TVE 15 032 juni

Examensarbete 15 hp
Juni 2015

Implementation of PID control using Arduino microcontrollers for glucose measurements and micro incubator applications

Hugo Andersson
Viktor Mattsson
Aleksandar Senek

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Implementation of PID control using Arduino microcontrollers for glucose measurements and micro incubator applications

Hugo Andersson, Viktor Mattsson, Aleksandar Senek

The task is to build a low-cost thermostat and design necessary elements to perform a study on water mixed glucose-impedance at different temperatures and cell growth in a temperature-controlled incubator housing a magnetic field of up to 3 mT. The incubator was designed in solidworks and made to fit petri dishes of two relevant sizes and necessary wiring. The coils designed to extend across the large of the incubator with six turns and a 4A current to yield a sixth of the required magnetic field, as field strength increases linearly with current and turns increasing either of these is advised, and a large enough homogenous field was observed to create a suitable environment for the study. A thermistor, temperature sensitive resistance, was used to get reading and a modified wheatstone bridge was used with a multiplying op-amp to stabilize and improve accuracy of readings. Using an arduino microprocessor utilizing a PID library to calculate the power needed from thermistor readings of ambient temperature and an H-bridge controller by PWM from the Arduino a thermostat capable of driving a peltier-cell was produced capable of raising, lowering and maintaining predefined temperatures.

Handledare: Dragos Dancila, Anders Rydberg
Ämnesgranskare: Viviana Lopes
Examinator: Martin Sjödin
ISSN: 1401-5757, TVE 15 032 juni

Populärvetenskaplig sammanfattning

Genom att använda en billig mikroprocessor som Arduino, en liten dator med ett inbyggt minne så att programmet körs konstant, och simulation- och modellering-program har vi skapat grunderna för att bygga spolar som ska skapa ett homogent magnetfält, en tålig och lufttät inkubator samt en termostat som kan både höja och sänka temperaturen efter behov. Arduino är en av många mikroprocessorer som finns att köpa men är en av dem billigaste och mest använda idag, framförallt på grund av sitt breda användningsområde. Med hjälp av den styr vi en analog koppling (H-brygga) som på kommando från Arduinon skiftar håll på strömmen, och levererar upp till 6A till en termoelektrisk-cell som agerar värmeaggregat. Eftersom riktningen på strömmen kan skifta så både kyler och värmer den vilket kan hålla en önskad temperatur stabilt med hjälp av en algoritm så kallad PID som beräknar nödvändig utgående strömstyrka. Tillsammans med en känslig spänningsförstärkare kan man mäta små förändringar i en termistor, en resistans som reagerar på temperaturförändringar, och avgöra vad temperaturen är med en osäkerhet mindre än $0.5\,^{\circ}$C. För att genomföra undersökningen designades två parallella spolar (Helmholtz-spolar) med hjälp av COMSOL, ett beräkningskraftigt fysik-simuleringsprogram, och beräknade nödvändig strömstyrka som krävdes i spolarna samt dimensioner för att uppnå ett konstant magnetfält på 3 mT. En lufttät modell gjordes i solidworks, ett modelleringsprogram, som skulle få plats med termoelektriska cellen, spolarna och nödvändiga kopplingar. Tillsammans skapar enheterna en stabil inkubator med ett homogent magnetfält som klarar temperaturer mellan $0\,^{\circ}$C - $60\,^{\circ}$C och separerat har man en temperaturreglerare kapabel att leverera en hög effekt och stabil omgivning.

Contents

1 Introduction

The project herein described was presented under bachelor thesis work done in Fasta Tillståndets Elektronik at Ångströmslaboratoriet. Our research was divided in two main sub-projects.

In the first project, we develop an innovative new method based on RF dielectric spectroscopy for glucose monitoring in collaboration with Ascilion AB. In this context, we need to implement temperature control for performing the measurements.

In the second project, a microincubator was developed to use in cell culture studies, consisting of a petridish to analyse cells which are illuminated by magnetic field. Results from earlier studies (reference) show that exposure of cells by magnetic fields can increase their proliferation, and therefore temperature regulation is also need here.

Overall, the aim of our work was to implement PID control using Arduino microcontrollers for both glucose measurements and microincubator applications.

1.1 Glucose measurements

As glucose affects mood and awareness it has become an important part in not just physical health but also mental. And as more raw sugar is consumed in the modern world diabetes has become a growing problem with an estimate of 8 % of the world's population suffering from complications related to blood glucose levels[1]. Almost every product up to 2015 has required perforating the skin in order to draw samples of blood to get stable readings of glucose levels, which take time and fidgeting[2]. With recent advances an attempt is made from different angles to reduce the complexity of performing a measurement without the need for any blood samples and one such is with high-frequency signals (RF signals). To calibrate the sensor correctly mimicking the body's temperature and its variation during the night and day for accuracy is important

1.2 Micro- incubator

Incubators are an age-old matter and are known to date back to Egyptians that used artificial heat in order to stimulate hatching of eggs without the help of hens, today incubators are more complex often with thermoelectric cells that are capable of adding and removing heat depending on the direction of the current[3]. The need of versatility of one unit has increased and commercial incubators often come in standards shapes and sizes as they have a broad range in both temperature and stability but also gas delivery, which can ensure a proper environment regardless of subject. The price has instead become a limiting factor with variety of available incubators on the market being cumbersome alternatives for smaller projects where only a few thousands of cells need to be investigated. Instead micro-incubators, where single petri-dishes and plates can be stored in a controlled environment, become important for small-scale experiments with a shorter timespan and therefore a low cost budget.

1.3 Cell growth in magnetic field

There have been studies indicating that human cells may grow faster while exposed to a magnetic field (B-field) and if this is the case a person with a broken leg could, for example, return to work and a normal routine faster if placed in a B-field[4]. One way of creating a homogenous magnetic field is by using two identical coils placed one radius apart and driving a constant current in the same direction in both, a setup called Helmholtz-coils. To test the validity of the hypothesis control measurements in steady environments, control samples will be placed in the incubator without a current and left for a period of time and then compared to cells left in the same temperature and time period but with a current through the coils in the incubator.

1.4 Arduino microcontroller

Arduino is a microcontroller board with open-source software and hardware. There are several different Arduino boards on the market. However, these different Arduino boards share a common feature, easiness to develop programs for them. With the Arduino you can make robots, thermostats and other electronic applications. For this project the Arduino is the perfect hardware since it can read the voltage across a thermistor and then incorporating PID-algorithm it can calculate how high is the PWM signal for regulate maximum current through the peltier element. The Arduino MEGA 2560 is shown in Fig. 1. The PWM pins are the pins on the top row and the analog pins are the pins on the bottom row.

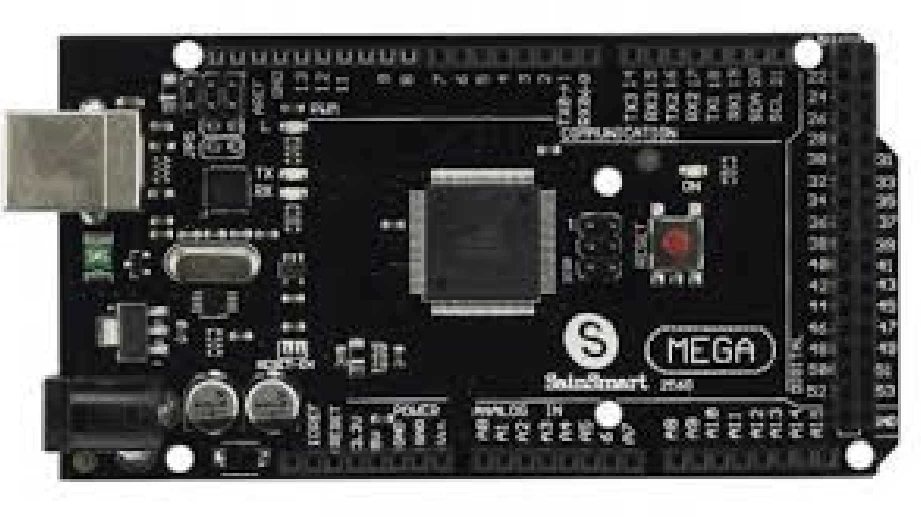


Figure 1: The Arduino MEGA 2560 board with nothing connected to it.

1.5 Arduino IDE

IIn this project, the Arduino integrated development environment (IDE) is used to write programs for the Arduino. The IDE is derived from open-source programs like Processing and it is written in Java. In the Java environment there is a text editor where users can write their code. For users who want to start programming this environment is perfect because it is easy to use, only it needs two functions to make an executive program, the first is *setup()* where you initialize variables and is only running once when you start the program and the second function is *loop()* that executes its contents until the Arduino board is disconnected. The Arduino is connected to the computer with USB and through the USB the sketches are uploaded to the Arduino. The code is written in C or C++. Standard constructs supported by a C/C++ compiler is working on Arduino.

The IDE also contains a compiler that is used before uploading the code to the Arduino. If there is something wrong in the code, an error message will occur with an explanation of the error. With serial communication through the USB it is possible to send commands when the program is running. There is a window called “Seriell monitor” where the user can display results and send commands to the Arduino. While writing program there is an opportunity to include open-source libraries from third-party developers and there are several libraries available on the Arduino website to download.

1.6 Processing

The Processing development environment (PDE) is an environment to write processing sketches. As mentioned in section 1.2.2 the Arduino IDE is based on the PDE, and therefore the PDE and Arduino IDE are similar. But in the PDE the sketches are written in other programming modes like Java or JavaScript. The language is Java but with simplified syntax. In this project the Java mode is used. The main reason to choose "Processing" in this project is the possibility to display the results graphically. In Java mode you write programs that display your results in a window as a graph for example. The graphical window makes it possible to draw a graph based on the results from the Arduino IDE. The Arduino IDE and PDE are communicating with serial ports.[5]

1.7 Labview

We chose to use Labview for programming the communication between the computer that reads the temperature, using the Arduino, and the VNA. Unlike other programming tools, for example Matlab, which are text-based, Labview is a graphical programming tool, where users instead of writing long segments of code, they can connect different sub VI’s to each other.

Labview consists of two parts, the front panel and the block diagram. It is in the block diagram where the programming takes place by placing sub-VI's, called blocks. The programming is done by wiring different blocks to each other, and it also possible to connect also three different controls to the ports on the block. The different controls are: constant, control and indicator. For example, if users want to display a value at one of the blocks, an indicator should be chosen. A segment of our block diagram from the program we made can be seen in Fig. 2.

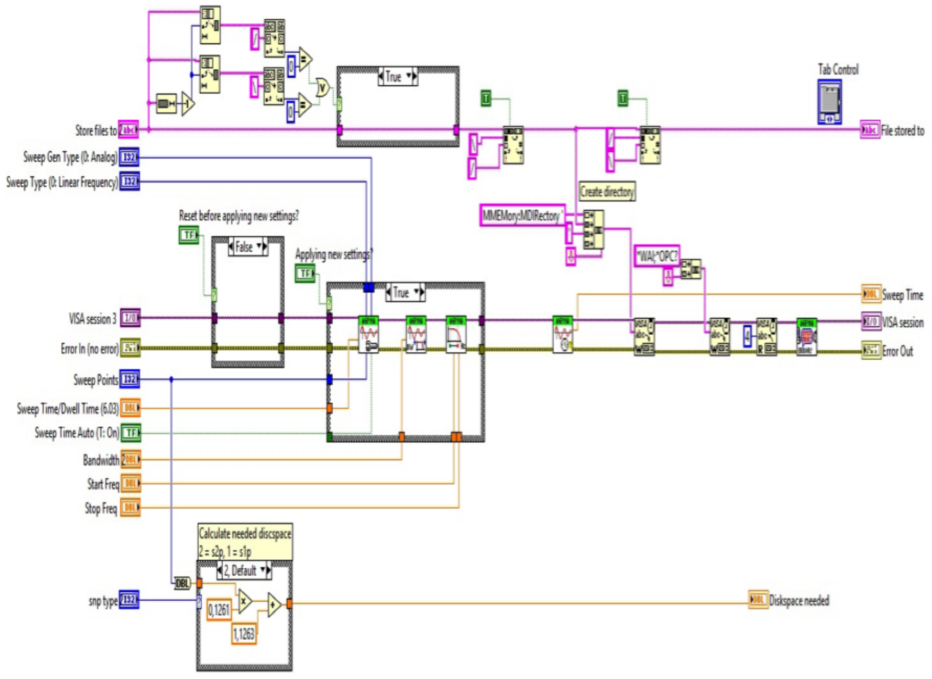


Figure 2: Part of our block diagram, this part creates the directory where we will save the measurement and sets the settings for the VNA.

The second part of Labview is the front panel. It is in this window that users execute the program, the controls and indicators, which is placed in the block diagram shown in the front panel. The control, as the name indicates, control the value of a variable at one point and the indicator shows the value at a certain point. The different controls can be customized, we use both a graph and a simple text box to view the temperature over time as well as the current temperature.

1.8 SCPI

SCPI stands for Standard Commands for Programmable Instruments and is, as the name states, a standard form of commands for communication between devices. It is via these commands our Labview VI communicates with the VNA. The SCPI were from the beginning created to work for GPIB the standard can now be used for various instruments such as RS-232, Ethernet and USB to name a few.[6]

The commands are often made up of multiple keywords where the first is the name of the larger group and the following is the name of a smaller subgroup and the last always specifies the name of the function. An example of an SCPI command we use is MMEMory:MDIRectory. MMEMory is the name of the branch which contains all the memory related function whereas MDIRectroy is the command for creating a directory, it must be followed with an exact location where to create the directory. Important thing to note is that the lower case characters can actually be left out and we would only use the capital characters.

There is documentation of all the SCPI commands available online which specify all commands needed and examples are also given.[7]

1.9 COMSOL

COMSOL is a multiphysics tool which lets the user decide which parts of the physical spectra to be evaluated, such as magnetic fields and currents or tensions and heat flow, and create accurate designs by either importing a Solidworks model or creating one in a built-in modeller. In the GUI choices such as physical constants connected to permeability and elasticity can be manually altered but also chosen from a large set of known material properties such as annealed copper, perfect electrical conductor and air. Boundary conditions are set, such as the simulation domain, which in our case is rather small in the range of 10-100 mm, and physical laws to be applied such as Maxwell's Equations and Ampere's law. The program uses mesh grids of suiting shape e.g. triangles or hexagonals to render a model fitting for estimation by solvers in the style of finite element methods by using patented solvers that process the data until convergence is reached.

1.10 Temperature measurement

There are several ways of measuring temperature like infrared detectors or old fashioned thermometers that take advantage of expanding liquids such as mercury; however the easiest way to translate a change in temperature to an electronic circuit is to "speak" in terms that an analog reference can quantify. This can be objectified by placing a thermistor, an electrical component that changes resistance in correlation with temperature, into a closed circuit and then mapping the change in resistance by measuring the voltage across it.

The three most common thermistors are NTC (Negative Temperature Coefficient), PTC (Positive Temperature Coefficient) and RTD (Resistance Temperature Detector). PTC has a logarithmic curve as typical resistance curve, which means as temperature increases so does the resistance. NTC has a similar setup however based on a negative temperature curve and RTD's are most commonly linear positive in a large portion of the temperature span. The biggest advantage of having a RTD is that they are commonly more accurate than the NTC and PTC but with the downside of being affected by the current that runs through them causing self-heating.

In order to keep the temperature within The peltier cell is a small element that uses n- and p-doped semiconductors to transfer heat, energy, from one side to the other, resulting in one hot and one cold side. The area of the peltier cell is around 16 cm2 and it can drive a 6 Ampere current in order to heat or cool a sample to minus degrees up to more than 100 degrees., there was a need for a RTD-thermistor, which had a self-heating error less than 0.5 °C per 1 mW, using a 5V V_{cc} the series resistance would have to be at least 1500 Ω.

1.11 Peltier control

The peltier cell is a small element that uses n- and p-doped semiconductors to transfer heat, energy, from one side to the other, resulting in one hot and one cold side. The area of the peltier cell is around 16 cm^2 and it can drive a 6 Ampere current in order to heat or cool a sample to minus degrees Celsius up to more than 100 °C. An h-bridge can drive a current in two directions. In order to use the peltier cell, the peltier cell has to be connected to the h-bridge. The peltier cell will heat if the current goes in one direction and cool if the current goes in the other direction. To control if the peltier cell will heat or cool, a pulse-width modulation (PWM) signal will be send from the Arduino. The PWM signal gets a value between 0 and 255 (8 bit) from the PID-controller. If the measured temperature is far away from the setpoint temperature, the PWM will get the value 255 in order to drive as much current as possible through the h-bridge and the peltier cell in order to get closer to the setpoint. The value between 0 and 255 means how much time of the duty cycle the current will be running. 255 requests a 100 % duty cycle, and 127 requests a 50 % duty cycle (current is running 50 % of the time). The PID-controller is used to regulate the temperature to the setpoint temperature. The P is the proportional term and depends on present error, I is the integral term and depends on accumulation of previous errors and D is the derivative term and is a prediction of future errors[8]. In the Arduino IDE there is a library containing a PID-controller. The only thing to do is to give the design parameters good values and place the method *Compute()* in the code where you want to execute the PID algorithm. Then *Compute()* returns an output value to the PWM signal.

1.12 Voltage divider

Using the Arduino's built-in voltage pin of 5V the initial model for the transistor was a voltage divider with a resistance in series with the thermistor, which can be seen Fig.3 as the resistance R1 and Rt with Vcc across, the relation of voltage in between follows equation 1, with the voltage in between becoming the signal to be scaled and then amplified by A1. For optimization purposes we consider the average temperature, the average operational temperature for which the incubator will be used, 37 °C knowing that the charge across the thermistor to be max 0.339V at 115 Ω to fulfil the requirements to avoid self-heating. Using thissetup, we determine the series resistance with equation 1 to fit this voltage and picking from the E12-series using a 1600 Ω resistance. The setup now produces approximately 0.93 mW reaching the condition of power for a stable resistance.

$$V_2 = V_{cc}\frac{R_t}{R_t+R_1} \tag{1}$$

The use of voltage followers, meaning a voltage to be replicated on the positive port and a short-circuit between negative input and output, are a necessity because if the pin on the positive input of A1 would be connected to R5 there would be a parallel connection of Rt and the sum of R5 and R6 changing the current running through the thermistor decreasing our voltage. An op-amp virtually has infinite input impedance and low output impedance meaning that no current will go through the positive input of A1 and change the results, the same goes for A2 to some extent.

1.13 Amplifier circuit

The voltage amplifier takes the difference between V_1 and V_2, that can be seen in Fig. 3, and multiplies it with a coefficient depending on resistances R5 and R6, this gives us equation 2.

$$V_a = \frac{R_6}{R_5}(V_1 - V_2) \tag{2}$$

The output from A_3 will be read using an analog pin in the Arduino, a 1.1 V reference helps turning a analog signal into a digital value between 0 and 1023 with 1023 being 1.1 V, from which we later can deduce the change in resistance. Maximum possible multiplication will be known once we know the largest voltage we can subtract in the multiplier without receiving a negative output by the op-amp possibly destroying our Arduino, this is calculated by realizing the smallest voltage possible from A1 is at the lowest temperature to be measured. If we want to use the linear part of the thermistors range this means that the lowest temperature will be 0 °C and lowest resistance 100 Ω; using equation 1 lowest voltage is 0.294 V.

To use the extent of our 1.1V span, we need a 0.294 V from the second voltage support (A_2), a second voltage divider connected to the same feeding pin as the thermistor (5V) is connected to the positive input of A_2 by a voltage divider with the necessary quote using equation 2. This improvement will provide a constant charge and will be connected to the negative input of A_3.

The difference between this and the voltage across the thermistor is obtained by multiplied with a constant and taking the thermistor's resistance at the highest temperature - yielding the greatest difference at the input of A_3 - and see that this must not exceed 1.1V resulting in

$$\frac{R_6}{R_5}(V_{1max} - 0.294) = 1.1. \tag{3}$$

With V1maxoccurring at maximum temperature, set to be approximately 79 °C, which corresponds to a R_t of 130 Ω and a voltage across of 0.375 V. Inserting this into equation 3 gives a maximum amplification factor R_6/R_5 no greater than 13.46. If the final circuit amplification >13.46 then this value needs to be considered.

As R_5 and R_6 grow in magnitude, the noise in the diagram will decrease as currents decrease, but as a trade-off that displacement voltages on ports of A_3 become of greater significance, a suitable range for R_5 and R_6 are 10-100 kΩ.

The resolution with which we can measure is calculated by taking the change in temperature per bit; with a span $0 - 79$ °C divided across 10 bits this gives a theoretical resolution of 0.08 °C, however this is limited by the stability of the thermistor which ensured an accuracy of less than 0.5 °C.

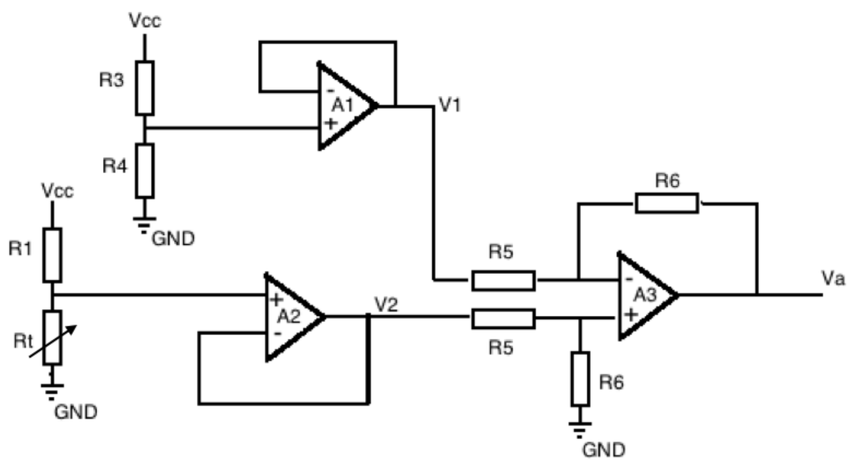


Figure 3: Diagram of the thermistor circuit, V_a is the output from the circuit, V_{cc} is a 5V feeding charge. The op-amps have a +5V feeding voltage not shown in the diagram, the rectangles are resistors of different ohms, same name denotes same resistance. V_1 is the scaling voltage, to set the voltage at 0 °C to 0V, V_2 is the voltage across the thermistor.

1.14 Incubator

It is essential that cell culture experiments are done under carefully controlled conditions and it is important that the cells should be kept stable so the results can be reproduced by others. Physical factors such as temperature and CO_2 variations, as well as mechanical vibration, can contribute to experimental artefacts [9]. In order to get that, a microincubator was developed and we focus in the need to be able to regulate the temperature and also control the airflow.

Ideally, we would like to have had a pair of Helmholtz coils as well in order to get an even magnetic field in the incubator, but unfortunately there was not possible to achieve it. The possibility to lower and raise the plate with the samples inside the incubator was considered however it would be very hard to implement it and it would just make the design unnecessary complex which was not that important for our study.

Concerning the air feature, we want to be able to regulate the airflow and have carbon dioxide in the incubator to create a natural environment for the cells cultured in laboratory. Important thing about the airflow is that it cannot have to high airflow because can alter the temperature in the incubator. The most interesting temperatures to study would be around 20 °C (room temperature) and 37 °C (body temperature).

2 Methods

2.1 Programming

Two similar programs were developed for the Arduino in order to regulate a temperature from a heat element to a desired temperature. The first program was adapted to Processing in order to display the temperature as a graph. The second program was adapted to Labview in order to do measurements with the VNA and also display the temperature in a graph.

2.2 Thermistor calibration

The thermocouples had to be calibrated before measurements. The setup contained a NTC thermistor, a thermometer with 0.1 °C resolution, a boiler, a cup and a digital multimeter or the Arduino. The setup is shown in the Fig.4. The thermistor and the thermometer were placed in the water filled cup. It was important that sensors of the thermistor and thermometer were close to each other in order to have a calibration as good as possible. Then the thermistor was connected to the digital multimeter. Then, we use boiler heater for having the water from 20 °C to 55 °C and at every degree Celsius the resistance displayed on the multimeter was noticed. Using Steinhart-Hart equation it is possible to convert the measured resistance as a temperature. The equation is suitable for NTC and PTC thermistors and the temperature in Kelvin is given by the inverse of

$$\frac{1}{T} = A + Bln(R) + Cln(R)^3 \tag{4}$$

where A, B and C are the coefficients to calculate and R is the resistance at that temperature. The method least squares was used with the data in order to calculate the coefficients. The calculations were made in Matlab and the equation with the calculated coefficients was plotted with the data to compare Steinhart-Hart with the measured values. To test, whether the Steinhart-Hart is good or not the process of heating the water, and notice the resistances were repeated for a few temperatures. The temperatures calculated from Steinhart-hart were compared with the temperatures measured with the thermometer.

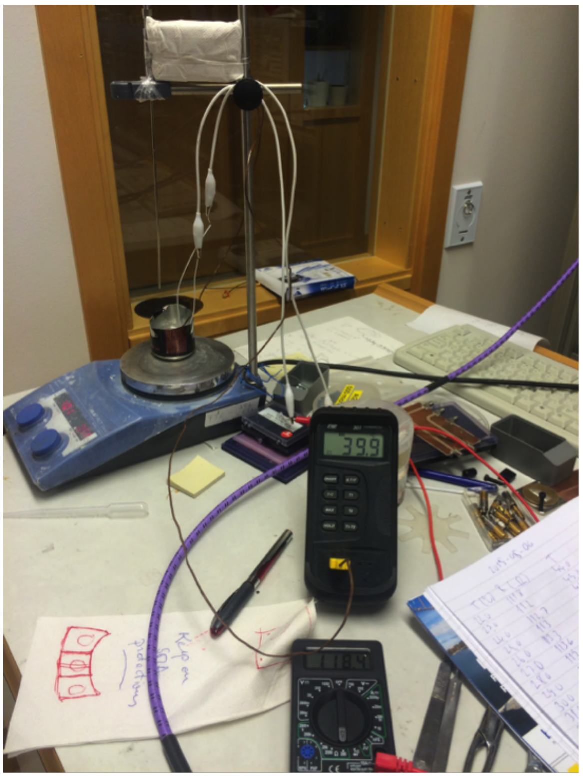


Figure 4: Setup of calibration with the cup placed on the blue boiler. The thermistor and thermometer are placed inside the cup. In the middle of the figure we se the thermometer and multimeter display temperature respectively resistance.

The used digital multimeter in the calibration was unstable. It was hard to notice correct value since the resistance varied a lot when the temperature was stable. The method to calibrate was repeated from the start, but with the Arduino instead of the multimeter. The Arduino does the same thing as the multimeter; it measures the resistance and displays it on the computer.

2.3 PCB test setup

The Helmholtz coils that will generate the magnetic field in the incubator will be placed under the incubator and on top of the incubator. The coils will be etched out to a PCB substrate. The PCB substrate is a plate with a layer of thin copper on both top and bottom. The material between the copper layers is plastic. A test was made in order to see if it is possible to heat an incubator with PCB substrate between the peltier cell and the incubator. The material of the incubator is aluminium, which is a good thermal conductor. The test contained three layers of PCB substrate, one peltier cell, one thermistor, one petri dish and one heat sink. The setup is shown in Fig. 5 where the thermistor is placed between PCB substrate and the petri dish. The peltier cell is not visible but it is placed under the PCB substrate.

The test was made without the incubator. Instead, the temperature is measured on the bottom of and inside the petri dish. As mentioned above, aluminium is a good thermal conductor and if the test works it should work to heat the incubator placed on PCB substrate.

The program written in Arduino IDE was used to drive the peltier cell and print the result. The setpoint for the test was 35 °C . Two runs were made, one with the peltier cell placed as shown in figure 4 and another with the peltier cell inside the petri dish filled with water.

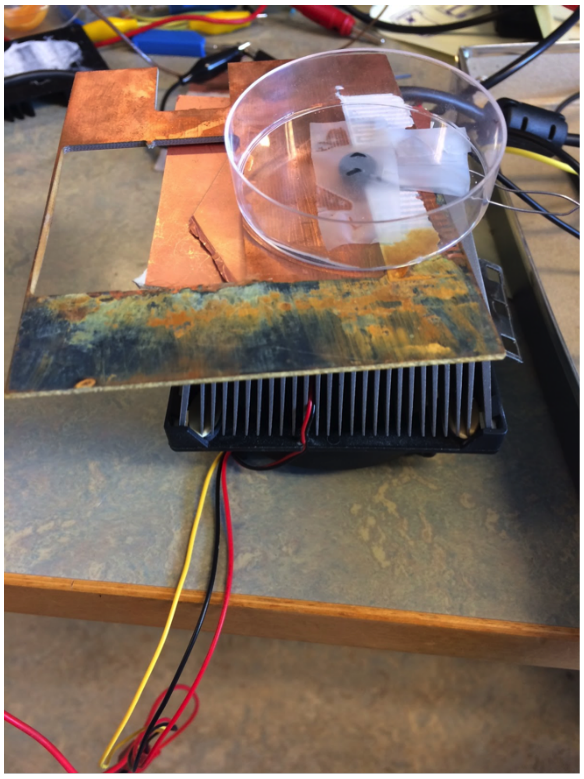


Figure 5: The setup of the test, where the heat sink is in the bottom and on the top of it the peltier cell is placed (not visible). The three layers of PCB substrate (Copper color) are placed on top of the peltier cell. The thermistor is the black dot between the PCB substrate and the petri dish (plastic bowl).

2.4 Design of coils

To create a homogenous magnetic field in the incubator, coils that fulfil requirements regarding dimension and power need to be built. However, instead of building a coil from costly materials, there are ways to create analogous situations of our planar coils using COMSOL that create a stable environment where currents in metal and the resulting B-field can be studied.

There are two main limits on the design. First,for the coils to become integrated in the incubator they need to be approximately within range of the containers dimension. Having petri dishes ranging in height from 20 mm to 40 mm we have to ensure that independent of designed coils, the plates meet the requirement at maximum distance from each other.

Second, a B-field up to 3mT is required. To create a fulfilling analogous situation in COMSOL we designed two coils, spaced by 4 cm apart with specific dimensions (inner radius of 23,5 mm, outer radius of 31,5 mm with a thickness of 30 mm). Inside these domain coils with six turns and a current of 4A were simulated using the “multi-turn coils” module in COMSOL 4.3 and the magnetic field (B-field [T]) was plotted.

3 Results

The Arduino, h-bridge and the amplifier circuit were mounted in a box as we can see in Fig. 6. The Peltier element and the thermistor will be connected to the box through the white terminal blocks. The thermistor will be also connected to the two left blocks, the peltier element in the two blocks in the middle. The second block from the right will be the positive voltage supply for the h-bridge and the next block will be connected to ground on the power supply.

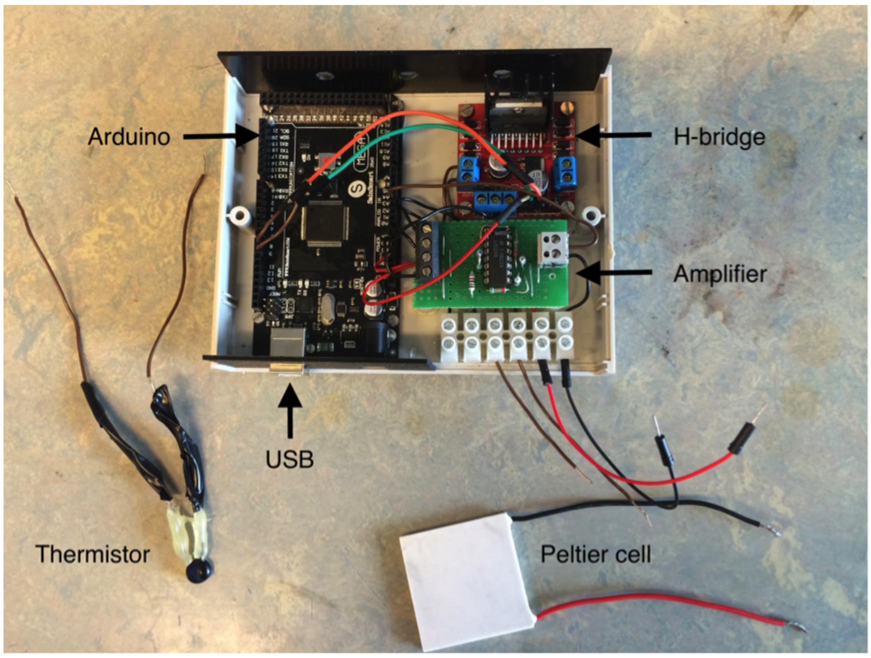


Figure 6: The box containing the Arduino h-bridge and amplifier circuit. The thermistor and the peltier element will be connected to the white terminal blocks in the middle of the figure.

3.1 NTC calibration

The Steinhart coefficients for the NTC thermistor are needed to make measurements with the thermistor. With the data containing resistances and their corresponding temperatures, the coefficients could be calculated. Least squares was the method applied to get the coefficients from the following equation system

$$\begin{bmatrix} 1 & ln(R_1) & ln(R_1)^3 \\ 1 & ln(R_2) & ln(R_2)^3 \\ \vdots & \vdots & \vdots \\ 1 & ln(R_n) & ln(R_n)^3 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \frac{1}{T_1} \\ \frac{1}{T_2} \\ \vdots \\ \frac{1}{T_n} \end{bmatrix}$$

where R1 to Rn are the resistances at corresponding temperatures T_1 to T_n. From the solved equation system the coefficients are:

$$A = 0.546066705965131 \cdot 10^{-3}$$

$$B = 0.349002469448578 \cdot 10^{-3}$$

$$C = -0.000290658290714 \cdot 10^{-3}$$

The resistances were then plotted against their corresponding temperatures. In Fig. 7 the red dots are the data from the calibration. The blue line is the fitted curve with the least square method.

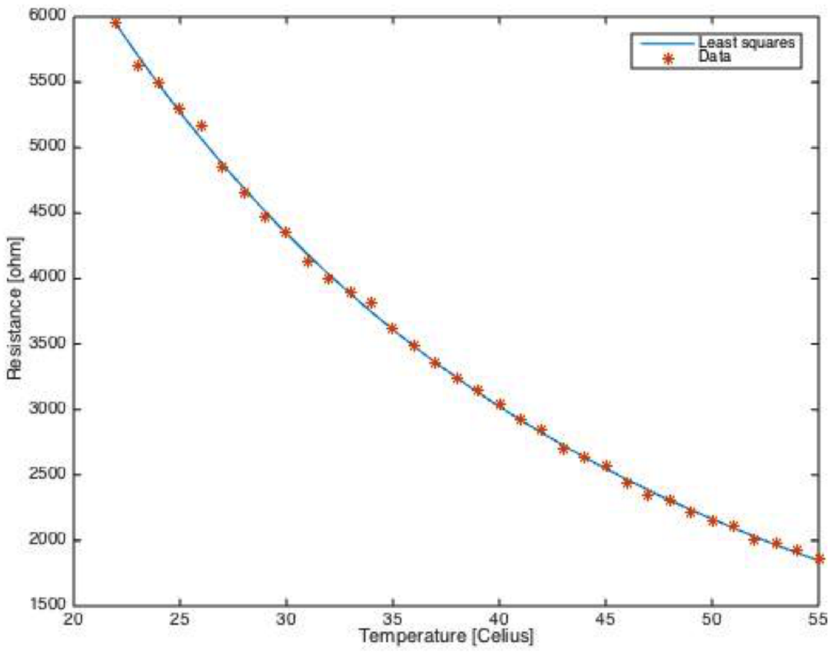


Figure 7: : Showing a least square fit adaptation to the data points from calibration, blue line is fitted line, and points are the data.

3.2 PCB test

The two test runs succeeded, meaning that the measured temperatures of the thermistor were the same as setpoint. The run with water inside the petri dish took about 50 minutes to reach the setpoint of 35 °C. The run with the thermistor under the petri dish was much faster, around 10 minutes to reach 35 °C. As shown in Fig. 8, the temperature does not increase during this part of the run. The temperature was not changed for several minutes during some parts of the run and that is why the red line is not increasing in Fig. 8.

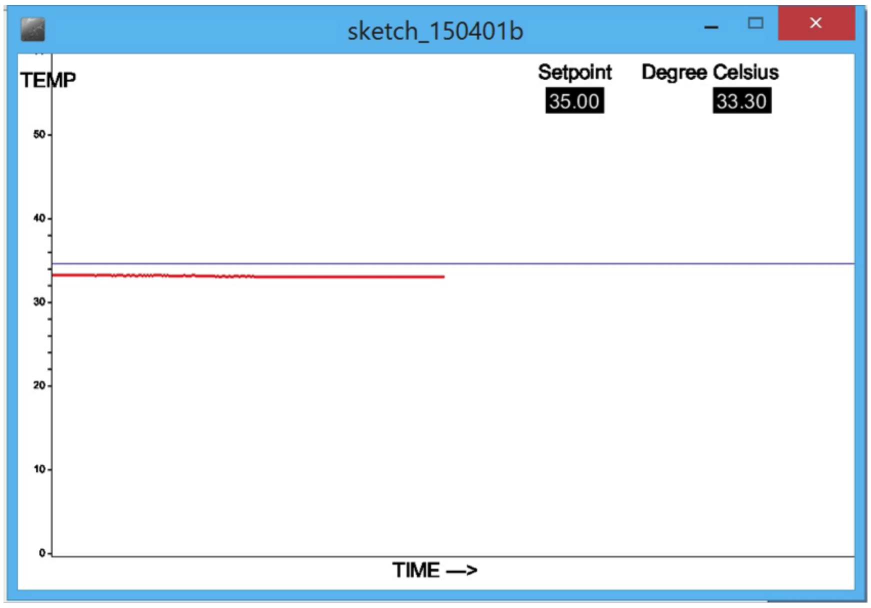


Figure 8: Screenshot of the temperature graph during a test run. The blue line is the setpoint temperature and the red line is the measured temperature. The y-axis shows the temperature in Celsius and x-axis is time without unit. The graph updates when the red line reaches the right side of the window.

3.3 Helmholtz coils

As seen in figure 8 and table 1 the field along a line close to the peripheral of the coils is homogenous and strong enough for scaling up to create the necessary magnetic field, being one sixth of the required 3 mT-field. In a line on the x-axis, the magnetic field varies greater with a variation of 0.1 mT which poses problems in the design of planar coils, minimizing the available volume with a homogenous magnetic field.

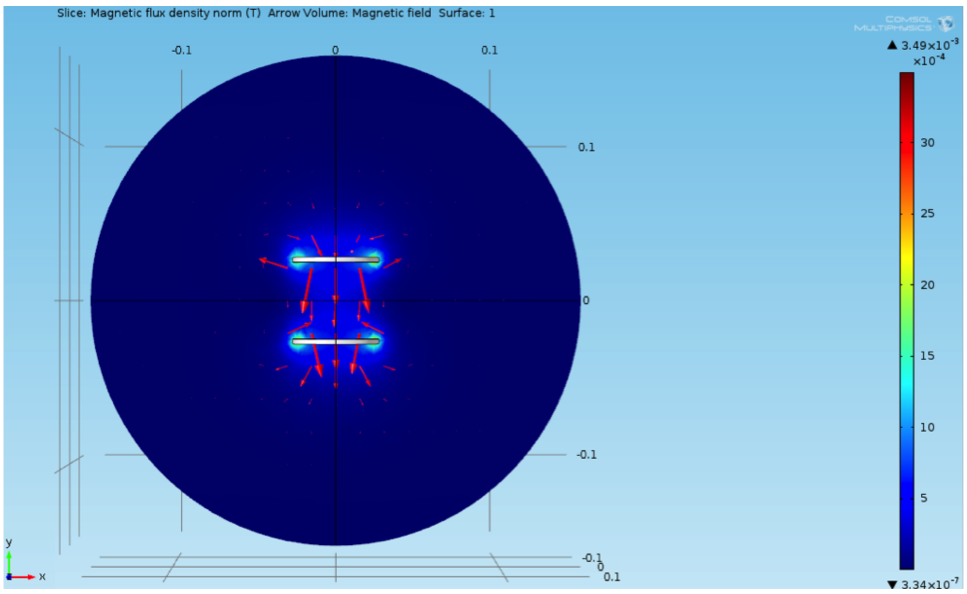


Figure 9: Figure showing our simulated results in COMSOL with the colorbar of the magnetic field on the right hand side, the scale on the bar is in Tesla, T. The blue circular plane is perpendicular to the z-axis and intersects (x,y,z)=(0,0,0). The arrows point in the direction of the magnetic field in that plane.

Table 1: The table shows the magnetic field in chosen points from Fig. 9.

x-axis (mm)	y-axis (mm)	B-field (mT)
16	0	0.30
0	0	0.41
-18	0	0.27
15	13	0.55
0	13	0.52
-17	13	0.49
16	-15	0.54
0	-15	0.53
-17	-15	0.54

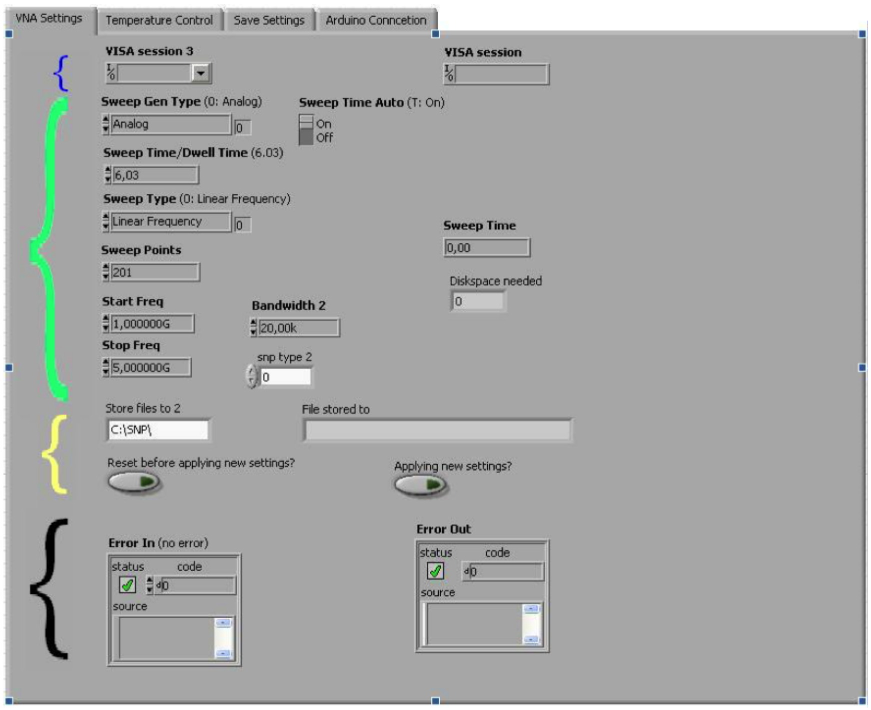
3.4 VNA VI

The VI we programmed to work with the VNA is made up out of four different tabs. The first part is setting the correct settings for the VNA and choosing, if necessary, create a directory to store our results. In Fig. 10, the settings tab are described, in the blue bracket the user choose the correct VISA session out of a list of available devices.

In the green part, user can choose the settings to be used by the VNA, usually it is only need to change the start and stop frequency.

In the yellow bracket, we create the directory as well as an option for reset traces, and any predefined settings from the VNA by ticking the button “Reset before applying new settings?” and making sure that it is on before starting the program. Moreover, if changes are need to be made in any parameter, then button “Applying new settings?”-button should also be on.

A “Error out” text may appear next to the black bracket displaying an error, reading something like “error in AGPNA(...) Read”. This is a mistake in the AGPA-protocol and does not affect the results.



VISA session 3

VISA session

Sweep Gen Type (0: Analog): Analog 0

Sweep Time Auto (T: On): On / Off

Sweep Time/Dwell Time (6.03): 6,03

Sweep Type (0: Linear Frequency): Linear Frequency 0

Sweep Time: 0,00

Sweep Points: 201

Diskspace needed: 0

Start Freq: 1,000000G

Bandwidth 2: 20,00k

Stop Freq: 5,000000G

snp type 2: 0

Store files to 2: C:\SNP\

File stored to

Reset before applying new settings?

Applying new settings?

Error In (no error): status ✓ code d0 source

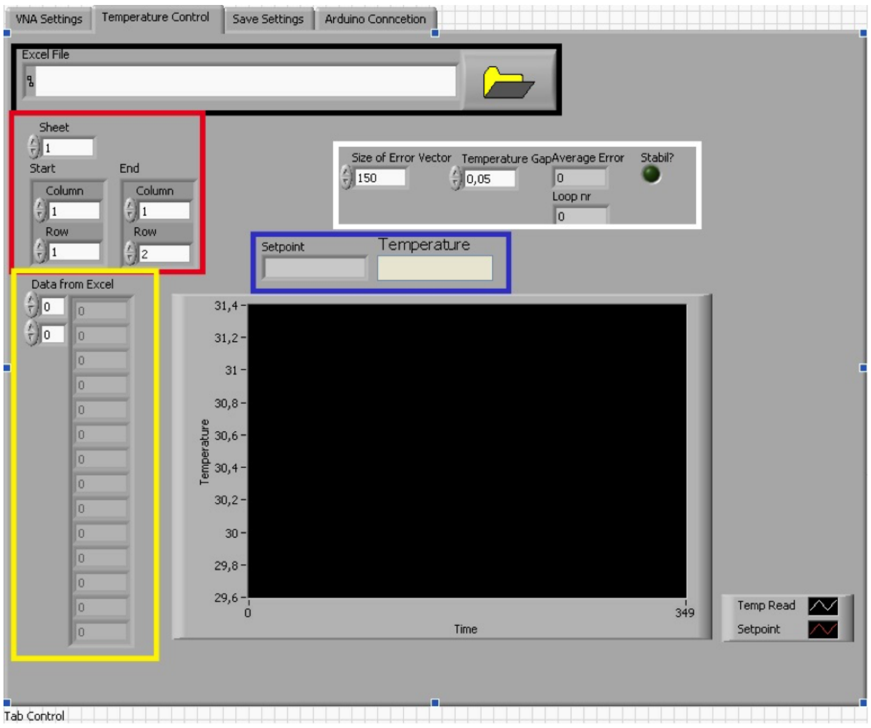
Error Out: status ✓ code d0 source

Figure 10: Settings tab color coded for easy explanation.

The second tab of our VI is called Temperature Control, this is where pick our temperatures to measure, using Excel and keeps track of temperatures values; it also be used to select the conditions for when to do a measurement. The black and red box in Fig. 11 shows where the user can choose the Excel file and specify the rows and columns. In addition, the yellow box displays the values read from

Excel so the user can confirm the values data are correct. The blue box displays the current setpoint and temperature. The graph displays the temperature and setpoint over the last 350 measurements.

The white box determines when the VNA will do and save the measurement. In the temperature gap box, the user put in how far from your setpoint you can be to find it acceptable, i.e. standard deviation. For example a setpoint of 27 °C and a temperature gap of 0,05 means that the temperature is acceptable if it is between 26,95 and 27,05. The VNA will do the measurement when the average error is below the temperature gap. The program works by saving the number of values set in the "Size of Error Vector" box and for each value we subtract the setpoint taking the absolute value of that and then the mean value of that vector, which we call "Average Error". The "Loop nr" box displays the amount of times the program have measured the temperature, it resets for each setpoint. The "Stabil?" button will light up when the conditions for doing a measurement meet the criteria and turnoff automatically once the data has been saved.



Excel File

Sheet 1

Start Column 1 Row 1

End Column 1 Row 2

Data from Excel 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0

Size of Error Vector 150

Temperature Gap 0,05

Average Error 0

Loop nr 0

Figure 11: Temperature Control color coded.

The last two tabs in our VI are called Save Settings and Arduino Connection and both are pretty basic. In save settings, you have to choose the VNA out of the list of devices and making sure the "snp type" and "Store files to" are correct.

In the Arduino Connection, you just have to choose the Arduino from the list of devices. However there is a bug that sometimes causes the Arduino to not show up. If so, refresh the list and if that did not work, close Labview and reconnect the Arduino and repeat the procedure. We believe this happened because there was a conflict between versions of Labview; for running VI/VNA we used Labview version 2009 while for programming it we used instead Labview 2014 version.

3.5 Incubator

Initially, we had a lot of different ideas of how the incubator would be designed. Figure 11 shows the first complete design, not made in Solidworks but in similar simpler free software called Sketchup.

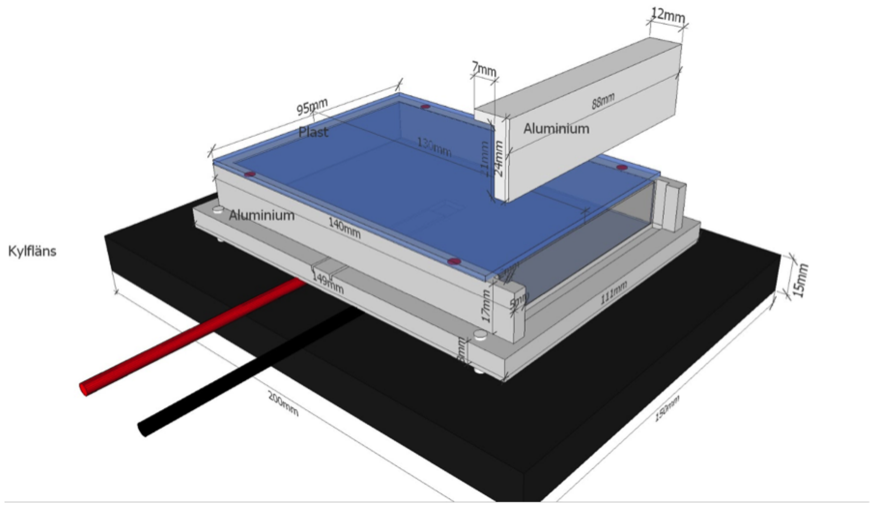


Figure 12: The first design of the micro-incubator.

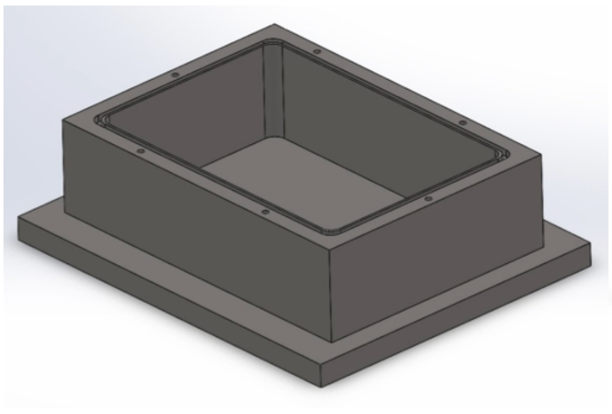
The black part in the bottom is not a part of the actual incubator as that is a heat sink for the peltier cell, necessary to use it efficiently, neither are the black and red cords part of the incubator as that is the cords for the peltier cell. For the incubator itself, the bottom and walls are made of aluminium with a plastic lid glued in place. In the incubator,we can fit different kinds of well plates (see Fig. 13, as an example of well plate). The different kinds of well plates are roughly the same size, they only differ in wells number and size.

Figure 13: One kind of well plate.

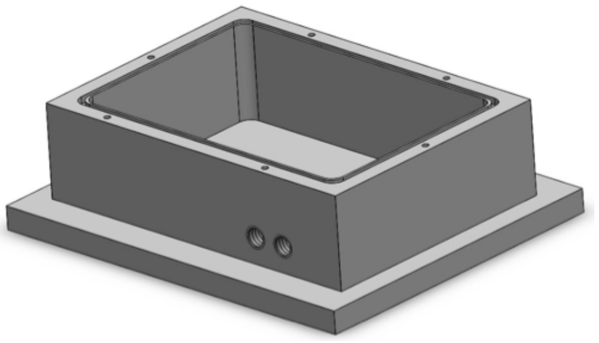
Of course, we need to be able to change the well plate so therefore we need an opening of some kind. So, we design a front cover opening. As seen in figure 11 the front cover is L-shaped at the top, the reason for this was to ensure it would air-tight secure with the lid.

In what would be our final design, we kept the basics from the previous design but changed how to open the incubator and also added additional features, as seen in Fig. 14a. This design was made using Solidworks instead of Sketchup software, and the major change was removed the possibility to open the incubator by removing the front cover and instead remove the lid to change the plates.

On top of the incubator there are the six screw holes so you can screw the lid in place. To make sure that it is completely air-tight we also cut a track all around the top in which we will fit an O-ring. The lid itself will be a simple transparent plastic lid. It must be transparent so that you can study the samples in the well plate under a microscope.



(a) The final incubator design.



(b) The incubator with holes for the valves.

Figure 14: Pictures of incubators

At last, create two holes in the side of the incubator. In the holes we can screw some valves to connect a tube for the carbon dioxide on one of them and use the other one as an outlet. We decided not to include them in the final design though, because we did not know what size they needed to be. They are, however, easily put back in. How the incubator would look with the holes can be seen in Fig. 14b.

The last task to do was to determine what material to use for the incubator. We decided to use aluminium because meets all the requirements for the incubator, for example it does not interfere with the magnetic field, and it is relatively cheap.

4 Discussion

4.1 Glucose measurements

Considering an average human temperature of 36.5 °C our scale of approximately 0-70 is ideal for measuring a broad range of temperatures above and below average proving useful for glucose measurement. Furthermore by changing parameters in the thermistor circuit one can narrow down, or widen, the temperature scale after pinpointing relevant areas from initial data.

4.2 Measurement accuracy

The accuracy of the measured temperature is a subject to discuss. There are some factors that affect the accuracy. Self-heating of thermistor, resolution of Arduino and resolution of thermometer at calibration affect the accuracy. In the datasheet for the pt100 can be seen that it will start to self-heat if too much current is running through. If the power is 1 mW, the self-heating will be less than 0.5 °C. It could be between 0 °C and 0.5 °C and that is hard to identify. The RF measurement and the incubator will be around 37 °C The circuit with the pt100 are designed so that the power generated at 37 °C will be less than 1mW. It is possible to design the circuit in a way that the power will be even lower. In order to do that the resistor in series with the pt100 has to be bigger. The bigger the resistor is, the worse the resolution of the Arduino becomes. You have to take both self-heating and Arduino's resolution in account when designing the circuit.

To calibrate the thermistors, a thermometer was needed as a reference. During the calibration of the NTC thermistor, the used thermometer had a resolution of 0.1 °C. That is not good if you want an accuracy of 0.1 °C or 0.01 °C. For the pt100, the calibration was not made since the pt100 accidentally was destroyed. But the resistance of the pt100 should be 100 Ω at 0 °C and 138.5 Ω at 100 °C and the curve is linear.

The NTC thermistor will not be used as the temperature sensor in this project. But we wanted to calibrate it anyway to see if it is good or not. The result was not that good and therefore the pt100 would be used as the sensor. The calibration for the NTC thermistor could have been better if we calibrated between 0 °C and 100 °C. The resistance changes a lot for every degree below 35 °C as we can see in Fig. 6. Therefore it would be better if the calibration could start from a lower temperature. But the boiler could not be cooler than 19 °C.

4.3 Cost

The cost for our components required to make this project was rather cheap. The Arduino Mega 2560 can be found on eBay for as little as 13 USD (110 SEK), the peltier cell costs about 7 USD (60 SEK) and the H-bridge motor costs no more than 5 USD (45 SEK). A pt100 temperature sensor is available for 20-40 SEK and finally we bought the box, in which we built in our components, for 60 SEK. An additional cost of around 20 SEK for the circuit we use to make the temperature accuracy better and for the terminal blocks in which we connect the peltier, thermistor and the power supply.

This gives us a total cost of about 400 SEK for the entire temperature controller and comparing that to commercially available products, for example Electron Dynamics TCM, which also is a temperature controller, costs a massively 7000 SEK and it basically does the same thing that our product does.

4.4 Temperature sensor

Our temperature controller can, unfortunately, only be used with a Pt-100 thermistor. That is because the voltage dividers in the amplifier circuit are designed after the specifications for the pt100. The same circuit could be designed to work with other thermistors, for example pt1000 or NTC, but you would have to calculate new values for the resistors at the voltage divider and at the differential amplifier stage. For the pt1000 this would be rather easy, you would just have to change the 1600 Ω resistor to one that is 10 times larger, 16 kΩ. That is because the pt100 and the pt1000 are both linear and have the same coefficient the difference being that the pt1000 has got a resistance that is 10 times larger than the pt100.

We could have added the possibility of having multiple thermistors by having multiple circuits next to each other on the circuit board, we would probably have room for three circuits inside the box, with them all being identical circuits but with different values for the resistors. We could have supply voltage to them all from the Arduino and having three different analog pins, one for each circuit, and in the code choosing which of the pins to read from. The major problem with this idea though is how to connect the thermistor and the easiest way would be to add more terminal blocks, having two blocks for each circuit and depending on which circuit we want to use we connect the thermistor to that circuits blocks. However, doing that would lead to a lot of blocks in the box, it would be too many really and also it would not look good.

The other idea would be to have two blocks for the thermistor and inside the box connecting the circuit we would like to read from to the blocks and simply leaving the other circuits disconnected, but this wouldn't be an ideal solution either because in other to change circuit one would have to remove the lid.

4.5 PID calibration

The PID in our Arduino program is not ideally calibrated but it is working. But we sometimes got a slow rise time and when we change the values we get a large overshoot instead. Calibrating the PID constants to get good depends on a number of factors. Firstly, the power supply used can make a huge difference, we used a supply that could deliver 600 mA, which is low level to drive the peltier efficientlyand we and we got some good measurements so it seemed as if our values for the PID were well balanced, no large overshoot and it stabilized after a short period of time. But when we switched to a power supply able to drive a higher current we got a large overshoot and the temperature didn't really stabilize.

Another feature to think about was where the thermistor is in relation to the peltier and what material is between them. When we were testing we used scotch tape to keep the thermistor directly on the peltier cell but when the measurements will be done it will be instead on top of the glucose sensor, made in metal, with the peltier underneath.

But changing the values for the PID is easily done in the code and it should not take too long find correct values. If we were to make further studies in this field we would probably be able to find suitable values.

5 Conclusions

The goal was to design, build and program a temperature controller to be used for different purposes, design a micro incubator for measurements on cells, constructing Helmholtz coils and being able to use the temperature controller together with a vector network analyzer. We have done all this but we did not have the time to get the incubator, we would send the design to someone who would build it for us. We could not construct the coils but we have made some tests and simulations on them.

We have also shown that it is possible to build a functional temperature controller with a fraction of the cost compared to other commercially available options. Our controller does not have good values for the PID but that is the only bug and also it is pretty easy to find good values.

We have had the opportunity to learn a number of different softwares, like Arduino IDE which is C/C++ and we all have prior experience with the similar language Java. But none of us had used Labview previously so that was a completely new experience for all of us, it was a little tricky at first but the more we used it the more comfortable we got. We also learned how to simulate coils using COMSOL and design the incubator in both the easy-to-use Sketchup and the more complex Solidworks.

6 References

[1] http://www.idf.org/worlddiabetesday/toolkit/gp/facts-figures[150614]

[2] http://www.nyteknik.se/nyheter/innovation/forskning_utveckling/article269028.ece [150614]

[3] https://en.wikipedia.org/wiki/Incubator_(egg)#History [150614]

[4] Lucia Potenza, Luca Ubaldi, Roberta De Sanctis, Roberta De Bellis, Luigi Cucchiarini, Marina Dachà, Effects of a static magnetic field on cell growth and gene expression in Escherichia coli, Mutation Research/Genetic Toxicology and Environmental Mutagenesis, Volume 561, Issues 1–2, 11 July 2004, Pages 53–62, ISSN 1383-5718, http://www.sciencedirect.com/science/article/pii/S1383571804001020

[5] https://processing.org/reference/environment/ [150519]

[6] http://en.wikipedia.org/wiki/Standard_Commands_for_Programmable_Instruments [150520]

[7] http://na.support.keysight.com/pna/help/latest/Programming/GPIB_Command_Finder/SCPI_Command_Tree.htm [150520]

[8] http://en.wikipedia.org/wiki/PID_controller [150519]

[9] 5 Kjell Hansson Mild, Jonna Wilén, Mats-Olof Mattsson, Myrtill Simko, Background ELF magnetic fields in incubators: A factor of importance in cell culture work, Cell Biology International, Volume 33, Issue 7, July 2009, Pages 755-757, ISSN 1065-6995, http://dx.doi.org/10.1016/j.cellbi.2009.04.004