Raspberry Pi Foundation. (2018, 30 April). *Setting up and Apache 2 webserver on Raspberry Pi* Retrieved from <https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md> accessed 18/08/18

Raspberry Pi Tutorials. (n.d.). *How to setup a Raspberry Pi Security camera Livestream* Retrieved from <https://tutorials-raspberrypi.com/raspberry-pi-security-camera-livestream-setup/> accessed 18/08/18

SD Association. (2018). *SD Memory Card Formatter* Retrieved from <https://www.sdcard.org/downloads/formatter_4/index.html> accessed 18/08/18

Stoffregen, Paul. (2018) *OneWire* Retrieved from [https://github.com/PaulStoffregen/OneWire accessed 17/08/18](https://github.com/PaulStoffregen/OneWire%20accessed%2017/08/18)

w3schools (1999) *Node.Js and Raspberry Pi* Retrieved from. <https://www.w3schools.com/nodejs/nodejs_raspberrypi.asp> accessed 10/12/18

Youngs. (n.d) *Brew Buddy Beer 40 Pints Home Brew Starter Kit* Retrieved from <https://www.amazon.co.uk/Brew-Buddy-Beer-Pints-Starter/dp/B001C54KQW/ref=sr_1_5?s=kitchen&ie=UTF8&qid=1542641698&sr=1-5&keywords=brew+buddy> accessed on 17/08/18

# 1.0 Appendix Figure 1 showing the pinout and wiring of the whole project.



# 2.0 Appendix Fig.2 showing a 2-channel relay board used for switching mains supply

