Challenge 2: Global Health care

*EEE-CS Specialist Team Final Report*

Authors: Akhere Ghinlan, Brandon Tan, Dali Zhao, Hassan Sharif, Jingyi Zhang, Marc Tan Teh, Noan Le Renard, Rikaz Rameez, Wan Hee Tang, Xiaowen Li and Yansong Huang

**1** **Introduction.................................................................................................................................... 2**

**2** **Subsystem Descriptions..................................................................................................................4**

**2.1 Heating Subsystem…………….........................................................................................4**

*2.1.1*  *Overview.................................................................................................................4*

*2.1.2*  *Heater Subsystem ...................................................................................................4*

*2.1.3* *Thermistor Subsystem…………………………..………………………………………..5*

*2.1.4 Code……………………………………………….………………………………………..5*

**2.2 Stirring Subsystem………….............................................................................................6**

*2.2.1* *Overview……......................................................................................................... 6*

*2.2.2* *Motor Subsystem.................................................................................................... 6*

*2.2.3*  *Photointerrupter Subsystem......................................................................... ……..6*

*2.2.4* *Code…………………...............................................................................................7*

*2.2.5* *Testing and analysing the stirring subsystem.........................................................7*

**2.3 pH Subsystem……………….............................................................................................8**

*2.3.1*  *Overview.................................................................................................................8*

*2.3.2*  *pH Probe Subsystem...............................................................................................8*

*2.3.3* *Pumps subsystem....................................................................................................8*

*2.3.4* *Code........................................................................................................................9*

*2.3.5*  *Results.....................................................................................................................9*

**2.4 UI Subsystem………………........................................................................................... 10**

**3 Overall System Integration and Summary…………………………………………………….11**

**4 Non-technical considerations of a real vaccine project in Mbarar..........................................12**

**5 Appendices.................................................................................................................................... 13**

**5.1 Pulse Width Modulation.................................................................................................13**

**5.2 Code .................................................................................................................................14**

*5.2.1* *Arduino code…………………………………………………………………………....14*

**5.3 Testing the application....................................................................................................17**

**5.4 Figures...............................................................................................................................18**

# **1 Introduction**

**Section Authors:** Noan Le Renard (Team Leader), Jingyi Zhang (Liaison Engineer)

In an increasingly developed world, tuberculosis kills 1.5 million people per year. Whilst the continuous increase of education, amongst other factors, will help reduce this number, this is a very long term plan and won’t save lives in the short term. To circumvent this, we are working with the World Health Organization in order to offer vaccines to Uganda, a country that is ravaged by this epidemic.

Our group is part of a team that grows the organism used in the antigen manufacturing of the typical vaccine production process. This team is composed of Biochemical Engineers, Chemical Engineers, Biomedical Engineers, Civil Engineers, Mechanical Engineers, Electronic and Electrical Engineers and Computer Scientists.

This project needed a suitable environment with strict parameters in order to grow the organism, this part was led by our group, composed of Electronic and Electrical Engineers and Computer Scientists. This environment needed these parameters: a temperature between 25-35°C within ±0.5°C of the set point, a required stirring speed range between 500-1500 RPM maintained to ±20 RPM, and to maintain the pH at 5.

**Figure 1: System Overview**

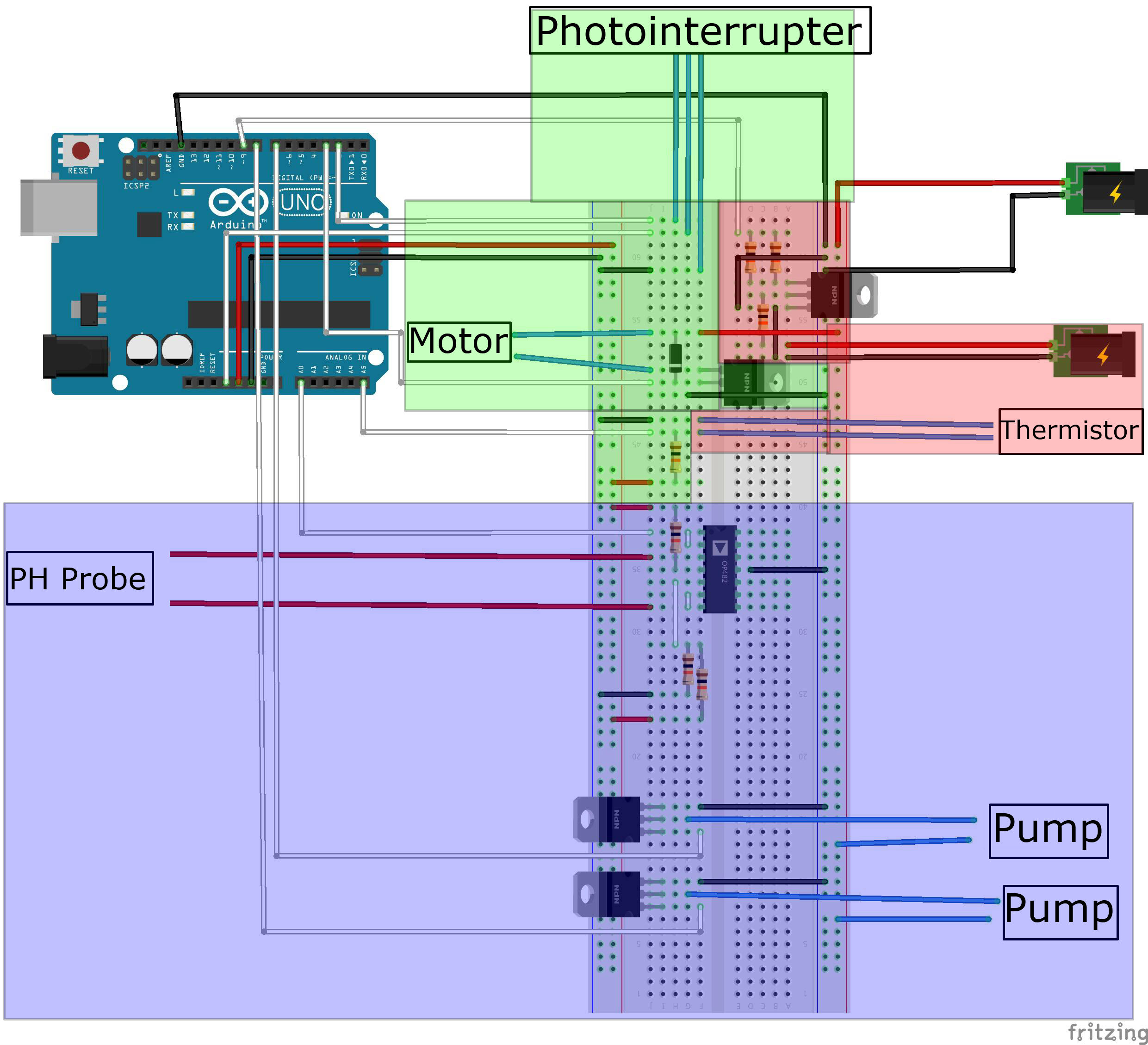


Fig1 represents the overview of our team’s systems. The subsystem highlighted in blue represents the pH subsystem, the one in red is the heating subsystem and finally the one in green is the stirring subsystem. This diagram shows the overall flow and the structure of our project. All our components are fitted onto the same breadboard, hence all the subsystems go together to make a complete system.

However, this project was conducted in a lab and without many considerations that are quintessential in a real life. Whereas this might have seemed minor for our case, many factors have to be discussed in projects of this magnitude. Bound to every idea is a budget, and the financial considerations for the creation and distribution of a vaccine are tremendous. From the first idea conceived to the assembly chain and finally having been flew by plane into Uganda, the logistics and monetary aspect of this project are paramount in order to obtain a successful product.

Furthermore, the paradigm of vaccines is already a controversial one. Albeit it saves lives, our documentation has shown that Uganda has a more “tribal” approach to medicine; favoring old fashioned natural treatments rather than antibiotics, for example. This is why through discussion we have come to the conclusion that increasing education is an important part of getting rid of tuberculosis. Education regarding the importance of modern medicine such as vaccines will drastically help the percentage of the population that is more willing to get vaccinated, and can help inform the population about the attitude to have in regard to tuberculosis, which will limit the number of new infections.

All in all, there are many different factors to take into consideration in a “real life” situation. Whilst one might say that the West sending vaccines to Africa is just making that region more dependent on westernized countries, one might say that we are just aiding their development as they had to survive harsher conditions. There are a plethora of more questions regarding the ethics and sustainability of this project which we will try and explore.

# 

# 

# 

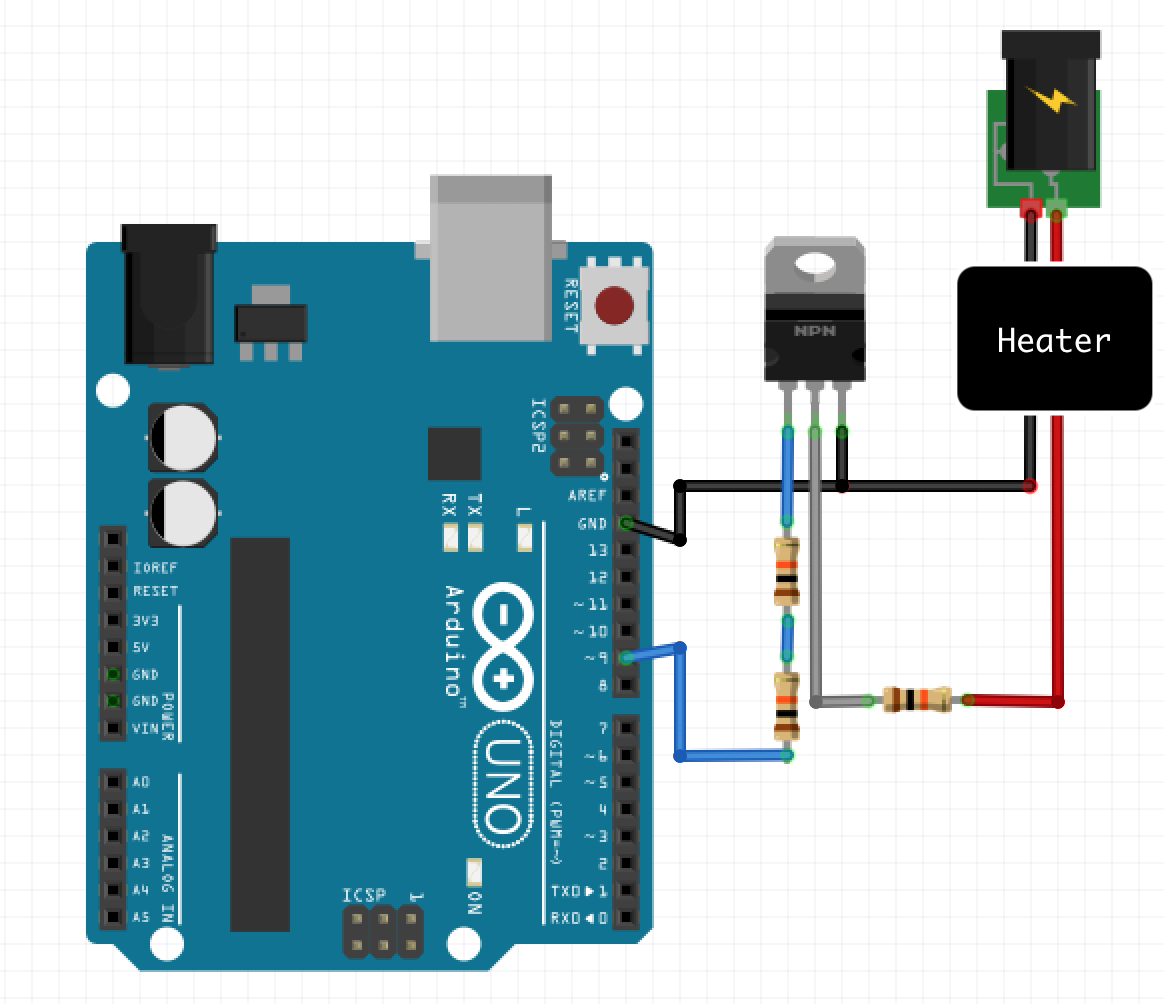
# **2 Subsystem Descriptions**

## **2.1 Heating subsystem = Heater + Thermistor**

Authors: Noan, Jingyi Zhang, Akhere, Hassan

**2.1.1 Overview**

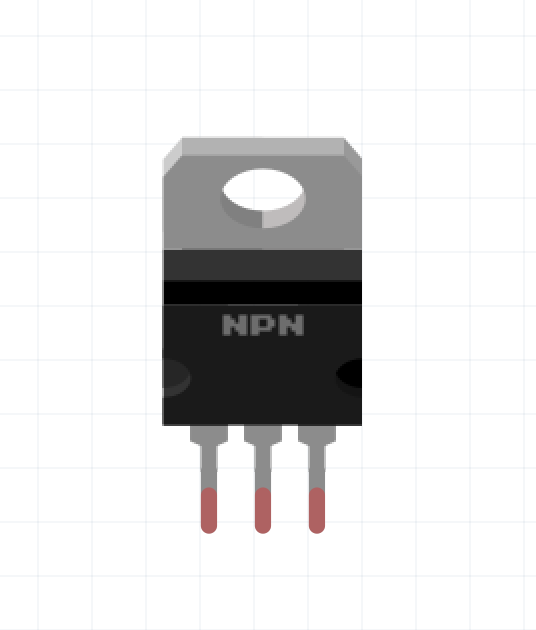
The heating system is made up of a heater subsystems and a thermistor subsystem. The circuit is built for each part separately and an accurate conversion method for the thermistor will be found, then the code is written and a graphical user interface is built.

**2.1.2 Heater subsystem**

The circuit for the heater consists of a transistor, three resistors with values of 10kΩ, 460 Ω and 3kΩ, an Arduino with pin 9 connected to the heater and then to ground, a heater and a power supply with 12V and 4A.

The negative side of the heater is connected with the negative power supply, ground and source of transistor. The positive side of the heater is connected with the positive power supply, a resistor of 460 Ω and then the drain of transistor. The gate of the transistor is connected with two parallel resistors of 10kΩ and 3kΩ, and then pin 9 on the Arduino. See figure 2.1.2a.

**Figure 2.1.2a**



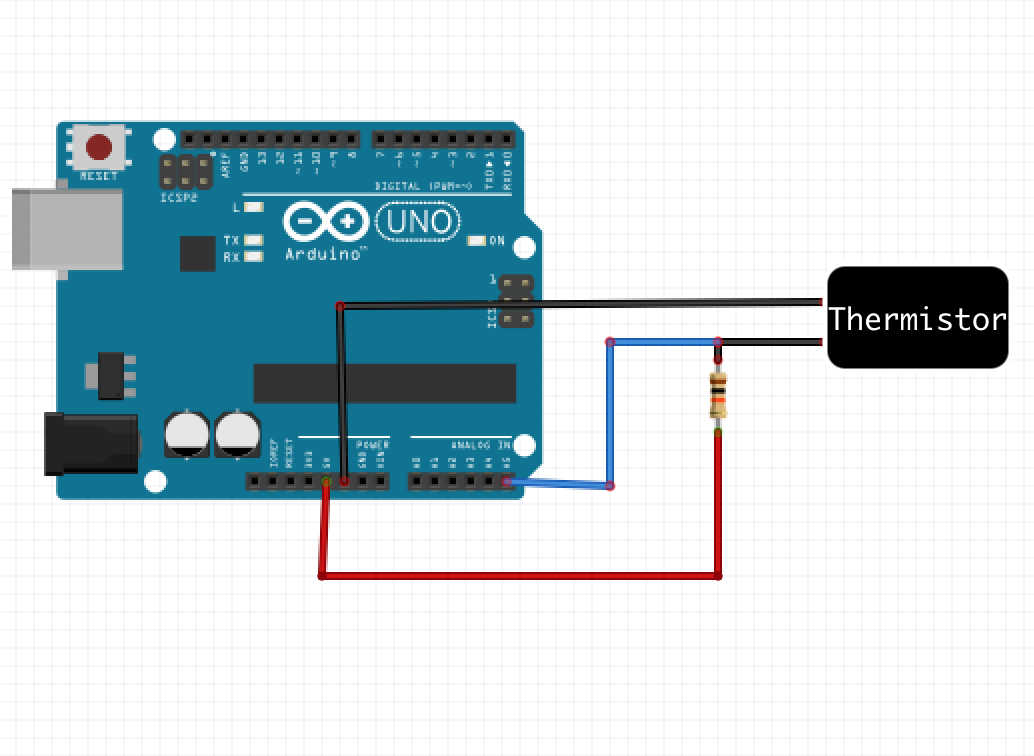
The transistor, shown in figure 2.1.2b, works as a switch to control whether to turn on or turn off the heater. It has three terminals labeled gate on the left hand side, drain on the middle and source on the right hand side. When the gate is written to high, current flows between drain and source and thus a closed circuit is formed. When the gate is written to low, current cannot flow between the drain and the source, so an open circuit is formed.

**Figure 2.1.2b**

Since the drain and source are linked to the positive and negative side of the heater, if there’s no current between these two terminals, the heater will be turned off and if there is current in between, the heater will be turned on. Therefore the heater could be controlled simply by Serial write high or low to the gate using the Arduino. A MOSFET transistor is used as it has a greater surface area which reduces the amount of joule heating so the power efficiency is greater. The excess heat can be dissipated quicker from the MOSFET since there is more surface area in contact with the surrounding air.

Resistors are used due to the high imbalance in voltage and current between the Arduino and power supply (12V, 4A). To protect the circuit and components, a 13kΩ resistance should be applied between the heater and Arduino as according to the data sheet, therefore two resistors of 10kΩ and 3kΩ are connected in series with their additive resistance effect. Furthermore, to reduce the voltage in the middle MOSFET leg, 460 Ω of resistance is used according to the data sheet in order to not burn the transistor.

Through testing, we noticed that even after turning the heater off, either manually or via code, the residual heat kept heating the cup for 5 degrees. Hence, we had to adapt our code to set the reference point 5 degrees lower, or else it would overshoot that target by 5 degrees.

**2.1.3 Thermistor subsystem**

The circuit for the thermistor consists of a thermistor, an Arduino (with pin A5, 5V and ground) and a 10kΩ resistor.

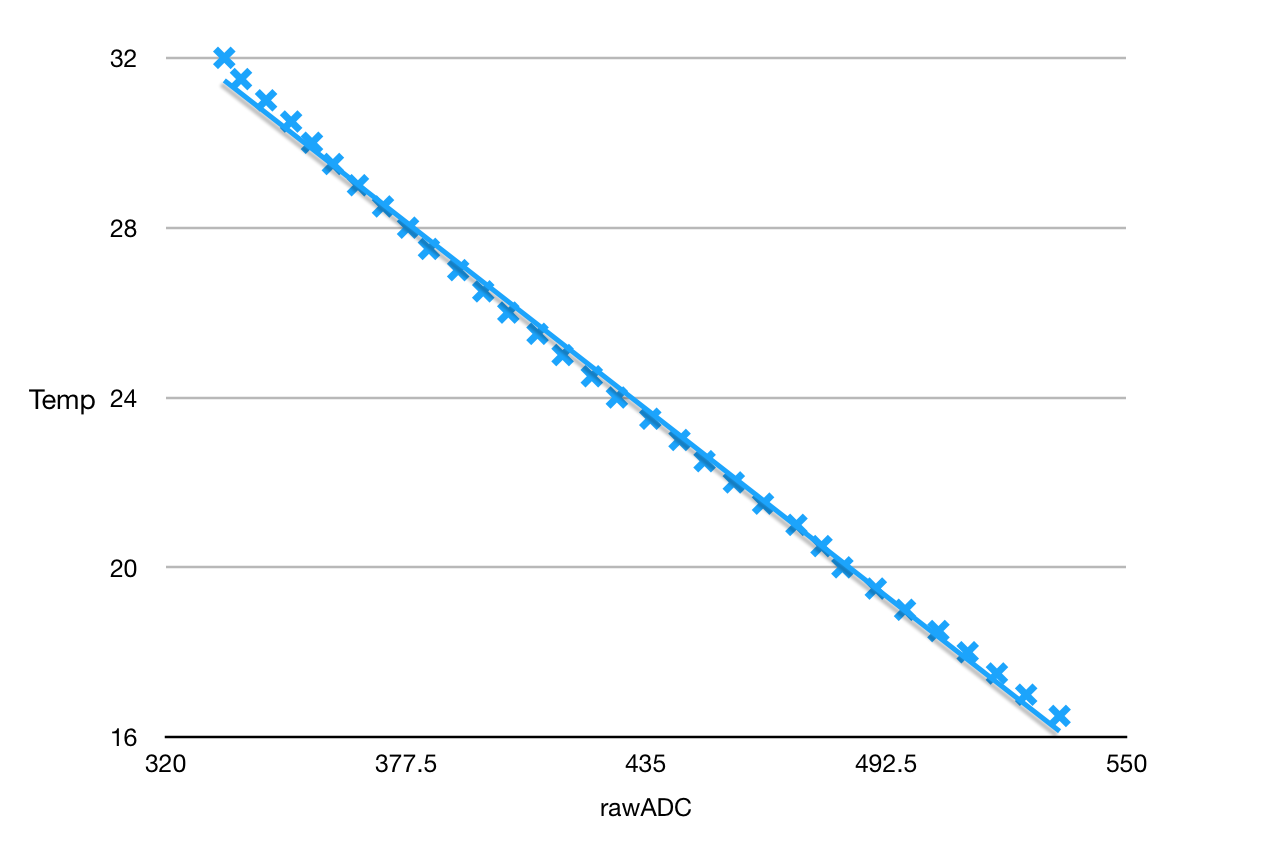
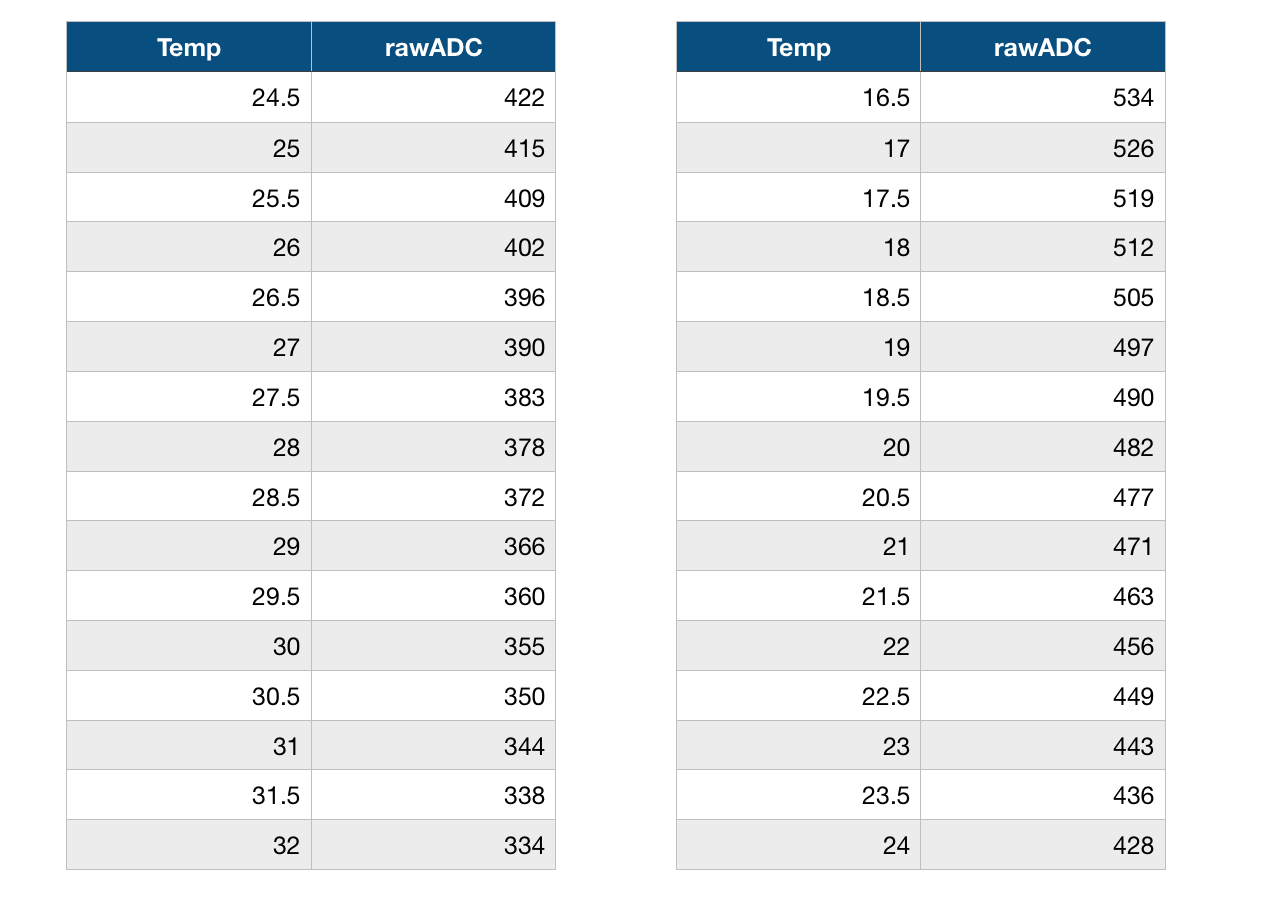
**Figure 2.1.3a**

The thermistor is connected to the ground on one side and connected to 5V on the other side through a resistor. The output of the thermistor is connected with pin A5 of the Arduino, as shown in Figure 2.1.3a. The resistor is used to form a potential divider to avoid a short circuit as the resistance of the thermistor might be too small at some temperatures.

**2.1.4 Code**

To produce the code of this system, an equation is needed to convert rawADC, which is an Arduino reading in range of 0, 1023, into Celsius. As there’s no suitable equations online, the temperature of liquid is measured by a thermometer and recorded with respected rawADC, as shown in figure 2.1.4a. The graph is plotted according to these data, figure 2.1.4b, and the best-fit-line shows a linear relationship between rawADC and temperature with equation:

**temp = - 0.0738 \* rawADC + 56.55**

The arduino code (flowchart in Appendix: 5.4 Figures: Figure 1 and code in appendix) that controls the heater and thermistor has a variable called lower. This variable is by default twenty five, but can be set by the user via the user interface code. When a reading is read from the thermistor, it is converted into Celsius via the equation stated above. If this temperature is above lower, the heater is turned off. If the temperature is below lower, the heater is turned on. The effect of this is that the temperature the user chose can be maintained. The variable lower is actually set to the value the user selected subtracted by five. As a result, the heater turns off before the desired temperature is reached so the water can reach that temperature from the residual heat. The temperature read from the thermistor is sent to the user interface where it can be presented in a graph. 

**Figure 2.1.4a**

**Figure 2.1.4b**

## 

## **2.2 Stirring subsystem = Motor subsystem + Photointerrupter subsystem**

Authors: Rikaz Rameez, Marc Tan Teh, Dali Zhao

**2.2.1 Overview**

The stirring subsystem consists of the motor and the photointerrupter. The motor is connected to pin 3 and the photointerrupter is connected to pin 2. The circuit diagram showing the entire subsystem is shown in Fig 2.2a.

### **2.2.2 Motor subsystem**

The circuit consists of a MOSFET, diode, motor, power supply and an Arduino. Diodes only allow current to flow in one direction. The anode of the diode is connected to the negative terminal of the motor and the drain of the MOSFET while the cathode is connected to the positive terminal of both the motor and the power supply. The diode here is placed across the motor in order to protect the components of the circuit from the back emf (electromotive force) generated by turning off the motor. The MOSFET used is an NPN-type IRF520 MOSFET. This MOSFET acts as an electronic switch that is controlled by the Arduino to drive the motor which requires a higher voltage and current that can be supplied by the Arduino. This is done using an external 6V power supply. However, because the motor can only withstand 3V, only 50% of the duty cycle is applied during Pulse Width Modulation (see Appendix 1) which means the maximum value for analogWrite is 127, which is approximately half of 255. This is used in place of a potentiometer to limit the amount of voltage supplied to the motor. The reason this MOSFET is used is because it can withstand the 3V voltage and 1A current required by the motor. The gate of the MOSFET connects to pin 3 of the Arduino and the source connects to ground while the drain connects to the negative terminal of the motor and the anode of the diode. Lastly, the photo interrupter sensor gives information about the RPM of the motor using the time period between each interrupt. The photo interrupter sends this information through its output pin to pin 2 of the Arduino. The negative terminal of the sensor connects to ground while the 3.3V pin of the Arduino powers the sensor through the positive terminal.

The motor speed is controlled by a voltage output that is controlled by the analogWrite function in the Arduino code. The RPM is calculated using the photointerrupter subsystem and the RPM that the user wants is inputted from the application (this is targetRPM). This value is compared with the current RPM (within a certain error range) and the voltage is lowered if the current RPM is higher than target and increased if current RPM is lower. If they are the roughly the same then the voltage is not changed.

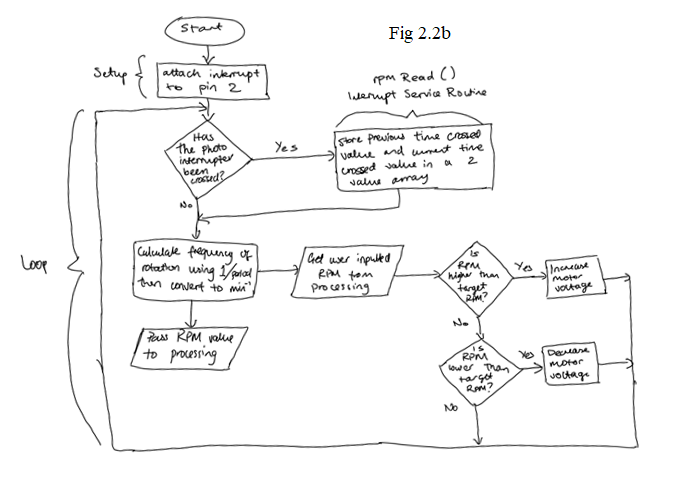
### **2.2.3 Photointerrupter subsystem**

The photointerrupter is outputs a digital signal which is high when the motor’s paddle crosses its path. There is an interrupt in the arduino code (using attachInterrupt) that detects when the photointerrupter is crossed and will execute an interrupt service routine that stores the current time (since the arduino started) and the time that the photointerrupter was previously crossed in a two element array (both are in microseconds, for accuracy). The difference between these two times are calculated (time period of half a revolution) and converted into RPM using the equation:

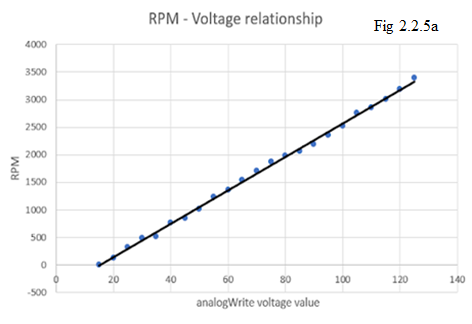
**rpm = 1000000.0/diff \* 60 \* 0.5**

One million microseconds (1 second) is divided by the time period to get frequency in Hz. This is then multiplied by 30 to get the RPM. It is multiplied by 30 since the time period was for half a revolution. This RPM is then sent via the Serial port to processing so that it can be used by the application to display information.

### **2.2.4 Code**

The code flowchart for the entire stirring subsystem is shown in Fig 2.2b. The full code is shown in Appendix 5.2. The explanation for the code is shown in each subsystem for the stirring subsystem.

### **2.2.5 Testing and analyzing the stirring subsystem**



The combined stirring subsystem was tested by measuring the RPM using a tachometer and checking if it matched the RPM values calculated using the photointerrupter. The analogWrite voltage values were also calibrated with the motor RPM to map the trend between them, as shown in Fig 2.2.3a

The trend between the voltage and RPM is linear so increasing the voltage will increase the RPM by a proportional amount. This made it clear to see how the voltage could be changed to control the motor RPM that is input by the user.

## **2.3 pH Subsystem = pH probe subsystem + pumps subsystem**

Authors: Brandon Tan, Wan Hee Tang, Xiaowen Li, Yansong Huang

**2.3.1 Overview**

The pH subsystem is made up of a circuit for the pH probe as shown in figure 2.3.1a and a circuit for the pumps shown in figure 2.3.2a. The flowchart for the code is shown in Appendix: 5.4 Figures: Figure 2. The reading for the pH probe is from pin 0 and the subsystem for the pumps is connected to pin 7 and 8.

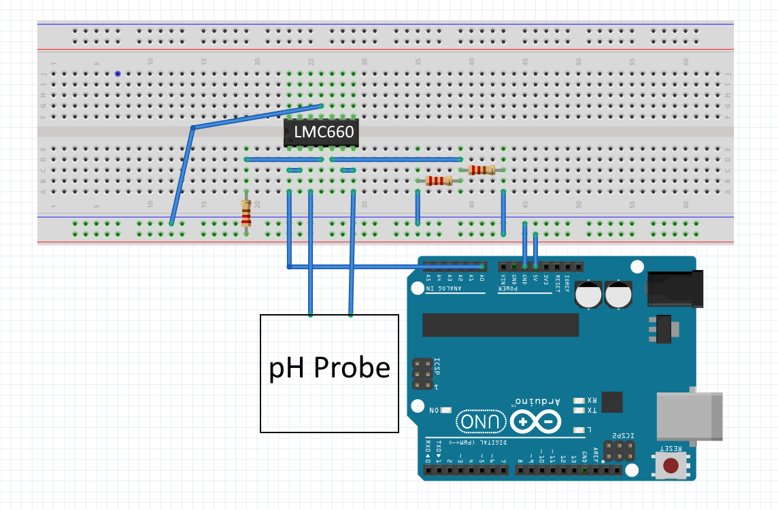
## **2.3.2 pH Probe subsystem**

Description of the subsystem**:** Reading from the pH probe and converting the ADC reading into its corresponding pH. Reading goes into analog pin 0 on the Arduino.

Specification of the subsystem: Since the pH probe gives both positive and negative readings according to the reference voltage, an op amp is used to offset the voltage to only positive values (e.g. from -500mV~+500mV to 0mV~1000mV) since an Arduino only reads from 0-5V. An ideal pH probe electrode gives 0V at pH 7. After the offset, the final output should be 500mV at pH 7. From our readings, we got a reading of 444.7703mV at pH 7, or roughly 445mV. The sensitivity of the pH probe varies according to temperature. With the assumption that our electrode is ideal, the pH can be calculated using:

**pH = (voltage – 444.7703 + 7 \* sensitivity) / sensitivity​**

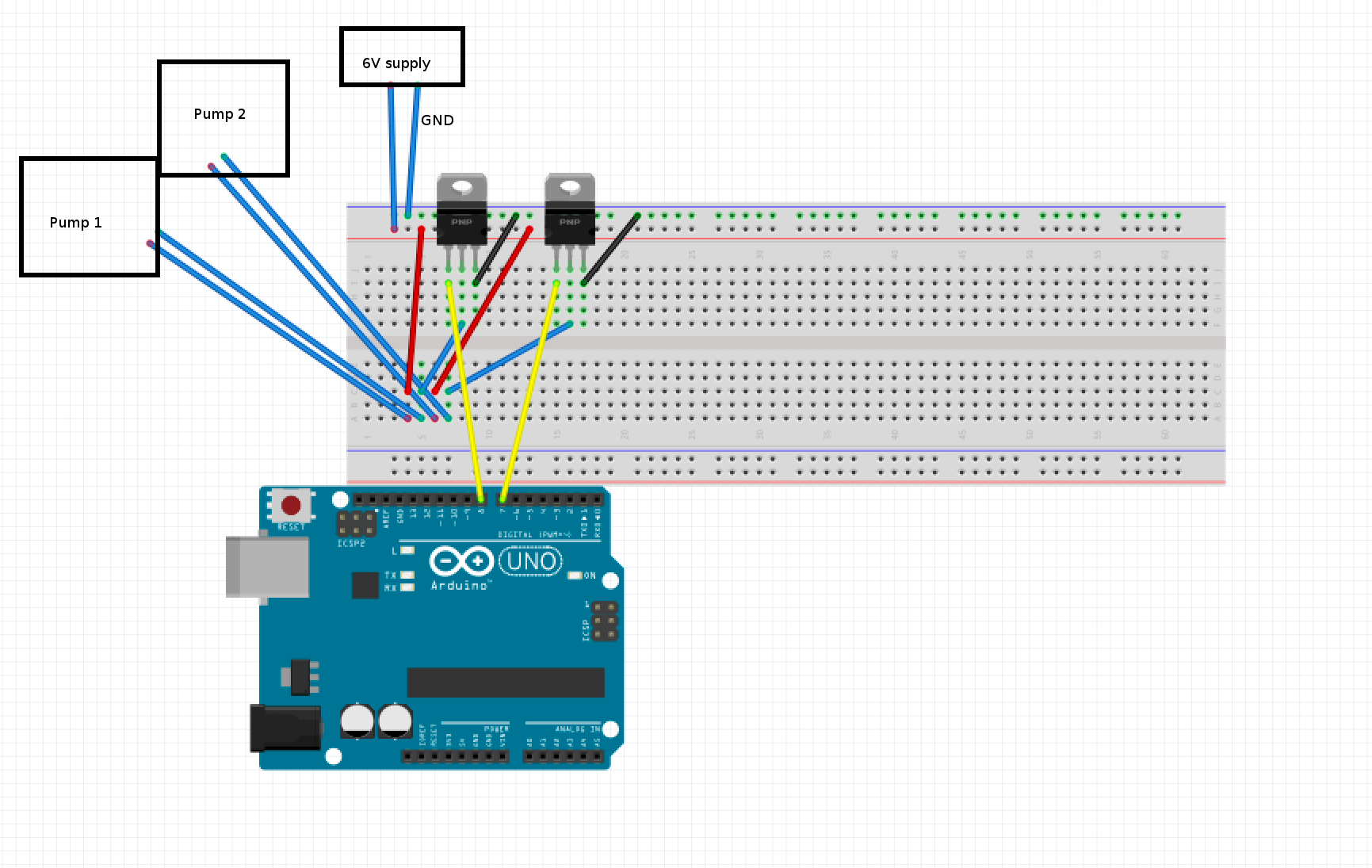
Where voltage is in mV, and sensitivity is -(R\*ln(10)\*temperature)/F, where R is the universal gas constant, F is the Faraday constant, and temperature is in Kelvin. Equation is derived from a graph with mV on the y axis and pH on the x axis.

This is how the probe subsystem was designed (shown in figure 2.3.2a):

A potential divider was used to lower the supply voltage from the Arduino, 5V to 0.5V and supplied it to an inverting Op-amp, LMC660. It was then connected to the input of the pH probe and the output of the pH probe is connected to another Op-amp which adds in 0.5V to all the readings from the pH probe since Arduino can only read positive values.

**Figure 2.3.2a**

**2.3.3 Pumps subsystem**

Description of the subsystem**:** Pumps should turn on and off when the corresponding digital signal (pins 7 and 8 for each pump) is sent. 

Specification of the subsystem: The peristaltic pumps used requires a 6V, 1A supply. Since an Arduino can only output a maximum of 5V, 100-200mA, a MOSFET is used to switch the current to a different supply, activated using the digital signals from pins 7 and 8 on the Arduino.

**Figure 2.3.3**

Here is how this subsystem was designed (shown in figure 2.3.2a):

The Gate of the MOSFET was connected to one pin on the Arduino, the Source was grounded and a wire connected the Drain and one port of the pump. By applying voltage (sending digital signal HIGH) at the Gate, it will have current flow through the channel between the Drain and the Source. Therefore, it generates a closed circuit to make the pump work.

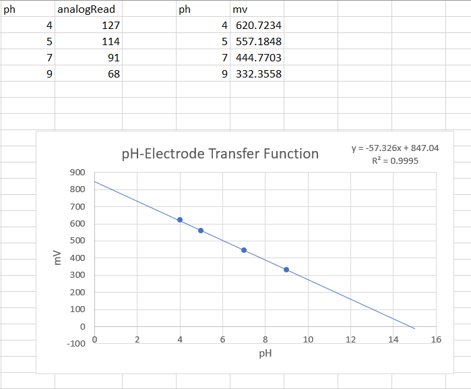
**2.3.4 Code**

Code flowchart shown in Appendix: 5.4 Figures: Figure 2

## Calculate pH from the pH probe reading

1. If the pH is within 0.05 of the setpoint, go to step 5
2. If the pH is higher than the setpoint, add acid. Go to step 6
3. If the pH is lower than the setpoint, add base. Go to step 6
4. Turn off pumps
5. Plot points on graph
6. Back to step 1

## **2.3.5 Results**

Using the sample solutions provided (pH 4,5,7,9) at 21℃, we plotted our transfer function of mV against pH.

Using the aforementioned equations:

Sensitivity (gradient) = -58.33596187 mV/pH

620.7234 mV: pH = 3.983797055

557.1848 mV: pH = 5.072981118

332.3558 mV: pH = 8.927018882

So values roughly match up with the actual pH values, shown in figure 2.3.5a.

**Figure 2.3.5a**

**2.4 UI subsystem**

Description of the subsystem: The aim of the user interface is to allow the user to monitor temperature, pH and RPM readings, and to provide a way to adjust values.

Specification of the subsystem:

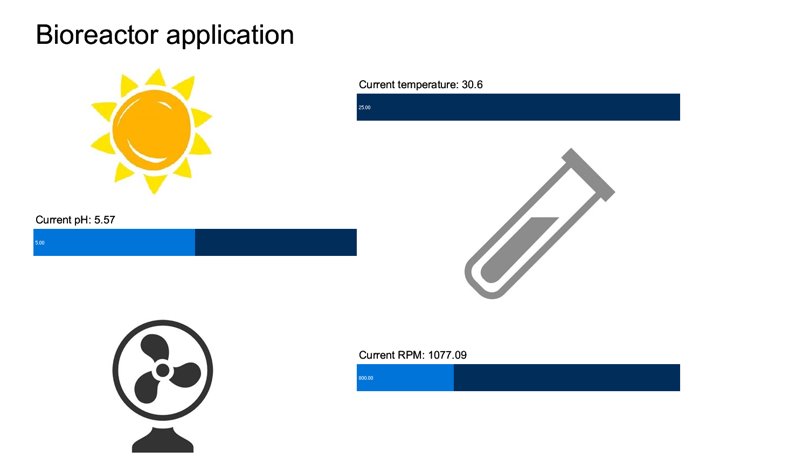
The user should be able to easily change setpoints for pH, temperature and RPM, as well as be able to see changes in readings.

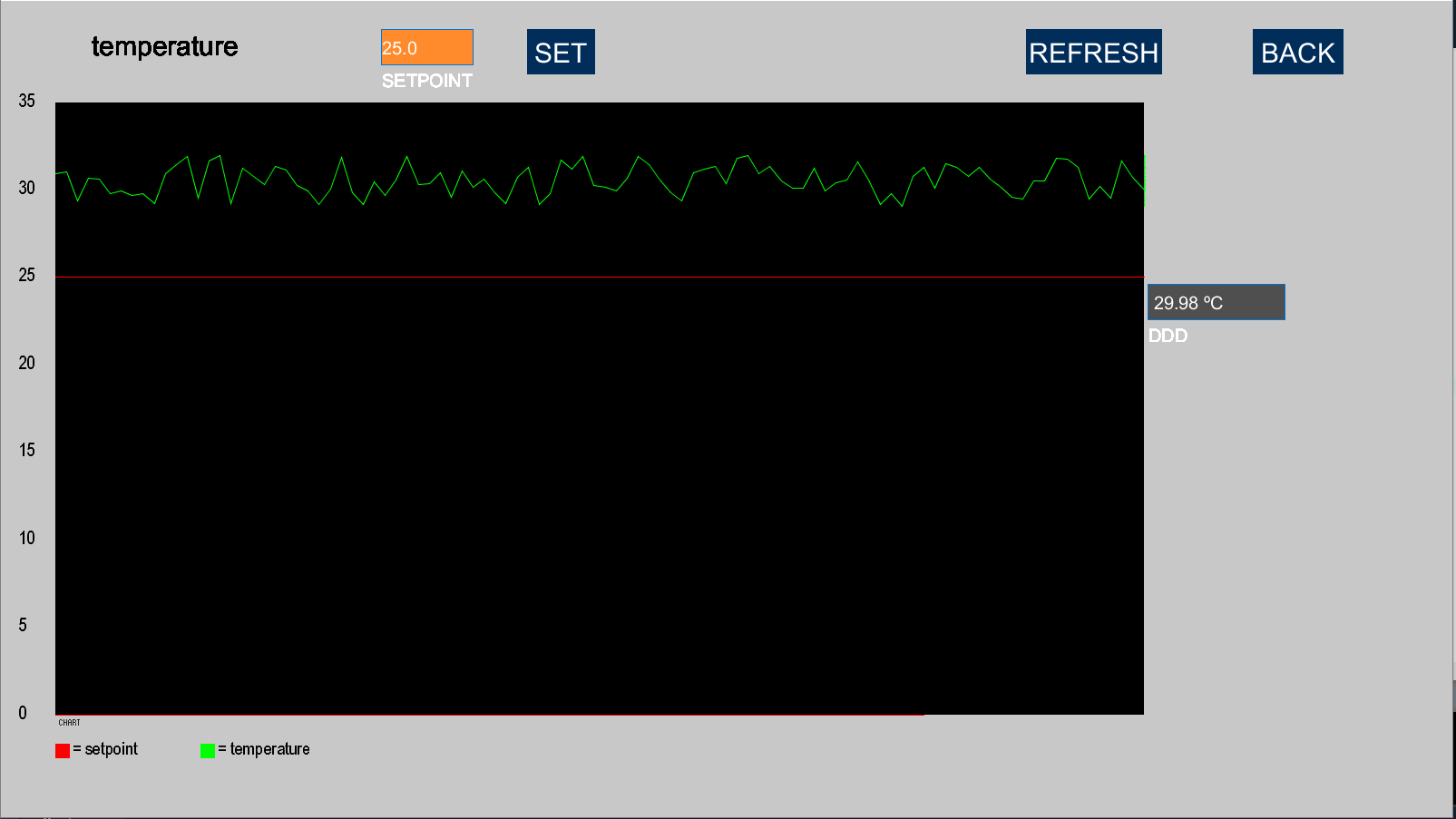
Readings are to be displayed on a graph (time on x axis), together with its current setpoint. The graph should scale according to the units e.g. if the readings only range from 0-10, the y axis should not be ranging from 0-1000.

Temperature, pH and RPM should each have their own separate graphs, and the user should be able to navigate between the three graphs. Setpoints can be adjusted from the graph page.

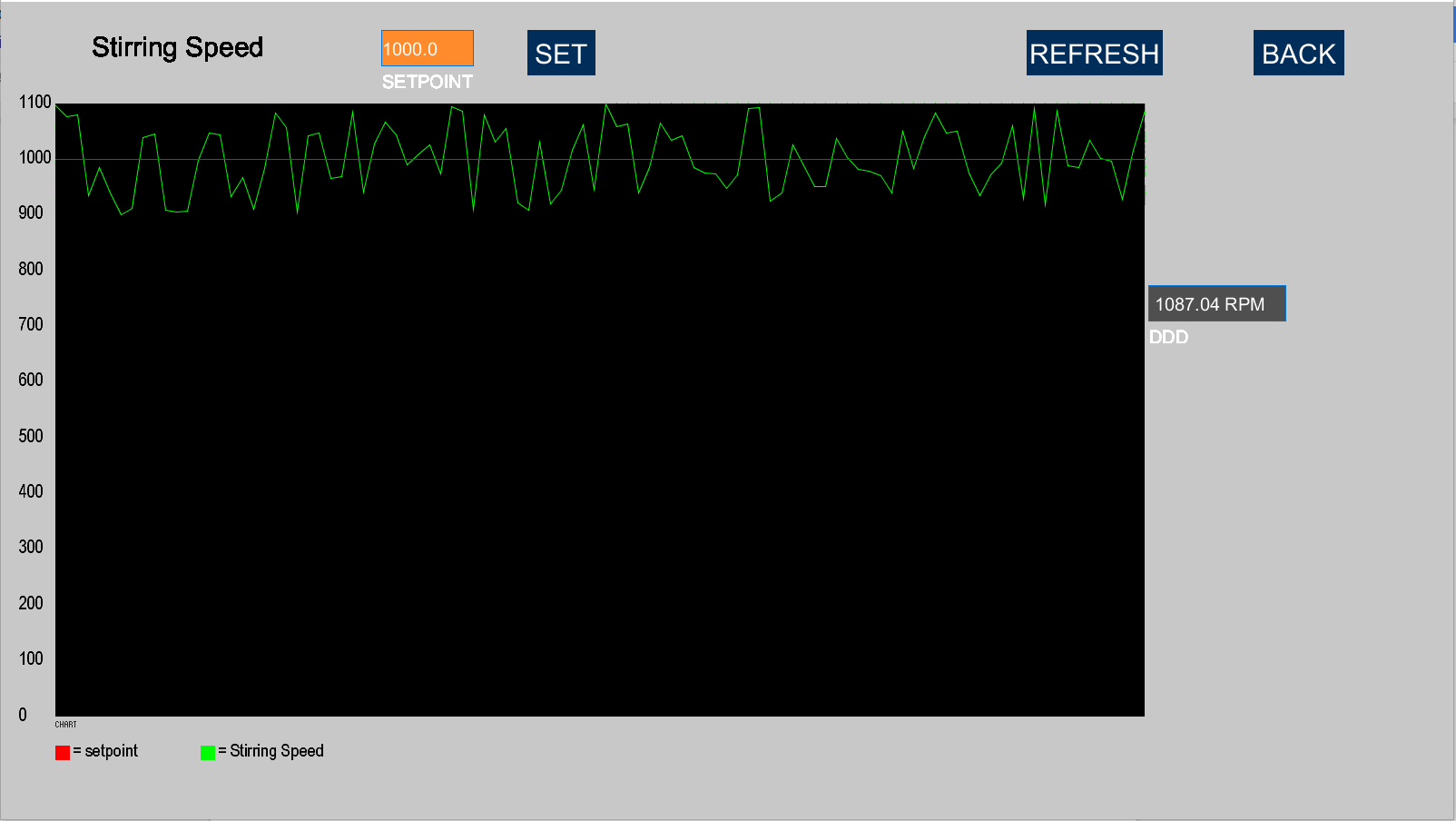
We did testing on this subsystem as shown in Appendix 4.3.

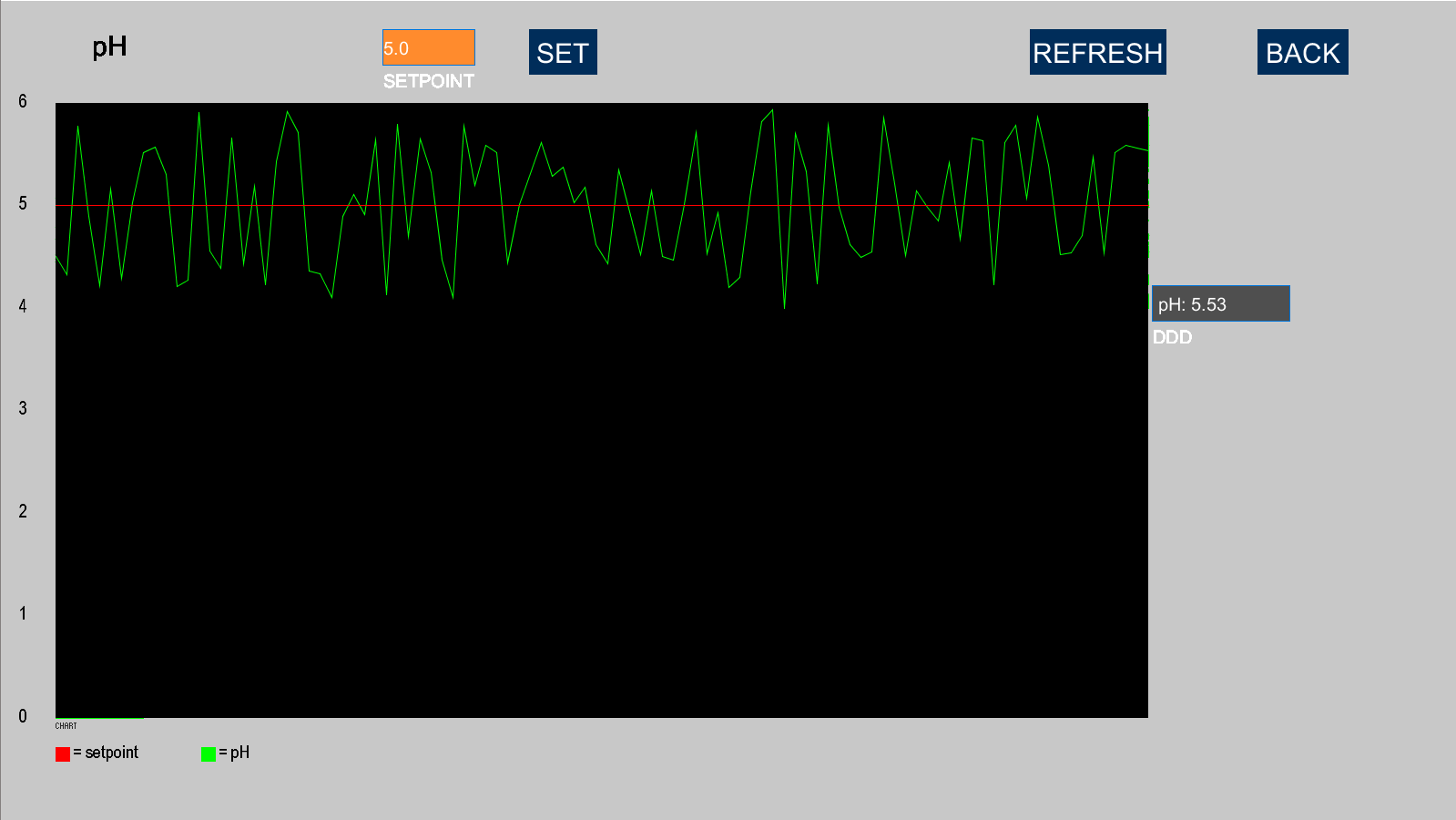
This is how the application was designed:

The main menu has three pictures that act as buttons, and three sliders. The sun button takes the user to the temperature graph, the test tube takes the user to the pH graph, and the fan takes the user to the RPM graph. The sliders to the left or right of the buttons can be user to adjust the setpoints. The current values of each subsystem is shown above each slider



When viewing the graph, readings are displayed using the green line, and the setpoint is displayed using the red line. The user can also set the setpoint from the graph menus. The refresh button refreshes the axes, and the back button takes the user back to the main menu.





**3 Overall System Integration and Summary**

The user interface displays pH, temperature and RPM readings, and allows the user to adjust the setpoint using sliders. The controlP5 Processing library was used to add components to make the GUI and the GUI communicated with the arduino code through the Serial library. The readings from each subsystem can be displayed on a graph by clicking on the pictures representing the subsystem. The graph shows the setpoint that the user wants the value to be and the actual value that the reading is getting from the sensor subsystems. All of the code used for the whole system (including the application) is shown in Appendix 5.2

However, setpoints reset to their default, hard coded values after a few seconds of running when changed from Processing, even after confirming that the Processing program is sending messages to Serial to change the setpoint. As well as that, due to inefficient code, the arduino’s small amount of memory fills up quick, resulting in the arduino freezing after a few minutes of running, and requires a reupload of the program.

During testing, equipment also malfunctioned. The motor had glue problems: the motor’s interrupting paddle kept popping off, causing the shaft to misalign, resulting in the shaft not spinning. The peristaltic pumps, even when powered (whirring sounds were audible) could not pump out liquids. The lack of operational material impeded our testing and created some inaccuracies with our reference points.

In addition, despite each subsystem working as intended when tested separately, there were a number of problems after combining all three subsystems. After connecting everything together, the voltage output went completely off, resulting in massively wrong readings for pH, temperature and RPM. Disconnecting a subsystem and reconstructing the circuit on a separate breadboard gives correct readings, but connecting the circuit back, again, gives wrong readings. Given more time, we would have liked to investigate this issue further, or to redesign our circuits. Our assumptions for the cause of these errors revolve around either a problem in the resistors used, or a subsystem drawing in excessive amounts of voltage.

**4 Non-technical considerations of a real vaccine project in Mbarara**

A big-scale project such as this one has many different considerations. However, the financial aspect is a very important one, but we don’t have agency of it in this case, so we won’t discuss it. Sustainability, on the other hand, is an essential part of such a project.

Whilst one might be doubtful on this aspect of a western-based project, measures have been thought about in order to make this project as effective as possible on the short term, and as sustainable as possible on the long term. For example, a goal has been to increase education in the region. On one hand, education for the people working in the factories, such as training local scientists that could develop, create and oversee the vaccines production and educate more locals so they are qualified to work in the factories. On the other hand, offering education for the people of Uganda on the benefits of vaccinations and how they are beneficial to them could change their mindset to make them accept this type of medicine. This will improve their lifestyle and change their hygiene habits to lower the rate of people getting infected by tuberculosis in the first place.

A certain level of skepticism is normal on our involvement, but instead of importing foreign workers, the goal is to avoid bringing in trained personnel from elsewhere, and instead set up a training program that stems from our newly established education program, so we can train and hire from the local talent that is available. This will also come as cheaper on the long term hence making it more sustainable.

Environment-wise, Africa’s local raw material market has most materials we might need, the end goal being to purely harness local materials. Pragmatism taken into consideration, there is a possibility of certain raw materials being scarce in the region, and most importantly, recognize that there is conflict of interest for certain resources.

Another problem arises for getting the vaccines to the local population: Whilst the urban population will have access to it through the already existing hospitals (albeit we need to take the slum population in mind), the rural population is extremely decentralized to these big institutions. Furthermore, the access to them is limited as there is an absence of effective transport roads to and from them. For example, a solution could be to utilize 4x4 convoys which will bring trained staff and fresh vaccines to remote areas that would not otherwise have access to them.

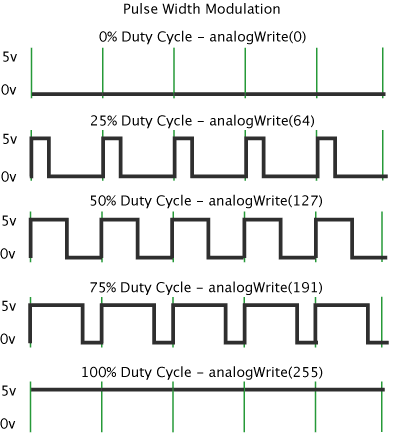
In the long run, as this isn’t particularly sustainable, we hope that our increase of the level of education will elevate the importance of health care to the government and locals, which would in turn promote the creation of more hospitals, which can become distribution centers for more remote areas.

In summary, the goal is to create a self-dependent ecosystem regarding the vaccine production and distribution in Uganda that doesn't require foreign intervention. We could also theorize the creation of marketing, and advertisement campaigns to spread the word of both the availability and importance of these vaccines in order to minimize and eventually eradicate the impact of tuberculosis in Uganda

# **5 Appendices**

## **5.1 Pulse Width Modulation**

A Pulse Width Modulation signal (PWM signal) is a signal that behaves like an analog signal but is in fact a digital signal. A high digital signal (1) is sent a certain percentage of a duty cycle (which is a time period that the PWM is measured between) and this percentage is the percentage of the highest voltage (in the diagram below this is 5v) that is used by a component.



## 

## 

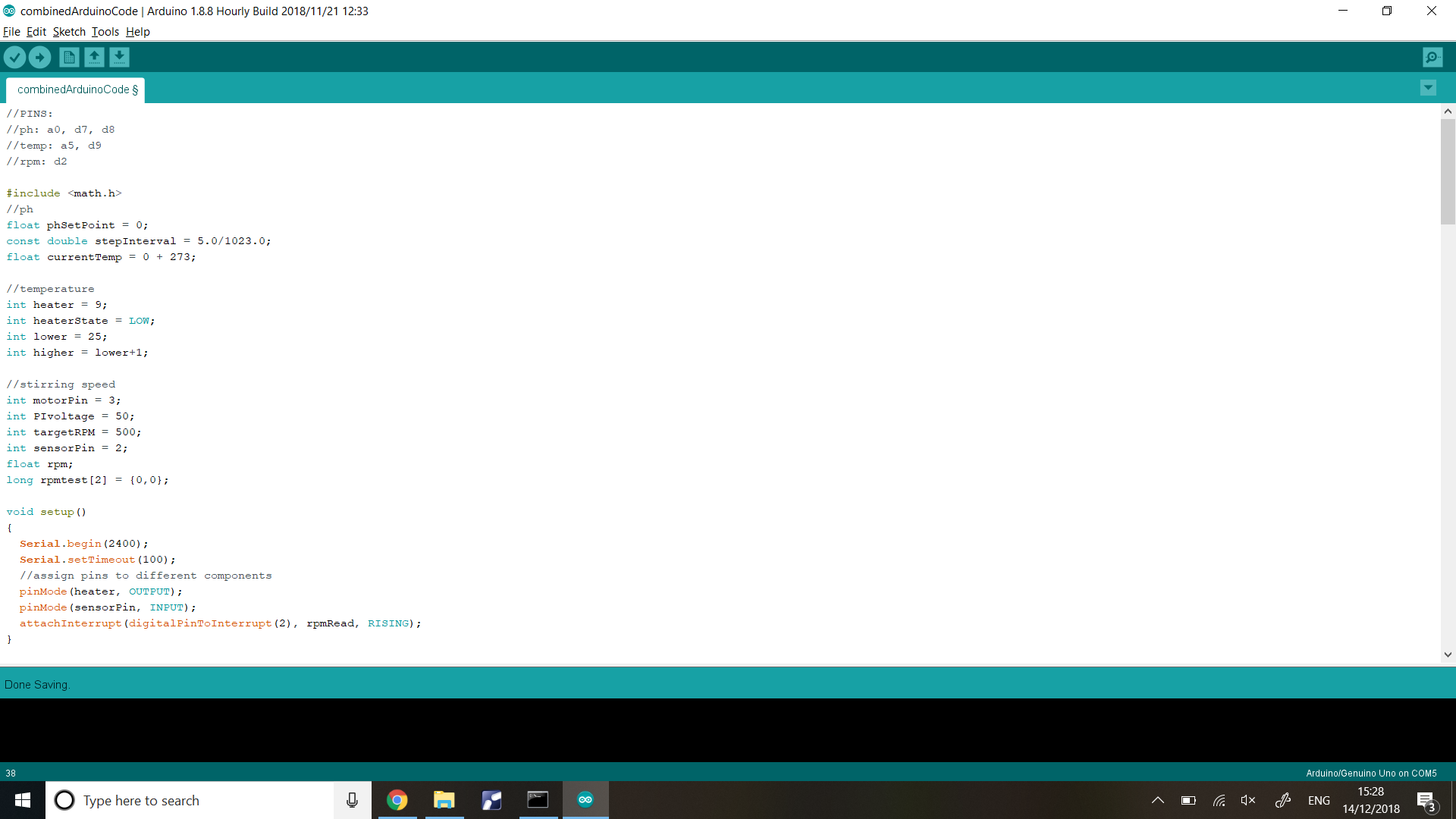
## **5.2 Code**

For more detail, look at the specific subsystems.

**5.2.1 Arduino code**

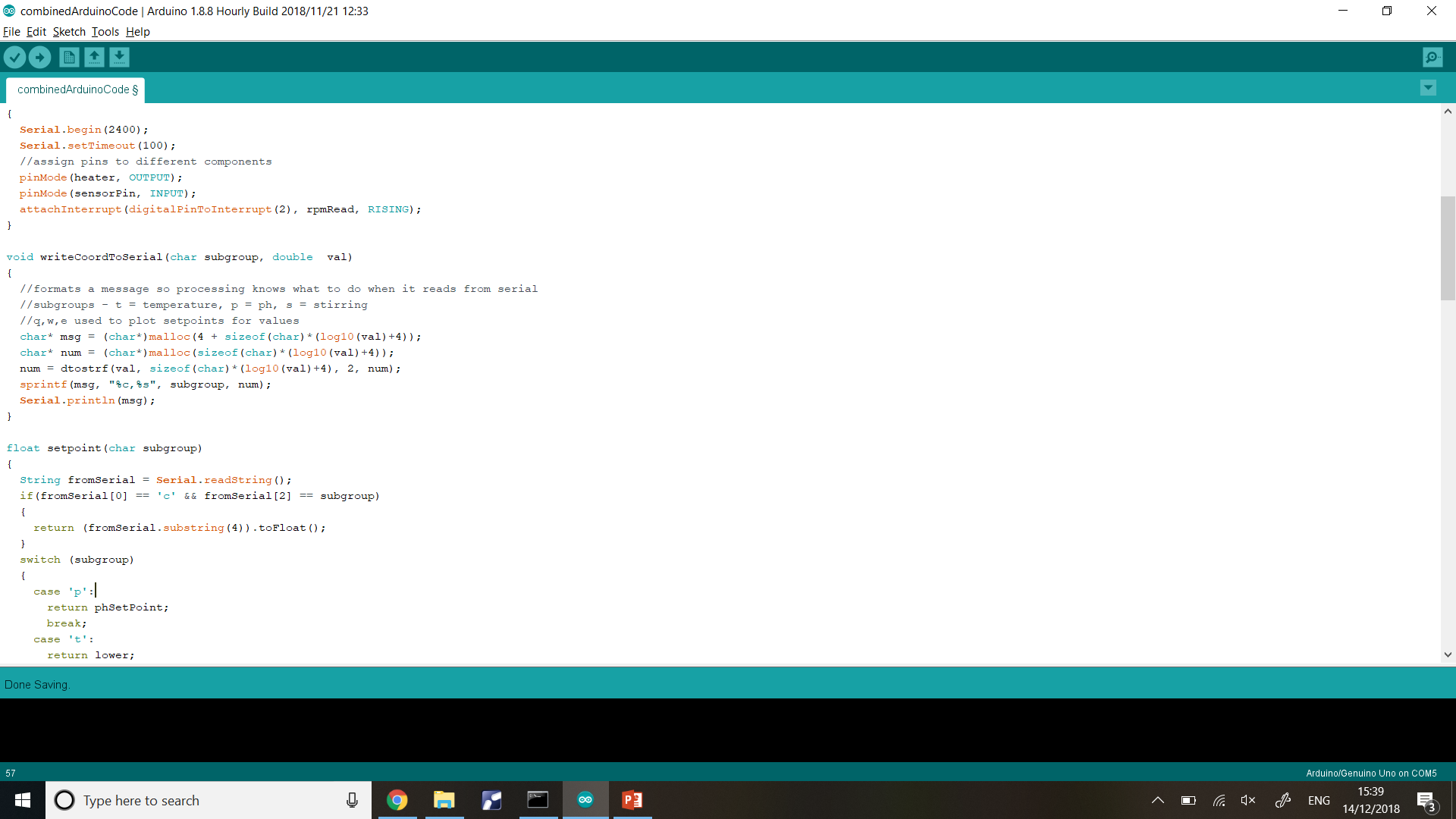
Variable declarations and setup

The variables are declared at the beginning of the code. In the setup function, the pins that need to be read from/ written to are declared. The interrupt is also attached to the photointerrupter pin here

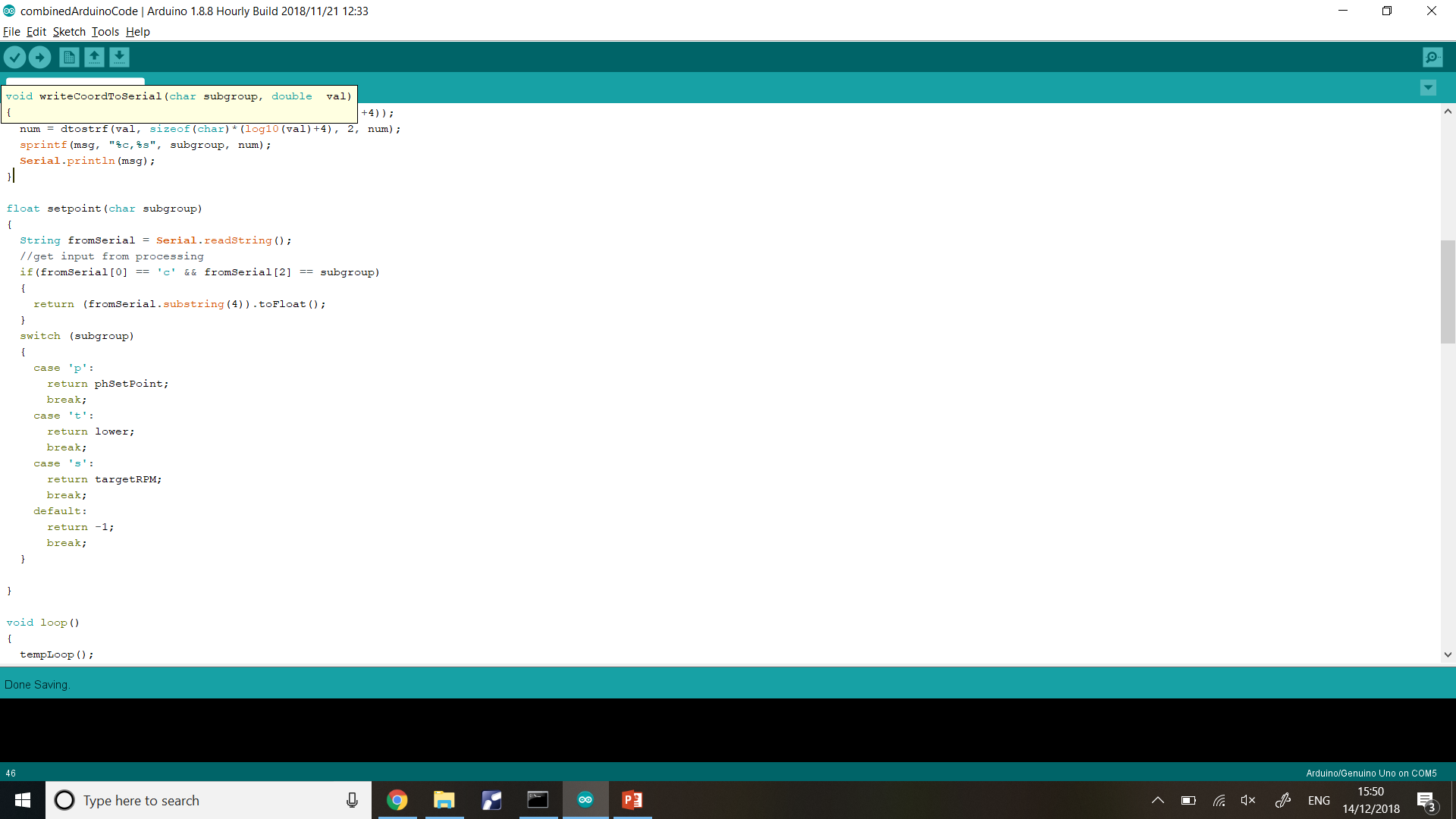
****

Writing to serial (writeCoordToSerial) and receiving from serial (setpoint)

The writeCoordToSerial function is used to encode messages that are sent to processing. E.g writeCoordToSerial('s',1000) writes "s,1000" to serial, and processing plots the reading of 1000 on to the [s]tirring graph

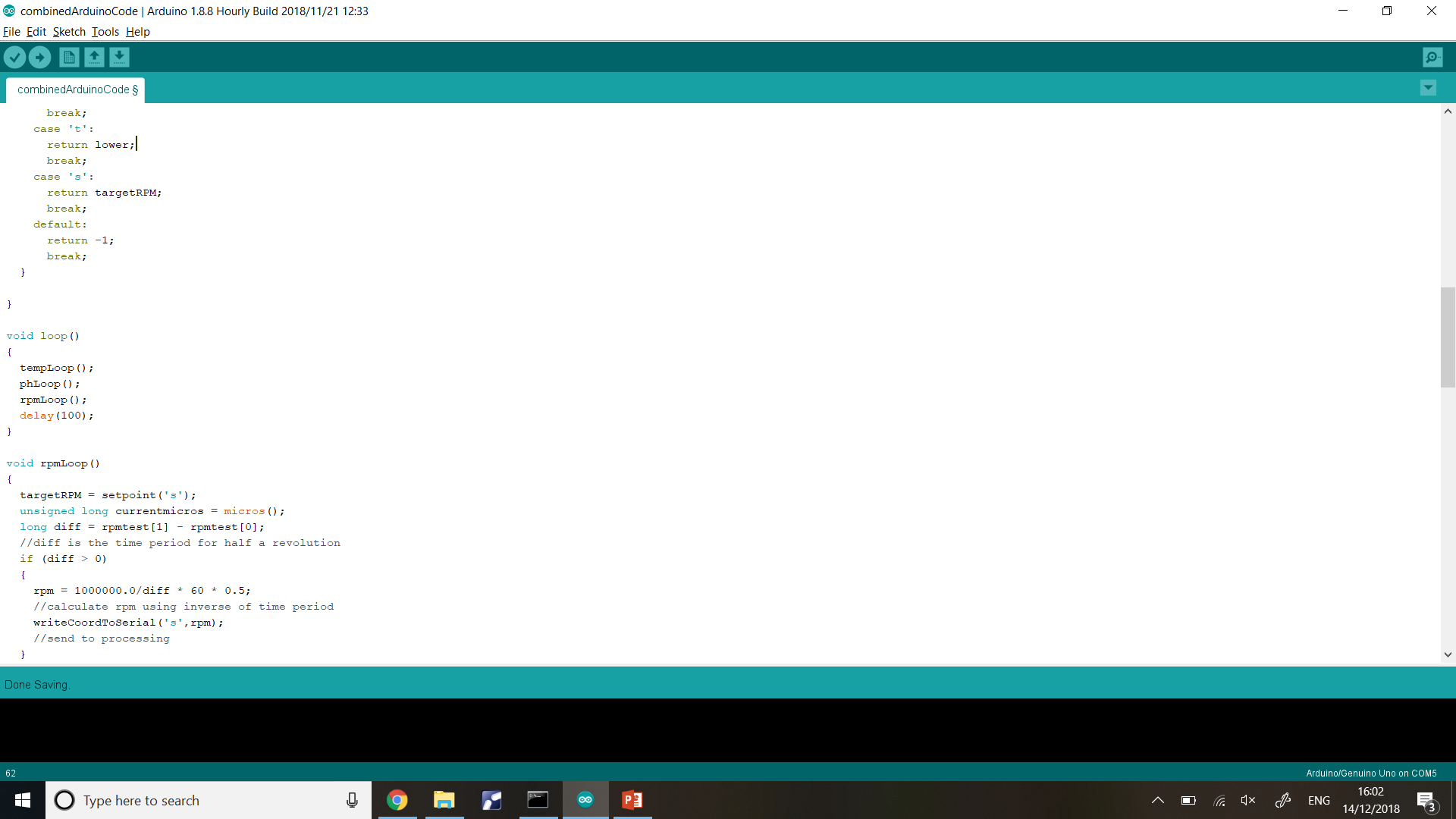


The setpoint function reads the serial port from processing and sets the setpoints for the corresponding value so that it can be changed by each subsystem



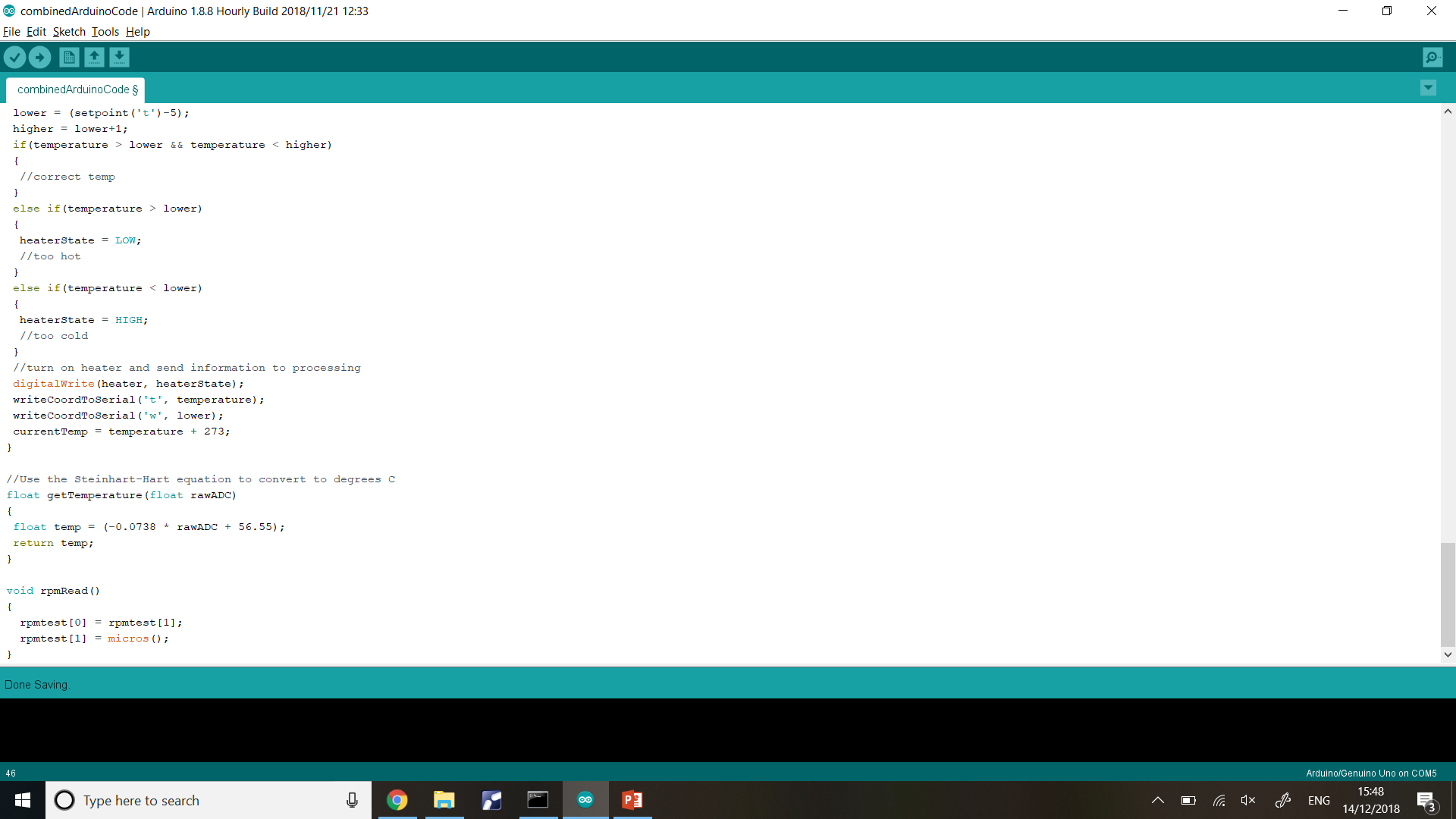
The loop

The loop is continually run and executes the loops for each of the subsystems

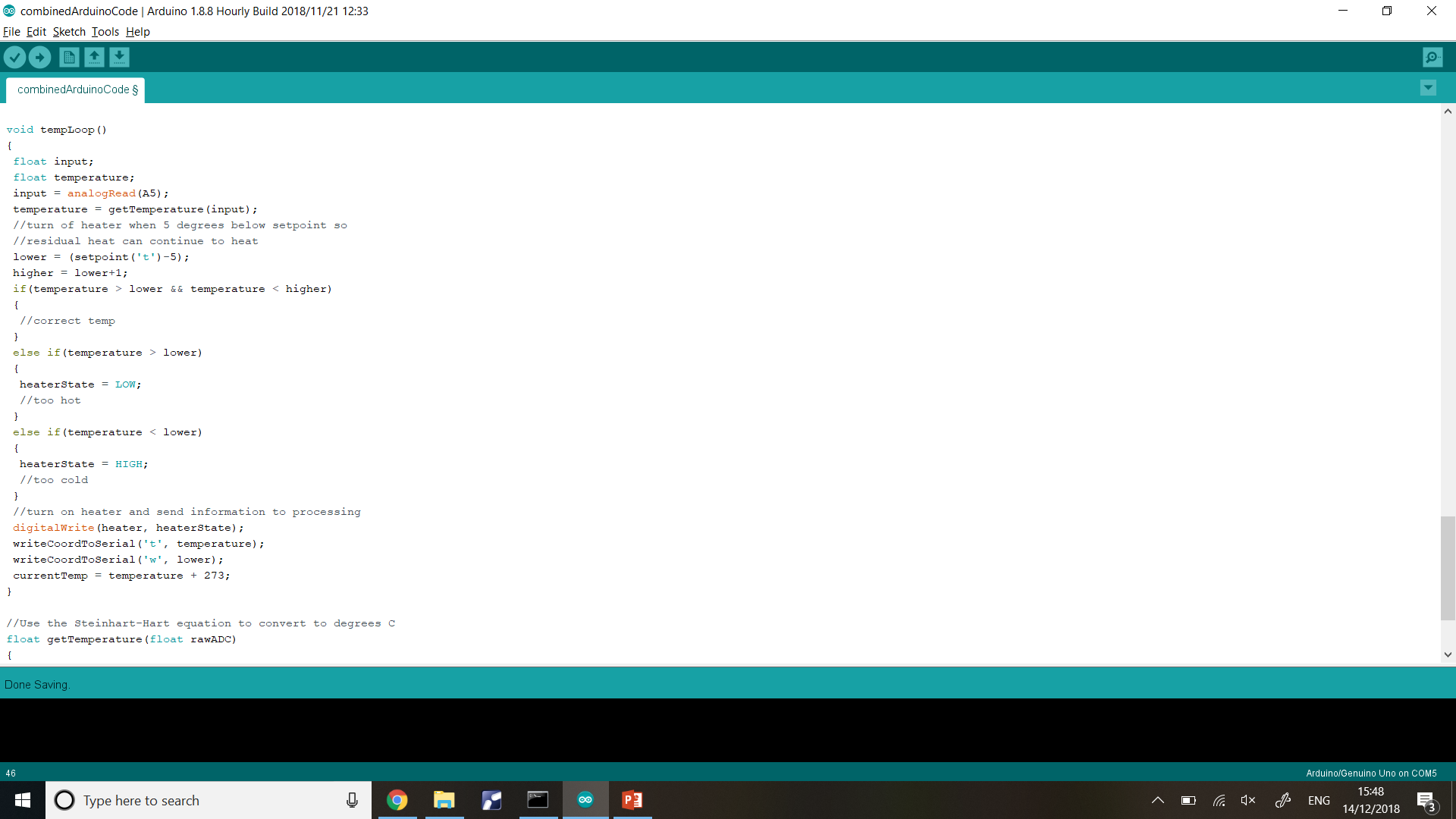


Heating subsystem

The getTempeature function converts the rawADC value (from the thermistor) and converts into degrees centigrade.

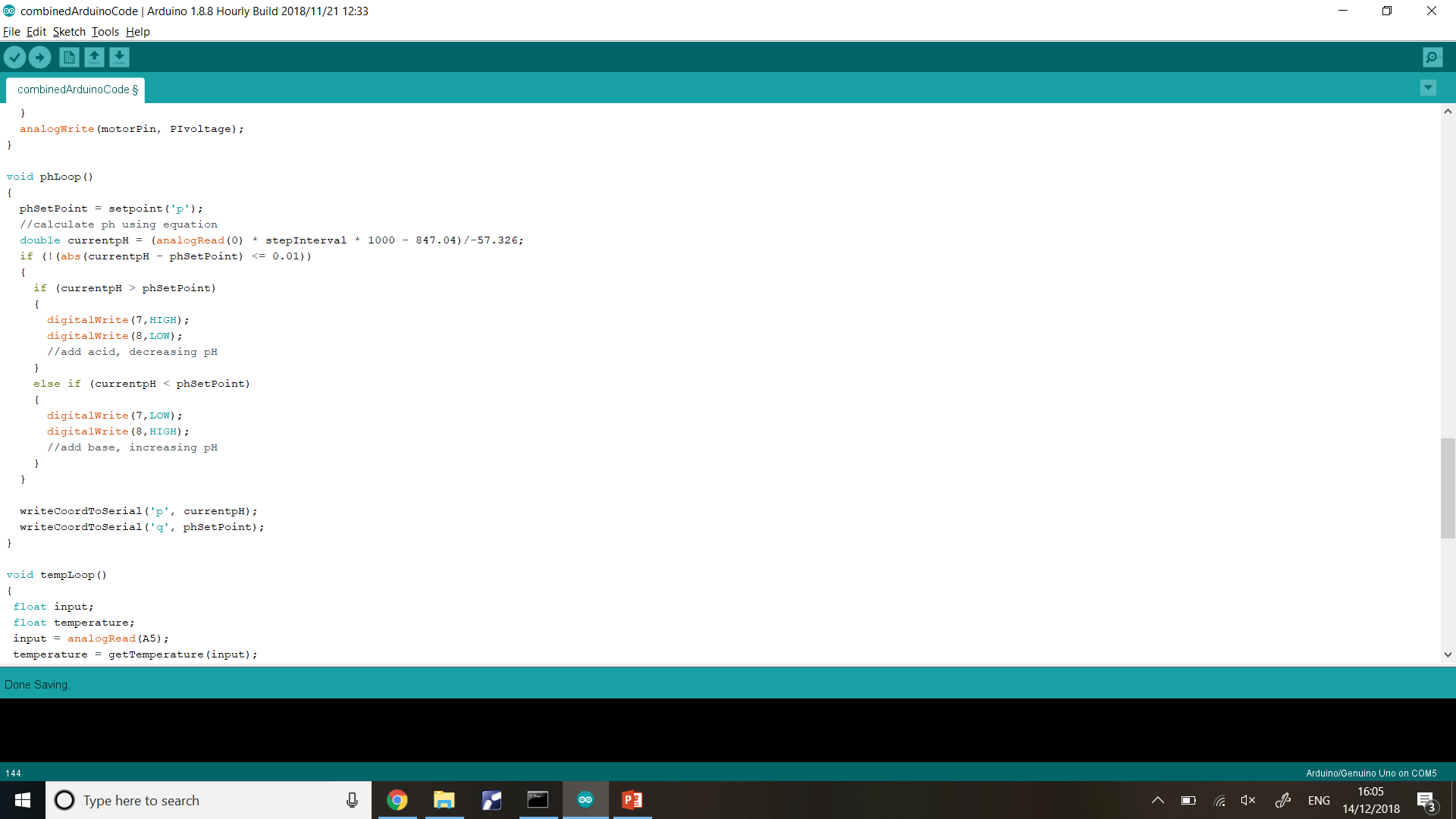


The tempLoop function turns the heater on or off to heat the bioreactor contents



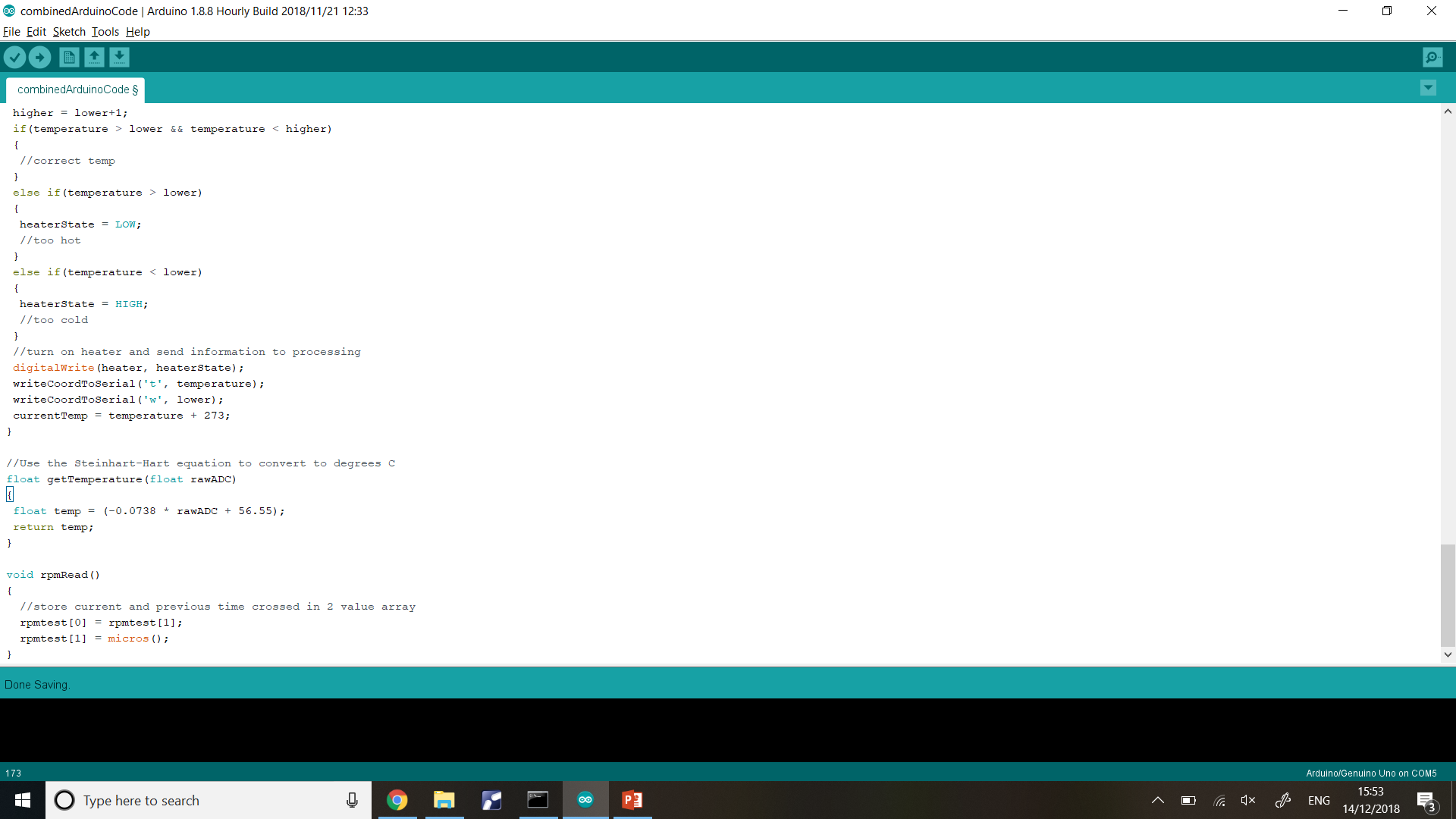
pH subsystem

The ph loop calculates the ph using an equation (shown in the ph section) and then turns on the appropriate pump (if any)

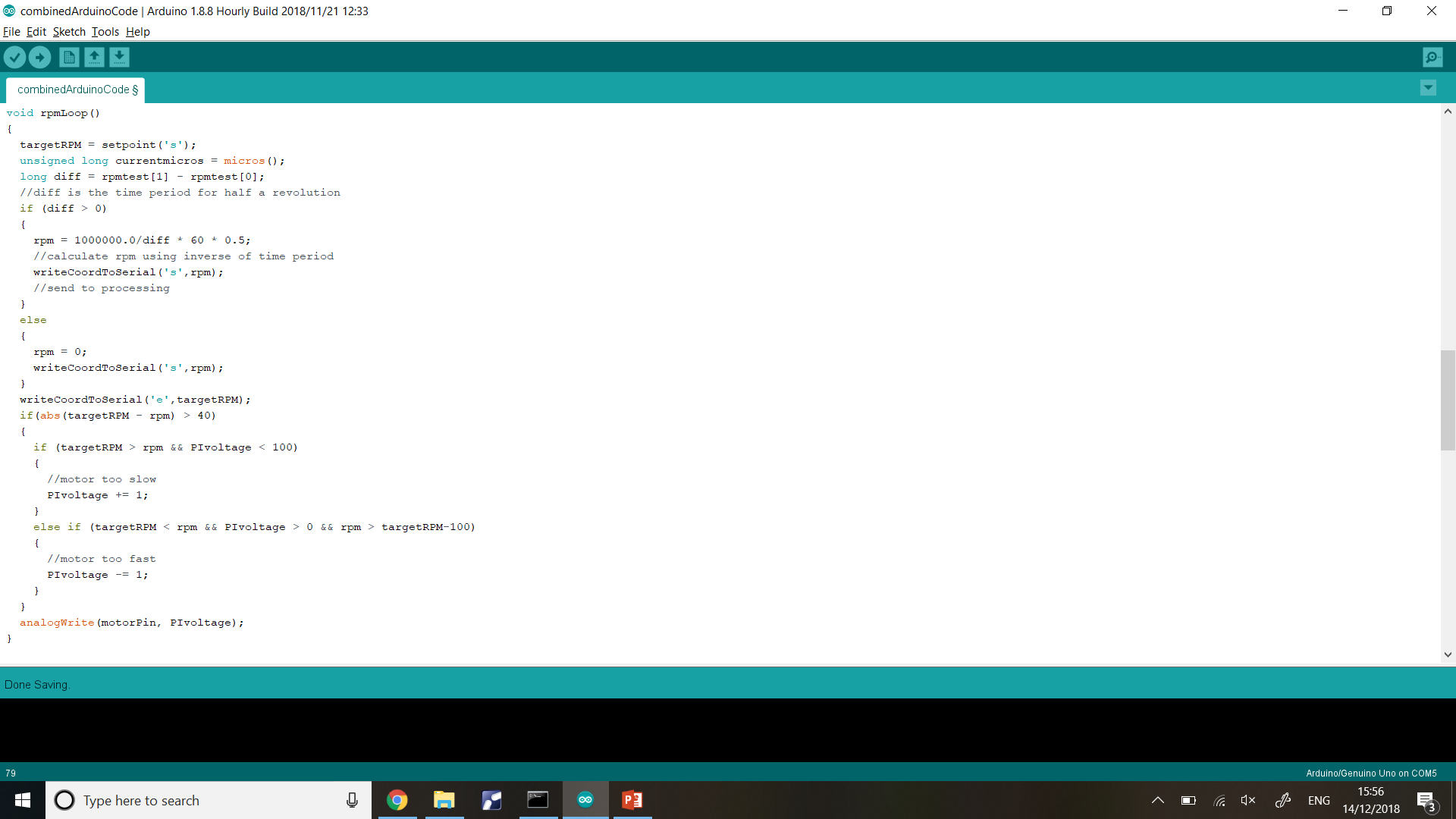


Stirring subsystem

In the setup function, an interrupt is attached so that when the photointerrupter is crossed (RISING), the rpmRead function is called.



The rpmLoop uses this information to calculate RPM and setup the motor speed



#### **5.3 Testing the application**

The processing application was tested using the whole system as well as with each individual subsystem (since we had problems with integrating the whole system together). The Arduino code was uploaded onto the Arduino beforehand and then the processing application was opened to be tested.

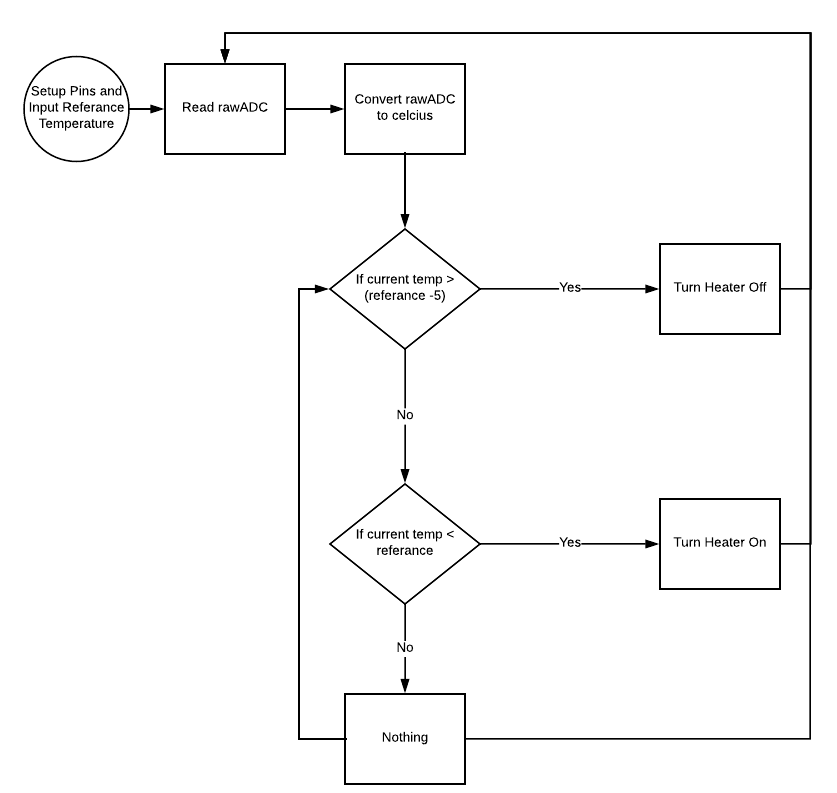
These are the tests that we ran:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Procedure** | **Expected outcome** | **Result** |
| The application shows the current values measured using sensors | Check if the current values are shown above the sliders | The values shown are the same as the values measured using specialist equipment (e.g thermometer) | The sensors were giving incorrect values when connected with the whole subsystem but was working separately |
| The application shows graphs when the pictures are clicked | Click on the pictures representing each subsystem | The graphs should be shown for that specific subsystem with correct values and setpoints | The graphs showed correct values sometimes but the sensors did not work when together so did not work with integrated system. The graphing itself did work but the information it was receiving was incorrect |
| The user can change the values of the sliders to change the system | Drag the sliders | The setpoints should change and the system should change to match these values | The setpoints changed but the sensor values were wrong so the application had no information that the value was correct |

Evidently, most of the tests were inconclusive and not correct. See section 3 for more elaboration on the problems with the whole integrated system

#### **5.4 Figures**

**Figure 1:**



**Figure 2:**

