



Development of a bioreactor system and programming of an Arduino control unit

Guðrún Kristín Einarsdóttir



Faculty of Computer Science
University of Iceland
2014

DEVELOPMENT OF A BIOREACTOR SYSTEM AND PROGRAMMING OF AN ARDUINO CONTROL UNIT

Guðrún Kristín Einarsdóttir

12 ECTS thesis submitted in partial fulfillment of a
Baccalaureus Scientiarum degree in Computer Science

Advisors

Hjálmtýr Hafsteinsson

Paolo Gargiulo

Faculty of Computer Science
School of Engineering and Natural Sciences
University of Iceland
Reykjavik, May 2014

Development of a bioreactor system and programming of an Arduino control unit
12 ECTS thesis submitted in partial fulfillment of a B.Sc. degree in Computer Science

Copyright © 2014 Guðrún Kristín Einarsdóttir
All rights reserved

Faculty of Computer Science
School of Engineering and Natural Sciences
University of Iceland
Hjarðarhaga 2-6
107, Reykjavík, Reykjavík
Iceland

Telephone: 525 4000

Bibliographic information:

Guðrún Kristín Einarsdóttir, 2014, Development of a bioreactor system and programming of an Arduino control unit, B.Sc. thesis, Faculty of Computer Science, University of Iceland.

Printing: Háskólaprent, Fálkagata 2, 107 Reykjavík
Reykjavík, Iceland, May 2014

Abstract

The object of this project was to continue the development of a flow perfusion bioreactor used for bone stem cell culture at the Icelandic Blood Bank. The bioreactor will be used to assess the effect of flow perfusion on osteogenesis (the formation of bone tissue) as it has been shown to assist cellular growth of bone stem cells [1].

The first part of the project consisted of improving the bioreactor to make it sustain a month in cell culture within a CO_2 incubator. This part proved to be challenging as the previous design of the bioreactor was very unstable and the motor got blocked after running for a few minutes. After multiple alterations on the design, the bioreactor was able to run for days without the motor blocking so it was considered ready for the cell culture. The second part of the project was the programming of a control unit for the bioreactor with an Arduino microcontroller. The requirements for the program were to have different functions to make the motor in the bioreactor run forward, backward and in pulse motion (frequently switching between forward and backward). The last part of the project was the programming of a graphical user interface for the control unit to make it available to use for people with non-technical background.

Útdráttur

Markmið þessa verkefnis var að halda áfram þróun á ræktunartanki (e. bioreactor) fyrir ræktun beinstofnfrumna í Blóðbankanum. Ræktunartankurinn verður notaður til að meta áhrif gegnumflæðis á beinmyndun þar sem sýnt hefur verið fram á að það geti bætt vöxt beinfrumna [1].

Fyrsti hluti verkefnisins fólst í því að bæta ræktunartankinn þannig að hann þyldi mánuð í frumuræktun í CO_2 hitaskáp. Þessi hluti verkefnisins reyndist erfiður þar sem fyrri hönnunin á tankinum var mjög óstöðug og mótorinn læstist eftir að hafa keyrt í fáeinar mínútur. Eftir margþættar breytingar á hönnuninni náðist að keyra ræktunartankinn í nokkra daga án þess að mótorinn læstist svo tankurinn var álitinn hæfur til frumuræktunar. Annar hluti verkefnisins var forritun Arduino örtölvu sem stjórnkerfi fyrir ræktunartankinn. Kröfur á forritinu voru að hafa mismunandi aðferðir til að keyra mótorinn í tankinum réttsælis, rangsælis og í púlshreyfingu (hröð skipting á milli réttsælis og rangsælis). Seinasti hluti verkefnisins var svo forritun grafíks notendaviðmóts fyrir stjórnkerfið til að gera fólki sem hefur e.t.v. ekki reynslu af forritun kleift að stýra tankinum.

Contents

1	Bioreactors	1
2	Biological Aspects	2
2.1	Tissue Engineering	2
2.2	Cell Culture	2
3	Design	4
3.1	Previous Design	4
3.2	Modifications	6
3.3	Electronic design	7
4	Programming	9
4.1	Arduino	9
4.1.1	Motor velocity control	9
4.1.2	Arduino code examples	10
4.2	Graphical User Interface	12
5	Discussion	17
5.1	Further Development	17
5.1.1	3D Printing	17
5.1.2	Flow Meter	17
5.1.3	Data Logging	18
5.1.4	Smartphone/ Tablet interface	18
	References	19

1 Bioreactors

A bioreactor is a container used to monitor and support biological culture. Bioreactors differ in complexity, depending on their purpose. In tissue engineering, cells are seeded on a scaffold (a three-dimensional supportive structure) which is then immersed in media inside the bioreactor for culture. The bioreactor is used to generate mechanical, electrical or chemical stimulation that affects the growing cells to enhance specific development of the tissue. This can be used for growing different types of cells for different applications, including growth of three-dimensional bone tissue that can be used for implantation [2].

In this project, the development of a flow perfusion bioreactor was continued for growing three dimensional bone tissue on titanium scaffolds. Flow perfusion bioreactors create a flow of the cell media through the pores of the scaffolds to enhance nutrient transfer, and therefore growth, through the thickness of the scaffold. The shear forces that apply on the scaffolds due to the media flow have also been shown to favour the expression of the osteoblastic (bone synthesizing cell) phenotype [2].

The purpose of the bioreactor in this project was to create a flow perfusion through the titanium scaffolds. The results would then work as a proof of concept that growing stem cells on scaffolds is more efficient in a perfusion system than in a static culture.

2 Biological Aspects

2.1 Tissue Engineering

Tissue engineering is the application of engineering and biological methods to develop replacements of biological tissue. It is a recent but growing field within bio-engineering with an interdisciplinary approach. Tissue engineering often concerns the generation of neotissue from cells on a scaffold. The scaffold is a porous structure used for mechanical support and works as an extracellular matrix for the cells. The scaffold must be made of bioabsorbable materials in order not to damage the tissue [3].

2.2 Cell Culture

The cell culture for this project took place at the Icelandic Blood Bank under the supervision of PhD student Sandra Mjöll Jónsdóttir. Half of the cells were grown on scaffolds inside the bioreactor while the other half was used as a control group grown in a static culture. The cells used were human embryonic stem cell-derived mesenchymal progenitor cells (hES-MP) which are stem cells from human embryos that can differentiate into different cell types, including osteocytes (bone cells) [4].

The cells are initially seeded on chitosan coated titanium scaffolds where they expand. The cell culture takes place inside a CO_2 incubator so the bioreactor must be able to withstand the temperature and humidity inside it. However, the control unit with the Arduino microcontroller was placed outside the incubator with a wire connecting it to the bioreactor. Figure 2.1 shows the bioreactor inside the incubator during cell culture. During the culture, the study group of the cells was transferred to the bioreactor system where the cells were differentiated towards osteocytes. Cell differentiation is the phenotypic change of a cell to a specialized cell and is obtained through changes in gene expression. This is done by immersing the cells in media that enhances the cell differentiation [4].



Figure 2.1: The bioreactor inside the CO₂ incubator.

3 Design

3.1 Previous Design

At the beginning of this project the bioreactor had already been built but had not proven successful when it came to stability. Figure 3.1 shows the bioreactor prior to the improvements made in this project. For the second version, the main emphasis was put on improving the design with respect to stability as the bioreactor has to be able to run for several weeks without interruption. The motor was very unstable which was likely because the wires connecting the motor to the Arduino microcontroller were vulnerable and prone to loss of contact.

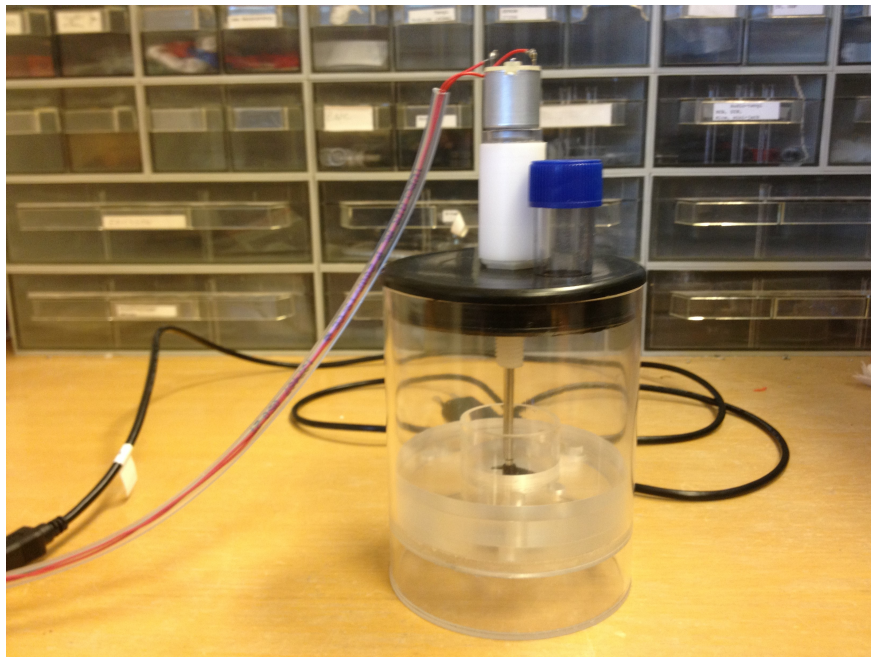


Figure 3.1: The bioreactor prior to improvements [5].

The bioreactor is made of a plastic container and a DC motor to which a steel rod is connected to turn a plastic propeller near the bottom of the container. Inside the container there are three plastic discs placed on top of each other with three vertical tubes in it, to contain the scaffolds. Two sheets of aluminium net are placed

between the discs so the scaffolds stay put when the liquid starts to flow through the tubes (see Figure 3.2 and Figure 3.3).

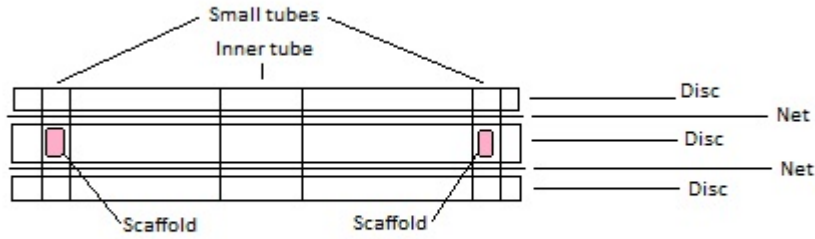


Figure 3.2: Schematic figure of the discs and aluminium nets inside the bioreactor [5].

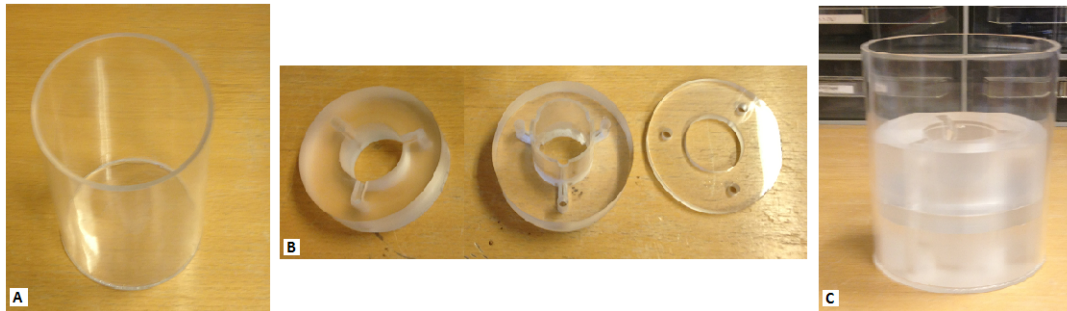


Figure 3.3: A) The plastic container. B) The three plastic discs. C) The container and the discs assembled [5].

When the motor starts spinning the propeller, the liquid inside the container starts flowing down the hollow cylinder in the middle of the bioreactor and up the three vertical tubes in the plastic discs as shown in Figure 3.4 or the other way around, depending on the spinning direction.

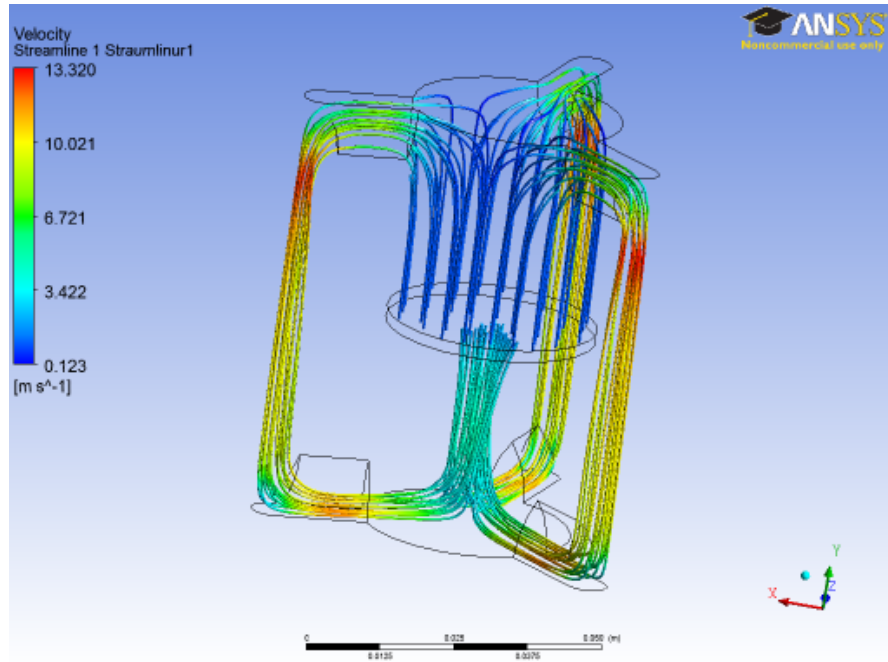


Figure 3.4: Flow simulation of the previous version of the bioreactor [5].

3.2 Modifications

It was clear from the start that modifications would have to be done on the motor-propeller system as the motor was unable to run for longer than a few minutes before it would get blocked. The stability of the motor became a major obstacle during the project and in the end, the motor-propeller system was replaced with a handle from an IKEA milk-frother, a new DC motor and the original steel rod connected to the handle. Additionally, a support for the handle was made by heating a piece of a soft plastic hose and then placing the handle within it to make perfect fit. These modifications proved successful even though inexpensive components were used that were available at the laboratory. Other modifications of the design include enforced wiring to prevent disconnection to the microcontroller and addition of silicon legs to minimize vibration of the system. Figure 3.5 shows the bioreactor after the improvements.

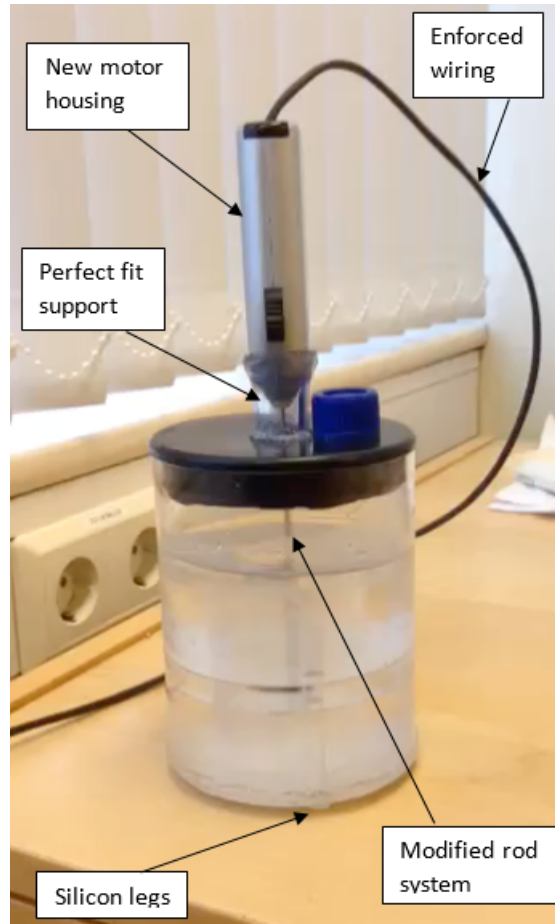


Figure 3.5: The bioreactor after the improvements.

3.3 Electronic design

The electronic design of the bioreactor was kept unchanged from the previous version. The motor in the bioreactor is controlled with an Arduino microcontroller through a quadruple half H-driver. The H-driver has four H-bridges but only one of them is used for this project. The H-bridge is used to enable voltage to be applied in either direction of the DC motor so it can run both forwards and backwards. The H-driver and the Arduino board are soldered on a circuit board as shown in Figure 3.6. A breadboard- and schematic diagram of the electronic circuit board were created using Fritzing, an open source software for documenting electronic prototypes. The diagrams are shown in Figure 3.7 and Figure 3.8.

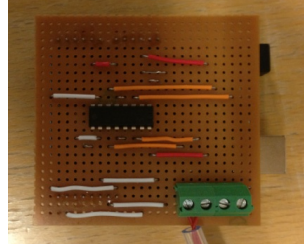


Figure 3.6: The circuit board with the H-bridge in the middle (black rectangle) and the Arduino on the other side [5].

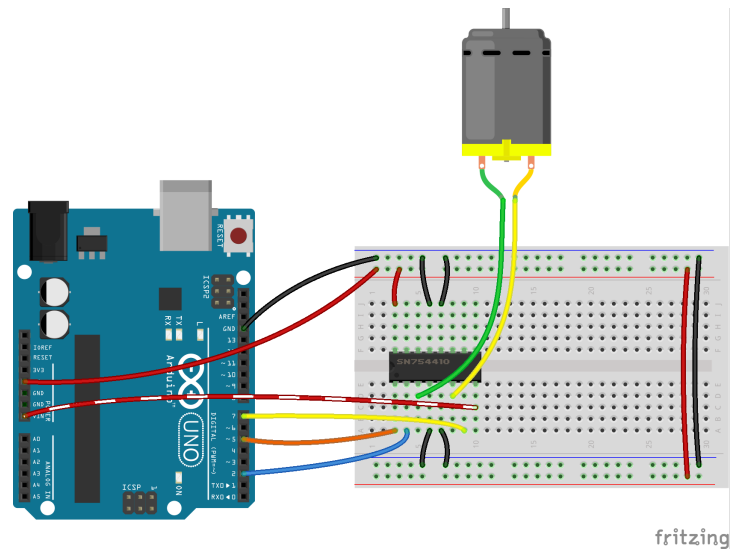


Figure 3.7: The breadboard design of the circuit.

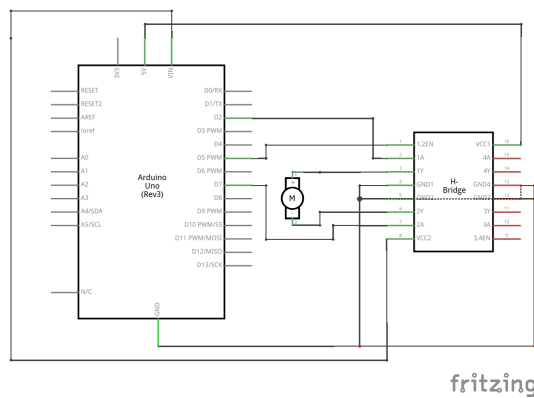


Figure 3.8: The schematic design of the circuit.

4 Programming

4.1 Arduino

An Arduino Uno microcontroller was used as a control unit for the bioreactor. The Arduino is an open-source electronics prototyping board with a boot loader for program uploading. The Arduino Uno board has 14 digital input/output pins and 6 analog input pins which operate at 5 volts. The program uploaded on the Arduino board is written in C/C++ in the Arduino integrated development environment (IDE). The input/output pins on the board that will be used in a program are defined at the start within a *setup()* function in the code. Each used pin must be given a name and defined either as an output or an input pin. A typical Arduino program loops while the Arduino board is turned on. The repeatable part of the program is placed within a *loop()* function in the code. When the Arduino program is uploaded to the board, an extra include header is added at the top along with a *main()* function at the bottom so it becomes a valid C++ program. [6]

4.1.1 Motor velocity control

The motor velocity is controlled with a Pulse Width Modulation (PWM) technique which creates analog output from digital input in the Arduino. The Arduino is able to send either 0 or 5 volts to its output pins so in order to control the velocity of the motor, a square wave of the output voltage is created. As an example, if 2.5 volts are needed at an output pin, then the Arduino sends 5 volts to the pin 50% of the time and 0 volts 50% of the time with a very frequent transition (around 2 milliseconds between transitions). Then the mean output voltage is 2.5 volts and the actual output voltage appears as a steady 2.5 volts because of the fast transition between 0 and 5 volts. In an Arduino code, the PWM values range between 0 and 255 so a PWM of 127 will give 50% of the maximum velocity [7].

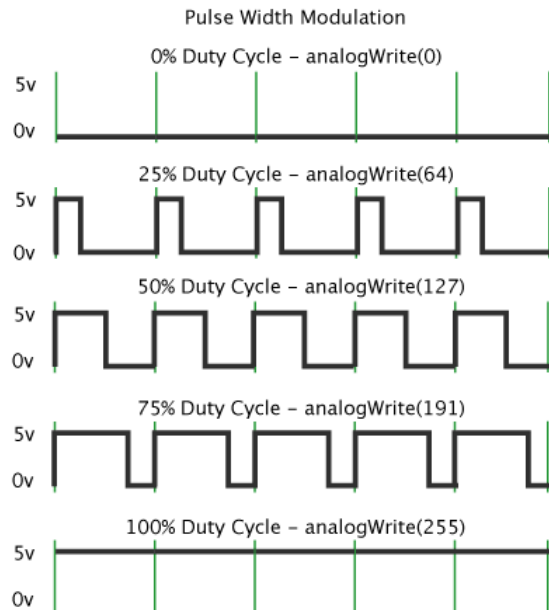


Figure 4.1: Motor velocity control with Pulse Width Modulation [7]

4.1.2 Arduino code examples

Below is the Arduino code for the `setup()` and `loop()` functions for the bioreactor program. In the `setup()` function, the three pins that control the motor are defined as output pins on the Arduino board. The last line in the code sets the data rate as 9600 bps for serial data transmission. In the `loop()` function, the `dailyProgram()` function which is shown later is called which performs the transitions of the motor running forward, backwards and in pulse movement. The `loop()` function runs repeatedly until the Arduino board is powered off.

Listing 4.1: Arduino code for the setup and loop functions

```
void setup()
{
  pinMode(motorVel, OUTPUT); // Motor velocity pin defined as output
  pinMode(motorForw, OUTPUT); // Motor forward pin defined as output
  pinMode(motorRev, OUTPUT); // Motor reverse pin defined as output

  Serial.begin(9600); // Sets the data rate in bits per second
                     // for serial data transmission
}

// Loops while the Arduino board is turned on
void loop()
{
  // Runs the daily function where the motor runs forward with 200 PWM,
  // backwards with 200 PWM and runs with pulse movement with 170 PWM
}
```

```

    dailyProgram(200, 200, 170);
}

```

The code below shows the *dailyProgram()* function which makes the motor turn forward for 30 seconds, stop for 10 seconds, run backwards for 30 seconds, stop for 10 seconds, run in pulse movement for 30 seconds and finally stop for 10 seconds. This function is called within the *loop()* function as shown above so these steps will be repeated until the Arduino board is powered off.

Listing 4.2: Arduino code for an example bioreactor program

```

// Pre:   velF, velB and velPulse are integers between 0 and 255
// Post:   The motor will turn forward with velocity velF, backwards
//         with velocity velB and in pulse with velocity velPulse
void dailyProgram(int velF, int velB, int velPulse)
{
    // The motor runs in pulse for 30 secs at velocity velPulse
    motorPulse(30000, velPulse);

    // The motor stops for 10 secs
    motorStop();
    delay(10000);

    // The motor runs forward for 30 secs at velocity velF
    motorForward(velF);
    delay(30000);

    // The motor stops for 10 secs
    motorStop();
    delay(10000);

    // The motor runs backwards for 30 secs at velocity velB
    motorBackward(velB);
    delay(30000);

    // The motor stops for 10 secs
    motorStop();
    delay(10000);
}

```

Below is the code for the *motorForward()* function which is called in the *dailyProgram()* function as shown above. Due to similarity of the functions *motorForward()*, *motorBackward()* and *motorStop()*, only the first is shown here to demonstrate the change of output voltage in the pins on the Arduino board using programming code. The function calls the *digitalWrite()* and *analogWrite()* functions which are built-in functions to change the voltage of digital and analog output pins on the Arduino board. The first argument of the *digitalWrite()* function is the digital output pin to be modified while the second argument is the corresponding new value

of the voltage. The parameter *HIGH* represents 5 volts while *LOW* represents 0 volts. Similarly, in the *analogWrite()* function, the first argument corresponds to the analog output pin and the second argument to the new voltage value. The voltage is an integer in the range 0 to 255 that corresponds to the PWM value of the output as explained in section 4.1.1.

Listing 4.3: Arduino code to run the motor forward

```
// Pre: pwm is an integer between 0 and 255
// Post: The motor runs forward with velocity pwm (in pulse width modulation)
void motorForward(int pwm)
{
    digitalWrite(motorForw, HIGH); // Gives the motor forward pin 5 volts
    digitalWrite(motorRev, LOW);   // Gives the motor backwards pin 0 volts
    analogWrite(motorVel, pwm);    // Sets the motor velocity as pwm
}
```

4.2 Graphical User Interface

Controlling the bioreactor directly via an Arduino microcontroller requires the user to manually change the code uploaded to the Arduino. To make the controlling of the bioreactor more accessible to people who might not be familiar with programming, a graphical user interface was implemented for the control unit. The user interface was designed and programmed with the Processing programming language which is a Java based open source language and easy to interact with Arduino programs. Using the graphical user interface to control the bioreactor requires the bioreactor to be connected to a computer inside the cell culture laboratory. This was not possible during the cell culture in this project so the control commands for the bioreactor had to be uploaded on the Arduino before starting the culture. When this became evident, the graphical user interface had already been programmed. Even though it could not be used at this point, it will likely be used in coming cell cultures performed in the bioreactor system.

The graphical user interface is shown in Figure 4.2 and consists of controls for the motor function, velocity and the duration of the motor running. The different functions are 'Forward', 'Reverse', 'Pulse' and 'Pause' which are selected with radio buttons while the velocity and the duration are selected with a slider. A function list was included as well, so a program of functions for the motor could be created. The interface creates a list of operational functions for the motor which are then processed by the Arduino and performed by the motor in the bioreactor. The user sets the variables of each function in the program and then pressing the button 'Add Function' adds the function with the chosen settings to the list 'Functions

to Perform’ which holds all the added functions that will be performed by the bioreactor. Figure 4.3 shows the interface when multiple functions have been added to the function list. The user can remove the most recently added function in the function list by pressing the button ‘Remove Most Recent Function’. Pressing the button ‘Clear Functions’ below the function list, removes all functions from the list. A basic program was predefined for the user interface that can be uploaded to the Arduino microcontroller with only few clicks. The button ‘Add Basic Program’ adds the basic program to the list of functions to be performed.

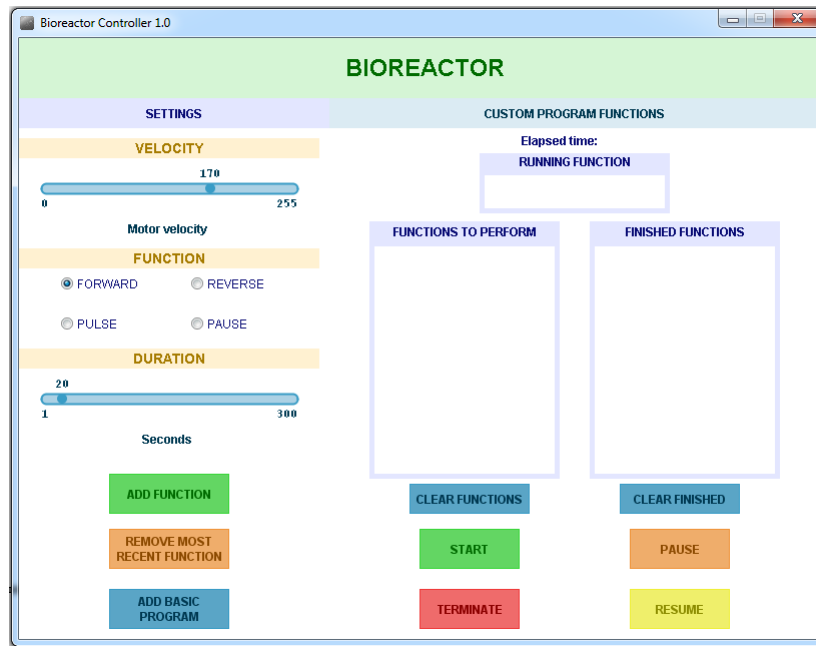


Figure 4.2: The graphical user interface

When the functions in the list are ready to be performed by the bioreactor, the user presses the ‘Start’ button. Then the first function is popped from the function list to the ‘Running Function’ box. When a function has finished running, it is moved from the ‘Running Function’ box to the ‘Finished Functions’ list and the next function is popped to the ‘Running Function’ box etc. Figure 4.4 shows the interface after some of the functions have been popped to the finished list and performed by the bioreactor. The user can press the ‘Pause’ button to pause the current function performed by the bioreactor. Then the motor is paused until the user presses the ‘Resume’ button. On resuming, the function finishes for the remaining time of its initial duration.

When all functions from the ‘Functions to Perform’ list have been performed by the bioreactor and popped to the ‘Finished Functions’ list, the function list is saved to a .txt file with the finishing date and time. Figure 4.5 shows an example of a log file created by the program. In this way, the functions performed by the bioreactor are automatically logged with the performing time. Pressing the ‘Terminate’ button

4 Programming

ends the program running in the bioreactor and saves the list of finished functions to a .txt file.

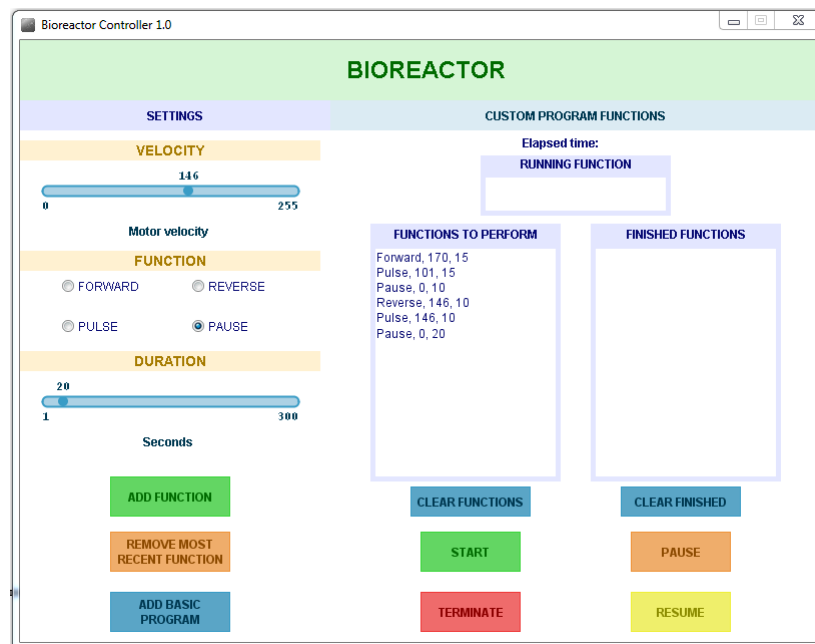


Figure 4.3: Multiple functions have been added to the 'Functions to Perform' list.

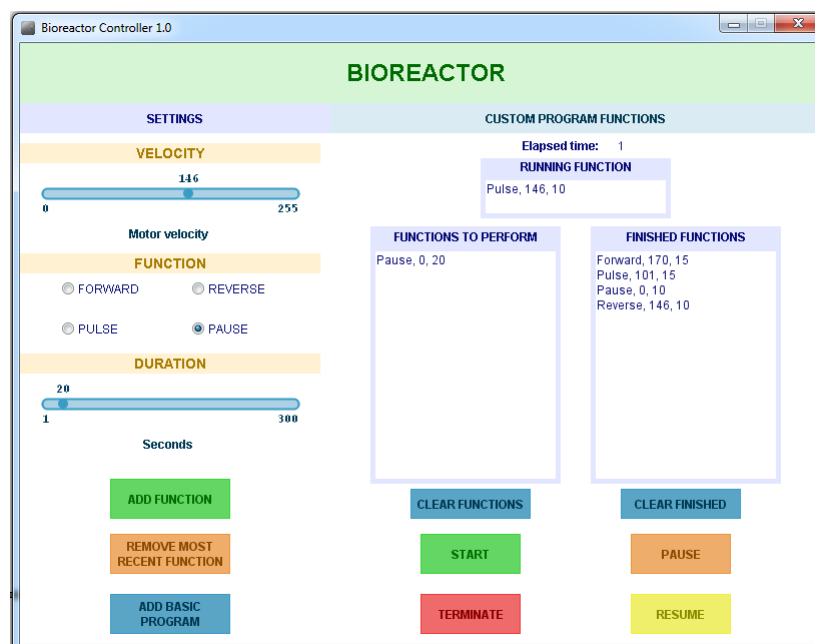


Figure 4.4: Some of the functions have been performed by the bioreactor and moved to the 'Finished Functions' list.

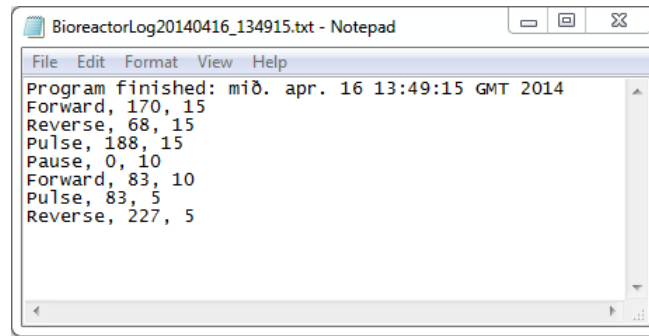


Figure 4.5: An example of a log file created by the graphical user interface program.

Below are shown code examples from the graphical user interface.

Listing 4.4: GUI code examples

```

Arduino arduino;    // The Arduino is imported as an object
int motorVel = 5;    // motor velocity pin number
int motorForw = 6;   // motor forward pin number
int motorRev = 7;    // motor reverse pin number
int svm = 170;       // motor velocity
int dur = 15;        // program duration
boolean isForward = true;

public void setup()
{
    size(400, 500, JAVA2D);
    createGUI();
    HeaderName.setFont(new Font("Arial", Font.BOLD, 24));
    motorVelocity.setFont(new Font("Arial", Font.BOLD, 14));
    motorDir.setFont(new Font("Arial", Font.BOLD, 14));
    duration_label.setFont(new Font("Arial", Font.BOLD, 14));

    // A new Arduino object is created in port 0 with 57600
    // bits per second for serial transmission
    arduino = new Arduino(this, Arduino.list()[0], 57600);

    // All the motor pins are defined as output pins
    arduino.pinMode(motorVel, Arduino.OUTPUT);
    arduino.pinMode(motorForw, Arduino.OUTPUT);
    arduino.pinMode(motorRev, Arduino.OUTPUT);
}

// Pre:  arduino is an Arduino object, motorForw, motorRev,
//        motorVel and svm are integers and isForward is a boolean
// Post: The motor now runs forward
public void goForward()
{

```

4 Programming

```
    arduino.digitalWrite(motorForw, Arduino.HIGH);
    arduino.digitalWrite(motorRev, Arduino.LOW);
    arduino.analogWrite(motorVel, svm);
    isForward = true;
}

// Pre: pulseTimer is a timer object
// Post: pulseTimer has started
public void goPulse()
{
    pulseTimer.start(dur/pulseTimer.getInterval());
}

// Pre: motorForward, motorReverse and motorPulse are buttons
// Post: the motor turns according to the button pressed
public void startMotor()
{
    if (motorForward.isSelected())
    {
        goForward();
    }
    else if (motorReverse.isSelected())
    {
        goReverse();
    }
    else if (motorPulse.isSelected())
    {
        goPulse();
    }
}
```

5 Discussion

After the cell culture in the bioreactor, it was evident that major changes need to be made on the reactor to make it work properly. For this project, the improvements from the previous version were limited as the same container had to be used. It was apparent after a month of cell culture that the bioreactor is made of too many parts which makes it easier for fungal growth to sustain in junctions between parts.

5.1 Further Development

For future improvements on this project, a new bioreactor would have to be built from scratch. The design has to prevent fungal growth as much as possible which could be achieved by having the container built in one piece, without junctions. All surfaces would have to be smooth as well and without sharp corners. The stability of the motor would have to be increased and one option is to place the motor beneath the container so the rod connecting the propeller to the motor could be omitted.

5.1.1 3D Printing

Three dimensional printing is now becoming a viable option for many applications. For the next generation of the bioreactor, 3D printing of the system could be an option to consider. With 3D printing, the whole bioreactor system could be built in one piece which would minimize the number of components. That would hopefully result in less risk of infection due to impurities getting between components.

5.1.2 Flow Meter

Currently, there is no accurate information available on the media flow through the tubes in the bioreactor. To be able to optimize the cell culture conditions, accurate data about the velocity of the media flow is essential. Then it will be possible to

grow cells with different velocity to see what effects it has on the growth. Adding a flow meter inside the bioreactor system would thus be of significant importance.

5.1.3 Data Logging

Logging the data from the performed functions on the bioreactor eases the analysis of different bioreactor settings and programs. The current data logging in the graphical user interface requires the bioreactor to be connected to a computer. To enable data logging without computer connection, additional memory might have to be considered for the Arduino microcontroller. The Arduino Uno, which is used in this project, has 32 kB flash memory which is used to store the uploaded Arduino program and is not used for data logging. The Arduino has 2 kB of SRAM memory as well as 1 kB of EEPROM memory. Typical cell culture in the bioreactor lasts for 4 weeks so if the bioreactor changes function every hour, the total number of performed functions is 672. A 700 line .txt file with the date and time information in the first line and bioreactor settings in all other lines takes 12.2 kB which is way more than the available memory on the Arduino. Data logging shields with SD card slots are available for the Arduino microcontroller which might be a viable option.

5.1.4 Smartphone/ Tablet interface

Future developments on the bioreactor system involve creating a smartphone and/or a tablet user interface to control the system. In the current version, the bioreactor has to be connected to a computer to use the graphical user interface. The bioreactor is kept inside a CO_2 incubator during the cell culture so it is not an option to use the user interface on a computer during the culture. It would though be of great value to be able to control the system with a graphical user interface during the culture. One possibility would be to program a smartphone and/or a tablet user interface that will connect wirelessly to the Arduino microcontroller to control the bioreactor system.

References

- [1] Wendt D, Marsano A, Jakob M, Heberer M, Martin I, *Oscillating perfusion of cell suspensions through three-dimensional scaffolds enhances cell seeding efficiency and uniformity*. Biotechnol Bioeng 2003; 84(2): 205-14.
- [2] Oragui, E., Nannaparaju, M., *The Role of Bioreactors in Tissue Engineering for Musculoskeletal Applications*, The Open Orthopaedics Journal; 2011, 5, (Suppl 2-M6) 267-270.
- [3] Ikada, Y., *Tissue Engineering: Fundamentals and Applications*. 2011, Elsevier Science.
- [4] Palsson, B.O., Bhatia, S.N., *Tissue Engineering*, 2004, Pearson Prentice Hall.
- [5] Czenek, A., Asmundsdottir, A.S., Magnusson, B., *Designing and Developing a Propeller Driven Perfusion Bioreactor*. 2013, Reykjavik University.
- [6] Arduino, *Arduino Uno* <http://arduino.cc/en/Main/ArduinoBoardUno>.
- [7] Hirzel, Timothy., *PWM* <http://arduino.cc/en/Tutorial/PWM>.