

Department of Engineering and Mathematics

**Matthew David King**

B5008338

BEng Computer Systems Engineering

**Remote Web-based Monitoring of the Brewing Process**

Dissertation submitted in partial fulfilment of the requirements of the degree of Batchelor of Engineering

# i Preface

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

The submission of the report is in accordance with the requirements for the award of the degree of Computer Systems Engineering under the auspices of the University.

# ii Acknowledgements

Throughout my project I have received valuable support and guidance I would first like to thank my supervisor Dr Alexander Kalashnikov who helped me define the scope of the project and supported me through some of the difficult phases.

To my Grandad who I miss dearly and has helped me achieve everything. Thank you, Grandad.

In addition, I would like to thank both of my parents for the support and guidance they have given me through this project.

# iii Abstract

The purpose of this project is to improve the productivity and technology of micro and hobby brewers by producing a low-cost solution for automation that fits their needs. This system consists of a temperature sensor controlled by a microcontroller that is connected to a single board computer by UART connection this data is then passed on to the website through the webserver by AJAX calls. A video feed is hosted on the webpage so that the brew can be monitored visually in case anything goes wrong that can’t be controlled electronically. Finally, a set of web controls the first of which is a button that gives instant updates of the temperature readings regardless of the timer on the webserver, then there are two checkboxes the first of these turns either on or off an LED in place of a relay for mains supply to a heating element and the second of these switches and LED on or off in place of a relay for mains supply to an LED strip that provides light for the camera in dark conditions. Finally, and emails will be sent to the user to notify them of impending action(s). Of these aims and objectives all were completed except for the email service which was only completed to a trial code state and the mains relays were changed for LED simulations due to safety reasons.

**Contents Page**

Contents

[i Preface 4](#_Toc6159930)

[ii Acknowledgements 4](#_Toc6159931)

[iii Abstract 4](#_Toc6159932)

[Nomenclature 6](#_Toc6159933)

[1.0 Introduction 7](#_Toc6159934)

[2.0 Literature Survey/Theory 7](#_Toc6159935)

[2.1 Choosing hardware 7](#_Toc6159936)

[2.2 Project Scope 10](#_Toc6159937)

[2.2.1 Just Add Water Kits 10](#_Toc6159938)

[2.2.2 Hobby Brewing Starter Kits 10](#_Toc6159939)

[2.2.3 Programming Languages 10](#_Toc6159940)

[3.0 Aim and Objectives 11](#_Toc6159941)

[4.0 Approach & methodology 12](#_Toc6159942)

[5.0 Project Development 12](#_Toc6159943)

[5.1 Hardware and wiring 12](#_Toc6159944)

[5.1.1 Arduino Pro Mini 12](#_Toc6159945)

[5.1.2 Raspberry Pi 13](#_Toc6159946)

[5.2 Software and Programming 13](#_Toc6159947)

[5.2.1 Arduino Pro Mini 13](#_Toc6159948)

[5.2.2 Raspberry Pi 14](#_Toc6159949)

[6.0 Project Outcomes and Progress 18](#_Toc6159950)

[7.0 Project Management 19](#_Toc6159951)

[8.0 Project Testing 20](#_Toc6159952)

[9.0 Conclusions 21](#_Toc6159953)

[9.1 Taking the Project Further 21](#_Toc6159954)

[10.0 References 23](#_Toc6159955)

[1.0 Appendix Figure 1 showing the pinout and wiring of the whole project. 28](#_Toc6159956)

[2.0 Appendix Figure 2 Showing the final Gannt chart at the end of the project 29](#_Toc6159957)

[3.0 Appendix Figure 3 showing a 2-channel relay board used for switching mains supply 30](#_Toc6159958)

# Nomenclature

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

|  |  |
| --- | --- |
| Abbreviation | Meaning |
| AJAX | Asynchronous JavaScript And Xml |
| CSS | Cascading Style Sheet |
| GB | Gigabyte |
| GPIO | General Purpose Input Output |
| HTML | Hyper Text Mark-up Language |
| IDE | Integrated Development Environment |
| LED | Light Emitting Diode |
| MCU | Micro Controller Unit |
| OS | Operating System |
| PCB | Printed Circuit Board |
| Rx | Receive (when discussing UART connections) |
| Tx | Transmit (when discussing UART connections) |
| UART | Universal Asynchronous Receiver-Transmitter |
| V | Volts |

# 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 throughout my university career.

When hobbyists or industrial brewers brew alcohol, temperature monitoring, especially for hobby brewers, is done manually by dipping a thermometer into the brewing liquid mixture. This procedure poses a variety of issues including the major health and safety concern because the boiling mixture temperature can get up to 100 degrees Celsius which can cause serious burns. Furthermore, if the brewer at a micro-brewery or industrial user may have very large vats of liquid mixture, she/he may need to stand on a ladder or walkway and lean over such a boiling vat again risking serious injury. This project will replace the above method with something more up to date by using a microcontroller and a single board computer to automate brewing process. Also, to help brewers know what to do and when, the system incorporates a timing system that will deliver email-based alerts so that brewers know when to take the next appropriate action(s).

# 2.0 Literature Survey/Theory

The process of brewing alcohol has three main stages. The first stage is called the mash, the second is called boiling and finally there is fermentation stage. At the last stage the brew sits in an insulated environment for an extended period of time when the yeast performs anaerobic respiration, producing alcohol, completing the brewing process. The project that will be produced will aim to monitor, and eventually regulate, the boiling stage of this process.

## 2.1 Choosing hardware

This project was constructed using an embedded system and a single board computer. For the choice of programmable microcontroller (MCU) 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 MCU 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.

|  |  |  |  |
| --- | --- | --- | --- |
| Name/Specification | Arduino Pro Mini | Arduino Uno | STM32F103C8T6 |
| Size L x 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 an MCU, 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. This was further reinforced by the fact that it could not communicate with the chosen MCU platform as it would need a UART connection using GPIO pins and the Intel Compute Stick does not have this functionality. 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 greater 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 much 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.

|  |  |  |  |
| --- | --- | --- | --- |
| Specification | 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 |
| Wi-Fi 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. As the big industrial brewers already have automated temperature monitoring and control tools in place.

### 2.2.3 Programming Languages

Having done research about the programming languages that are being used, I found that Python (Python Software Foundation, 2001) for the temperature update code was a bad choice as it overcomplicated the update system to the webserver and added an extra programming language to the project. Instead, JavaScript was used in Python's place - this then extended into using a JavaScript language Node.js (Node.js Foundation n.d.) and two modules one called onoff (onoff, n.d.) and the other Express (Node.js Foundation, 2017) 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 webpage elements both automatically and on user input. At the end of the build phase of this project one of the last things to be done will be to add and email alert system a tutorial on how to send emails for Node.js servers was found under the title Node.js Send an Email (w3schools, 1999) this research means that there is a ready made solution that will speed up the building process and make it easier to complete this section of the project.

# 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 webpage using 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 webpage.
4. Create a completed automatically updating webpage with video feed and a working webserver solution (prebuilt or custom) with port forwarding network permissions permitting
5. 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.
6. Time permitting a relay and a heating element could be added to be able to fully automate the temperature regulation of the brew
7. Time permitting create a light source for the camera that will automatically switch on in dark environments.

# 4.0 Approach & methodology

Design and build a temperature sensor based upon the Arduino Pro Mini and the DS18B20 one wire temperature sensor that sends data at regular intervals via a UART connection to a single board computer for data processing. Upon getting the data to the single board computer system which in this case was a Raspberry Pi, as discussed in the literature review section of this document, serves both as a webserver and the central brain for this project. The Raspberry Pi is connected to the internet via an ethernet cable and the web services accessible over the local network with the aim of port forwarding these services into the public domain and using a domain name service to make accessing these services easier. The webserver will also act as a central hub for the storage of the data and any actions that need to be taken these will be the email services and the lighting and heating services which will be controlled by the user and eventually automated.

# 5.0 Project Development

This section of the document describes and discusses all of the steps taken during the building of this project and all of the setbacks and holdups to the projects timetable along with the sections of the project that were completed ahead of the planned end date.

## 5.1 Hardware and wiring

The main development style, that was followed throughout the project, was a step by step process that included elements of work that were known in advance and elements of work that were not known at the time.

### 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 Appendix 1.0. 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 (Raspberry Pi Foundation, 2016) 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). Initially the program did not work, and this was due to an error in the fundamentals of writing the code. This fundamental error was due to a mistake in writing the time delays into the program. Using the built-in delay() function in the Arduino IDE clogs up to microcontroller for the amount of time in the function this means that the board can not multi-task to fix this a custom function was made that uses a millisecond counting system that does not interrupt the progress of the board and allows multi-tasking. The other major error that was stopping the progress were two of the needed libraries were not installed or included and the incorrect technique for reading data from the temperature sensor. Following this, the program was fixed, and two necessary libraries were added these were the OneWire (Stoffregen, 2018) and DallasTemperature (Burton, 2018) libraries. With the addition of these changes and a major rewrite of the code the program compiled. In order to write it to the Pro Mini an Arduino Uno was used with the main chip removed to act as a USB to serial converter to write the program to the Arduino Pro Mini.

### 5.2.2 Raspberry Pi

Moving on from the Arduino Pro Mini the other major element to this project is the Raspberry Pi the initial setup of the Raspberry Pi is simple yet time a little consuming. The first thing is to get an SD card of an appropriate size, for this project a 32GB card was selected, and format it the tool of choice for the SD card in use is SD card formatter (SD Association, 2018). The OS that I will be using is Raspbian (Raspberry Pi Foundation 2018) this disk image is written to the SD card and then the SD card is inserted into the Raspberry Pi for first boot and setup. The setup of the Raspberry Pi OS started the command used to start the wizard is, sudo raspi-config, the first step during setup is the resizing of the file system’s partition on the disk to take up as much available space as it could. Following this SSH was enabled on the Raspberry Pi so it could be run headless to reduce the space required throughout the development and the camera module was activated and then the wizard was exited and the Raspberry Pi rebooted.

Following this initial setup of the Raspberry Pi the Raspberry Pi update and upgrade commands were run, see bullet points below, these commands update the available packages lists for the Raspberry Pi allow it pull new software and updates for existing software. The second of the two commands checks these lists and applies any changes to the packages be it updates or new installs in this case it will just update existing packages, the -y option at the end of this command will allow it to bypass the question that it asks with a “yes” answer allowing the command to install any updates it finds.

* sudo apt-get update
* sudo apt-get dist-upgrade -y

Following the updates, I removed a number of unnecessary programs that take up disk space and that would never be used these were, Minecraft for the raspberry pi, the wolfram alpha package, scratch, scratch 2 and the libre office suite. These packages were not uninstalled one by one but were instead removed in batches so as to remove them faster but allowed for reduced human error when inputting the package names. Post uninstall a command was run that removed any redundant packages left over from the update and uninstallations. The commands that were used are detailed in the bullet points below.

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

The next step was to set the Raspberry Pi up for the project and there were two pieces of software that needed to be installed Apache2 webserver software (The Apache Software Foundation, 1997), which would ultimately be replaced by a custom solution, and Motion4.0 (Motion Project, 2018) a security camera streaming software. The next step was to set the Raspberry Pi’s local IP to static with help from a guide on the internet (ModMyPi LTD, 19 April 2016) as this is essential for port forwarding and greatly helps during any testing that needs to be done. The following two bullet points are the install commands that were used in the console to acquire the software discussed above.

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

Following these installs a brief test of the Apache2 server was completed by opening a web browser and typing the local IP address of the Raspberry Pi which in this case was 192.168.1.128 this test brought up the Apache2 test landing page and confirmed that it was working properly now that Apache2 was up and running the next setup was Motion 4.0. Setting up Motion 4.0 requires getting into its configuration file and modifying it to give the desired outcome. Before modifying the configuration file for Motion 4.0 the camera module was added to the /etc/modules file this was done by adding “bcm2835-v4l2” to the bottom of the file saving and exiting, this change would come into effect after a reboot. The list of following bullet points shows the command to open the Motion 4.0 configuration file and changes made to it.

* 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

The first of the indented bullet points details a change that allows Motion 4.0 to run as a daemon meaning that it is a background process and will be started on boot by the OS. Following this change the dimensions of the camera’s were used as the width and height properties for Motion 4.0 so that it captures the full view of the camera. The frame rate was then changed from one frame per second to sixty although the camera is not capable of this it allows the camera to have some headroom and run at its highest possible frame rate it is limited however by the ability of the Raspberry Pi’s hardware and can be slightly delayed while on stream. Changing the output\_pictures option to off stops Motion 4.0 outputting stills wherever motion is detected on the screen this allows disk space to be kept free of clutter. The next bullet point changes the port that the video is streamed on changing this from the default of zero, which means that it wont stream anything, to 8081 an open port that can be used for the video feed stream. Finally, the ability to stream on the local network is a project requirement and therefore the steam\_localhost option is changed from off to on after these changes are implemented and a reboot performed checking the local IP address of the Raspberry Pi with the port 8081 shows the video feed meaning that it has worked.

Continuing with the setup of the Raspberry Pi’s OS and software packages a new IDE was installed called Geaney (Brush Matthew, Hopf Dominic, Lanitz Frank, Treleaven Nick, Tröger Enrico and Wendling Colomban.,2006) this was installed to make code writing easier as Geaney can detect and check multiple languages and show where errors are users can also include breakpoints and step through code line by line, this is not possible when writing code in the terminal or writing code in IDEs that come with languages as many of these are simplistic and some do not come with an IDE at all.

The design of the website was done independently to the webserver. Initially, when the webserver solution was Apache 2 the html and CSS files were stored under the /var/www/html folder, this was moved to a under /home/pi/node.js/Public/ and the CSS was given its own folder under this directory after the change from Apache 2 to a customised solution was made, the website code houses the containers for the video feed, an iframe, and the updating website code, in this section there was a button and the two checkboxes for the heating and lighting controls. The CSS file contains information that talks about the style of the webpage, where things are on the page and their other properties.

Once the website and stream working together a python script was written to interface with the Arduino Pro Mini as a proof of concept for functionality this code was an adapted version of an online guide I had read (emmshop, n.d.). During the writing of the python script a config option for the Raspberry Pi was changed, this option was to disable console over serial but keep the hardware for serial interfacing active this way there could be no interfering software running that could affect how my programs worked. However, this produced a problem that Apache 2 couldn’t solve simply so a new approach to the problem was adopted and a custom webserver was the way forward that was chosen.

Having done research into JavaScript and how it can make webpages work better and be more user friendly it was apparent that using more than one programming language over complicated this project and made it more complicated to recreate and understand. So, a change from Apache 2 and python scripts was made this change was to remove these two elements and replace them with one single solution this custom webserver housed all the code needed to run everything and it tied the whole project together.

This changeover required the installation node.js, which is a JavaScript webserver package with a very wide variety of addons that allow for the rest of the project all to run from one file greatly increasing the projects readability and efficiency. The next two bullet points are the commands 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 proceeded correctly

The first steps in writing the new webserver were to rebuild the serial communications element of the program with the aid of what was written in python and by using an article called Simple and Easy method to host Node.js Webserver with express for static website (Opencodez, 2018) the article explained how to write the code that hosted the webpage and the static content, both the JavaScript and CSS file were static content, on the local network.

Following the initial functionality of the webserver, the next step in development was to get the AJAX elements of the system working and this took a long time as everything that was being done had to be learned from scratch. The first AJAX element to be programmed in was a button press that retrieves temperature data from the webserver and displays it on the webpage in degrees Celsius. This was done to build up the layers of code that would be working and so that it was easily testable this code on the server side was put onto a try catch error handling method. When the button was programmed and worked a switch block was added to the AJAX code and this allowed for clean code and better readability and for multiple different situations in a simple solution. The second entry into the switch block was a timed request response that sent data whenever there was a request for it the JavaScript on the webpage had a ten second timer between each of these timed requests and the returned data was displayed on the webpage in the same way as the button did so that the button could be an immediate update and in case the user wanted an update more regularly than the ten second window.

With the data now on the webpage along with the video stream all that was left to do was add in the controls for the relay circuits now simulated due to the lack of time. For the web controls checkboxes were used as the relays will either be on or off for periods of time and checkboxes were the best fit for this task. When the state of the checkbox changes an AJAX, request is sent with the new state this then switches the LED, standing in for the relay, either on or off depending on its previous state.

Following this a start was made into the email service however the only code that was written that got working was test code that sent an email one the program start with simple greeting text following this the code that was written was tidied up and everything was made to start up on system boot so the user doesn’t need to get stuck trying to remember or learn Linux commands.

# 6.0 Project Outcomes and Progress

Although this project was not fully completed major progress was made into completing the build, more building was scheduled in this project than could feasibly be completed within the given time constraints. Looking back at the aims and objectives of this project and comparing that with the work completed you can see that the majority of this work is completed and that this project is almost at a finished stage.

In terms of the main aim of this project to produce a system that can be used to monitor the brewing process remotely from a webpage via a temperature sensor and a web enabled camera. This can be considered complete for the following reasons.

In the case of the first objective of this project to create a working temperature acquisition system this objective was completed because an Arduino Pro Mini with a DS18B20 and a UART connection was used to acquire temperature data at a rate of one data reading per second and send it to a secondary device over the attached UART connection.

Following this the second objective, Setup Raspberry Pi for data acquisition from the Arduino system, was met because the Raspberry Pi was setup and running on Raspbian OS with a scripts that checked the serial interface and every time that there was a fresh piece of data it would write that data to the console. This section of the project developed into a section of the webserver that read the data from the serial connection and sent data to the webpage every ten seconds.

The third objective, create a completed webpage, was met because a webpage written in HTML and styled with an external CSS file this was stored in the appropriate folders.

The fourth objective, create a completed automatically updating webpage with video feed and a working webserver solution (prebuilt or custom) with port forwarding network permissions permitting, was completed by building the webpages automatically updating functionality was done both client and server side the code on the client side was written in jQuery using the AJAX technique allowing for areas of the webpage to be updated without the update of the entire page being less disruptive to the user. The video feed was added into the webpage through the use of an iframe and the webserver, originally prebuilt, was a custom design by the end and was completed as it could serve the webpage and static content, the CSS and JavaScript, to the client device and send data to the webpage every ten seconds or on demand by a button press on the webpage. Unfortunately, due to the lack of administrator privileges on the network the port forwarding part of this objective wasn’t possible.

The fifth objective, 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, was only partially completed as the code to send an email was implemented however the functionality for it to update the user of impending changes was not implemented due to a lack of time to implement it. Furthermore, a system for sending SMS messages was not implemented due to a lack of time.

An addition to the project was planned and attempted however broke down during the build and could not be completed. This addition was a data logging element using and SQL database called SQLite (SQLite, n.d.) at the time of it working the webserver could write to the database and log data with a timestamp however it could not pull the data from the database and ceased functioning due to a file corruption with the database file no further work was done due to a lack of time.

Finally, due to a lack of time at the end of the building of the project only LED markers were implemented for the last two objectives that were controlled by checkbox by the user on the website’s user interface using AJAX calls to the webserver allowing the user to turn either one or the other or both of the controls on at any given time.

# 7.0 Project Management

Throughout out the project a rough adaptable timetable has been used in order to attempt to keep the project flowing and keep it from stalling. The type of project management that was used was a Gannt chart and it was created in the Gannt Project software (GanntProject, 2019) this software allows for easy creation of tasks and the options of dependency on previous tasks or not and the length of time the task takes along with the percentage of completion. The Gannt chart produced for this project (Appendix 2) made it easier to track the rate of progress in this project and the project’s sticking points. During the earlier stages of this project progress moved swiftly and the time that was left over from these early successes was easily allocated to other sections of the project that were more challenging. This was especially useful when learning about the JavaScript elements of this project and the allocation of extra time to it made them easier to complete on time. However even with the extra time this project was not completed, and some small elements remain uncompleted due to their complexity.

# 8.0 Project Testing

The testing procedure for this project was done systematically and aimed to test the elements of the project that could fail or have something go wrong with them and produce errors.

The first test will test the temperature sensor and how it should react to changes in temperature for this the sensor will be submerged in cold water and the reading allowed to stabilise then it will be quickly moved to hot water, straight from the kettle, where it should show the new temperature of the hot water after allowing for it to adjust to the sudden change should read at near 100°C. During this test the temperature sensor was placed in cold water and stabilised at 15°C it was then moved to hot water, fresh from the kettle after boiling, where the temperature readings rose to 85°C this proves that the sensor is working as expected both on the webpage and tested through a UART reading python script. This test was repeated on the webpage and the results were the same as the UART reading script for each update of the webpage’s calls to the server

Following this test, I checked that the camera feed was working first as a standalone solution and then embedded in the website. In the first instance there should be a video feed on port 8081 on the Raspberry Pi’s local IP address testing this on an external device’s browser brought up the video feed surrounded by a black boarder. This was then tested on the website. The website was hosted on the custom webserver solution on port 8082 accessing this brought up the webpage for the project and embedded on the page was the video feed running and had an acceptable framerate.

Whilst on the website the next thing to test was the checkbox controls and when checked the webserver should light up the LED corresponding to that checkbox so for a checked heater checkbox the heater LED should light up. This should also be true for the lighting checkbox. Upon testing this both checkboxes worked as expected. However if the webpage was closed at any point while either or both of the LEDs was lit then the next time the user loaded up the website the checkboxes were desynchronised from the hardware so the checkbox would be uncheck and the LED still this is a code issue and needed solving by querying the webserver to check if the LEDS are lit and then setting the state of the checkbox based upon the current state of the LED.

# 9.0 Conclusions

Throughout this project I have learned a great deal especially on the topic of project and time management. Before completing this project, I had no real knowledge of project management or the importance that it played in the role of ensuring that all elements of the project were completed as a unit. Furthermore, time management had a very large role to play in this project and the way that it turned out.

I also learned new skills in programming in JavaScript due to the fact that I was not able to use the Python programming language in the way that I thought that I could. I also learned, from scratch, creating and maintaining a webserver and website including adding new features and upgrading obsolete ones, programming in HTML, CSS and various forms of JavaScript.

Furthermore, I learned a lot about organising file systems in order to keep projects tidy. I also learned a lot about the usefulness and capability of version controlled external storage as it allowed for easy storage and peace of mind when playing around with new features without the worry of losing a stable working version of code.

A lot was learned about combining multiple smaller and simpler project elements, ideas and systems to create a fully functioning computer system that was fit for purpose and the workload that goes into creating such a system.

## 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 (Appendix 3) 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](https://www.ebay.co.uk/itm/Enhancement-V2-Pro-Mini-8MHz-3-3V-5V-adjustable-MEGA328P-Arduino-Compatible-/321413432145%20accessed%2010/06/18)

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

GanntProject (2019) *GanntProject* <https://www.ganttproject.biz/> accessed on 25/09/18

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

Node.js Foundation. (2017). *Express* <https://expressjs.com/> accessed 10/12/18

Node.js Foundation. (n.d.) *Node.js* Retrieved from <https://nodejs.org/en/> accessed 10/12/18

Node SerialPort. (2019). *Getting started with serialPort* Retrieved from <https://serialport.io/docs/en/guide-usage> accessed 05/02/19

(n.d.) *onoff* <https://www.npmjs.com/package/onoff> accessed 10/12/18

Opencodez (2018, 25 August) *Simple and Easy method to host Node.js Webserver for static website* Retrieved from<https://www.opencodez.com/java-script/static-website-with-node-js-webserver.htm> accessed 15/02/19

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

RandomNerdTutorials.com. (2019). *SQLite database on a raspberry pi* Retrieved from <https://randomnerdtutorials.com/sqlite-database-on-a-raspberry-pi/> accessed 15/02/19

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* Retrieved from <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

Raspberry Pi Tutorials. (n.d.) *How to setup a Raspberry Pi Node.js Webserver and control GPIOs* Retrieved from <https://tutorials-raspberrypi.com/setup-raspberry-pi-node-js-webserver-control-gpios/> accessed 31/01/19

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

SQLite. (n.d.) *SQLite* Retrieved from <https://www.sqlite.org/index.html> accessed 14/11/18

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

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

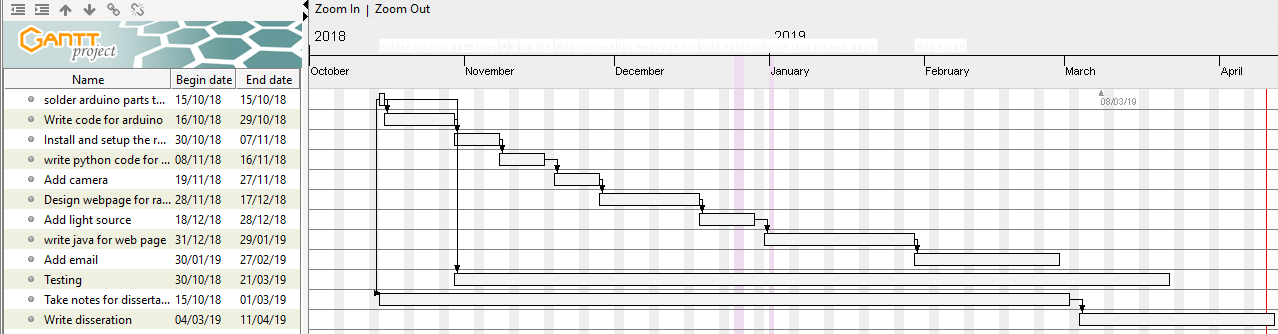
w3schools .(1999) *Node.js Send an Email* <https://www.w3schools.com/nodejs/nodejs_email.asp> accessed 20/02/2019

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 17/08/18

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



# 2.0 Appendix Figure 2 Showing the final Gannt chart at the end of the project



# 3.0 Appendix Figure 3 showing a 2-channel relay board used for switching mains supply

