Challenge 2: Global Health care

CS-EEE 32 Specialist Team Final Report

Authors:

1. Nathan Yuen
2. Bowen Wu
3. Nafisa Sarder
4. Yuehan Wang
5. Hannah Tan
6. Yiwen Chen
7. Lewis Quinn
8. Nadia Abdul Muiz
9. Peter Petrov
10. Seth Perera

Table of Contents

1 Introduction ***2***

2 Subsystem Designs ***3***

2.1 Heating Subsystem **3**

2.1.1 Purpose of the Subsystem 3

2.1.2 Technical Steps to Design the Subsystem 3

2.1.3 Results of technical testing 4

2.1.3 How the heating Subsystem Links with the Other Subsystems 4

2.2 Stirring Subsystem **4**

2.2.1 Purpose of the Subsystem 4

2.2.2 Technical Steps to Design the Subsystem 4

2.2.3 Results of Technical Testing 5

2.2.4 How the Stirring Subsystem Links with the Other Subsystems 5

2.3 pH Subsystem **6**

2.3.1 Purpose of the pH Subsystem 6

2.3.2 Technical Steps to Design the Subsystem 6

2.3.3 Results of technical testing 7

2.3.4 How the pH Subsystem Links with the Other Subsystems 7

2.4 Communication between Arduino and ESP32 ..... **7**

2.4.1 Purpose 7

2.4.2 Technical Design Steps 7

2.5 Communication between ESP32 and User Interface **8**

2.5.1 Purpose 8

2.5.2 Technical Design Steps 9

2.5.3 Visualisation 9

3 Overall System Integration and Summary ***9***

3.1 Testing **9**

3.2 Final Demonstration Outcome **10**

4 Appendices ***10***

1 Introduction

Across Uganda, there are many people developing tuberculosis. Although some people may experience symptoms of the tuberculosis, many others are carriers of the disease as there is only a 10% of the bacteria awakening. The overall challenge was to design, develop and test the circuitry, software and interfaces required to remotely monitor and control parameters of a small-scale bioreactor which will be used in a BCG vaccine manufacturing facility in Uganda.

The purpose of the system that we have designed is to remotely monitor system parameters, provide the user with real-time system information, allow the user to control and adjust system parameters and log the system parameters for process quality control. This is done by connecting the 3 systems – heating, stirring and pH to an Arduino which is connected to and ESP32 via a Bidirectional Logic Level Shifter (BLLS). The ESP32 which contains a Wi-Fi antenna connects to the internet via a Wi-Fi connection. Once connected to the internet it connects to the thingsboard where it can interact with the dashboard. This will allow data to be sent from the subsystems to the user interface.

There were 3 subsystem groups each working on one of the subsystems – heating, stirring and pH. Each of the groups were tasked with building the circuit for their subsystem and writing code which keeps the subsystems parameters within the system specifications. To achieve this each of the groups had to calibrate the code with the circuits to keep the system parameters at reasonably close range to the required range for the system. Figure 1 below shows a diagram of the three subsystems.

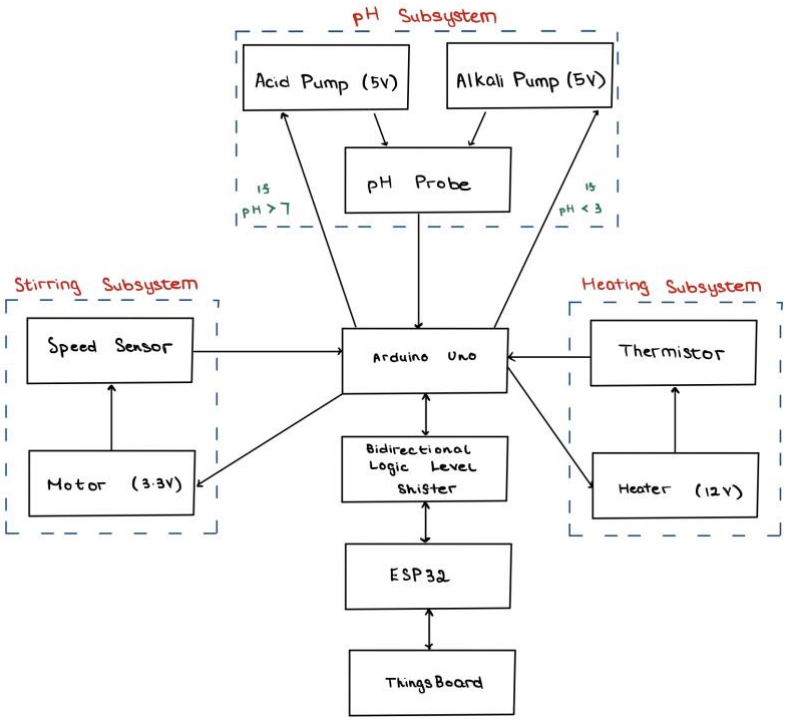

Figure 1: Diagram of the complete system

The system specifications are:

- The inlet water temperature is assumed to be in the range of 10-20°C
- The temperature control system must be able to maintain the temperature at a set point in the range 25-35°C to within ±0.5°C of the set point.

- The stirring subsystem must be able to maintain the stirring speed at a set point in the range 500-1500 RPM within ±20 RPM of the set point.
- The pH subsystem must be able to maintain the pH value at a setpoint in the sensing range 3-7.
- All subsystems should be able to:
    - Log data.
    - Provide a user interface for monitoring the parameters and allowing the set point values to be adjusted.

2 Subsystem Designs

2.1 Heating Subsystem

2.1.1 Purpose of the Subsystem

The aim of the heating subsystem is to heat and maintain the temperature of the solution at a set point in the range 25-35℃ to within ± 0.5 ℃ of the set point.. It is crucial to accurately maintain optimal temperature within the bioreactor for vaccine production.

2.1.2 Technical Steps to Design the Subsystem

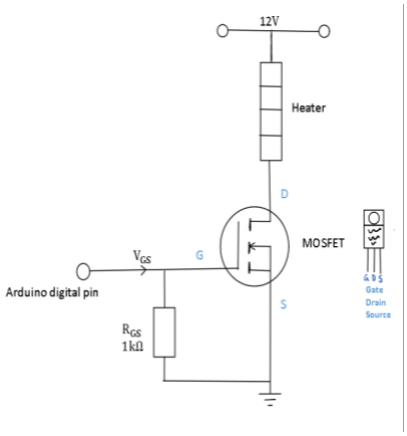

Figure H1: Actuator circuit of the heating system system as a potential divider

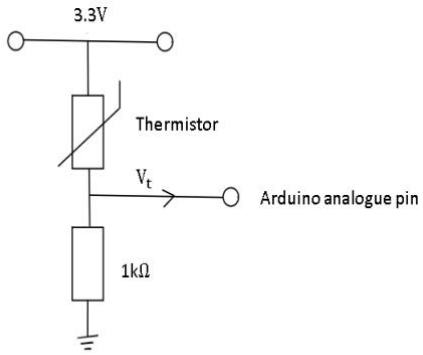

Figure H2: Shows the sensor circuit of the heating

Firstly, to control the heater, we decided to use a MOSFET (n channel) which will act as a switch to turn the heater ON/OFF as shown in figure H1. The gate to source voltage V_{GS} from Arduino will determine whether the heater is ON or OFF. The MOSFET gate quickly charges up as the supply voltage is applied, turning them on. Once the supply is removed from the gate, the MOSFET still stays ON, hence the heater is still ON. The gate to source resistor is to ensure that as soon as the supply is removed, the MOSFET can quickly turn OFF as it acts as a small capacitor to extract excess electric charge from the gate.

To determine when the MOSFET should turn the heater ON and OFF, we decided to calibrate the sensor(thermistor) to find out the relationship between the measured temperature of the solution and the sensor output signal (voltage across the thermistor V_t), using the circuit shown in figure H2. We obtained readings of voltages across thermistor and its corresponding temperature, which gave us an equation that relates the temperature and V_t. It is used in our Arduino coding to control temperature by converting the V_t to temperature. It is then compared with the set point temperature. If the temperature is below the setpoint, the Arduino sends HIGH voltage V_{GS} to turn ON the MOSFET hence the heater. If the temperature is above set point, the Arduino sends

LOW voltage V_{GS} to turn OFF the MOSFET hence the heater. Table H1 and Figure H3, given in the appendix, shows the data obtained and the plot of the data for calibration.

2.1.3 Results of technical testing

Procedure:
Set up both circuits shown in figure H1 and H2 and connect it to Arduino. Analogue pin is the input pin and digital pin is the output pin in our Arduino coding. Finally, turn on the power supply. Thermometer is used to measure the temperature.

Expected operation	Result
If the temperature of the solution is below 25°C , the Arduino sends HIGH voltage to turn ON the heater	The heater turned ON at 29°C
If the temperature is above 35°C , the Arduino sends LOW voltage to turn OFF the heater.	The heater turned OFF at 38°C

Table H2, shows the test results of the heating system

The result wasn't accurate and did not meet the expected operation hence failed to meet the specification of controlling temperature within the setpoint. We concluded that our calibration is not accurate which led to wrong conversion of voltage to temperature. This could be because of the malfunction of the circuits which led to inaccurate voltage readings.

2.1.3 How the heating Subsystem Links with the Other Subsystems

Heating subsystem can affect other subsystems. A constant temperature is required to stabilize the electrode's sensitivity to measure the pH value in the pH subsystem. Failure to maintain a constant temperature in the range will cause variation of pH electrode's sensitivity and therefore will lead to inaccurate measure of pH values. Also, temperature can affect the speed of reaction in stirring subsystems. An unexpected increase in temperature, above the setpoint, can lead to undesirable increase in rate of reactions. Moreover, an increase in speed of the motor in stirring system can lead to increase in heat lost to the solution, increasing its temperature. The heating subsystem will respond by turning the heater OFF if it goes above the 35°C .

2.2 Stirring Subsystem

2.2.1 Purpose of the Subsystem

The stirring system is responsible for controlling the reaction speed. When the motor is on and the blades of the propeller are in contact with the solution, there will be more frequent collisions between the reacting particles, which accelerates the reaction. Specifically, the stirring speed should be monitored between 500-1500 RPM to ensure the reaction will take place at a desired rate.

2.2.2 Technical Steps to Design the Subsystem

The motor circuit is needed to be built first, which includes an infra-red sensor (digital input) and motor (analogue output). The transistor acts as an amplifier which provides additional current that Arduino Uno can't. Connecting the diode and the motor in parallel protects the motor from being damaged by the back EMF that it generates from the coils inside. Finally, a 3.3V power supply will be applied to the motor and a 5V to the sensor. The motor circuit can be seen on Figure S1 and the speed sensor can be seen on Figure S2.

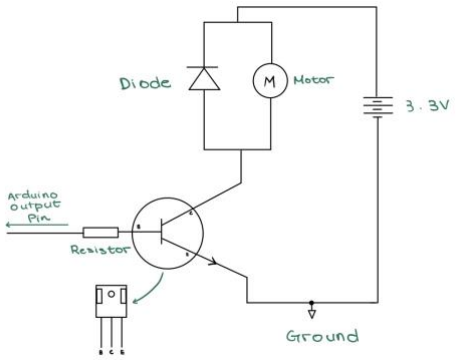

Figure S1: Motor Circuit

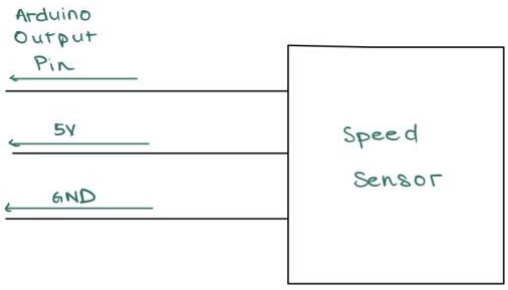

Figure S2: Speed Sensor

To measure the speed of the motor, the speed sensor would detect the speed at which the propeller is spinning. When the propeller is blocking the sensor, it would output a value of 1. When the propeller is not blocking the sensor, it would output a value of 0. As the propeller is spinning, the value outputted by the system will change constantly and time between the change of values is calculated. This value is then used to calculate the speed of the motor using an equation. If the speed of the motor is above the reference value, the motor will slow down and if the speed of the motor is above the reference value, the motor will speed up. This is done in small increments and will continue looping until the speed of the motor is equal to the reference value.

2.2.3 Results of Technical Testing

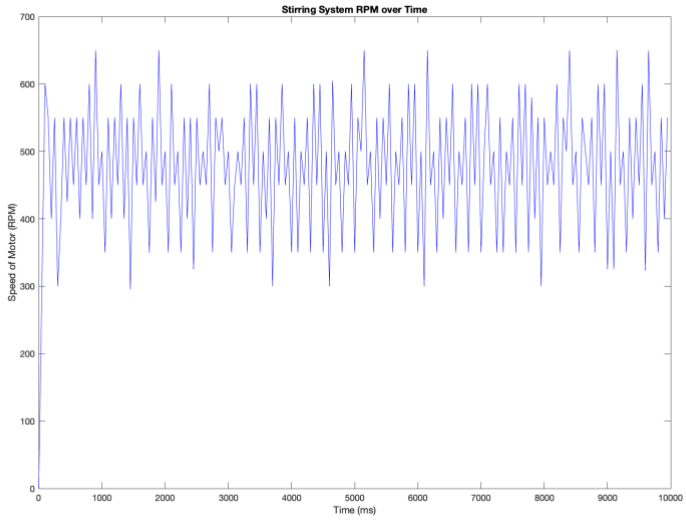

Figure S3: Graph of RPM over Time when the reference value is set to 500 RPM

Procedure	Expected operation	Result
Reference value input is in between 500 – 1500 RPM	The motor would spin at the reference speed with ±20 RPM	Motor span at the reference speed with ±200 RPM. At higher speeds, motor stops after a few seconds

2.2.4 How the Stirring Subsystem Links with the Other Subsystems

The stirring system itself can adjust the rate of reaction from the energy perspective, in other words, the stirring speed partially depends on the reaction rate generated by the

pH and heating subsystem. Take the case where the heating subsystem has increased in temperature more than the expected range, followed by an undesirable increase in the rate of reaction, the stirring subsystem handles this by slowing the stirring speed to a lower reference value. Oppositely, when the motor is spinning too fast which warms up the solution, the heating system can also respond to this by lower the temperature. For pH system, the stirring one will respond similarly but the variable is concentration of the solution instead the temperature, that is, when the solution becomes either too acidic or alkaline, motor speed will automatically adjust accordingly.

2.3 pH Subsystem

2.3.1 Purpose of the pH Subsystem

The main aim of this pH system is to control the pH value of the liquid to maintain at a setpoint in the sensing range 3-7. This means that the solution should be neither alkaline nor excessively acidic, establishing a harmless acid-alkaline balance for the vaccine.

2.3.2 Technical Steps to Design the Subsystem

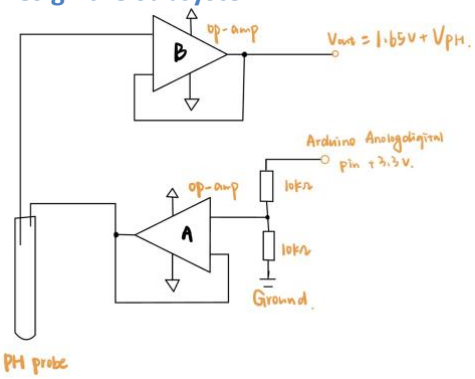

Figure pH1: Calibration circuit for pH system

Firstly, a calibration circuit should be set up as Figure pH1 shown below. According to the transfer function of the pH electrode, when pH value is larger than 7, the voltage produced will be smaller than 0, so an additional input voltage is needed to shift the bipolar pH-electrode signal to a single signal. Also due to the high impedance of the electrode, which will mess up the function of the potential-divider, so a high-input impedance buffer, op-amp, will be required between the potential divider and the pH probe. And the second op-amp is set up to buffer the output of the pH electrode. Output Voltage value is recorded for different values of pH [Figure pH3 Appendix] and plotted on a graph to find the relationship between the two variables.[Table pH2 Appendix] This will be used in the code to determine whether pump acid or alkali according to the voltage output value.

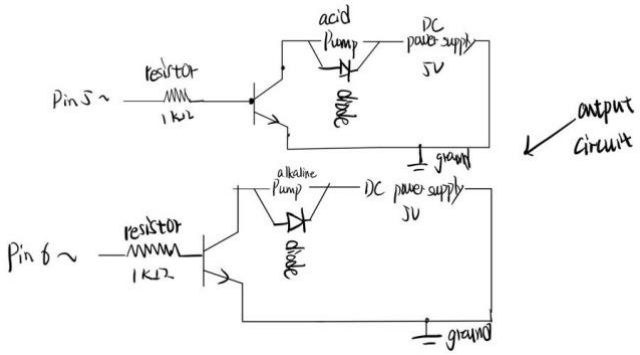

Figure pH4: Output circuit of pH system

The next step is the output circuit shown as Figure pH4, connected with two PWM pins of the Arduino board, so that the Arduino can output the program like analogue interface. The transistor acts as an amplifier which provides additional current than current supply by the Arduino Uno, and the diode is parallel to each pump to protect the transistor from any self-generated back-emf from pump. Two 5V DC power supply is applied to two pumps. Finally, when pH value of solution is above 7, the interface connected with acid pump will output 1, the interface connected with alkaline will output 0. When pH value of solution is under 3, the interface connected with alkaline pump will output 1, the interface connected with acid pump will output 0.

2.3.3 Results of technical testing

Procedure	Expected operation	Result
Put the pH probe in the solution with pH value of 10	The motor of acid pumping turns on	Pump acid
Put the pH probe in the solution with pH value of 4	The motors of acid and alkali pumps both stay off	No pump operates

Table pH5: pH subsystem test results

2.3.4 How the pH Subsystem Links with the Other Subsystems

Since the pH electrode's sensitivity varies over temperature according to the transfer function of pH probe, the temperature of the measured solution must be maintained almost constant by the heating system to stabilize the electrode's sensitivity to measure the pH value.

2.4 Communication between Arduino and ESP32

2.4.1 Purpose

The simplicity of the Arduino offers great prototyping speed, however at the cost of certain necessary elements for it to interface with other devices outside of one serial line which is typically used for debugging. As such the usage of the esp32 comes in, considering the system will be an IOT devices to allow for remote control, the esp32 offers great support in IOT device development with its built in Wi-Fi capabilities.

Unfortunately, we can't just glue the two microcontrollers together to allow for data transfer and we can't even use the UART, which is a very common cross-device communication protocol, as that is the very serial line that is taken up for on-site debugging. However, there exists such protocol for this scenario, I2C. Which is a protocol that allows for one master device and many slave devices and only uses two wires, keeping development simple and easy to connect.

2.4.2 Technical Design Steps

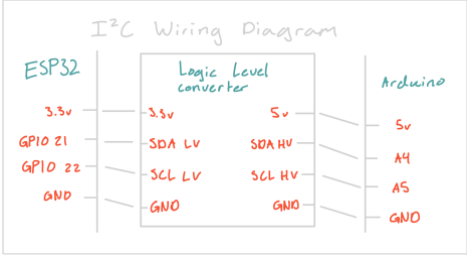

Figure 0xf3 I2C Wiring Diagram

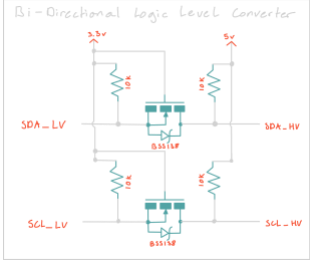

Figure 0x4b Bi-Directional Logic Level Converter

The first step in the process of setting up the I2C communication is creating the connection between the ESP32 and the Arduino. The connection points are shown in Figure 0xf3, it can also be noted that the connection although uses few wires, is not quite as straight forward as initially thought. Unfortunately, due to the different operating voltages of the microcontroller, a logic level converter must be placed in between the connections to switch the incoming voltages to the right level. Conveniently, such circuits exist for this exact purpose, which is shown in the Figure 0x4b. The circuit on either side is powered by the respective voltage and takes in two inputs, allowing for a bi-directional link between two microcontrollers which operate at 3.3v and 5v. It must also be noted that this circuit has a specific direction in which it must be hooked up to work properly, it is labelled on the device which the high and low sides are, and the microcontroller with the lower voltage connects on specific side and the microcontroller with the higher voltage connects to the opposing side.

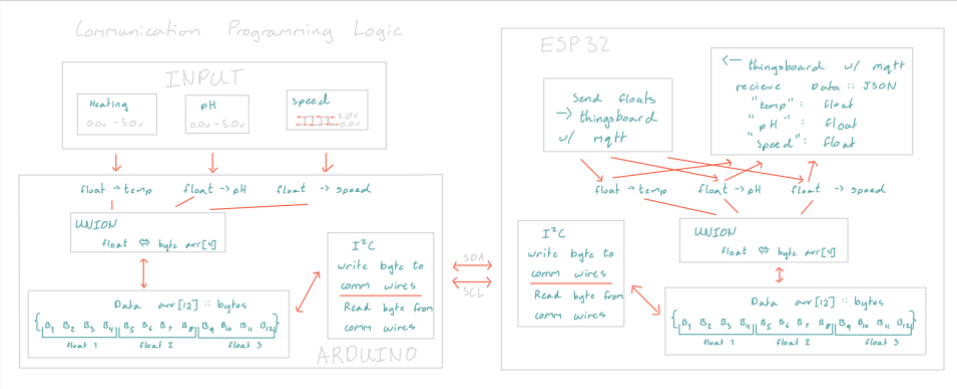

Figure 0x32 Communication Programming Logic

Following the connections being made between the microcontrollers the code which runs on these controllers can be constructed and compiled. In the Figure 0x32 the block diagram shows the logical directions of the program. The essentials for the functioning of the I2C protocol are the unions between floats and byte arrays, and the creation of the complete data byte array to allow for the transfer of all three float values relevant to the whole system. Because floats cannot be sent across the I2C protocol the data must chunk into bytes that can be sent across (I2C allows for single byte transfer). The union data structure in the Arduino language allows for the computer to interface with the float type and its corresponding bytes so that they can be sent across the I2C SDA line without having to use a function that maps the float value to set of bytes which then has to be decoded on the other side. The only data that has to be known ahead of compilation is the position of each float in the complete byte array which is alright considering our system only contains three modifiable variables.

2.5 Communication between ESP32 and User Interface

2.5.1 Purpose

The main purpose of the ESP32 is to connect to the Internet and send the data it has received from the Arduino Uno to ThingsBoard. ThingsBoard is the dashboard used to view data on the three subsystems and control the setpoints of the temperature, pH and RPM which is then sent back to the ESP32. This is both done using the MQTT protocol.

2.5.2 Technical Design Steps

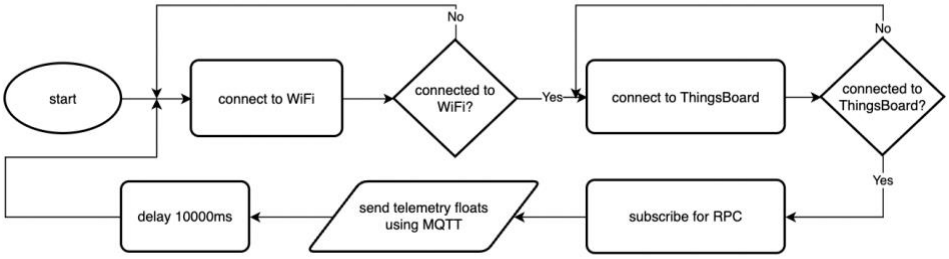

Figure 2.5.1 Sending Data to ThingsBoard Flowchart

The above figure displays the process of sending data on the current readings of pH, temperature, and RPM from the ESP32 to ThingsBoard by subscribing to Remote Procedure Calls (RPC). There are two types of RPC, which are client-side and server-side RPC. The bioreactor uses server-side RPC, which works by sending requests from the platform to the ESP32. A getUserVals method is used to receive user values using conditional statements depending on which specification is chosen, and ThingsBoard sends a request to the ESP32 to change the setpoints of the temperature, pH or RPM using the setUserVals method. The method for setting user values involves parsing through a float data array and inserting the values for temperature, pH, and RPM in their respective variables.

2.5.3 Visualisation

At the start of the project, the User Interface (UI) as seen in Figure 2.5.2 was created and presented during the initial proposal presentation. This UI features a drop-down menu to select different specifications, plus and minus buttons to change setpoints and displays the current reading.

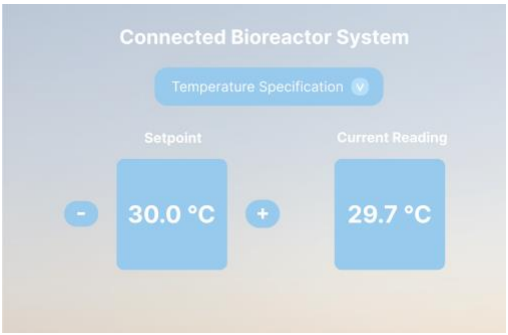

Figure 2.5.2 Proposed User Interface

Figure 2.5.3 ThingsBoard User Interface

At the end of the project, the UI in Figure 2.5.3 was used as the final UI as created on ThingsBoard. On this dashboard, the user can view the current values for the temperature, pH and RPM as well as see how the values fluctuate on the Timeseries Line Chart. If needed, the user could adjust the setpoints of the values using the control knobs in the middle.

3 Overall System Integration and Summary

3.1 Testing

Connectivity - All the tests were conducted from a command line on things board which was used to send requests to the Arduino via the esp32. Test values for RPM, pH and

Temperature were sent and received successfully. This is evident by console logging the received values via the Arduino's serial output.
This test proved the connectivity of the system to be at level deemed acceptable. Although the front end lacked a GUI this would not be difficult future implementation.

Heating Subsystem - A target temperature of 0°C was chosen which was below ambient temperature and so a HIGH signal should be detected on the heating output pin to turn the heater. This did not happen due to the thermometer reading a value of 77°C (evident from the serial output on Arduino uno). A similar problem occurred when the target was set to 100°C to turn of the heater.
Tests to see if the HIGH output signal would turn on the heater on to begin with were not conducted due to time constraints and negligence.

pH Subsystem - Target pH of 0 was set. This input would expect a HIGH output to the alkaline pump pin which was not true. This was fixed when it was noticed the wrong pin was being referenced in the code. After this the alkaline pump worked based on the target input of 0. Likewise acid pump turned on after a target pH of 10 was chosen.
The pH probe also constantly fluctuated at a value between 5 and 8 which was deemed to be due to random error in the probe which was in an acceptable margin of error.

Stirring Subsystem - An RPM of 500 was chosen and the motor increased to this RPM as expected although the RPM was fluctuating at around ±100. When the reference RPM was changed to 1000 RPM, the motor did not change speed, although the value received was the reference RPM. However, when the code is reuploaded, the motor started to spin at 1000 RPM, but stop after a few seconds.
Although we were able to input a reference value for the motor to spin, we were not able to change it while it was spinning at did not spin at around ±20 RPM of the reference value. To improve this, a PID controller could be used to cause less fluctuation in the RPM.
We are able to control the motor, however if it starts to stir beyond the range of ±20 RPM, this could affect the rate of reaction in the bioreactor

3.2 Final Demonstration Outcome

A common problem with all the three subsystems is the communication. Although the stirring and the pH system had moderate success, the value of the temperature could not be read. The calibration, electrical or logical errors occasionally worked but would often either give incorrect readings for the temperature. This meant the logic would respond with bad signals causing the heater to turn on and off at inappropriate times.

4 Appendices

pH value	Output voltage to Arduino (V)
4	1.83
7	1.65
10	1.47

Table pH2: Calibration of pH System

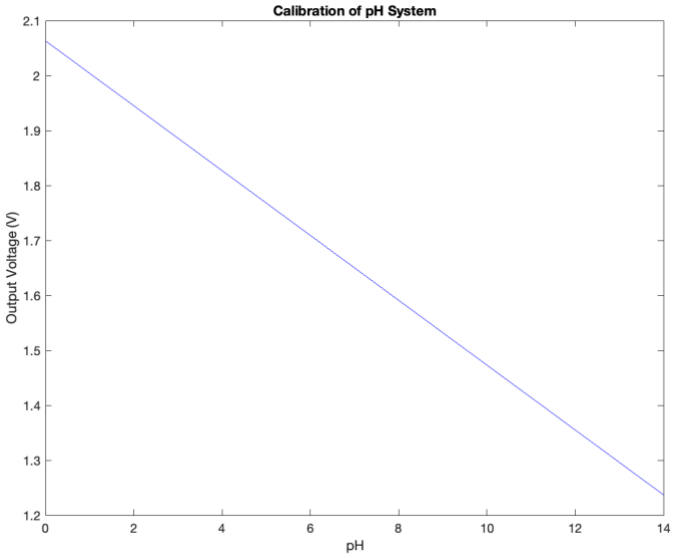

Figure pH3: Graph plotted to obtain the relationship between pH and voltage

Temperature °C	Voltage across thermistor (V)
20	2.15
21	2.13
22	2.11
23	2.09
24	2.07
25	2.06
26	2.04
27	2.03
28	2.01
29	1.99
30	1.975
31	1.957
32	1.94
33	1.922
34	1.905
35	1.89
36	1.87
37	1.85
38	1.835
39	1.818
40	1.8

Table H1, shows the data obtained for calibration of the heating system

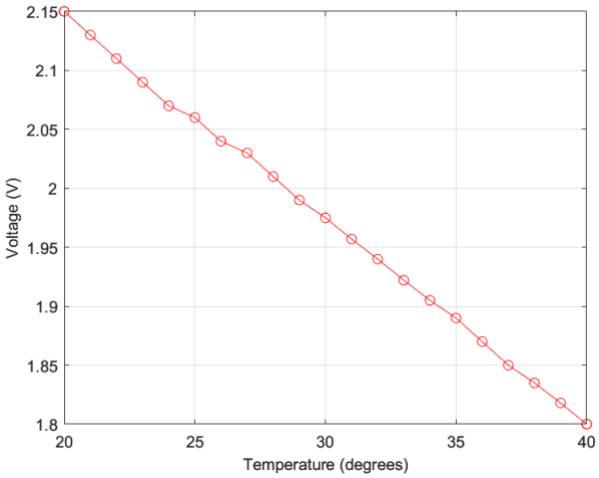

Figure H3, shows the plot of temperature against voltage across thermistor for calibration of the heating system. (Equation of the line y=-0.0172403x+2.49064)