Challenge 2: Global Health Care

CS-EEE Specialist Team Final Report

- Authors:
  - Faisal Hassan
  - George Ghetiu
  - Qayyum Erywan
  - Peichen Li
  - Alexio Nugroho
  - Simon Zhu
  - Jeremy Man
  - Daniel Daramola
  - Darius Jankunas

1. Introduction **2**

2. Subsystem Descriptions **3**

2.1 Subsystem: Heating 3

2.2 Subsystem: Stirring 5

2.3 Subsystem: PH 6

2.4 Communication between Uno and ESP 7

2.5 Communication between ESP and cloud, plus data logging and visualisation 8

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

4. Appendices **11**

5. References **13**

1. Introduction

Authors: Peichen Li

Tuberculosis (TB) is an infectious disease and is among the top 10 causes of death worldwide, affecting Africa in particular. It is a disease caused by *Mycobacterium tuberculosis* (MTB) bacteria and it generally affects the lungs. Tuberculosis is a chronic and slow-onset infectious disease that mainly affects young people. The disease is mainly spread when individuals with active TB cough, spit, speak or sneeze (airborne particles). According to the *Global Tuberculosis Report 2022* by the World Health Organisation (WHO), there were 10.6 million new cases of TB worldwide in 2021 and 1.6 million out of these were deaths. In Uganda in particular, in 2021, tuberculosis caused 14 thousand deaths. Tuberculosis deaths have already fallen significantly in recent years due to the World Health Organisation actively promoting the BCG vaccine in Uganda.

The vaccine primarily used against the TB bacteria is the Bacillus Calmette-Guérin (BCG) vaccine. For the WHO to be able to offer BCG vaccination in Africa, a sufficient and affordable supply of vaccines is needed, and hence more vaccine plants have to be built.

For the team's Engineering Challenge 2, the aim was to design a BCG vaccine production plant as part of the BCG vaccine manufacturing facilities in Mbarara, Uganda. As CS-EEE students, the task was to design, develop and test the circuitry, software and interfaces required to remotely monitor and control the parameters of a small-scale bioreactor control system.
The overall system is shown in figure 1:

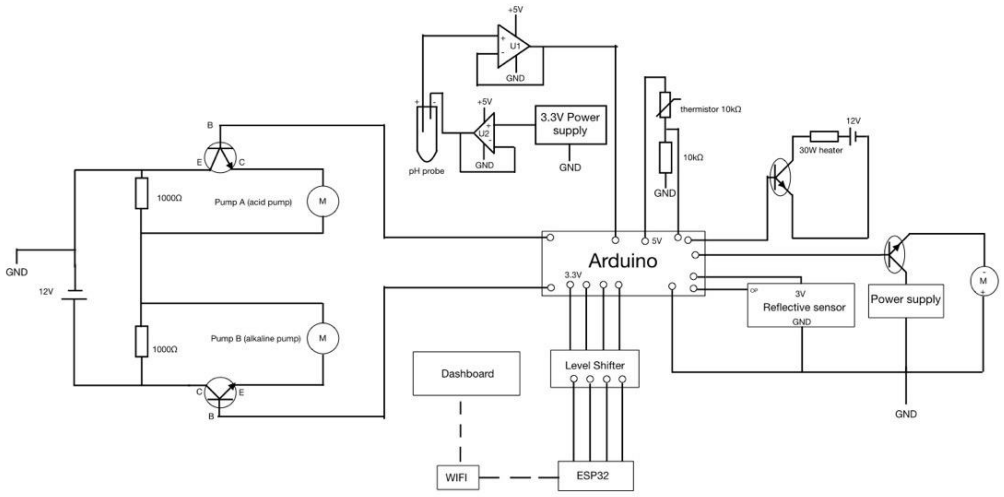


Figure 1: Overall diagram for the whole system

The overall system designed by the team fits the solution to the challenge perfectly. The subsystems involved are heating, stirring and pH. The circuit of the heating subsystem constantly takes readings and sends information to the microcontroller to keep the temperature within the desired range and precision. The IR LED and phototransistor in the stirring subsystem is arranged as a reflective sensor at the gear wheel. This will output 2 pulses per revolution which need to be calibrated in the Arduino to adjust the stirring speed. In the pH subsystem, alkaline/acidic liquid will be sent into the main chamber at a rate dependent on how

far the desired pH value is from the measured value to adjust the pH value of the system. Furthermore, a user interface was designed to provide the user with real-time system information and allow the user to control and adjust system parameters.

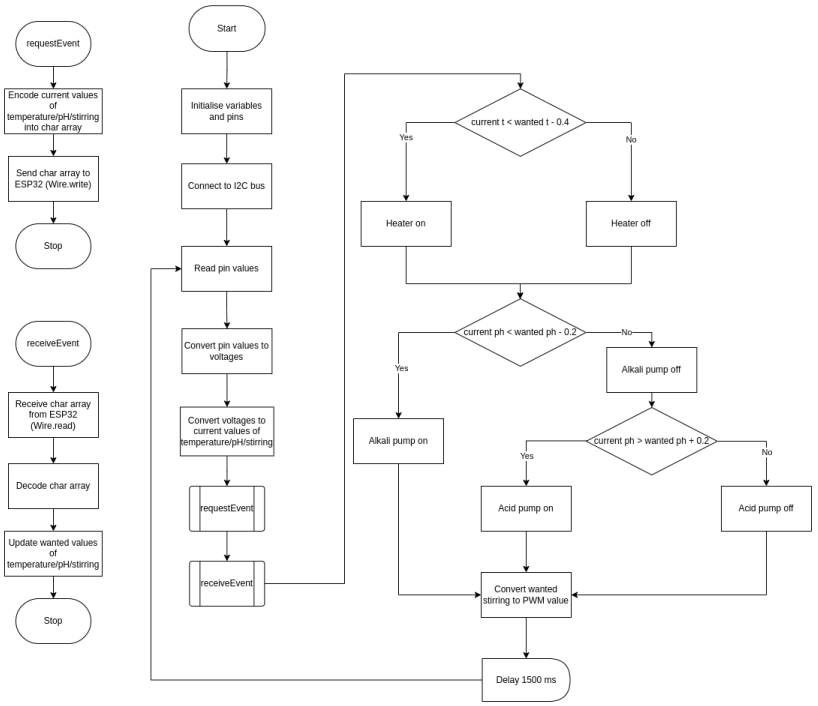


Figure 2: A flowchart representing the working mechanism of the Arduino

2. Subsystem Descriptions

2.1 Subsystem: Heating

Authors: George Ghetiu, Simon Zhu

Purpose of the heating subsystem and specification :

The heating subsystem is used to monitor and regulate the temperature of the water, in order to provide the optimal conditions for yeast growth. The subsystem is designed so that the user can select what temperature they want the water to be, and the system would respond by setting the water to that desired temperature. The range that the user can select is between 25°C and 35°C and the subsystem has to sustain the desired temperature with an accuracy of ±0.5°C.

Circuit design and how the subsystem works :

The circuit for the heating subsystem consists of two parts: the temperature sensing and measurement, and the heating. The two parts are connected through an Arduino microcontroller.

The left-hand side of figure 3 is responsible for sensing and measuring the temperature of the water. This part consists of a voltage divider, with a 10kΩ NTC thermistor in series and a 10kΩ resistor, powered using the 5V pin on the Arduino. The voltage across the resistor is measured

by a pin on the Arduino. Since it is an NTC thermistor, the resistance of it will decrease with temperature. So as temperature increases, the voltage across the resistor (and therefore the voltage into the read pin) would increase.

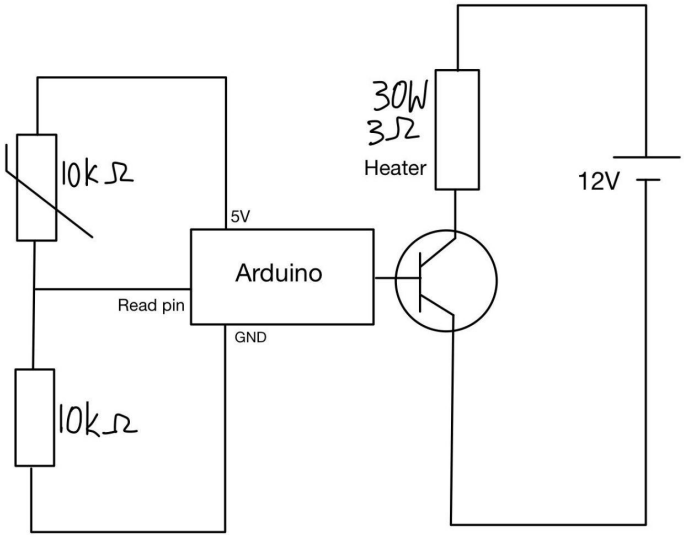


Figure 3: A circuit diagram representing the heating subsystem

The right-hand side of figure 3 is responsible for heating the water. It consists of a transistor which is used as a switch to turn the heater either on or off. The 30W, 3Ω heater is powered by a 12V power supply. The signal to determine whether the transistor is on or off is sent by the Arduino. This is a result of comparing the actual temperature of the water and the desired temperature of the water. If the desired temperature is higher than the actual temperature by 0.4 °C or more, a signal would be sent to the transistor for it to turn on. Consequently the heater would turn on. Otherwise, no signal would be sent and the heater would stay off.

The conversion is performed by the Arduino's ADC, or Analog-to-Digital Converter. This is a device that converts an analog voltage (a range of continuous values) into a digital value (a range of discrete values). The ADC has a defined number of "steps" that it can convert within the voltage range. For example, if an Arduino has a 10-bit ADC with a reference voltage of 5 volts, it will be able to measure a voltage range of 0 to 5 volts in 210 (1024) steps. This means that each step will represent a change in voltage of 5/1024 = 0.0049 volts.In order to calculate the actual p.d. across the resistor, the function (5.0 x ADC voltage) / 1023.0 is applied to the ADC value that had been read by the Arduino.

Subsequently, an experiment was carried out to map temperature (independent variable on the x-axis) to a p.d. of the resistor (y-axis). A straight line of best fit of the form y = mx + c was used to calculate the values of m and c. The function was then implemented in the Arduino code to return the temperature for any voltage, according to the aforementioned function.

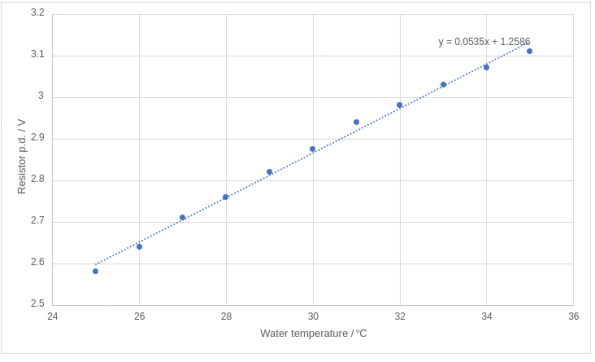


Figure 4: A graph used to map out the calibration of the heating subsystem

The reasoning behind the 0.4 °C temperature difference is that the heating element would still be hot, even with the power turned off. As such, the water temperature would increase by 0.4 °C on average before stabilising. In addition, as the temperature starts to drop, the heater would turn on when the actual temperature is 0.4 lower than the desired temperature, keeping it within the 0.5 °C range that is required.

2.2 Subsystem: Stirring

Authors: Daniel Daramola

Purpose of the stirring subsystem and specification :

The task of the stirrer is to ensure the contents of the bioreactor are well mixed. This is needed as the heater and pH pumps are unable to evenly distribute heat and acid/alkaline solutions to the vaccine. It is important that an adequately concentrated solution is achieved, as there are several chemical reactions taking place in the bioreactor. Each component of these reactions (temperature/pH change) need to be congruous throughout the whole solution for efficient reactions to occur and to achieve the required solution. Additionally, the stirrer helps to provide oxygen for microorganisms present in a bioreactor which is essential for their growth and metabolism. To ensure the solution is stirred appropriately, the speed of the stirrer is adjustable to a set speed from a range of 500-1500 rotations per minute, this is to preserve the chemical integrity of the solution and reduce the risk of damaging it.

Circuit design and how the subsystem works :

The stirring subsystem consists of a motor and a sensor.

In order to drive the motor, an external power source is used as the microcontroller (Arduino Uno) is not able to provide enough current to drive the motor. A transistor is used as a switch to control the voltage being supplied to the motor from the power supply. By altering the voltage supplied to the transistor the speed of the motor can be controlled. To vary the voltage at the transistor a PWM digital pin on the Arduino Uno is used, this is done by utilising the analogWrite() function where the pin will output a voltage between 0-5V. In order to input the correct PWM value for a selected rotations per minute, test PWM values were used to map to the rotations per minute they produce, which was measured by a tachometer. A function was then made to map the correct PWM value for the desired rotations per minute (using the graph shown in figure 6).

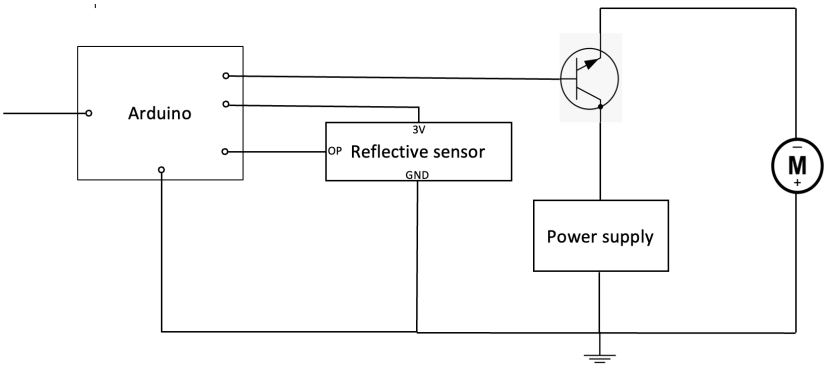


Figure 5: A circuit diagram representing the stirring subsystem

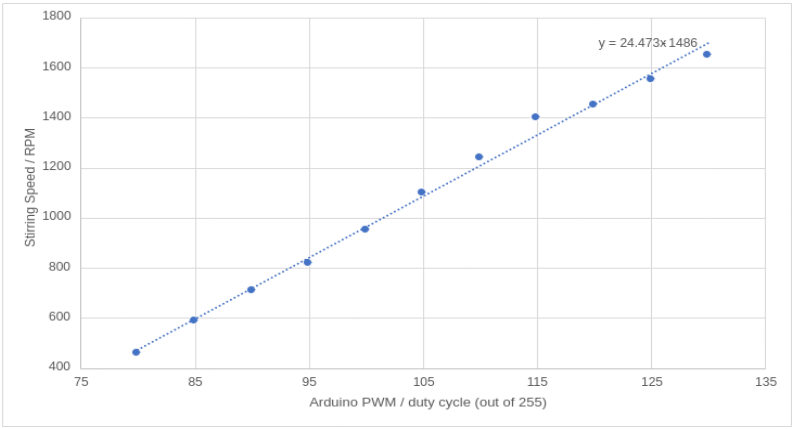


Figure 6: A graph used to map out the calibration of the stirring subsystem

The sensor is composed of an Infra-red LED and phototransistor which are arranged to form a reflective sensor that is supplied with 3V by the Arduino. The reflective sensor sends a pulse (a square wave where the signal is formed of a HIGH and LOW state) to the microcontroller every time light is reflected on it, so it is placed facing the gear wheel on the output shaft in such a way that it outputs two pulses per revolution. Using the pulseIn() function, the time taken for the HIGH state and LOW state of each pulse is calculated and by doing further calculations, the rotations per minute of the motor can be found.

2.3 Subsystem: PH

Authors: Jeremy Man, Darius Jankunas

Purpose of the pH subsystem and specification :

The pH subsystem is used to measure the pH of the solution and maintain the pH in a certain range. The acidity of the substance can be altered to provide the optimal conditions in the bioreactor. The default pH is 5, but the pH setpoint is adjustable by the user. The possible range is 3-7. The subsystem consists of a pH sensor and pH pumps. The pH sensor takes the measurements of the current value. There are two pH pumps - acid and alkali. These are used to adjust and regulate the pH of the main solution.

pH Sensor :

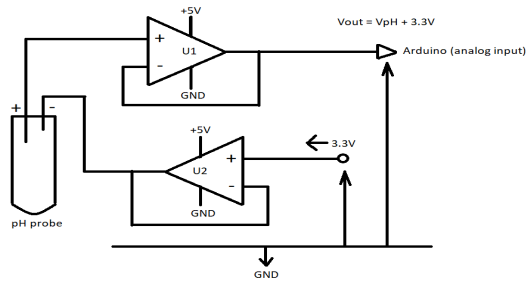


Figure 7: A circuit diagram representing the pH sensors

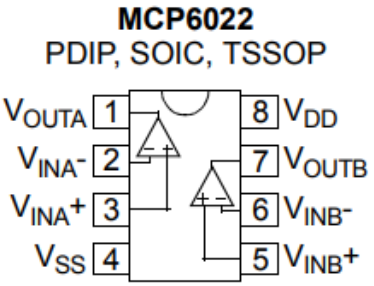


Figure 8: MCP6022

System Components:

- MCP6022 dual operational amplifier
- pH probe (PH-100ATC Probe)

How it works :

The pH sensor measures the pH by recording a voltage. The voltage can be in the range -500mV to 500 mV depending on the concentration of positive H^+ ions and negative OH^- ions in the solution. As the arduino can only map input voltages between 0V and 5V, it cannot take negative voltage input which means that a pH greater than 7 cannot be measured. Therefore, an input offset is required. Here, an offset voltage of 3.3V is supplied to the pH probe through the negative terminal so that the output voltage of the pH probe at the positive terminal is always positive. There are two operational amplifiers, U1 and U2 (inside MCP6022) in the circuit. They act as buffer amplifiers so the voltage gain is zero. The purpose of buffer amplifiers is to provide a much lower impedance at the output. This allows a higher precision measurement. The output voltage of the pH voltage is then read by the Arduino Uno as an analog input. The actual pH

sensor value would be the analog input (pH voltage) - 675 (3.3V) as the analog-to-digital conversion resolution of the arduino of 10 bits. The sensor value is mapped into pH values using the equation acquired in the calibration.

The sensor is calibrated using solutions with known pH. Solutions used:

- Buffer solution with pH = 4.0 (Red)
- Buffer solution with pH = 5.0 (Colourless)
- Buffer solution with pH = 7.0 (Yellow)
- Buffer solution with pH = 10.0 (Blue)

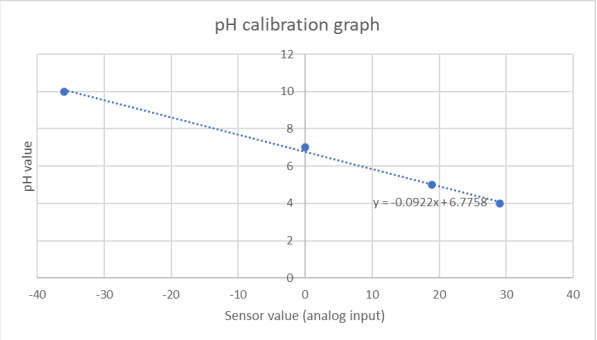


Figure 9: A graph used to map out the calibration of the pH subsystem

pH pumps :

System Components :

- 2*1000 ohm resistors
- 2*IRLB8748 transistors
- 1*12V power supply (battery)

Circuit design and how it works :

The pH pumps are used to control the pH of the solution by pumping either an alkaline or acidic solution into the main solution based on how far away the desired pH value is from the measured value. If the desired pH value is more than 0.2 away from the measured value from the pH sensor, depending on whether the pH of the solution is too high or low to the setpoint, the acid or alkali pump will turn on respectively.

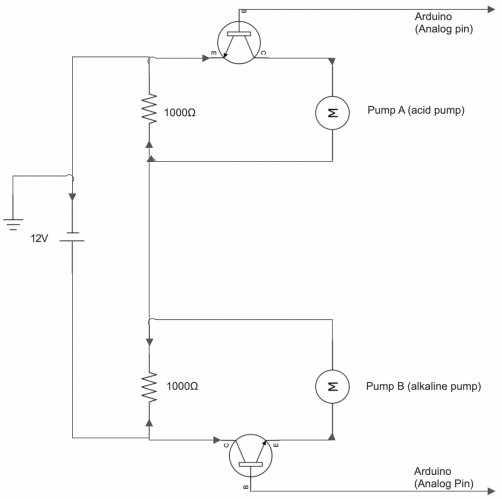


Figure 10: A circuit diagram representing the pH pumps

The system is powered using a 12V power supply with the voltage split evenly between the two pumps (Pump A and Pump B) using a voltage divider system. Each pump is powered by a brushed DC motor. Each motor is also connected to a IRLB8748 transistor, which when sent the signal from the Arduino closes and opens as defined by its algorithm. Within 0.2 of the desired value there is no signal sent to the pH pumps, however this level of accuracy is adequate for this application.

2.4 Communication between Uno and ESP

Authors: Qayyum Erywan

Purpose of Uno - ESP communication and specification :

To allow communication of data and maintain consistency between the Arduino Uno and ESP32. Examples of such data are: Readings from each subsystem and New values to adjust each subsystem.

How it works:
To allow communication between the ESP32 and Arduino Uno, the team implemented the I2C Communication protocol. This is because the I2C protocol has enough functionality to achieve what is needed, and both the ESP and Uno already have an existing library to make use of it (i.e, Wire library). By using the I2C protocol, the ESP is set as the Master while the Uno is set as the Slave.

Figure 11: ESP32 to Arduino connectivity

However, the ESP and Uno both operate at different voltages which are 3.3V and 5V respectively. Connecting them directly to each other would damage the ESP as it operates at a lower voltage. Thus, the use of a Bidirectional Logic Level Shifter is required. This will act as an intermediate between the two microcontrollers. The ESP is connected to the Low-voltage side (3V) while the Uno is connected to the High-voltage side (5V). This will allow the signals being sent to shift from 5V to 3V and vice versa maintaining their consistency across boards. The level shifter will have a total of 4 pins which are the power, ground, SDA and SCL. SDA or serial data is used to transmit data and SCL or serial clock is the clock used to synchronise communication between the Arduino and ESP..

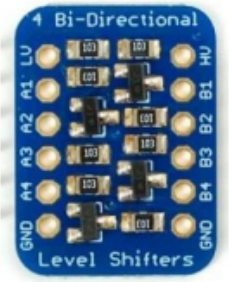


Figure 12: Bi-directional Level Shifter

Master:
The ESP requests data from the Uno where it receives readings from sensors. The data is concatenated into a single string (i.e. T25.00P5.00S500 where T is temperature, P is pH and S is stirring) which the ESP will decode and allocate the individual readings to their respective variables. The ESP also sends data to the Uno whenever the wanted output of a subsystem is changed. This data is sent in the same format as the data received.

Slave:
These strings containing sensor current values are concatenated into a single string to be sent over the I2C protocol when the master requests for it. The Arduino also receives the same format of data where it will have to decode the data from the master and adjust the desired values of subsystems for the proper function of the algorithm.

2.5 Communication between ESP and cloud, plus data logging and visualisation

Authors: Alexio Nugroho

Purpose of ESP - cloud communication and specification :

The dashboard was used to display the results of the three subsystem sensors in a way that is easy and pleasant to look at. In addition, the dashboard is used by users to change the desired values of the bioreactor.

Visualisation:

Figure 13 is the initial wireframe used to have a rough outline of the dashboard. Meanwhile the figure 14 is the final image of the dashboard that is actually used in the live demonstration. The design has changed due to the limitations of what is available on Thingsboard. As seen above, there are three card widgets to display the reading as it is and three chart widgets were used to display values of the last two minutes. To change the desired values, a control knob, horizontal slider and plus/minus buttons were used for temperature, stirring and pH respectively.

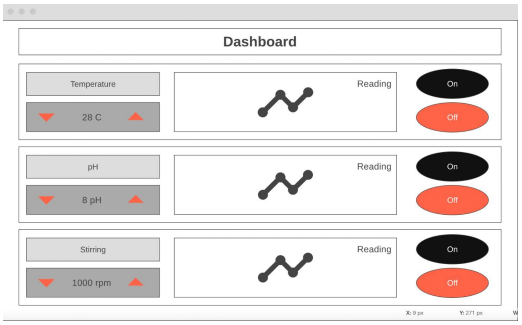


Figure 13: Initial Wireframe

ESP32 to Thingsboard to ESP32:

After receiving data from the Arduino, the ESP32 will send data to the Thingsboard through a WiFi connection (eduroam). To find the device on the internet, each device on the Thingsboard will have a randomly generated access token. In the ESP32 code, there will be a define TOKEN and a define THINGSBOARD_SERVER (thingsboard.cloud). The ESP32 will then connect to that device and send the sensor readings as telemetry data. In the bioreactor, there will be three telemetries which are Temperature, Stirring and pH. The data will be displayed in a number value and in a line chart.

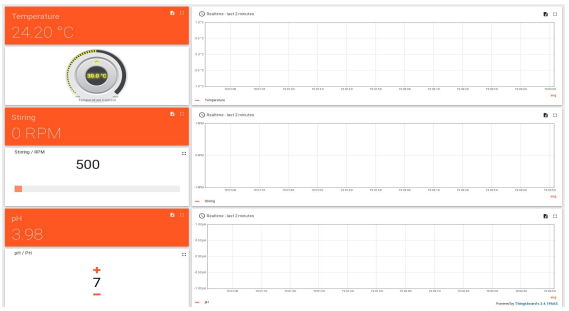


Figure 14: Thingsboard Dashboard

Control widgets are then used to send the user desired value back to the ESP32. These control widgets use Remote Procedure Calls (RCP) to set and get values. Each control widget will have two methods. One to change the actual data and one to return the actual data value. In the bioreactor design, only the set method is used as only the control widgets send data to the ESP. Each control widget method has a different name to differentiate between them. These are: setTemperature, setStir and setPH. The ESP32 will first need to subscribe to the RPC and request a value from the Thingsboard. The RPC method will get called and transmitted to the ESP whenever there is a change in the control widget.

3. Overall System Integration and Summary

Authors: Faisal Hassan

➔ **Heating** :

The heating subsystem was successful in the demonstration, although it was somewhat slow in reaching the desired temperature because of the use of the analogWrite() function with an argument of 60.

Encountered problem and Solution :

The transistor was originally getting too hot to the point of melting the breadboard since digitalWrite() was used to send a signal to the transistor. To solve this issue, analogWrite() was used instead with a value of 60. This value supplied the transistor with enough power so it would turn on, but not so much that it would overheat.

➔ **Stirring** :

In the final demonstration, the stirring subsystem was not working as planned. Thus, this subsystem has not fully met the conditions of the specification.

Encountered problem and Solution :

The rotations per minute measured on the motor would not stay at a set value but vary at a range of about +/- 50 RPM from the desired speed of the user. This is due to the fact that the supplied voltage of 6V from the battery is too high for the motor. To resolve this, a PID control can be implemented in the code to regulate the input voltage to the transistor. It would work by calculating the error of the measured speed of the motor. Then it would apply the necessary calculations to obtain a value to re-enter into the analogWrite() function. As a result, the speed would be adjusted accordingly to give the correct RPM.

➔ **pH** :

Like heating, the pH subsystem can be considered successful, as it was able to bring the solution in the bioreactor to the desired acidity. From the observations made during the demonstration, the error range was determined to be around 0.1, making the subsystem accurate.

Encountered problem and Solution :

A problem with the pH pumps was the overheating of the transistors used. Initially the team used ZTX450 transistors. However this small transistor was not meant for the higher currents involved in driving larger DC motors and overheating was experienced as a result of overdriving the smaller transistor above its rated specifications. IRLB8748 transistors were used instead and this issue was resolved. The larger IRLB8748 not only has a larger surface area than the ZTX450, but also has an aluminium heatsink, allowing for a much higher rate of heat dissipation.

To further increase accuracy, the conditions for starting and stopping the pumps have to be more detailed, because there is the risk of the pumps overshooting. To solve this, the pumps are not activated when the difference is less than ±0.1, which is reasonably likely in the pH subsystem (due to the fluctuation of the sensor reading).

➔ **Connectivity** :

The connectivity between Arduino, ESP and Dashboard was working as expected in the definitive version of the bioreactor. Values were being sent both ways with great efficiency and this allowed the smooth functionality of all the other subsystems, integrating the various parts of the project. The connectivity solution meets the assignment requirements.

The overall integration was successful and the team's implementation is suitable for the Ugandan bioreactor needs if the specified change is made to the stirring subsystem. The other two conditions (temperature and pH) can be controlled reliably with the current solution. Thus, the vaccine plant would meet its demand by implementing the team's design.

4. References

"Global Tuberculosis Report 2022." *World Health Organization*, World Health Organization, https://www.who.int/teams/global-tuberculosis-programme/tb-reports/global-tuberculosis-report-2022.

MCP6022 - Smart | Connected | Secure | Microchip Technology. https://www.microchip.com/en-us/product/MCP6022.

Thingsboard. "Widgets Library." *ThingsBoard*, Thingsboard, https://thingsboard.io/docs/user-guide/ui/widget-library/.

"Connectivity." *Bioreactor Connectivity Overview*, UCL, https://moodle.ucl.ac.uk/mod/resource/view.php?id=4307628.

5. Appendices

Raw data for experiments:

[1]- Heating

Temperature	Voltage (resistor)
25	2.58
26	2.64
27	2.71
28	2.76
29	2.82
30	2.875
31	2.94
32	2.98
33	3.03
34	3.07
35	3.11

[2]- Stirring

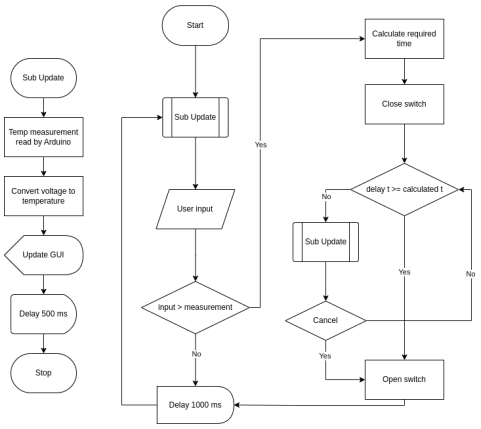
Stirring Speed	PWM value
460	80
590	85
710	90
820	95
950	100
1100	105
1240	110

1400	115
1450	120
1550	125
1650	130

[3]- pH

Sensor value	pH
29	4
19	5
0	7
-36	10

[4]- Flow chart used previously (first stages)



[5]- Code used for all subsystems : https://github.com/FaisalBinHassan/Bioreactor