university of groningen

University of Groningen
Faculty of Science and Engineering

Development of an automation control application (Arduino) for a bioreactor system

IE&M Bachelor Thesis

Author: Jasper Stein S3495329
Tel: 0648760073
Adr: Oosterhamrikkade 18B
Pc: 9714 BC Groningen
Email: j.w.stein@student.rug.nl

Supervisors:
S. Achinas
G.J.W. Euverink
G.H. Jonker
1-3-2021

Abstract

The object of this project was to develop an automation control system application for a miniaturised bioreactor operation based on Arduino technology. The miniaturised bioreactor is designed in a multi parallel setup to produce biogas through anaerobic digestion. The research is conducted to develop a suitable Arduino control application that can sufficiently control, monitor, and visualise process parameters while being less expensive than the currently known control systems for bioreactors. The first part of the project consisted of analysing the setup of the miniaturised bioreactor system and identifying relevant process parameters that could be controlled, monitored, and visualised. The second part of the project was to acquire the necessary Arduino parts to realise the physical aspect of the control system. The Arduino parts were chosen by specifying whether the parts were inputs or outputs and what function the Arduino parts should full fill within the control system. The third part of the report was to translate the Arduino parts into a written code using the Integrated Design Environment (IDE) and the C/C++ language of Arduino. This proved to be quite difficult because of the need to convert each individual code into one code that could achieve full control of all the necessary Arduino parts. The fourth part of the project was to program the graphical user interface for the control unit to visualise and monitor the relevant process parameters. The last part of the project contains a short mention of possible market options for future investments in the control unit. The results of the report show that Arduino is suitable for monitoring relevant process parameters of a miniaturised bioreactor. However, adjustments should still be made in order to make Arduino sufficient enough as a controller. Furthermore, Arduino also needs the help of Processing to visualise the parameters that are measured by the pH electrode and the thermometer.

Content

1.Introduction **6**

2. Materials and Methods **8**

2.1. System Setup *8*

2.2. Parameters *10*

2.2.1. PH 10

2.2.2. Temperature 10

2.2.3. Agitation 11

2.2.4. Biogas flowrate 11

2.2.5. Feed 11

3. Arduino Hardware **12**

3.1. Standard parts *13*

3.2. Inputs *15*

3.2.1. Sensor accuracy 15

3.2.2. pH value 16

3.2.3. Temperature value 17

3.2.4. Biogas flowrate 18

3.3. Outputs *19*

3.3.1. Peristaltic base pump 19

3.3.2. Activate/deactivate alarm for change in temperature 20

3.3.3. Stirring speed 21

3.3.4. Feed 22

4. Arduino Software **23**

4.1. Base pump calculations *25*

5. Graphical User Interface **28**

6. Final Arduino control application **29**

7. Validation **32**

8. Market options **36**

9. Discussion **38**

10. Conclusion **43**

11. References **45**

Appendix A **49**

Appendix B **51**

Appendix C **53**

Appendix D 57

1. Introduction

Since the development of bioreactors, scientists have found many ways to apply the technology in several fields within engineering. Examples of these fields include wastewater treatment, cell culture, and tissue engineering in healthcare, production of high-value pharmaceuticals and bulk chemicals in industrial biotechnology, and even cultivation of algae for oxygen generation in space exploration [2]. Due to ongoing evolvement in the biotechnology scene, the demand for various applications actively increases the progress in bioreactors structures. One of the latest developments in bioreactor structures is the micro bioreactor system. These micro bioreactor systems (MBR) are much smaller than the commonly known bioreactors. The microbioreactor is a downscaled bioreactor that holds a capacity ranging from circa 0.1 mL to approximately 100 mL [3]. The MBRs were initially designed to achieve acceleration in the processing speed by utilising a high-throughput capability since the high-throughput capability results in the opportunity to perform many cell cultivations in parallel [3].

The group of products and processes for biotechnology (GPPB) of the University of Groningen has developed its own bioreactor system that contains four multi parallel miniaturised bioreactor operations. The design of this bioreactor system can be seen in figure 1. The GPPB wants to work with 96 operations at once. This means that the system, which is shown in figure 1, will be used 24 times. The system shown is estimated to have a production range of approximately 1 ml – 40 ml of biogas a day [1]. The bioreactor operations will be used to study the simulation behaviour of cell cultures by microbial fermentation processes. Within these processes, growth and production rates of the cells are strongly dependent on several process parameters, namely: pH, stirring speed, and temperature [4]. These parameters can be monitored and controlled in the microbioreactors system. In this project, the focus will be on one miniaturised bioreactor system with its relevant process parameters.

In order to control and monitor these relevant process parameters, the GPPB wants to develop an automation control application that is less expensive compared to the control systems that are more commonly used for normal scaled bioreactors. Besides the interest of the GPPB in this project, other institutes and university could also have interest in the possible designed control application for their own future research. The problem of the relatively high costs that occur for normal control systems can be countered by using Arduino as the micro controller. Arduino is a Printed Circuit board that is especially designed for it being cheap and easy to use.

Furthermore, the ARDUINO board also has many other electronic components that are needed for the microcontroller to function or to extend its capabilities [5]. The programming language that is used within Arduino is a simplified C++ language, in a processing based Integrated Development Environment (IDE), which is easily accessed from the internet. Arduino uses a programming code named (*ino) files called sketches that can be compiled and uploaded to the Printed Circuit Board by using an USB cable that connects the Arduino PCB to the device where the IDE has been downloaded [6].

Control systems for mini bioreactors are scarce products. One that can be found on the world wide web is the bioreactor made by Tommi Lintiläv [7]. The bioreactor in his process controls the stirring speed for a rotary cell culture system bioreactor. With the use of Arduino, the costs of this RCCS are approximately €65, -. However, the RCCS only controls one parameter. In

this project, several parameters will be controlled. Therefore, an estimation can be made that the costs for this project will exceed the €65, - that is used for the RCCS [7]. Furthermore, it can be found in research from Marinescu G. C. and Popescu. R. G. that a control system that controls five parameters costs approximately 477 USD. Accordingly, the price estimation for the control system that is developed in this project will be between €65, - and €388, - [8]. This price estimation is important for possible future investors such as Applikon Biotechnology, which facilitates the funding of the development of the miniaturised bioreactors. Applikon Biotechnology is a world leader in developing and supplying advanced bioreactor systems for industrial biotechnology. Applikon can guide a customer from initial screening up to full-scale production, using the same platform [9]. Therefore, an application that could control mini bioreactors could be interesting as a future investment for Applikon.

The use of an Arduino board to control and monitor the relevant process parameters of the bioreactor operations has more advantages than its simplicity and the relatively inexpensive components. First, the use of computers (i.e., through a DAQ device/card) in the bioreactor processes enables the opportunity to store data and convert the data to a format that can be used by spreadsheets or other software packages for more extensive analysis. Another advantage is the obtainment of real-time data for monitoring and control purposes with the possible realisation of a fully automated microbioreactor setup which can ease the future research with bioreactor processes [10].

Arduino is already being used as a controller for bioreactors. Husain [10] gives a good example of the use of Arduino in the bioreactor scene. In this article, Husain especially designed the controller for a miniaturised bioreactor, which is the same in this report. The results of Husains' article indicate that the development of an application based on an Arduino simulator is achievable.

The eventual goal of this project is to develop an Arduino application that can simulate a cell culture. The application should thereafter be able to return relevant process parameters from the cell culture. Furthermore, the application should also provide a graphical user interface where the relevant process parameters can be specified, and simulation behaviour can be controlled while operating the microbioreactor.

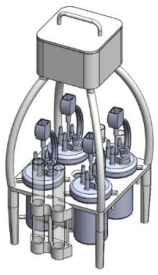

Figure 1: Four multi-parallel miniaturized bioreactor system.

2. Materials and Methods

In order to decide what Arduino components must be used to control the system, information about the developed miniaturised bioreactor and its relevant process parameters has to be acquired. In the following two sections, the system setup and the relevant parameters with their operating ranges will be addressed.

2.1. System Setup

Before starting with the explanation about the setup of the miniaturised bioreactor operation, an explanation should be made about how bioreactors are used and which of these processes is used in case of the miniaturised bioreactor designed and fabricated by the GPPB.

Currently, bioreactors are used for several different processes. Mentionable examples of these are cultivating mammalian cells (such as CHO), microbial cells (such as E. coli), and yeast or small plant cells (such as moss) [11]. The principle of these processes is that the cells produce the desired compound by transforming nutrients into high-value biopharmaceutical products. The transformation of the nutrients happens by fermentation [12]. The fermentation process can be conducted in two ways, namely anaerobic and aerobic. The biggest difference between these two is the use of oxygen. With anaerobic processes, a process is conducted without the use of oxygen. In the case of the miniaturised bioreactor, an anaerobic process is used. A benefit of the anaerobic process is that it requires low energy. Furthermore, small quantities of by-products (sludge) will be produced [11].

The developed miniaturised bioreactor uses the anaerobic digestion process in its design. The cross-section and the top view of the miniaturised bioreactor can be found below in figure 2. Important parts are indicated with an arrow and a description. From the figure, four parameters can be distinguished that need to be considered when designing the mini control system. The parameters are pH, temperature, agitation and, biogas flowrate. The miniaturised bioreactor is specially designed to measure these parameters. Visualised with the top view, the nut and port for the pH probe and the stirrer can be easily accessed for control purposes. Furthermore, to measure and control the biogas flow rate, the gas outlet is used. This gas outlet will lead to a flask with water. Due to the pressure of the gas up top, water will exit the flask. The difference in weight of the flask can thereafter indicate how much gas has been produced during the bioreactor operation. Figure 3 shows the combination of the bioreactor process and the flask to measure the biogas flowrate [1].

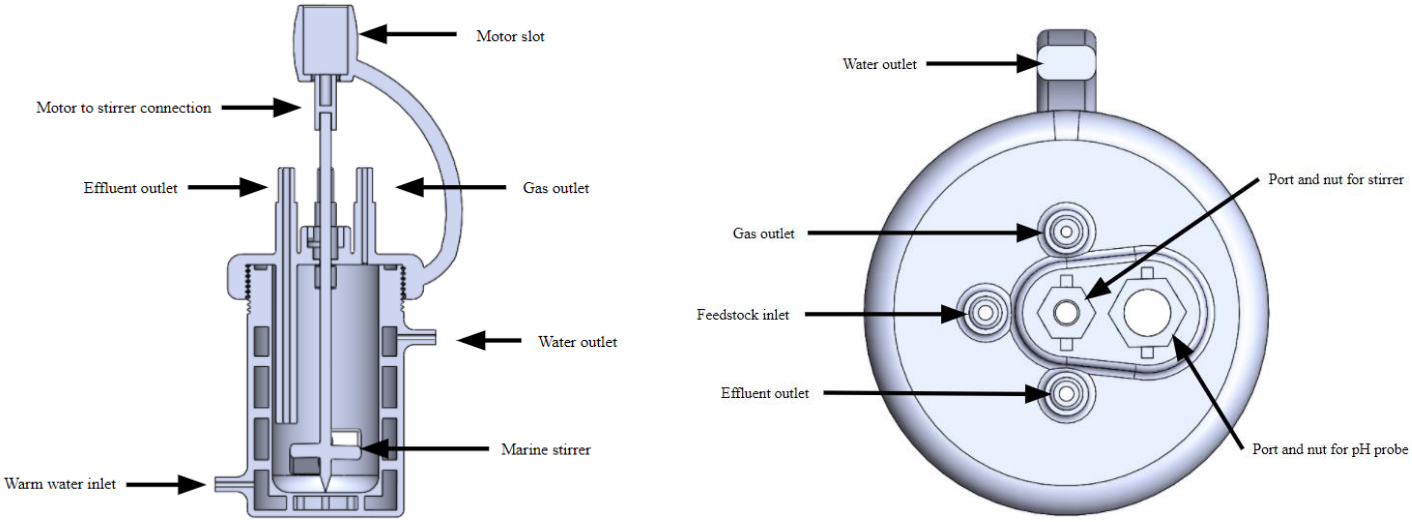

Figure 2: Cross-section and top view of the developed miniaturised bioreactor

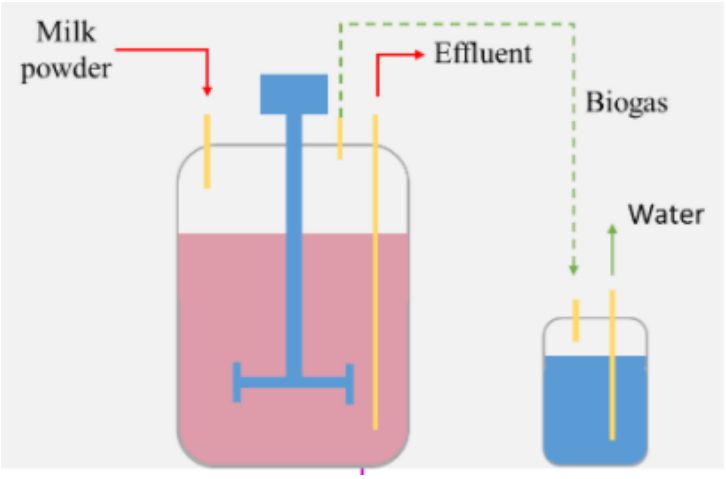

Figure 3: biogas flowrate regulation

2.2. Parameters

In bioreactor operations, one of the essential tasks is to control key parameters. The key parameters can optimize the process by maintaining the cells in a desired physical and chemical environment. The parameters that must be controlled and/or measured are pH, temperature, agitation and, biogas flowrate. Each of these parameters will be addressed on their importance and operating ranges.

2.2.1. PH

The pH is one of the most important parameters to be controlled. Without control of pH, cell growth can be hindered or even stopped altogether because of a too acidic growing medium [13]. Drifting pH values can negatively influence the product's yield in operations. Therefore, keeping the pH in the correct operating range will impact both cell viability and the product's quality [12]. The solution for the problems mentioned can be to set the pH at a pre-defined set-point. Normally, the bioreactor should have an operating range of a pH value between 6.7 and 7.4, whereas the methanogenic activity fails when the pH value decreases below 6.5 [14]. In order to maintain the pre-defined set-point, a pH probe should be used. This pH probe should measure the pH value within the miniaturised bioreactor. If the pH tends to be too acidic, the base solution should be added to stabilise the pH.

2.2.2. Temperature

Another important parameter to control is the temperature. In anaerobic processes, two significant temperature zones can be distinguished. The cause of the different temperature zones is the difference in the type of micro-organisms that are being worked with [15]. The micro-organisms can be separated in mesophilic and thermophilic organisms. The mesophilic organisms have an optimum working rate at a temperature that lies approximately around 35 °C, while the optimum working rate of thermophilic organisms is at around a temperature of 55 °C [15]. In case of the miniaturised bioreactor, the tank is heated to 35 °C. The control of temperature is important because of several reasons. Temperature higher than 37 °C can quickly harm cell viability, while a lower temperature can result in slower cell metabolism. Furthermore, a homogenous and constant temperature should be strived to when operating the bioreactor. Otherwise, the process still cannot be conducted efficiently [16].

2.2.3. *Agitation*

The agitation in a bioreactor increases the efficiency of the temperature and pH control. The agitation system of a bioreactor consists of an impeller, a drive mechanism, and a motor. The agitation system's purpose is to deliver a power input into the culture medium by mixing the substance in the bioreactor to get a homogeneous distribution of the pH, temperature, and a proper biogas flowrate [16]. The effectivity of the distribution depends closely on the impeller size, type, and location due to hydrodynamic and aeration and their potential effect on the cells and the process. However, the study shows that, if the digester is continuously stirred, different mixing intensities only have a minimal effect on the bioreactor. The only effect can be found in the initial start-up. Intense mixing in the initial start-up harms the performance. To prevent digester failure in the initial start-up period, intense mixing should be avoided [17].

2.2.4. *Biogas flowrate*

The product of the process is the production of the biogas. The measurement of the amount of biogas produced is required to calculate the efficiency of the process. The biogas produced can be an energy source that can be valorised as electricity, heat, biofuel or can be injected into the natural gas grid. The main composition of biogas that is generated from anaerobic digestion is methane and carbon dioxide 50%~70% and 25%~50% respectively [18]. However, biogas also contains other trace gases, moisture, particulate matters (PMs), and contaminants such as volatile organic compounds (VOCs), sulphur compounds, siloxanes, and ammonia [19]. Currently, the GBBP uses gas chromatography to determine the composition of the biogas. Biogas is taken from the bioreactor using a syringe. Thereafter, the biogas is investigated by separating the chemical components with carriers to detect and determine the presence or absence and/or how much is present of the desired components [20].

2.2.5. *Feed*

In order to efficiently generate biogas, regulation of the culture conditions is imperative. Depletion of nutrients, along with build-up of metabolic waste are major contributors to senescence and metabolic changes that reduce the quality of the produced biogas [21]. The use of feeding systems can eliminate fluctuation in glucose levels and improve biogas production. This can be achieved with two actions. Firstly, by feeding key nutrients into the miniaturised bioreactor so the biological reactor within the bioreactor does not stagnate due to low supply of desired nutrients. Secondly, by subtracting the metabolic waste and unusable bio-matter [22]. The optimal feeding strategy in bioreactors depends on the structure of the reaction kinetics and the interaction between the different reactions [23]. In order to regulate the feed in the miniaturised bioreactor, it has been determined that the inlet and outlet will consist of 2 ml /day. In addition, the composition of the feed will be milk powder in distilled water.

3. Arduino Hardware

The parameters and the system set-up have now been determined. Therefore, sensors and actuators can be allocated to each function and parameter. In order to start with the allocation of sensors and actuators, the standard parts needed to work with Arduino should be chosen. Thereafter, the determination on which extra Arduino components are needed has to be done. Furthermore, with the help of Fritzing and Circuito, schematic overviews of the Arduino circuits will be given [31]][32]. To achieve this, the inputs, transformation processes and outputs have to be distinguished. The transformation processes can be full filled using the right coding. However, the inputs and outputs have to be performed by Arduino components. The following was found:

Inputs:

1. pH value
2. Temperature value
3. Biogas volume
4. Feed

Transformation processes:

1. Analyse pH value: if lower than 6.7, start the motor to add a base to the bioreactor.
2. Analyse temperature value: if lower than 34, start alarm until the lab worker stops the alarm after the heater is examined. If the temperature is higher than 36, start alarm until the lab worker turns the alarm off after the heater is examined.
3. The gas that is produced pushes the same volume of water out of the system. Thereafter, calculate the volume of the biogas by using the difference in weight. For example, 1 gram equals 1 ml which is 1 cm^3.

Outputs:

1. Start Base pump
2. Activate/deactivate alarm to indicate when the heater does not full fill production standards
3. Stirring speed
4. Start pumps for subtracting and adding the feed to the miniaturised bioreactor system.
5. Graphical user interface

3.1. Standard parts

In order to make a working control system, basic parts needed in every Arduino controller have to be added. In this section, basic parts will be addressed with a short description of what the basic parts entail.

Arduino Uno Rev 2 Wi-Fi board:

The Arduino Uno Rev 2 Wi-Fi board is one of the most used Arduino boards when working with a connection to the internet. The Arduino Uno Wi-Fi is functionally the same as the Arduino Uno Rev3, but with the addition of Wi-Fi / Bluetooth and some other enhancements. It incorporates the ATmega4809 8-bit microcontroller from Microchip and has an onboard IMU (Inertial Measurement Unit) LSM6DS3TR. It has fourteen digital input/output pins (of which six can be used as PWM outputs), six analogue inputs, a sixteen MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button [24]. The board can be easily connected to a device with the use of a USB printer cable or with the use of the internet. Each Arduino component that will be worked with has to be connected to the power source (5V or 12 V) of the Arduino board. Furthermore, each circuit must end in the GND port to close the circuits [5].

Breadboard:

One of the essential parts when prototyping with a new Arduino control system is the breadboard, which can be used to prototype your circuits. The possibility to temporarily use components rather than committing them to a circuit and soldering them in place is especially useful in the design stage of the project. The setup of breadboards is common. The outside consists of a plastic case with rows and columns of holes. Underneath these rows and columns tracks of copper can be found which enable the possibility to connect components electrically with each other [5].

Multimeter:

The multimeter is a device that helps with the inner understanding of the Arduino circuit that is worked with. A multimeter is a meter that measures volts, amps, and resistance. It can show the electrical values of different components and what is going on with the circuit. With this device, components can be checked on whether they work properly [5].

Jump wires:

The jump wires are used in combination with the breadboard. The insulated wire can be used to connect the Arduino board to components and the breadboard. In the case of Arduino, the wire used most often is insulated equipment wire. This type of wire is generally used on a small scale for low-voltage electrical applications.

The jump wires can be separated into two different varieties, which are single-core and multi-core wires. Single-core jump wires are a single piece of wire that is mostly used in situations where the wire is laid on the breadboard neatly without moving it too much. The multicore wire consists of more little wires that are twisted together, which gives the multicore wire more strength and resistance to bending than a single core wire has. This results in the opportunity to move the multicore wires more often. In the case of this project, multicore wires will be used [5].

Relay shields/Mosfet:

The relay shield is not used in every Arduino system. However, in the system that is developed in this project, the relay shield is essential. The Arduino Relay Shield is a solution for driving high power loads that cannot be controlled by Arduino's digital IOs, due to the current and voltage limits of the controller. Examples of these are the stirrer motor and the base pump. The relay shield can manage the high-power loads by separating the incoming voltage in two sections. The first section is the command section, which is only powered by the 5 Volt which can switch the relay between two pole changeover contacts (Normally Open and Normally Closed). The second section supplies the 12 V needed to activate the pump or the motor [25].

The MOSFET, which stands for Metal Oxide Semiconductor Field Effect Transistor, is a semiconductor device that can be used for amplification of electronic signals in electronic devices. The MOSFET can therefore be used the same as a relay shield. The MOSFET is a four-terminal device which contains terminals for a source, gate, drain and body. MOSFET is generally considered as a transistor and employed in both the analogue and digital circuits [26]. The MOSFET can be used in combination with the actuators that are needed in the Arduino system that will be developed.

3.2. Inputs

The input is the information retrieved from the miniaturised bioreactor that will be read by the Arduino board. Several sensors must be connected to the input ports of Arduino Uno. Each of these sensors uses voltage to translate the information found into usable language for the Arduino Uno. Furthermore, the voltage level is also used to determine the accuracy of the sensors.

3.2.1. Sensor accuracy

The accuracy of each sensor can be calculated using millivolts and the bits the millivolts relate to. The microcontroller inside of the Arduino Uno has a circuit that is called an analogue-to-digital converter. The changing voltage that occurs during the use of the Arduino Uno can be read by the ADC, which thereafter converts the changing voltage to a number between 0 and 1023. The range 0 – 1023 can be explained using the bits. The Arduino Uno has a resolution of 10 bits. This results in a binary number between 00 0000 0000 to 11 1111 1111, which indicates a decimal number of the 0 -1023 as said before. This indirectly means that 0 indicates that 0 V is present in a pin of Arduino Uno and that 1023 indicates that 5 V is present in a pin of Arduino Uno. In between 0 and 5 V, analogRead() returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to a pin of Arduino Uno. Having this knowledge, the accuracy of the Arduino Uno itself can be explained. According to the Arduino Uno datasheet, the accuracy of the ADC in the Arduino Uno is around +/- 2*LSB, which means 5 V / 1023* 2 = 9.8 mV. Compared to the bits, an error of +/- 2*LSB is 2 bits (4 decimal) in 10 bits (1023 decimal), which is a maximum error in case of the accuracy of the converter is 4 /1023 or approximately 1 / 256 which is equal to 0.25% [27]. Having this knowledge, the accuracy of each sensor can be calculated to check if the sensors can efficiently measure the relevant process parameters.

3.2.2. *pH value*

For pH reading, the pH electrode from VWR international will be used. The pH electrode is connected to an Arduino BNC receiver, which can send the measured values to the analogue PIN 2 of Arduino Uno. Thereafter, the analogue readings can be translated into a pH value between 1 and 14. Moreover, the accuracy of the pH sensor can now be calculated. When the total pH value (14) is divided by the 256 (error margin) an accuracy of +/- 0.055 can be found. The pH electrode has to be calibrated with the use of three-point calibration. The reason for this to check whether the pH electrode is accurate between each of these points. With the right code, the pH electrode has to be put in a solution with a pH value of 7. The code has to be adjusted in such a way that the measured pH value corresponds with the solution of pH 7. Thereafter, the same has to be done with a solution with a pH value of 4 and a solution with the pH value of 10. If the measured values do not differ +- 0.1 from the given solutions, the pH electrode is calibrated properly. The components that are used to measure the pH value can be found in the Bill of Materials below. When these items are acquired, the Arduino circuit in figure 4 can be made. It can be seen from figure 4 that the BNC receiver is connected to the power source and the GND port of Arduino UNO [28].

BreadBoard - Half Size	1x
Arduino Uno	1x
USB Cable A to B	1x
pH electrode	1x
BNC receiver Arduino	1x
Jumper Wires Pack - M/M	6x

Table 1: Bill of Materials pH electrode

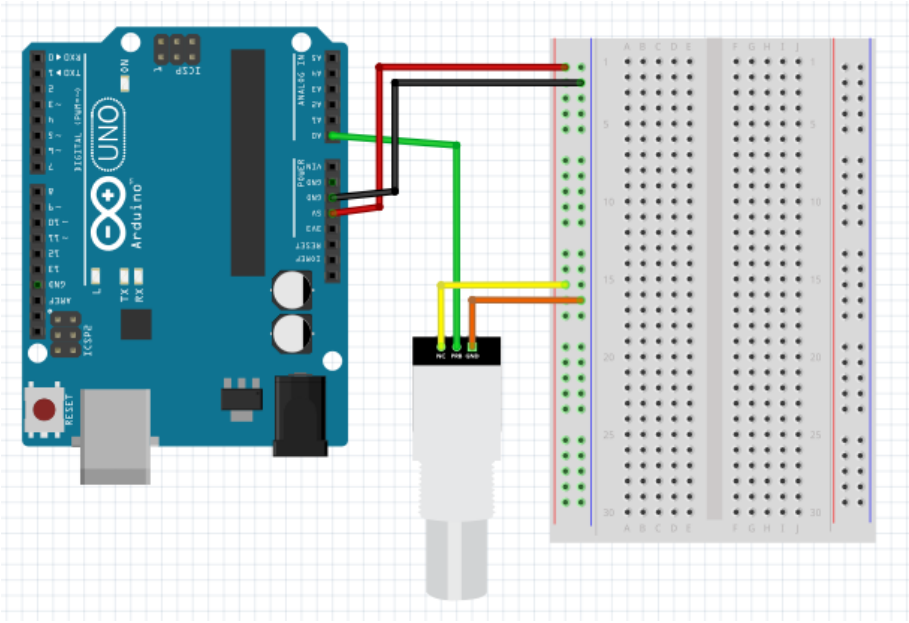

Figure 4: Arduino circuit of the pH electrode

3.2.3. *Temperature value*

In order to measure the temperature value of the water basin, the one-wire bus digital Thermal Sensor DS18B20 will be used. One wire bus means that the Thermal Sensor only requires one data line (and GND) to communicate with the Arduino board. The Thermal Sensor can be powered by the Arduino board due to the power supply range of 3.0 V to 5.5V. Furthermore, the operating ranges of the Thermal Sensor are between -55°C to 125°C. This means that it can handle the temperature that has to be measured ~35 °C with ease. The accuracy of the Thermal Sensor at around 35 °C can again be calculated by dividing the total temperature range (180 °C) by 256 which results in an accuracy of +/- 0.7 °C. However, it is stated in the literature that the Thermal Sensor is more accurate around 35 °C compared to the accuracy at around the boundaries of the Thermal sensor. Therefore, it can be estimated that the accuracy in case of this project is +/- 0.5 °C. The materials that are needed for the thermal sensor to function is shown in the bill of materials below. Besides connecting the Thermal Sensor to the 5v and the GND port, it also has to be connected to any digital or analogue pin [29].

BreadBoard - Half Size	1x
Arduino Uno	1x
4.7K Ohm Resistor	1x
DS18B20 temperature sensor	1x
Jumper Wires Pack - M/M	5x

Table 2: Bill of Materials DS18B20 temperature sensor

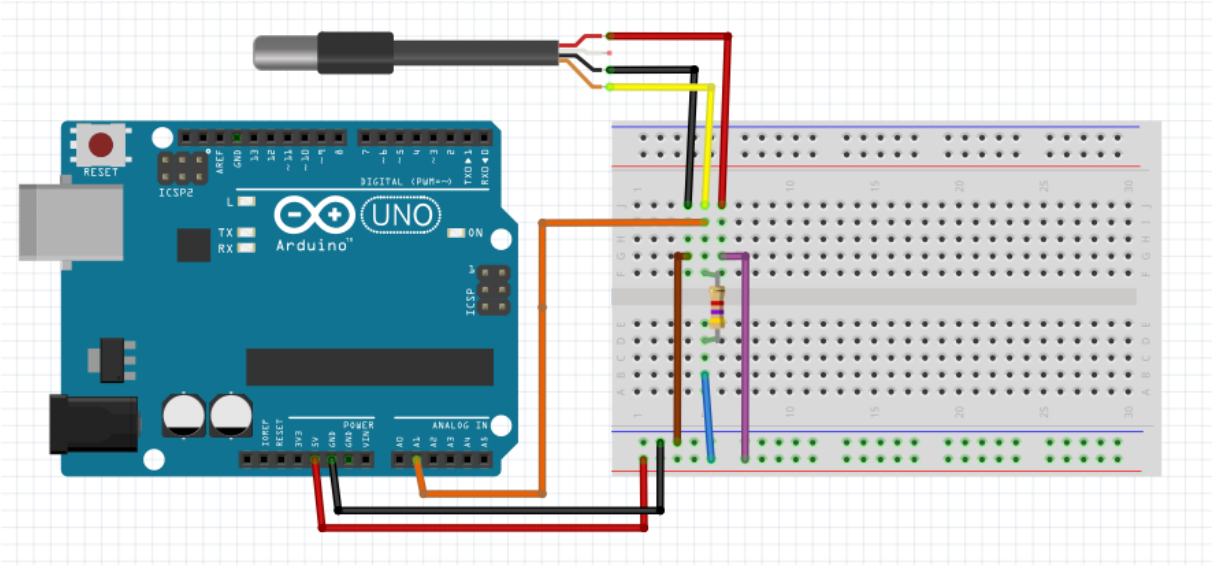

Figure 5: Arduino Circuit DS18B20 temperature sensor

3.2.4. *Biogas flowrate*

The biogas flow rate will be measured according to the difference in weighted water from the system that can be found in figure 3. The Arduino component used for this measurement will be an Arduino load cell bar 100 g combined with the SparkFun HX711 Load Cell Amplifier. These parts are chosen to sufficiently manage the production range of the biogas, which is approximately 1 ml – 40 ml a day [1]. In search of a valid sensor, normal gas flow rate sensors that were found were not accurate enough for the production range. The process range of the flow rate sensors that were found needs a minimum of 0.25 L/min to function. In this project, the biogas has an approximated range of 0.04 ml/h to 1.6 ml/h. This is much lower than the minimum for the gas flow rate sensors. Therefore, the choice has been made to use a Load cell bar with a maximum weight of 100g, whereas the load cell can accurately measure a weight between 0 g to 100 g. The calculation of the accuracy can be done by dividing the total weight range (100g) by 256, which results in an accuracy of approximately +/- 0.4 gram. Therefore, the accuracy of the load cell is accurate enough to measure the biogas that is produced within one day. To translate the measurements of the Load Cell, the SparkFun HX711 will be used. The SparkFun HX711 needs two digital pins to function. The Arduino circuit needs the following parts shown in table 3. The Male Headers Pack- Break-Away are needed to be attached to the SparkFun HX711 to efficiently use the jump wires for connection purposes. The schematic overview of the Arduino circuit of the Load Cell Bar can be found in figure 6.

BreadBoard - Half Size	1x
Arduino Uno	1x
USB Cable A to B	1x
SparkFun HX711 - Load Cell Amplifier	1x
Load Cell Bar 100 g	1x
Jumper Wires Pack - M/M	6x
Male Headers Pack- Break-Away	3x

Table 3: Bill of Materials Biogas flowrate

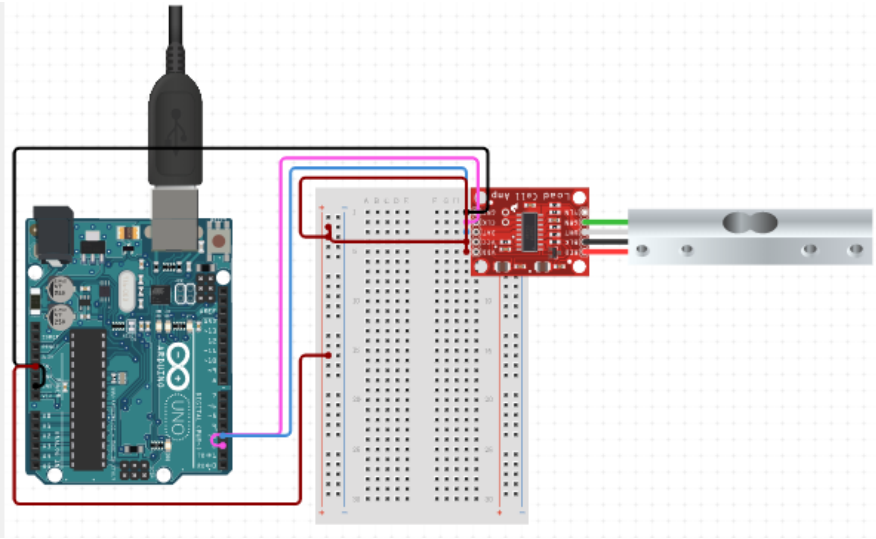

Figure 6: Arduino circuit load cell and SparkFun HX711

3.3. Outputs

The output is the information/commands sent from the Arduino board to actuators that must perform the commands. Each actuator has to be connected to an output port. The commands that need a pump or a motor (e.g., Base pump) also need a relay shield or a MOSFET to process high power loads.

3.3.1. Peristaltic base pump

In order to keep the pH level on the desired value, a peristaltic pump will be used that can pump base into the miniaturised bioreactor. As can be seen from Table 5 and figure 8, the peristaltic base pump needs extra components to work. The N-Channel MOSFET 60V 30A is already mentioned previously. However, the Wall Adapter Power Supply – 12VDC 2A is not mentioned before. The Adapter Power Supply is needed whereas the Arduino UNO/Arduino mega can only supply 5V of power into the system. The 5V is not enough to power the peristaltic pump for it to work at full capacity. The Wall Adapter Power Supply can guarantee enough voltage for the peristaltic pump. The pump that will be used is the Peristaltic Pump with a 12V micro motor. This pump has a max flow rate of 2-3 litre per min and has a maximum head of 2 meters. The bill of materials and the Arduino circuit can be found in table 4 and figure 7, respectively. The Arduino circuit is similar to the arduino circuit of the DC motor used for agitation. This will be shown in figure 9.

BreadBoard - Half Size	1x
Arduino Uno	1x
Peristaltic Pump 12V Micro Motor Pump	1x
Wall Adapter Power Supply - 12VDC 2A	1x
N-Channel MOSFET 60V 30A	1x
10K Ohm Resistor	1x
Jumper Wires Pack - M/M	6x

Table 4: Bill of Materials peristaltic base pump

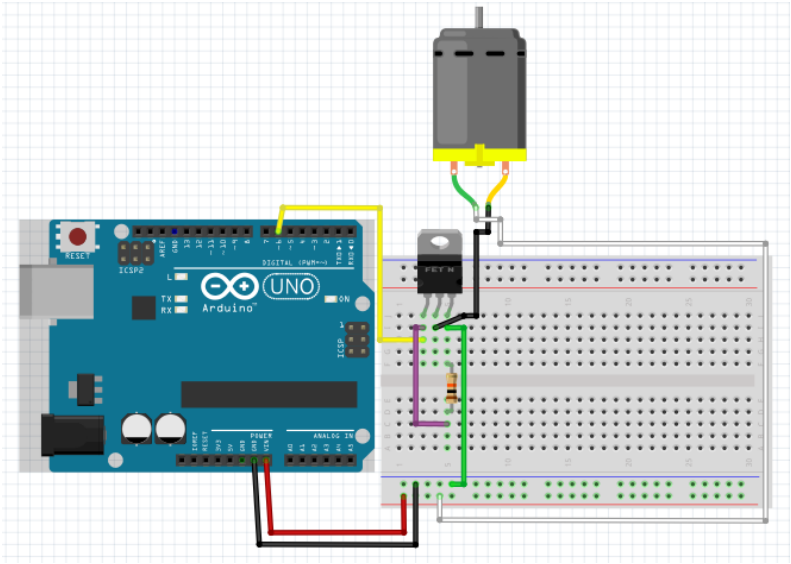

Figure 7: Arduino circuit peristaltic pump

3.3.2. *Activate/deactivate alarm for change in temperature*

The temperature will be managed by a heater that needs interaction from the lab worker. This heater is a separate PID controller that manages the temperature of the water bath. However, when the heater has a malfunction or it suddenly stops, a lab worker has to be alarmed. To alarm the lab worker if the heater does not work properly, an alarm should be made. The alarm that will be developed is based on a Piezo. A Piezo is a sound device which can be programmed to make different noises in different circumstances. The Piezo can easily be connected to the Arduino board by using one of the digital pins. Thereafter, with the right programming, the Piezo can make noise if the temperature moves out of its spec. The parts needed for the connecting of the Piezo can be found in the Bill of Materials below. Together with that, the Arduino circuit necessary for the Piezo to work is shown in figure 8.

BreadBoard - Half Size	1x
Arduino Uno	1x
USB Cable A to B	1x
Piezo	1x
1K Ohm Resistor	1x
Jumper Wires Pack - M/M	4x

Table 5: Bill of Materials Piezo alarm

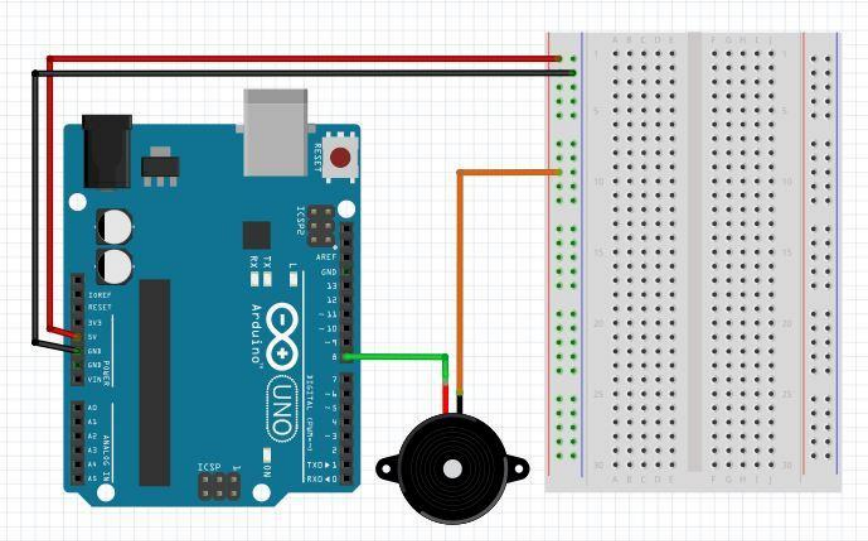

Figure 8: Arduino circuit piezo alarm

3.3.3. *Stirring speed*

The stirring speed that will be managed by the Arduino board is performed by a Micro Spur Gearmotor (6-12V). This motor perfectly fits in the top of the miniaturised bioreactor system that is already developed. To let the motor work, the power supplied by the Wall Adapter Power Supply used for the peristaltic pump is again needed. Moreover, the motor also needs a MOSFET to amplify the power supply from the adapter. The Micro DC Gearmotor is chosen for its size. The dimensions of the DC motor are 10x12x24mm with a 20mm axle, which allows placing the DC motor on top of the miniaturised bioreactor. The DC motor can deliver a speed of 100 to 150 RPM which can be adapted by the Arduino board. The bill of materials is shown in table 4 below. With these materials, the Arduino circuit in figure 9. Also, the DC motor and the peristaltic pump have the same circuits.

BreadBoard - Half Size	1x
Arduino Uno	1x
Micro DC Gearmotor - (6-12V)	1x
Wall Adapter Power Supply - 12VDC 2A	1x
N-Channel MOSFET 60V 30A	1x
10K Ohm Resistor	1x
Diode Rectifier - 1A 50V	1x
Jumper Wires Pack - M/M	9x

Table 6: Bill of Materials Dc spur motor

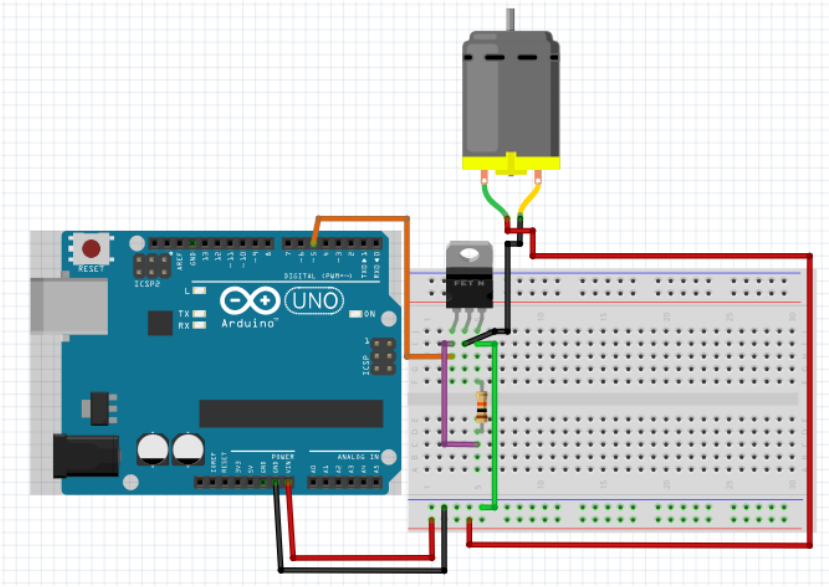

Figure 9: Arduino circuit DC spur motor

3.3.4. Feed

In order to regulate the feed, 2 ml/day has to be pumped in and out of the system to add essential nutrients and extract unusable bio-matter. To do so, two identical peristaltic pumps will be used. Just as the pump and the dc motor, the two identical pumps need a 12 V power supply to work. In addition, the working environment of the pumps needs to be between 0 and 40 degrees Celsius with a relative humidity of <80%. The flow range of the pumps is 0-100 ml/min. Compared to the other peristaltic pump for pumping the base into the system, these pumps have a lower flow rate which results in them being more accurate to pump small quantities of liquid in and out of the miniaturised bioreactor. However, these pumps will be regulated through a relay shield, which differs from the MOSFET's used for the DC motor and the peristaltic base pump. The choice for the relay shield is because the two feed pumps need to work exactly the same for the feed to be successful. Furthermore, the breadboard that manages the actuators is not able to manage more MOSFET's and their electrical circuits due to its size. The relay shield is an Arduino compatible smart module with 2 mechanical relays providing an easy way to control high voltage. The Feed is also the only parameters that make use of both breadboards used in this project (explained in a later section). One of the breadboards will be used as the control of the relay shield, in which 5 V from the Arduino Uno is used to switch the relay from Normally closed to common. The other breadboard will thereafter supply the 12 V that is needed to activate and deactivate the pumps simultaneously. The code will be made in such a way that the pumps will be activated after a determined number of milliseconds. The pumps will work for 1200 milliseconds to add and subtract the desired amount of nutrients and unusable bio-matter respectively. The bill of materials can be found in table 7. Thereafter, the schematic overview of the Arduino circuit for the two peristaltic pumps and the associated relay shield can be found in figure 10.

BreadBoard - Half Size	1x
Arduino Uno	1x
Peristaltic pump 12V	2x
Wall Adapter Power Supply - 12VDC 2A	1x
Relay shield 2 channel 5V	1x
Jumper Wires Pack - M/M	6x
Alligator wire clip	2x

Table 7: Bill of Materials Feed

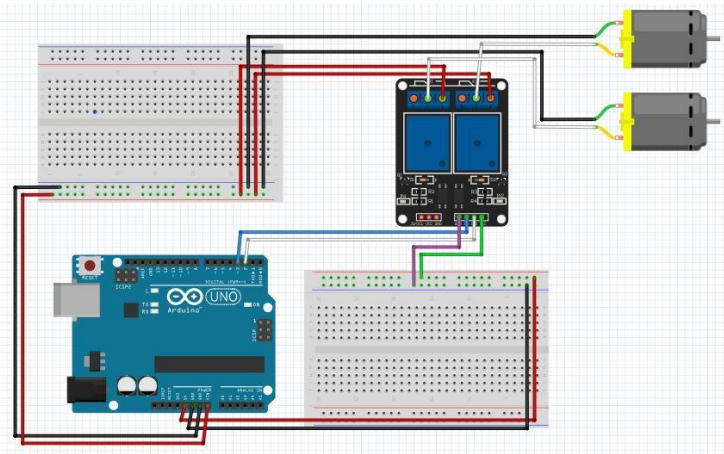

Figure 10: Arduino circuit of the peristaltic pumps and the associated relay shield

4. Arduino Software

After the hardware is assembled, the necessary software should be determined. Within the Arduino IDE, sketches with programming codes can be uploaded to the Arduino mainframe. These sketches consist of basic commands and statements that need to be understood for the sketches to work. In the following section, the basic commands and statements that are used in the code necessary for using the control system are explained. The different basic commands and statements can be categorised in four distinct sections, which are: Comments, Declarations, Void Loop, and Void Setup.

Comments:

Comments in Arduino can be separated into two different sections, which are multiline and single-line comments. The multiline comment normally starts at the beginning of the code. The comment is enclosed within the symbols /* and */. In most cases, the multiline comment is used to provide an explanation or comment on the code before the code is displayed. Comments are completely ignored by the software when the sketch is compiled and uploaded. The single-line comment works almost the same as the multiline comment. However, these comments can be found inside the setup and loop functions. The single-line comment is initialised with symbols // and is used for explaining pieces of code that are harder to understand on itself. Any written code after the // symbols are also ignored during the uploading of the code to the Arduino mainboard [5]

Declarations:

The declarations in the arduino code are values that are stored for later use by the program. The declarations that are used for the development of the control system and their purpose are the following:

- Define, (#define) allows the programmer to give a name to a constant value before the program is compiled. The compiler can replace references to the assigned constants with the defined value.
- Include, (#include) is used to include outside libraries in your sketch. The outside libraries are prewritten pieces of code that will assist in performing the code with the arduino board.
- Integers, (int) are the primary datatype for number storage. The Arduino Uno can store a 16-bit value into an integer. Thereafter, when recalling this integer in a later phase of the code, the Arduino will use the stored 16-bit value.
- Constant, (const) is a variable qualifier that can modify the behaviour of the mentioned variable, making a variable “read only” which means that the variable cannot be changed during the performance of the code. Const and #define can be used for the same purpose.
- Float, (float) is a datatype for floating-point numbers, which are numbers that have a decimal point in them. The float function is often used in place of the integer function due to the greater resolution that the float function holds in comparison with the integer function [5].

After the declarations have been made, the code enters the setup function

Void Setup:
The void setup function only runs once. The purpose of the void setup is to prepare the Arduino board, assigning values and properties to the board that do not change during its operation. The setup function uses curly brackets, { and }, to contain its contents. Each function in Arduino needs a matching set of these brackets. Otherwise, when the number of curly brackets does not match the number of functions, an error will be given.

One of the functions that can be found in the void setup is the pinMode function, which configures a specified pin either for input or output. The pinMode function includes two parameters, namely pin and Mode. Pin stands for the number of the pin whose mode you want to set, and Mode stands for if the pin has to be used as either an INPUT or an OUTPUT. Another used function in void setup is Serial.begin, which sets the communication between the Arduino and the Serial monitor. Serial.begin passes a determined value to the speed parameter. The communication between the Arduino and the Serial Monitor will thereafter exchange messages at a data rate of x bits per second. The value that is most used is 9600 bits per second [5].

Void Loop:
Void loop is like void setup, but instead of running one time, the loop runs continuously until the reset button of the Arduino board is pressed or the power is removed. Within void loop, actual commands can be stated that will thereafter be performed by the Arduino board. One of these commands is digitalWrite, which sends a digital value to a pin. Digital pins have only two states: on or off. In the case of digitalWrite, HIGH or LOW is used. The function includes two parameters, which are: pin, the number of the pin whose mode has to be set, and the choice between HIGH and LOW. Another function that is used repeatedly within void loop is delay. The name of the function speaks for itself. Delay stops the program for an amount of time in milliseconds. During this time, nothing happens [5].

Two other statements that can be found in the script that is needed for the control system to work are the ‘ if ’ statement and the ‘ for ’ statement. These statements are also used in other programming languages due to their clear usability. The if statement checks for a condition and executes the following statement or set of statements if the condition is ‘true’. The if statement is useful when the information required from the measured parameters has to be transformed into action from the pump and the piezo alarm. The for statement can be used to repeat a block of statements. In the develop control system, the for statement can be for example used with measuring different pH value samples to smooth the value of the found pH [5].

4.1. Base pump calculations

Special mention should be made for the code that is necessary to establish a PID controller for the base pump. However, first an explanation should be made about what a PID controller is. PID is an acronym for proportional, integral, and derivative. A PID controller is a controller that includes elements with those three functions. It entails the method of feedback control that uses the PID controller as the main tool [39]. In order to design the PID controller, calculations should be performed about the adjustment of the pH by the base pump. The base that is used by the pump is a NaOH solution (NatriumHydroxide). Several steps have to be followed to calculate the amount of NaOH that has to be added after predetermined intervals. These steps are:

1. Calculate the initial and final number of moles of hydrogen ions in the tank.
2. Subtract the initial number of moles by the final one to obtain the number of moles of hydrogen ions neutralised by hydroxide ions.
3. Since mole ratio of hydrogen ions to hydroxide ions to natrium hydroxide is 1 to 1 to 1 in neutralisation, the number of moles of hydroxide ions and thus NaOH equals the difference calculated in Step 2.
4. Multiply the amount of NaOH by its molar mass to get its mass.
5. Divide the mass of NaOH with the concentration of the base in order to find the amount of mL to add to the bioreactor system.
6. Divide the mL to add with the flowrate of the base pump to determine the amount of seconds the pump needs to be on in order to pump the amount of mL into the system.

pH	pH diff	g of NaOH	mL to add	Sec pump on
7	0	0	0.00	0.00
6.95	0.05	1.46E-08	0.29	0.18
6.9	0.1	3.11E-08	0.62	0.37
6.85	0.15	4.95E-08	0.99	0.59
6.8	0.2	7.02E-08	1.40	0.84
6.75	0.25	9.34E-08	1.87	1.12
6.7	0.3	1.19E-07	2.39	1.43
6.65	0.35	1.49E-07	2.97	1.78
6.6	0.4	1.81E-07	3.63	2.18
6.55	0.45	2.18E-07	4.36	2.62
6.5	0.5	2.59E-07	5.19	3.11
6.45	0.55	3.06E-07	6.12	3.67
6.4	0.6	3.58E-07	7.15	4.29

Table 8: Results base pump calculations

The steps above will be performed for different intervals. It will be assumed that the volume of the liquid within the miniaturised bioreactor is 30mL. This equals 0.03 dm^3 of liquid within the bioreactor. Furthermore, the molar mass of NaOH can be found with the following information. $M_{m.Na}$ = 23, $M_{m.O}$ = 16, $M_{m.H}$ = 1. The combined molar mass is then 40. The flowrate of the base pump is equal to 1.67 mL/s. Table 8 displays the results that have been found.

The amount of NaOH that has to be added can be calculated by following the first four steps that are stated above. First, calculate the initial and final number of moles of hydrogen ions in the tank. The initial moles of hydrogen can be calculated with the following formula, where in.pH is the initial pH and V the volume of the miniaturised bioreactor:

1. $Initial\ moles\ of\ hydrogen = 10^{-in.pH} * V$

Furthermore, the final moles of hydrogen can be calculated with the same formula. However, the initial pH has to be switched with the final pH (fin.pH):

2. $Final\ moles\ of\ hydrogen = 10^{-fin.pH} * V$

The second step is to subtract the initial moles of hydrogen by the final one to obtain the number of moles of hydrogen ions neutralised by the hydroxide ions. Simply combine formula 1 and 2 in order to do so.

3. $Neutralised\ hydrogen\ ions = (10^{-fin.pH} * V) - (10^{-in.pH} * V)$

The third step of the calculations is to determine the moles of NaOH that are needed in order to stabilise the bioreactor system. Since mole ratio of hydrogen ions to hydroxide ions to natrium hydroxide is 1 to 1 to 1 in neutralisation, the number of moles of hydroxide ions and thus NaOH equals the difference calculated in Step 2.

4. $H\ (Hydrogen) : OH\ (Hydroxide) : NaOH\ (Natrium\ Hydroxide) = 1 : 1 : 1$

The fourth step is to calculate the mass of NaOH that is needed in order to change the pH of the system to a preferred value. This can be done by multiplying the amount of NaOH by its molar mass to get its mass. The molar mass of NaOH is indicated with $M_{m.NaOH}$ and equals 40 $\frac{g}{mol}$.

5. $Mass\ of\ NaOH = ((10^{-fin.pH} * V) - (10^{-in.pH} * V)) * M_{m.NaOH}$

The fifth step of the calculations is to find the amount of mL of base that has to be added in order to stabilise the bioreactor system. To do so, the mass of NaOH has to be divided with the concentration of the base. However, the concentration of the base has to be determined. The density needed in order to create a logical amount of mL is determined to be $5 * 10^{-5}$ $\frac{g}{L}$. The concentration is indicated with C_{base}.

6. $mL\ of\ base\ to\ add = \frac{((10^{-fin.pH} * V) - (10^{-in.pH} * V)) * M_{m.NaOH}}{C_{base}}$

Thereafter, the sixth step can be performed in order to determine the amount of seconds the pump needs to be on. By dividing the mL to add with the flowrate of the base pump, the amount of seconds can be found. The flowrate is indicated with Q_{pump}.

7. $Seconds\ pump\ needs\ to\ be\ on = \frac{((10^{-fin.pH} * V) - (10^{-in.pH} * V)) * M_{m.NaOH}}{C_{base} * Q_{pump}}$

The results that are acquired with the given steps are visualised in figure 11. However, before this data can be implemented into the necessary code for the base pump, tests should be performed in order to determine the noise of the pH electrode. After the noise is determined, the code can take into account the noise to make the PID more accurate. The figure below can determine on which points, and how long, the base pump should be active.

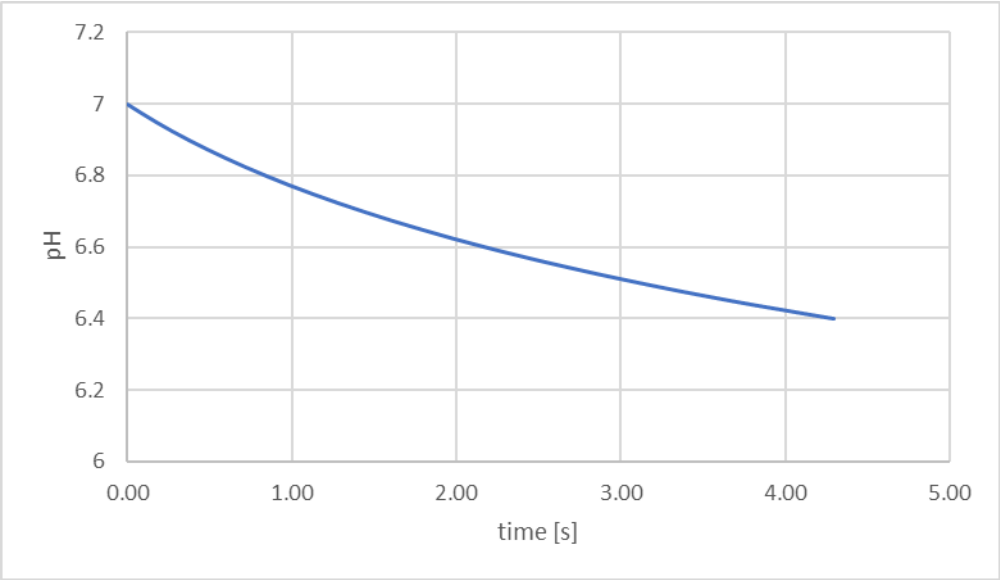

Figure 11: Time the base pump should be on to stabilise the MBR

5. Graphical User Interface

In order to establish a graphical user interface, the software of Processing is used. Processing is a flexible software that, just like Arduino, uses sketches and the C++ language for learning how to code within the context of the visual arts. Furthermore, Processing is free to download on almost any device. Processing and Arduino are compatible with each other, which allows transferring the measured values from the Arduino board to the command window of Processing. In Processing, values can be translated into adjustable meters, which can be achieved by downloading the meter library that can be easily found in the library manager of Processing [30]. Figure 12 shows what the meters can look like when properly used.

Processing is used for several purposes. The first purpose is to visualise the measurement values of the Temperature and the pH. The code that is used for this purpose can be found in Appendix B. Each of the variables (e.g., font type, colour, or size) can be adjusted according to the wishes of the user. The second purpose of Processing would be to, just as with Arduino, alarm the user if the temperature or the pH becomes too low or too high. The alarm is made with the use of the Sound library, which makes it able to upload and play any mp3 sound file within the Processing command window. However, the sound chosen will be played on the device that is used to monitor the process parameters. Therefore, one alarm will sound from the Arduino control system itself and one alarm will sound from the device with the command windows of Arduino and Processing. The code for the alarm can be found in Appendix B. The sound code is not yet added to the rest of the Processing software.

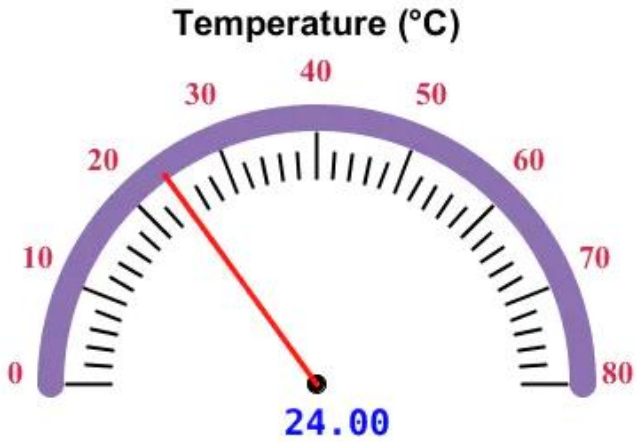

Figure 12: Example of the Temperature meter made with Processing

6. Final Arduino control application

The final Arduino control application consists of all the previously given arduino circuits. Some of the Arduino circuits can be combined in order to minimize the number of parts used. For example, both the arduino circuit of the peristaltic liquid pump and the DC motor use the Wall Adapter Power Supply. Therefore, the parts will be separated using two Breadboards. One Breadboard will manage the components that need extra voltage in order to work, and the other Breadboard will manage the sensors and the piezo alarm. The components needed in order to build the Arduino control system can be found in table 9 below. Thereafter, with the help of Fritzing, a schematic overview of how the control system is built is given in figure 13. It should be mentioned that all the actuators (DC motor, Feed pumps, and the Base pump) are visualised as dc motors. In order to avoid confusion, the names of the actuators and sensors are added to the schematic overview.

Arduino Uno	1x
Breadboard - Half Size	2x
USB Cable A to B	1x
Wall Adapter Power Supply - 12VDC 2A	1x
N-Channel MOSFET 60V 30A	2x
10K Ohm Resistor	2x
4.7K Ohm Resistor	1x
1K Ohm Resistor	1x
Relay shield 2 channels 5V	1x
DS18B20 temperature sensor	1x
Micro DC motor 100 - 1000 RPM (6-12V)	1x
Piezo	1x
Peristaltic Liquid Pump	3x
Silicone Tubing	6x
pH electrode	1x
BNC receiver Arduino	1x
Jumper Wires Pack – M/M	n
Jumper Wires Pack – M/F	n
Jumper Wires Pack – F/F	n
Male Headers Pack- Break-Away	n

Table 9: Bill of Materials complete control system

In order to build the schematic overview above, many steps have to be performed. To perform these steps in a clear and logical order, a manual has been made to clearly indicate which steps have to be taken to assembly all the individual pieces into one working control system. The manual can be found in Appendix C. The pieces and their assembly steps are divided by the breadboard the pieces use. Furthermore, some of the parts had to be soldered to assure a better connection between wires and the actuators and sensors. The components that needed soldering are the three pumps, the dc motor, and the thermometer.

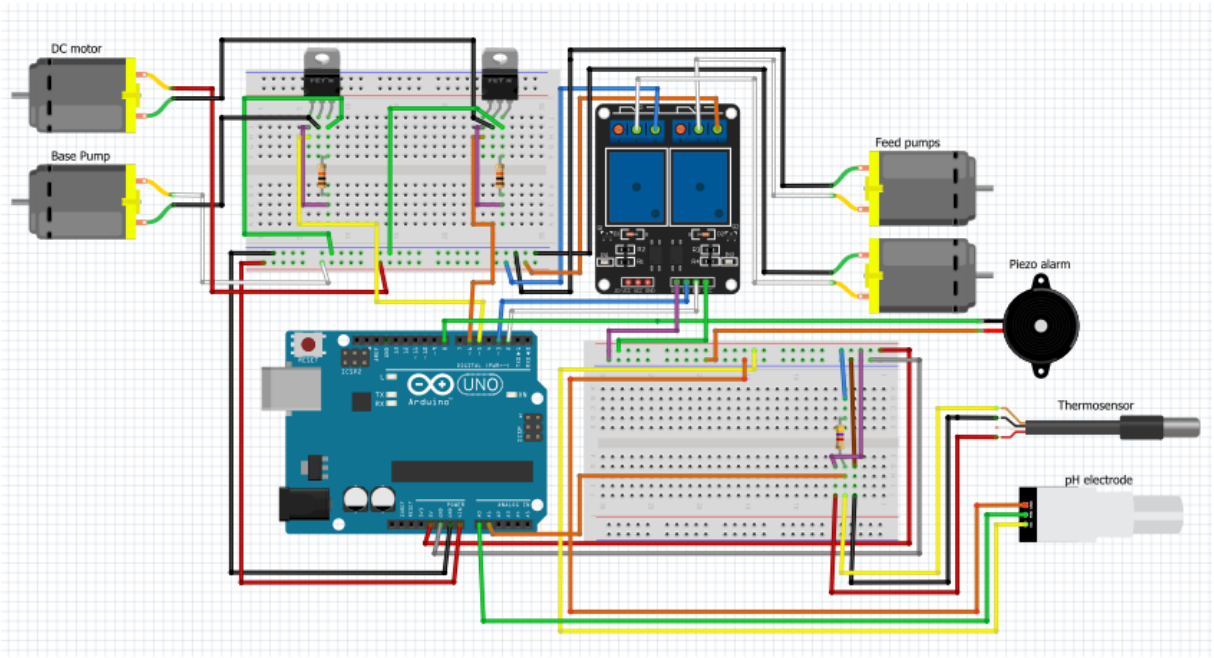

Figure 13: Schematic overview of the final Arduino circuit

The developed control system was built around Arduino UNO rev 2 Wi-Fi which, as stated earlier, provides 14 digital input/output pins and 6 analogue inputs pins. The application uses 5 digital inputs/outputs to control the three pumps (one for the base and two for the feed), the piezo alarm, and the micro-DC motor. Furthermore, 2 analogue inputs are used to read the values that the temperature sensor and the pH electrode measure. However, the weight to determine the biogas flow rate will not be measured. This has no further effect on the remaining components of the control application. Reason for not measuring the weight in the final control system is that the Load cell could not be read and calibrated properly. Therefore, the software and code will not be included in the final control system. However, the code will be shown in Appendix D. Figure 14 shows pictures of the control application assembled in a portable box.

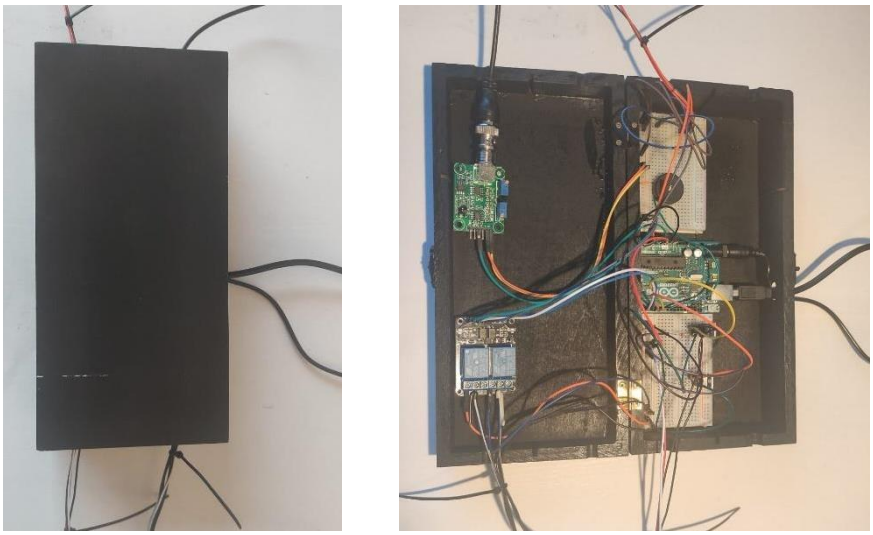

Figure 14: Closed and open picture of the Arduino control application

In order to create a graphical user interface, the Arduino Uno is directly connected to Processing. The command window of Processing can display desired meters explained in the previous section. The parameters displayed are the pH, the temperature, and the on/off situation of the motor. The final layout of the visualisation can be found in figure 15. The Wi-Fi module on the Arduino Uno rev 2 Wi-Fi was intended to be used to upload and control the Arduino system without the use of the USB-B cable. However, the Arduino Uno rev 2 Wi-Fi only uses the internet module to send data to a pre-determined IP address. Information to link the incoming data on the web browser of the IP address to Processing could not be found. Therefore, the choice has been made to aim for wireless connection in the future development of the control system.

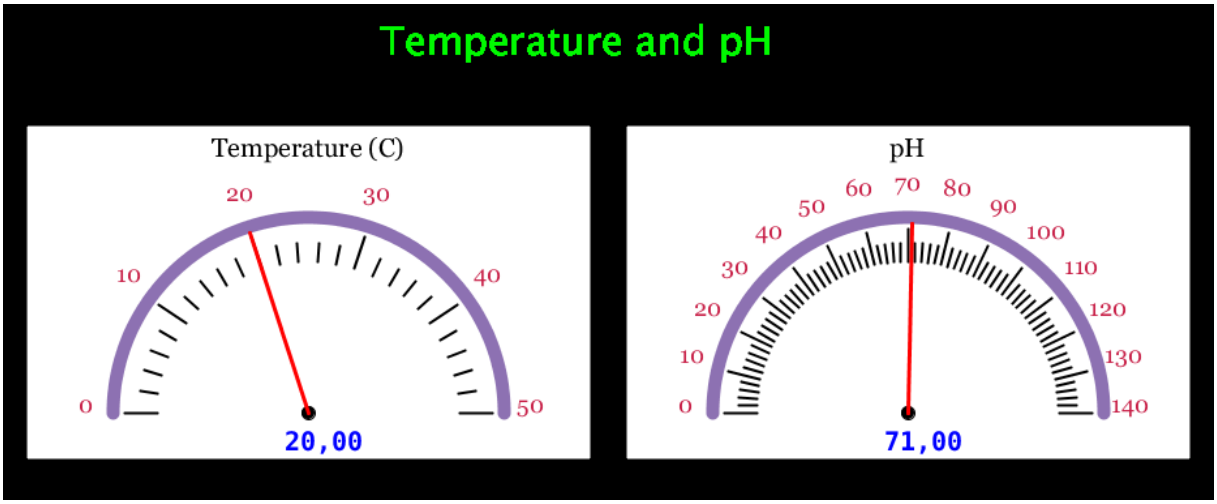

Figure 15: Visualised parameters in Processing

Despite the Arduino community having many open-source well-documented code examples available, writing, and testing the possible combinations of code was a time-consuming process. The current functional version totals 102 code lines including comment and annotations. The code is developed based on the codes that were found while working with each component. However, to combine each piece of code in one sample proved to be difficult. This resulted in a couple of unforeseen consequences which will be talked about in the discussion. The Arduino software for the final control application can be found in Appendix D. Additionally, the code that has been used for Processing can also be found in Appendix C. The code made for processing contains 92 lines of code.

7. Validation

In order to determine whether the control application full fills the need of measuring and controlling the relevant process parameters, simulations have been done. The thermosensor and the pH electrode are tested on several aspects. Each of these aspects are tested over a timeline of five minutes (300 seconds). Firstly, tests were performed on the accuracy of the two sensors. In case of the thermosensor, measurements were performed on water that was adjusting to room temperature. The water has been measured three times to reduce the change of inconveniences. By doing so, it can be seen in figure 16 that the value of the temperature adjusts itself with a difference of approximately 0.065 ˚C. The steps that can be seen are caused by the millivolts that are attached to a predetermined amount of bits explained earlier. These steps could be reduced by taking the average of for example ten measurement values and plotting the found averages against time. An example of this can be found later when analysing the accuracy of the pH electrode.

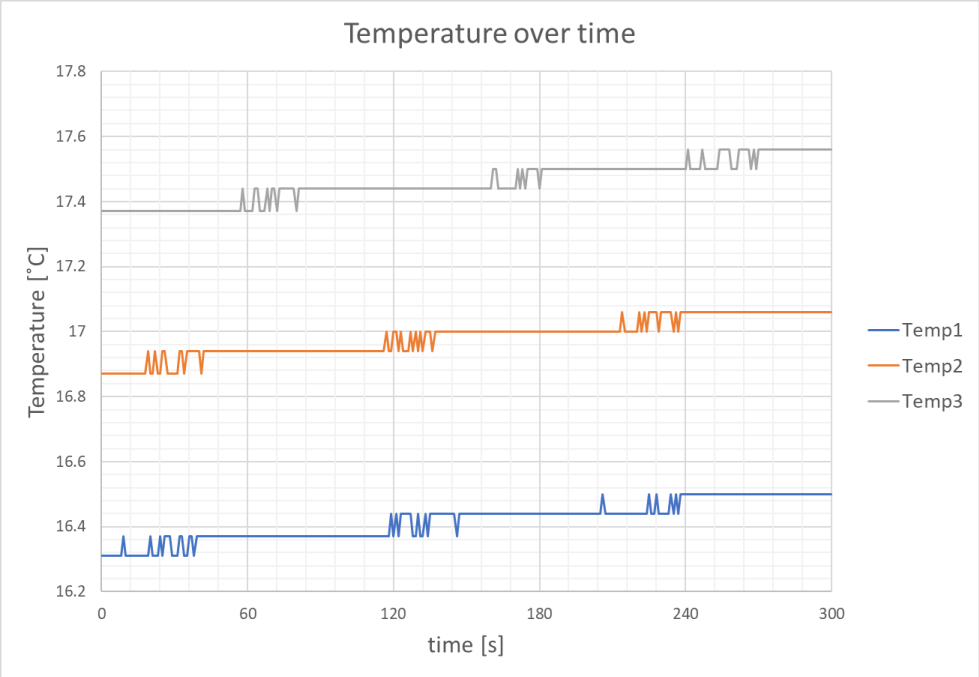

Figure 16: Measured temperature of water adjusting to room temperature

Besides measuring the changing temperature of water, the influence of a sudden temperature change of the water has also been measured and analysed. In two cases, after one minute, 100 mL warm water and 100 mL cold water was added to a flask that contained 200 mL of water to measure the sudden increase and decrease of the temperature. After adding the water, it can be analysed how fast the thermometer can efficiently measure the new temperature of the water. Figure 17 shows the increase and decrease of the temperature respectively. From the results, it can be seen that the temperature sensor is able to measure the new temperature after a delay of 30 seconds.

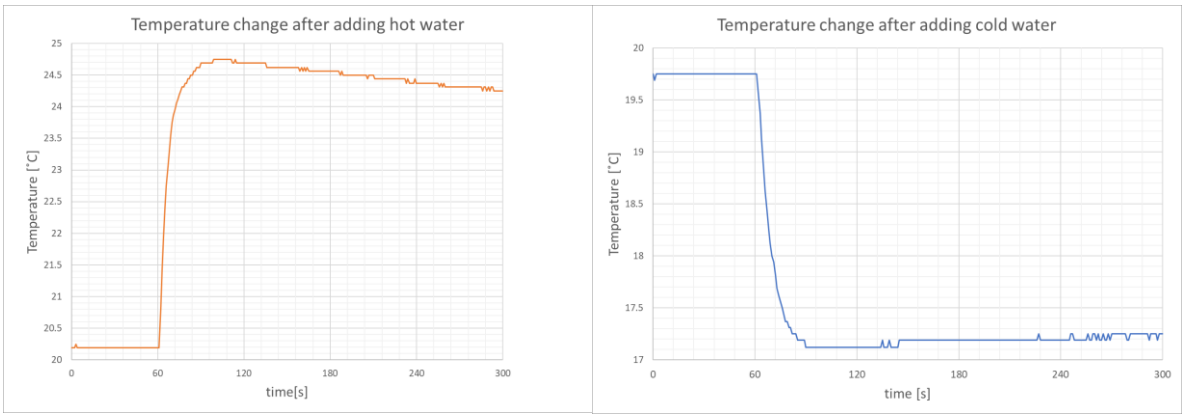

Figure 17: Temperature increase and decrease of the water

The pH value will be measured with a program in which a "for – next loop" is used. This for statement takes ten measured values and will put them in one average value in order to smoothen the measurements mentioned in previous paragraph. The first two pH measurements were performed on a neutral solution with a pH of 7. The first situation was to measure the pH value when the liquid was continuously stirred. The second situation was to measure the pH value when the liquid was not stirred. As can be seen in figure 18, two different results occurred. The first test, which performed the measurement while continuously stirring the liquid, has a relatively constant value at around pH 7. However, the second test, which performed the measurement while the liquid was not stirred, shows a decrease of the measured pH value over time. The decrease in measured pH when not stirring the liquid is caused by the concentration of H^+ around the pH electrode that becomes higher when not stirring the liquid.

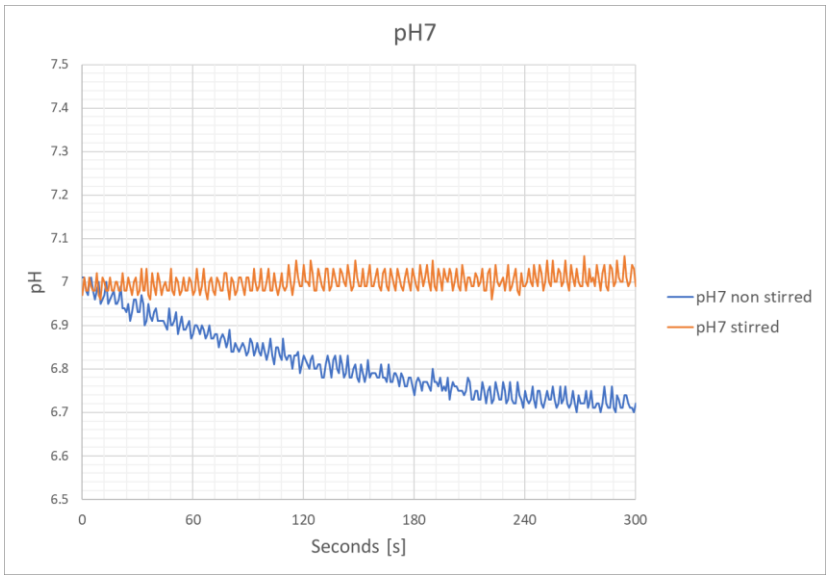

Figure 18: pH measurement of a stirred and non-stirred solution

The values that have been found when stirring the liquid can be further analysed by creating a Signal to Noise ratio (SNR) over time. Figure 19 shows the SNR which indicates how much a measured value divert from what the measured value should have been. What can be seen, is that the noise does not exceed more than 1% of the desired output. In addition, it even shows that 80% of the measured values does not even exceed 0.5% of the desired output.

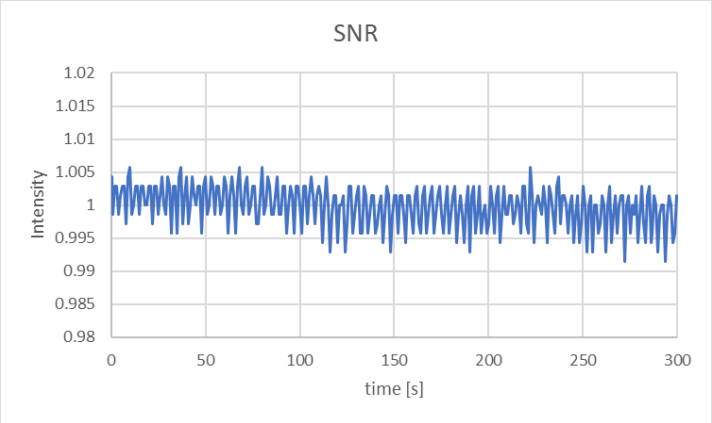

Figure 19: Signal to Noise ratio for the pH electrode

In order to determine how much time the pH electrode needs after sudden pH increase or decrease occurs, two scenarios have been tested. The first scenario was to add an acidic solution after one minute of measuring, which results in the left side of figure 20. What can be seen is that the pH rapidly drops from approximately a pH of 7 to a pH of 5. Thereafter, the pH electrode slowly adjusts to 5.15 where the measured values become stable again. The pH electrode takes approximately 90 seconds to become stable again. In scenario two, the same principle will be used with the addition of a base solution. What can be seen on the right of figure 18 is that the pH increases to 9.15. After this, the measured value becomes relatively constant after 60 seconds at around a value of 9.05.

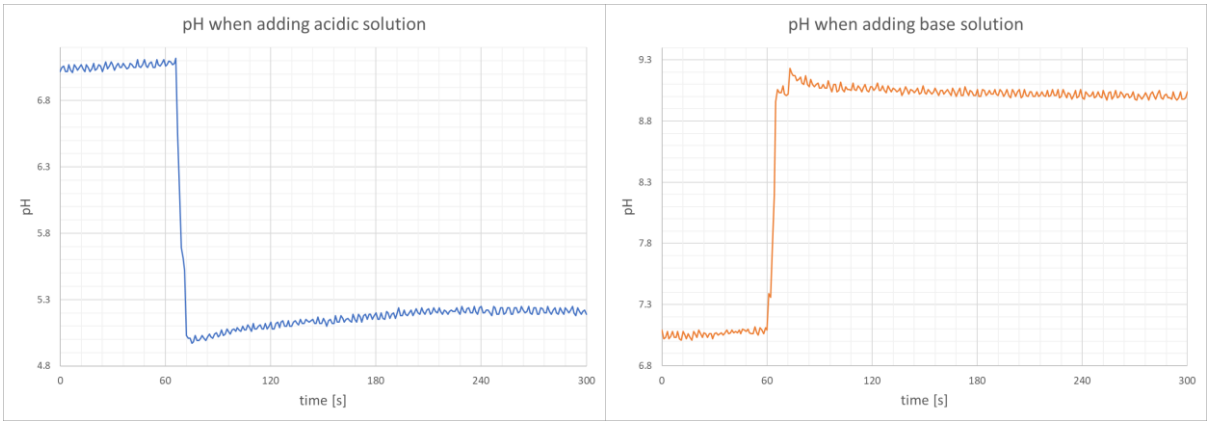

Figure 20: Increase and decrease of the pH value when adding base and acid

The last test that has been performed was the test of the behaviour of the pH electrode in combination with the base pump. Three tests have been done, which consists of two normal tests and one control test. Firstly, the concentrated base was used to gather information about how fast the pH would increase after adding 1 mL. As can be seen in figure 19, after adding 1 mL of base (black arrow), the pH rose immediately to 9.6. Thereafter, this diluted base was used in the next two tests, which had to indicate how the pH would change after adding the new composition of base. The orange line in figure 19 shows that after one minute, the base is added to the liquid (black arrow). What can be seen is a short increase in pH from 6.5 to approximately 7.1 pH. After 100 seconds, the pH had decreased again to 6.8 pH. Again, 5 mL of the thinned base is added to increase the pH (orange arrow). What could be seen after this

is that the pH would become relatively constant 20 seconds after the second sample of diluted base was added. In order to verify if the increase of the pH was not coincidence, a control test has been performed. The grey line in figure 21 indicates the pH increase after adding the base after 60 (black arrow) seconds and 180 seconds (grey arrow) respectively. It can be seen that the grey line follows the same principle as the orange line. Therefore, it can be stated that the pH electrode is able to sufficiently measure the pH difference after base is added.

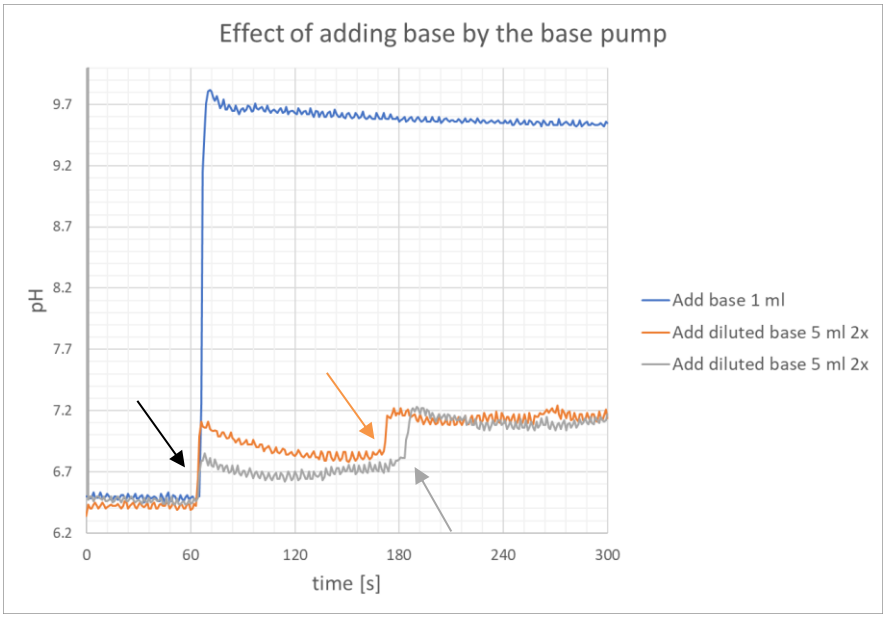

Figure 21:Testing the base pump

8. Market options

The development of the control application for the miniaturised bioreactor also comes with a couple of expenses. To make a clear overview of what the expenses are, table 10 shows the product, quantity, price in euros, and the Company where the product is bought. The prices are based on the price of one individual piece. Hereafter, the total expenses can be calculated. The total expenses are €304.94.

Product	**Quantity**	**Price €**	**Company**
Arduino Uno Wi-Fi Rev2 (Arduino Wi-Fi)	1x	46.95	B0l
Peristaltic pump	1x	14.90	Ben electronics
BNC receiver Arduino	1x	30.90	Bits & parts
PH electrode	1x	116.00	VWR™
DS18B210 thermometer	1x	7.40	Ben electronics
DC motor	1x	6.05	Matronics
USB printer cable	1x	4.94	Allekabels
PHPoC Wi-Fi shield for Arduino	1x	12.48	Amazon
Jump wires m/m m/f f/f (150 tot)	1x	9.48	Amazon
Breadboard half size	2x	5.45	Bits & parts
Adapter 12v 2a dc power supply	1x	13.90	Ben electronics
Peristaltic liquid pump 12 V	2x	27.85	hobbyelectronics
Relay shield 2 channel 5 V	1x	6.95	Kiwi Electronics
Piezo alarm	1x	1.49	Ben electronics
Resistors	4x	0.05 p.p.	Conrad

Table 10: Expenses control system

The price for the control system can be compared to the price for the open bioreactor control system that is found in the project of Marinescu G. C., et al. [8]. In addition, the open bioreactor control system is also the only control system that was currently found to be the only functional implementation simple and ready to be used. Most of the other open-source projects are in beta/incomplete stage and not maintained for years. However, the controller designed in the research of Marinescu G. et al. [8] is specially designed for expressing bacterial protein growth in a bioreactor, while the control system developed in this project is designed for producing biogas by using anaerobic digestion. Therefore, it could be stated that similar products with the same purpose are not yet effectively developed.

Consequently, if investors would invest in and start to sell the control system, the revenue could be positive. However, to achieve positive revenue, several factors have to be considered. Firstly, the product should be optimized by the hand of tests and advice from researchers that will probably work with the control system and the miniaturised bioreactor. Secondly, the product should be adjustable for several purposes. For example, the control system is now developed for the use of anaerobic digestion to produce biogas. The product is currently too specific, which could cause only a small number of customers who would be interested in the control system. A solution for this would be to determine a possible core control system, where several other features (e.g. calibration module for the pH electrode) can be added to the core. In this way, the developed control system can work for example both for anaerobic digestion, bacterial protein expression, or stem cell development. Thirdly, a concise manual should be made on how to assemble the different components into one working control system and how the software has to be used for each part. Thereafter, everyone should be able to assemble the control system which could cause more customers to buy the product whereas there are no technical barriers left.

9. Discussion

The results of this project show that Arduino is a usable tool to monitor, control, and visualise the relevant process parameters of a miniaturised bioreactor system. Tests in a simulated environment have been performed to validate this argument. However, due to the ongoing COVID-19 virus, a real test with the actual working miniaturised bioreactor system could not be performed. Therefore, the effectivity of the control system cannot be fully determined in the given circumstances.

In addition, several difficulties and inconveniences were found in each of the sections while creating the control application. First, the voltage loss while performing tasks with the Arduino control system. Due to the connections being made, a small voltage loss occurs throughout the control application. Several points of the control application have been tested with a multimeter. The measured values indicated that some of the components are operating at a lower voltage level than intended. However, the voltage loss only accounts for 2% for the 5V power supply (0.1V) and approximately 2.1% for the 12V power supply (0.25V). These values can slightly change the output of the measured values. The cause of the voltage loss can have several reasons. The first reason would be that the laptop only delivers a voltage of 4.9V and the wall adapter power supply only 11.75V. However, the chance that both devices have a deviation of approximately 2% is unlikely. The second reason could be the resistors that are being used are a bit to strong to be used in the electrical circuits. This would explain the consistent loss of voltage throughout the controller. The last reason could be the connections that are being made with the use of the jumper wires. The jump wires are only connected with one pin/head to another pin/head. The voltage loss could occur due to a small misdirected flow of current. The voltage loss has been measured with the use of a multimeter. However, the accuracy of the multimeter should also be taken into account.

Secondly, another problem that was found while working with the hardware of Arduino was the Wi-Fi shield. As advertised, the Arduino Uno rev 2 Wi-Fi shield should have the ability to wirelessly connect to the internet of the desktop that is being worked with. However, while doing extra research on the Wi-Fi shield, it seemed that the wireless connection can only be made in one direction. The Wi-Fi shield first needs a connection to the desktop to download the software that is being performed. Thereafter, the Wi-Fi shield can upload the measured values to a determined IP address. This means that a desktop should be attached to the Arduino controller at any given moment. Another desktop could be used in order to read the values that are sent to the determined IP address. However, it could not (yet) be found if the possibility is present to translate the values of the IP address to the processing command window to visualise these values. Further research on this matter should be conducted in order to see if a wireless connection is possible in both directions.

Thirdly, difficulties were found while performing the development of the Arduino software. As is explained in a previous section, the Arduino language works with void setup and void loop. These commands can only be performed once. In order to let the control system work, each of the individual pieces of code with their own void loop command had to be combined in one piece of code. The issues that occurred after combining the pieces of code was the delay function. With the delay function, several other functions of the Arduino also delay the performance of their tasks. This indirectly means that, if the temperature is measured to be

below or above the operating range, a delay of 60 seconds will occur in which the pH cannot be measured and adjusted. The same counts for the opposite, if the pH value is measured to be below the operating range, the temperature cannot be measured. A possible solution for this problem is to adjust the operating ranges so that a small buffer arises. An example of this would be to set the pH operating range from 6.7 to 6.8. When the temperature drops below 34 degrees Celsius, a minute of delay will start. Before the start of the delay, the pH will still be above 6.8. The chance that the pH will be below 6.7 after the minute of delay is very small. Again, the same can work the other way around. In this way, the problem is minimized to a factor that is almost no problem for the miniaturised bioreactor operation. This problem could be solved in several ways. First of all, an extra arduino shield could be added to the system, which makes it possible to manage different tasks on different arduino shields so that the tasks do not interfere with each other. Furthermore, apiece of interval code, which is also used for the feed pumps, could be made in order to make the codes for measuring the pH and the temperature more independent from the loop function.

Another problem that occurred when combining the code and the hardware of arduino was the PID control of the base pump. The results from the calculations for the PID of the base pump indicate that the pump should be active for a couple of seconds to add base into the miniaturised bioreactor system. The amount of seconds that the base pump should be active is approximately 0 to 4.5 seconds. However, the calculated values are based on a concentration for the Natrium Hydroxide of approximately $5.0 * 10^{-5} \frac{g}{L}$. These calculated values do not adequately describe the chemical possibilities that have to be taken into account. The concentration of the NaOH is to such extent not able to be used for this process. In order to achieve such low concentration, the initial NaOH has to be diluted several times before the concentration comes even close to the desired concentration values. If the dilution of the NaOH is not sufficient enough to such extent, other solution should be found. One of these solution would be to change the hardware for the base pump. The base pump has a flow rate of approximately $1.67 \frac{mL}{s}$. If the flow rate can be decreased with a factor 1000, the concentration level could increase with a factor 1000. A concentration of $5.0 * 10^{-2} \frac{g}{L}$ would already be much easier to be diluted from a NaOH solution. Furthermore, what can also be seen from the results of the calculations is that the amount of mL that have to be added to the miniaturised bioreactor exceeds 4 mL in some of the cases. More than 4 mL on a total volume of 30 mL could affect the performance of the bioreactor negatively. Another reason to change the concentration to reduce the amount of mL that have to be added to the miniaturised bioreactor. What should also be mentioned about the base pump calculations is that they are based on a water environment. Normally, when the pH is adjusted in a system, the liquid in the system is not on waterbase but has a buffer of already used chemicals which can affect the effect of adjusting the pH within a miniaturised bioreactor. Therefore, it can be assumed that a higher concentration then the one found in the results of the base pump calculations can be used in order to adjust the pH of the liquid in the miniaturised bioreactor.

Furthermore, another problem that was found during the development of the software was the combination of the Arduino software and the processing software. The measurements that are found with Arduino, have to be translated in such a way that Processing can read the measured

values. However, several aspects of the translation have to be known beforehand to run it smoothly. First, Processing cannot be used when the monitor in Arduino writes the measured parameter values of the miniaturised bioreactor. Secondly, within Arduino, the measured parameter values have to be printed in a string value that separates the values with a comma, which can thereafter be sent to Processing. The string value can then be separated within Processing by assigning each of the individual string values to their own meter. However, in some cases, Processing experiences difficulties in translating the string value which results in an Array index out of bounds exception. This error immediately stops Processing in processing the values from Arduino. The stop of processing can cause future mistakes in monitoring the relevant process parameters. Another problem that occurred while using Processing was the limitation of the m.updateMeter(int(value)) command, which can only update the meter with an integer value. The result of this is that the meter only displays the values of the pH and the temperature as integers. Consequently, the pH can only be shown as 6 or 7. In order to achieve precise measurement, the pH will be multiplied with 10. In this way, the pH will be shown as an integer between the values 65 and 75, which is far more accurate than only 6 and 7.

In section 8. Market options, a clear list of all the expenses can be found. These results are in line with those of previous studies. However, the expenses are based on the components that are needed for one individual miniaturised bioreactor. The GPPB aims for a control system that could also possible control the four multi-parallel miniaturised bioreactor operations. Further development of the control system to control four multi-parallel bioreactor operations, will not influence the price linearly. Reasons for this are the following. When switching from one to four miniaturised bioreactors, the Arduino Uno board should be replaced by the Arduino Mega board. The Arduino Mega board has more digital inputs/outputs and more analogue inputs that can easily control and monitor the four multi-parallel bioreactors. The Arduino Mega board is more expensive than the Arduino Uno board. Furthermore, the expenses for the pH electrode and the micro dc motor can be multiplied by four whereas each miniaturised bioreactor needs their own pH electrode and micro dc motor. However, the expenses of the thermosensor, piezo alarm, and the 100g Load cell stay the same. The thermosensor and the alarm control an external water bath. The water from this bath is transported to each of the four multi-parallel miniaturised bioreactors. In addition, after the production of the biogas has been performed, the biogas will be collected in a flask above the same load cell, which thereafter calculates the combined production yield. When this information is combined, it can be stated that the control system that can be developed for four multi-parallel miniaturised bioreactor operations is relatively cheaper than the control system for one individual miniaturised bioreactor operation. Furthermore, when more parts are needed, the opportunity arises to buy parts in bulk, which can also result in lower total expenses.

The results of this project can also be compared with other examples of control systems and control software. The paper of Chandra, J. A. P., & Samuel, R. D. S. [33] describes the possibility to control a bioreactor system with the use of LabVIEW. The parameters that are being controlled in their research are comparable with the ones in this project, namely: pH, temperature, and agitation. In order to control the parameters, Chandra, J. A. P., & Samuel, R. D. S. use a mathematical approach in which they design models which can simulate the

behaviour of the various parameters. LabVIEW uses basic built-in functions and a control simulation module from which rich features and easy development of control algorithms and system modelling is possible [34]. Furthermore, several minor inconveniences are mentioned while using a PID controller within LabVIEW, but these are easily solved by changing the software. However, the research does not talk about the hardware that is compatible with the control software that is being developed. Arduino and LabVIEW are compatible. Therefore, due to the limitations of Processing, future research in the combination of Arduino as a controller and LabVIEW as the GUI could be interesting.

The combination of Arduino and LabVIEW is not an unknown principle. Husain, A., et al. [10] provide a clear overview of how an Arduino mega board can be linked to LabVIEW to control a miniaturised bioreactor. The parameters that are being controlled in Husain, A., et al. are the temperature and the motor speed. Due to the relatively low number of parameters monitored and controlled, the control system developed is a bit more limited in its application compared to the control application developed in this report. However, the control application developed in Husain, A., et al. shows that has higher accuracy and a better working capability compared to the control application developed in this project. This shows that higher effectivity of the control application designed in this project is possible and should be strived to in future research.

Another example of the use of mathematical control software can be found in the article of Rajinikanth, V. and Latha, K. [35]. The article presents a novel method for modelling and model based PID controller implementation for a nonlinear bioreactor. The software that is being used in this article is POLYMATH, which can generate software data to start with creating an optimal model for controlling the bioreactor. Just as in the previous paragraph, Rajinikanth, V. and Latha, K. talk about creating a mathematical model for optimizing the control of the bioreactor's parameters, without linking it to hardware. However, the mathematical models developed can effectively be used to determine optimal circumstances for the bioreactor system in this project to be even more accurate in determining the optimal operating ranges of the relevant process parameters from the anaerobic digestion in the miniaturised bioreactor.

Further literature analysis indicates that the software of Profibus can also be used as control software. An example of this is shown by Liljequist, V. [36]. Profibus (or PROFIBUS) is an open Fieldbus standard for serial communication with a three-layer protocol structure; Physical, data link and application [37]. In the thesis of Liljequist, V. uses Profibus and .NETframework to control and monitor bioreactors with a volume of 10-2000 L, which is much more than the miniaturised bioreactor studied in this project. However, Liljequist, V. shows that the control application that is designed is suitable for bioreactors systems of such scale. With a change of actuators and sensors, Profibus (and .NETframework) could be suitable software in order to control and monitor the miniaturised bioreactor. However, it is not certain if the accompanying hardware of the miniaturised bioreactor can be combined with the software of Profibus.

The last control software that will be addressed is Lucculus by Hofer A., et al [38], which can implement sophisticated control tasks easily with its Process Information Management

Software. Hardware that is needed to control and monitor the relevant process parameters can be easily put into the Lucculus control systems. Applikon biotechnology [9] is an example of a company that sells the Lucculus software in combination with its developed control systems. However, these control systems are much more expensive than the control application that is developed in this project. Nevertheless, the control systems of Lucculus are much easier to use and are a lot more developed in the way of controlling and monitoring parameters of a bioreactor process compared to the control application that is designed in this project.

10. Conclusion

In this project, a control application for a miniaturised bioreactor system that produces biogas using the Arduino open electronics platform was developed. The control application that is developed, is suitable for visualising and monitoring the relevant process parameters. However, minor adaptations should be made in order to make it suitable for controlling the miniaturised bioreactor system. First, the control application proved effective in monitoring the determined parameters. The control of the relevant process parameters was mathematically suitable. Nevertheless, a chemical point of view suggested that the PID of the base pump should be adjusted in order for it to be sufficient to control the pH value within the bioreactor system. The visualisation of the relevant parameters with the designed graphical user interface proved to be successful. Furthermore, the presented control system is easy to build and cost-effective alternative compared to other control systems that are available for normally scaled bioreactors.

The final Arduino control system has been developed according to several steps that have been followed, namely: determining the system setup, Arduino hardware, Arduino software, and the graphical user interface. Each of these steps proved to consist of its own level of difficulty. The system setup was relatively easy to determine, whereas most literature concerning bioreactors and miniaturised bioreactors can be found directly on the world wide web. Furthermore, the system setup was already predetermined by the GPPB using the information about the designed miniaturised bioreactor. The Arduino hardware and software were a bit more difficult to determine. In case of the Arduino hardware, unforeseen stagnation of the work on the control system occurred due to delivery of not working parts and late delivery of parts. This was managed by focusing on what was already known, and thereafter prepare for what had to be done to make the design process relatively easier to conduct. Hereafter, the Arduino software was determined. Each piece of software from the different components was found and created, after which each of these individual pieces had to be combined in one piece of code. Combining these pieces resulted in a couple of minor issues that were already mentioned in the discussion. Due to late/wrong delivery of the internet shield, the graphical user interface was determined in a very late stage of the designing process. Consequently, the graphical user interface does not have a wireless connection to the Arduino controller. Further research should indicate whether reading the values from IP address and transport them to Processing is a valuable future improvement of the design.

In addition, possible future work on the topic should include several extra aspects, of which one is the following. The tests that are done with the pH electrode should be repeated. However, it should now be done in a miniaturised bioreactor which holds a liquid that is similar in characteristics compared to the eventual liquid that will be used for anaerobic digestion. These tests will properly indicate how the pH can be adjusted with a given concentration of NaOH (NatriumHydroxide).

Furthermore, the control system of the miniaturised bioreactor at its current state (i.e., equipped with temperature, pH, mixing speed, and biogas flow rate measurement) may have a limited application. In order to expand the functionality of the control system, additional features could be added. Examples of these are cell density measurements, a load cell, or even DO level. An example code and setup for the load cell can be found in Appendix D. By adding

these extra features, the capability of the control system will increase. In addition, with the extra features, the control system could be used for several more research purposes which in case increases the interest of possible investors. The second recommendation for future work on the control system would be to upgrade several sensors and actuators. The currently used sensors and actuators full fill the purpose with good accuracy. However, better versions of each component could be used/bought to increase the accuracy even more, which could thereafter also result in a better production rate of the biogas.

Another important aspect of the project was to build a control system that was relatively less expensive than the control systems that are available for bioreactor systems. The total expenses that were found for the control system were determined to be €305.92. This price corresponds to the prediction about the price that was made beforehand. However, when the control system is optimized, the price could become higher due to extra expenses.

Conclusively, experiments show that Arduino is partly suitable as a resource for a control application that can monitor, control and visualise relevant process parameters of a miniaturised bioreactor system. The relevant process parameters can be monitored with a relatively high accuracy. In order to make the Arduino also suitable for controlling the miniaturised bioreactor, different aspects of the control of the pH value should be changed. An example would be to change the hardware for the base pump in order to create a lower flowrate. In addition, Arduino needs the help of Processing in order to visualise the relevant process parameters. Future research and development should be conducted to make the control application more in demand by adding extra features. Thereafter, the control application could be desirable for possible investors whereas the price of the control application does not exceed comparable open-source bioreactor controllers.

11. References

1. Achinas, S., (2020) *PowerPoint slides miniaturised bioreactor system and other general information*. Groningen, s.n.

2. Wang, B., Wang, Z., Chen, T., & Zhao, X. (2020). *Development of novel bioreactor control systems based on smart sensors and actuators*. Frontiers in Bioengineering and Biotechnology, 8, 7. doi:10.3389/fbioe.2020.00007

3. Betts, J. I., & Baganz, F. (2006). Miniature bioreactors: Current practices and future opportunities. *Microbial Cell Factories, 5*(1), 21. doi:10.1186/1475-2859-5-21

4. Shuler, M. L., Kargi, F., & DeLisa, M. P. (2017). *Bioprocess engineering* (Third edition ed.). Boston: Prentice Hall.

5. Nussey, J. (2013). *Arduino for dummies*. Somerset, UNITED KINGDOM: John Wiley & Sons, Incorporated. Retrieved from http://ebookcentral.proquest.com/lib/rug/detail.action?docID=1183913

6. D'Ausilio, A. (2011). *Arduino: A low-cost multipurpose lab equipment*. Behavior Research Methods, 44(2), 305-313. doi:10.3758/s13428-011-0163-z

7. Lintilä, T. (2016). *Bioreactor 0.1*. Retrieved from https://wiki.aalto.fi/display/MechP/Bioreactor+0.1

8. Marinescu G. C., Popescu. R. G., (2018). *Open-source bioreactor controller for bacterial protein expression*. PeerJ Preprints, doi:10.7287/peerj.preprints.27150v1

9. Applikon Biotechnology, 2019. About Applikon Biotechnology. [Online] Available at: https://www.applikon-biotechnology.com/en/about-applikon/[Accessed 10 October 2019].

10. Husain, A. R., Hadad, Y., & Zainal Alam, Muhd Nazrul Hisham. (2016). Development of low-cost microcontroller-based interface for data acquisition and control of microbioreactor operation. *Journal of Laboratory Automation, 21*(5), 660-670. doi:10.1177/2211068215594770

11. Abdelgadir, A., Chen, X., Liu, J., Xie, X., Zhang, J., Zhang, K., . . . Liu, N. (2014). *Characteristics, process parameters, and inner components of anaerobic bioreactors*. BioMed Research International, 2014, 1-10. doi:10.1155/2014/841573

12. Hamilton. (2018). *Biopharma PAT quality attributes, critical process parameters & key performance indicators at the bioreactor*. Retrieved from https://www.sentinelprocess.com/wp/wp-content/uploads/2019/03/How-Does-Process-Analytical-Technology-PAT-Apply-to-the-Bioreactor.pdf

13. Griswold, A. A. (2004). *pH control in a miniaturized bioreactor*. Retrieved from http://hdl.handle.net/1721.1/32812

14. Franco A., Mosquera-Corral A., Campos J.L., Roca E. (2007). Learning to operate anaerobic bioreactors Retrieved from file:///C:/Users/Gebruiker/Downloads/Learning_to_Operate_Anaerobic_Bioreactors.pdf

15. Jain, S., Jain, S., Wolf, I. T., Lee, J., & Tong, Y. W. (2015). *A comprehensive review on operating parameters and different pretreatment methodologies for anaerobic digestion of municipal solid waste*. Renewable & Sustainable Energy Reviews, 52, 142-154. doi:10.1016/j.rser.2015.07.091

16. Collignon, M., & Williams, A. (2020). *Controlling the key parameters of a bioreactor*. Retrieved from https://www.pall.com/en/biotech/blog/bioreactor-process-parameters.html

17. Hoffmann, R. A., Garcia, M. L., Veskivar, M., Karim, K., Al-Dahhan, M. H., & Angenent, L. T. (2008). *Effect of shear on performance and microbial ecology of continuously stirred anaerobic digesters treating animal manure*. Biotechnology and Bioengineering, 100(1), 38-48. doi:10.1002/bit.21730

18. Anukam, A., Mohammadi, A., Naqvi, M., & Granström, K. (2019). *A review of the chemistry of anaerobic digestion* Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-74884

19. Kuo, J., & Dow, J. (2017). *Biogas production from anaerobic digestion of food waste and relevant air quality implications*. Journal of the Air & Waste Management Association (1995), 67(9), 1000-1011. doi:10.1080/10962247.2017.1316326

20. Henrichon, P., (2020). *Gas chromatography: History, methods, and applications* Nova Science. Retrieved from http://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9781536173529

21. Weegman, B. P., Essawy, A., Nash, P., Carlson, A. L., Voltzke, K. J., Geng, Z., . . . Firpo, M. T. (2016). *Nutrient regulation by continuous feeding for large-scale expansion of mammalian cells in spheroids*. Journal of Visualized Experiments, (115) doi:10.3791/52224

22. Faust, G., Janzen, N. H., Bendig, C., Römer, L., Kaufmann, K., & Weuster-Botz, D. (2014). *Feeding strategies enhance high cell density cultivation and protein expression in milliliter scale bioreactors*. Biotechnology Journal, 9(10), 1293-1303. doi:10.1002/biot.201400346

23. CLAES, J. E., GEERAERD, A. H., & VAN IMPE, J. F. (1999). *Heuristic feed rate profiles for optimal yield and productivity of fed-batch bioprocesses*. Chemical Engineering Communications, 172(1), 189-216. doi:10.1080/00986449908912770

24. Arduino, *Arduino Uno Rev2 wifi* http://arduino.cc/en/Main/ArduinoBoardUno

25. How relays work. (2001, May 1,). Plant Engineering, 55, 80. Retrieved from https://search.proquest.com/docview/2210069812

26. Agarwal, T., (2018) *What is a MOSFET : Working and its applications*. Retrieved from https://www.elprocus.com/mosfet-as-a-switch-circuit-diagram-free-circuits/

27. Errington, J., *Precise voltage measurement with the arduino board*. Retrieved from http://www.skillbank.co.uk/arduino/measure.htm

28. Parida, D. (2020). pH meter using arduino uno and LCD display. Retrieved from https://circuitdigest.com/microcontroller-projects/arduino-ph-meter

29. Dimetrov, K. (2016). DS18B20 (digital temperature sensor) and arduino. Retrieved from https://create.arduino.cc/projecthub/TheGadgetBoy/ds18b20-digital-temperature-sensor-and-arduino-9cc806

30. Reas, C., & Fry, B. (2014). *Processing (2. ed. ed.)*. Cambridge, Mass. [u.a.]: MIT Press.

31. Fritzing [computer software] (2020). . https://fritzing.org/:

32. Hadadi, A. *Circuito*. Retrieved from https://www.circuito.io/app?components=512,11021

33. Chandra, J. A. P., & Samuel, R. D. S. (2010). Modeling, simulation and control of bioreactors process parameters - remote experimentation approach. International Journal of Computer Applications, 1(10), 103-110. doi:10.5120/216-365

34. LabVIEW Control Design Toolkit user manual. (2007). National Corporation Instruments.

35. Rajinikanth, V., Latha, K., "Modeling, Analysis, and Intelligent Controller Tuning for a Bioreactor: A Simulation Study", International Scholarly Research Notices, vol. 2012, Article ID 413657, 15 pages, 2012. https://doi.org/10.5402/2012/413657

36. Liljequist, V. (2017). *Development of a bioreactor simulator for supporting automation software test and verification*. Uppsala universitet, Avdelningen för systemteknik.

37. Tan, K. K., & Putra, A. S. (2011). Drives and Control for Industrial Automation (1. Aufl. ed.) Springer Verlag London Limited. Retrieved from http://ebooks.ciando.com/book/index.cfm/bok_id/203600

38. Hofer, A., Kroll, P., Von Rotz, S., & Neutsch, L. (2019). Lucullus® PIMS for event-based process control. Retrieved from https://securecell.ch/en/lucullus-pims-for-event-based-process-control/
39. Johnson, M. A., Moradi, M. H., & Crowe, J. (2005). PID control. London: Springer London, Limited. doi:10.1007/1-84628-148-2 Retrieved from https://ebookcentral.proquest.com/lib/[SITE_ID]/detail.action?docID=303713

Appendix A

Bill of materials for each individual variable.

BoM Dc motor

BreadBoard - Half Size	1x
Arduino Uno	1x
Micro Gearmotor - 90 RPM (6-12V)	1x
Wall Adapter Power Supply - 12VDC 2A	1x
N-Channel MOSFET 60V 30A	1x
10K Ohm Resistor	1x
Jumper Wires Pack - M/M	9x

BoM Peristaltic base pump

BreadBoard - Half Size	1x
Arduino Uno	1x
Peristaltic Liquid Pump with Silicone Tubing	1x
Wall Adapter Power Supply - 12VDC 2A	1x
N-Channel MOSFET 60V 30A	1x
10K Ohm Resistor	1x
Jumper Wires Pack - M/M	6x

BoM DS18B20 temperature sensor

BreadBoard - Half Size	1x
Arduino Uno	1x
4.7K Ohm Resistor	1x
DS18B20 temperature sensor	1x
Jumper Wires Pack - M/M	5x

BoM Piezo alarm control

BreadBoard - Half Size	1x
Arduino Uno	1x
USB Cable A to B	1x
Piezo	1x
1K Ohm Resistor	1x
Jumper Wires Pack - M/M	4x

BoM pH electrode

BreadBoard - Half Size	1x
Arduino Uno	1x
USB Cable A to B	1x
pH electrode	1x
BNC receiver Arduino	1x
Jumper Wires Pack - M/M	6x

BoM Biogas flowrate

BreadBoard - Half Size	1x
Arduino Uno	1x
USB Cable A to B	1x
SparkFun HX711 - Load Cell Amplifier	1x
Load Cell Bar	1x
Jumper Wires Pack - M/M	6x
Male Headers Pack- Break-Away	3x

BoM Feed

BreadBoard - Half Size	1x
Arduino Uno	1x
Peristaltic pump 12V	2x
Wall Adapter Power Supply - 12VDC 2A	1x
Relayshield 2 channel 5V	1x
Jumper Wires Pack - M/M	6x

Appendix B

Manual for building the control system

Breadboard 1

Component:	From:	To:
Jumpwire M/M (red)	Arduino Uno, 5v	Breadboard 1, 1st plus hole
Jumpwire M/M (black)	Arduino Uno, GND	Breadboard 1, 2nd minus hole
Jumpwire M/M (orange)	Arduino Uno, A1	Breadboard 1, H5
Jumpwire M/F (green)	Arduino Uno, A0	BNC connector, Po
Jumpwire M/M (white)	Arduino Uno, 2	Relay shield, IN2
Jumpwire M/M (blue)	Arduino Uno, 3	Relay shield, IN1
Flat Long Jumpwire (green)	Arduino Uno, 8	Breadboard 1, A22
Jumpwire M/M (purple)	Breadboard 1, 3th plus hole	Breadboard 1, D4
Jumpwire M/M (brown)	Breadboard 1, 4th minus	Breadboard1, G6
Flat Short Jumpwire (blue)	Breadboard 1, 5th plus hole	Breadboard 1, B5
Jumpwire M/F (yellow)	Breadboard 1, 12th plus hole	BNC connector, V+
Jumpwire M/F (orange)	Breadboard 1, 13th minus hole	BNC connector, G
Flat Short Jumpwire (orange)	Breadboard 1, 16th minus hole	Breadboard 1, A18
Jumpwire M/M (green)	Breadboard 1, 24th plus hole	Relay shield, VCC
Jumpwire M/M (purple)	Breadboard 1, 25th minus hole	Relay shield, GND
Piezo alarm	Breadboard 1, E18	Breadboard 1, E22
4.7K Ohm Resistor (YPRG)	Breadboard 1, D5	Breadboard 1, G5
Jumpwire M/M (black)	Breadboard 1, J4	DS18B20 black wire
Jumpwire M/M (yellow)	Breadboard 1, J5	DS18B20 yellow wire
Jumpwire M/M (red)	Breadboard 1, J6	DS18B20 red wire
BNC connector	BNC connector	pH electrode

Breadboard 2

Component:	From:	To:
Wall Adapter Power Supply - 12VDC 2A	230 Volt Wall Outlet	Arduino Uno
Jumpwire M/M (red)	Arduino Uno, V_{in}	Breadboard 2, 1st plus hole
Jumpwire M/M (black)	Arduino Uno, GND	Breadboard 2, 2nd minus hole
Jumpwire M/M (yellow)	Arduino Uno, 5	Breadboard 2, H6
Jumpwire M/M (orange)	Arduino Uno, 6	Breadboard 2, H24
wire M/M (wit)*	Arduino Uno, 6th plus hole	Peristaltic pump, +
Jumpwire M/M (black)	Arduino Uno, I7	Peristaltic pump, -
Jumpwire M/M (green)	Breadboard 2, 7th minus hole	Breadboard 2, I8
Jumpwire M/M (purple)	Breadboard 2, C8	Breadboard 2, I6
10K Ohm Resistor (BrBlOS)	Breadboard 2, D8	Breadboard 2, F8
N-Channel MOSFET 60V 30A	Breadboard 2, J6	Breadboard 2, J7 & J8
Jumpwire M/M (red)	Breadboard 2, 11th plus hole	DC motor, +
Jumpwire M/M (black)	Breadboard 2, I25	DC motor, -
Jumpwire M/M (green)	Breadboard 2, 12th minus hole	Breadboard 2, I26
Jumpwire M/M (purple)	Breadboard 2, C26	Breadboard 2, I24
10K Ohm Resistor (BrBlOS)	Breadboard 2, D26	Breadboard 2, F26
N-Channel MOSFET 60V 30A	Breadboard 2, J24	Breadboard 2, J25 & J26
Jumpwire M/M (blue)	Breadboard 2, 22nd plus hole	Relay shield, NO1
Jumpwire M/M (black)	Breadboard 2, 23rd minus hole	Feedpumpin, -
Jumpwire M/M (orange)	Breadboard 2, 24th plus hole	Relay shield, NO2
Jumpwire M/M (black)	Breadboard 2, 25th minus hole	Feedpumpout, -
Wire (white)	Relay shield, COM1	Feedpumpin, +
Wire (white)	Relay shield, COM2	Feedpumpout, +

Appendix C

Arduino Code

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define pHsensor A0              //pHsensor connected to anaolog pin 0

const int Thermometerpin = A1;   //Thermometer connected to analog pin 1
const int Feedpumpin= 2;         //FeedPumpIn connected to digital pin 9
const int Feedpumpout = 3;       //FeedPumpOut connected to digital pin 10
const int motorPin = 5;          //DCmotor connected to digital pin 5
const int basepump= 6;           //Pump connected to digital pin 6


float tempCelsius;                // temperature in Celsius
unsigned long previousMillis = 0;
unsigned long int avgValue;       //Store the average value of the sensor feedback
const long interval = 432000;      // interval at which to start pump half a day milliseconds)

int buf[10],temp;

OneWire oneWire(Thermometerpin);     // setup a oneWire instance
DallasTemperature sensors(&oneWire); // pass oneWire to DallasTemperature library

void setup()
{
 pinMode(Feedpumpin, OUTPUT);        //sets the digital pins as output
 pinMode(Feedpumpout, OUTPUT);
 pinMode(motorPin, OUTPUT);
 pinMode(basepump, OUTPUT);
 Serial.begin(9600);
 sensors.begin();                    // initialize the sensors
}

void loop(){

unsigned long currentMillis = millis();

if (currentMillis - previousMillis >= interval) {
    digitalWrite(Feedpumpin, LOW);
    digitalWrite(Feedpumpout, LOW);
    delay(1200);
   previousMillis = currentMillis;
   }
   else {
    digitalWrite(Feedpumpin, HIGH);
    digitalWrite(Feedpumpout, HIGH);
   }

   digitalWrite(motorPin, HIGH);

for(int i=0;i<10;i++)       //Get 10 sample value from the sensor for smooth the value
  {
    buf[i]=analogRead(pHsensor);
    delay(10);
  }
  for(int i=0;i<9;i++)        //sort the analog from small to large
  {
    for(int j=i+1;j<10;j++)
    {
      if(buf[i]>buf[j])
      {
        temp=buf[i];
```

```
          buf[i]=buf[j];
          buf[j]=temp;
        }
      }
    }
    avgValue=0;
    for(int i=2;i<8;i++)                      //take the average value of 6 center sample
      avgValue+=buf[i];
    float phValue=(float)avgValue*5/1024/6; //convert the analog into millivolt
    phValue=-6.7*phValue+28.9;      //convert the millivolt into pH value


sensors.requestTemperatures();             // send the command to get temperatures
tempCelsius = sensors.getTempCByIndex(0);  // read temperature in Celsius
  Serial.print(tempCelsius);               // print the temperature in Celsius
  Serial.print(",");
  Serial.println(phValue,2);

  delay(500);

if (tempCelsius < 17 || tempCelsius > 22 || phValue < 6.7){
    tone(8, 300);
    delay(10000);
    noTone(8);
    delay(20000);
  }
  else {
    noTone(8);
  }

if (phValue < 6.7){
  digitalWrite(basepump,HIGH); //turns the pump on
  delay(100);
  digitalWrite(basepump,LOW);
  delay(30000);
  }
  else {
  digitalWrite(basepump,LOW);
  }

}
```

Processing Code

```
import meter.*;
import processing.serial.*;

PImage img1;
PImage img2;

Serial port;
String[] list;

Meter m, m2;

void setup() {
  size(950, 450);
  background(0);

  port = new Serial(this, "COM3", 19200);

  fill(120, 50, 0);
  m = new Meter(this, 25, 100);
  // Adjust font color of meter value
  m.setTitleFontSize(20);
  m.setTitleFontName("Georgia");
  m.setTitle("Temperature (C)");
  m.setDisplayDigitalMeterValue(true);

  // Meter Scale
  String[] scaleLabelsT = {"0", "10", "20", "30", "40", "50"};
  m.setScaleLabels(scaleLabelsT);
  m.setScaleFontSize(18);
  m.setScaleFontName("Georgia");
  m.setScaleFontColor(color(200, 30, 70));

  m.setArcColor(color(141, 113, 178));
  m.setArcThickness(10);
  m.setMaxScaleValue(50);

  m.setNeedleThickness(3);

  m.setMinInputSignal(0);
  m.setMaxInputSignal(50);

  // A second meter for reference
  int mx = m.getMeterX();
  int my = m.getMeterY();
  int mw = m.getMeterWidth();

  m2 = new Meter(this, mx + mw + 20, my);
  m2.setTitleFontSize(20);
  m2.setTitleFontName("Georgia");
  m2.setTitle("pH");
  m2.setDisplayDigitalMeterValue(true);

  String[] scaleLabelsH = {"0", "10", "20", "30", "40", "50", "60", "70", "80", "90", "100", "110", "120", "130", "140"};
  m2.setScaleLabels(scaleLabelsH);
  m2.setScaleFontSize(18);
  m2.setScaleFontName("Georgia");
  m2.setScaleFontColor(color(200, 30, 70));

  m2.setArcColor(color(141, 113, 178));
  m2.setArcThickness(10);
  m2.setMaxScaleValue(140);

  m2.setNeedleThickness(3);

  m2.setMinInputSignal(0);
  m2.setMaxInputSignal(140);

}

public void draw() {

  textSize(30);
  fill(0, 250, 0);
  text("Temperature and pH", 300, 50);

  if (port.available() > 0) {

    String val = port.readString();
    list = split(val, ',');
    float temp = float(list[0]);
    float pH = 10*float(list[1]);


    println("Temperature: " + temp + " C  " + "pH: " + pH + "pH");

    m.updateMeter(int(temp));
    m2.updateMeter(int(pH));

    delay (1000);

  }
}
```

Processing sound code

```
import processing.sound.*;
SoundFile file;

void setup() {
  size(640, 360);
  background(255);

  // Load a soundfile from the /data folder of the sketch and play it back
  file = new SoundFile(this, "sample.mp3");


  if (temp < 18 || temp > 21 || pH  < 6.7) {

  file.play();
  delay (10000);
  }
```

Appendix D

Calibration software

```
#include "HX711.h"

#define LOADCELL_DOUT_PIN  3
#define LOADCELL_SCK_PIN  3

HX711 scale;

float calibration_factor = -10000; //-7050 worked for my 440lb max scale setup

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on scale");
  Serial.println("Press + or a to increase calibration factor");
  Serial.println("Press - or z to decrease calibration factor");

  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale();
  scale.tare(); //Reset the scale to 0

  long zero_factor = scale.read_average(); //Get a baseline reading
  Serial.print("Zero factor: "); //This can be used to remove the need to tare the scale. Useful in permanent scale projects.
  Serial.println(zero_factor);
}

void loop() {

  scale.set_scale(calibration_factor); //Adjust to this calibration factor
  Serial.print("Reading: ");
  Serial.print(scale.get_units(), 3);
```

```
  Serial.print(" kg"); //Change this to kg and re-adjust the calibration factor if you follow SI units like a sane person
  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();

  if(Serial.available())
  {
    char temp = Serial.read();
    if(temp == '+' || temp == 'a')
      calibration_factor += 10;
    else if(temp == '-' || temp == 'z')
      calibration_factor -= 10;
  }
}
```

Weight software

```
#include <HX711_ADC.h>
#include <EEPROM.h>

//pins:
const int HX711_dout = 2; //mcu > HX711 dout pin
const int HX711_sck = 3; //mcu > HX711 sck pin

//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
unsigned long t = 0;

void setup() {
  Serial.begin(9600); delay(10);
  Serial.println();
  Serial.println("Starting...");

  LoadCell.begin();
  float calibrationValue; // calibration value (see example file "Calibration.ino")
  calibrationValue = 696.0; // uncomment this if you want to set the calibration value in the sketch
#if defined(ESP8266)|| defined(ESP32)
  //EEPROM.begin(512); // uncomment this if you use ESP8266/ESP32 and want to fetch the calibration value from eeprom
#endif
  //EEPROM.get(calVal_eepromAdress, calibrationValue); // uncomment this if you want to fetch the calibration value from eeprom

  unsigned long stabilizingtime = 2000; // preciscion right after power-up can be improved by adding a few seconds of stabilizing time
  boolean _tare = true; //set this to false if you don't want tare to be performed in the next step
  LoadCell.start(stabilizingtime, _tare);
  if (LoadCell.getTareTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
```

```
    while (1);
  }
  else {
    LoadCell.setCalFactor(calibrationValue); // set calibration value (float)
    Serial.println("Startup is complete");
  }
}

void loop() {
  static boolean newDataReady = 0;
  const int serialPrintInterval = 0; //increase value to slow down serial print activity

  // check for new data/start next conversion:
  if (LoadCell.update()) newDataReady = true;

  // get smoothed value from the dataset:
  if (newDataReady) {
    if (millis() > t + serialPrintInterval) {
      float i = LoadCell.getData();
      Serial.print("Load_cell output val: ");
      Serial.println(i);
      newDataReady = 0;
      t = millis();
    }
  }

  // receive command from serial terminal, send 't' to initiate tare operation:
  if (Serial.available() > 0) {
    char inByte = Serial.read();
    if (inByte == 't') LoadCell.tareNoDelay();
  }
```

Expenses load cell

Load cell 100g	1x	15,78, - xx	Grandado
Sparkfun load cell amplifier	1x	14,45, - xx	Kiwi electronics