MICROCONTROLLER CONTROLLING A SHAKER FOR PROTOZOA

DIPLOMA WORK

STUDENT: ANDREA FERNÁNDEZ FERNÁNDEZ

STUDY PROGRAM: ELECTRONIC ENGINEERING

MENTOR: SUZANA URAN

MARIBOR, JUNE 2018
ACKNOWLEDGEMENT

I would like to thank all the people who have relied on me during these 4 years.

To my family, specially my parents, for always having a word of encouragement and always believe in me.

To the “Pisito”, for being the motor during this stage of my life, and to the Minidorm team, for so much in so little.

Thanks to the University of Cantabria for giving me the opportunity to live the Erasmus experience.

And to Suzana Uran, mentor of my thesis, for her time, patience and dedication.
# INDEX

**CHAPTER 1.- INTRODUCTION** ......................................................................................... 5

**CHAPTER 2.- DEVICES** ............................................................................................. 7

2.1.- MICROCHIP PIC18F25K22 .................................................................................... 7
2.2.- MODULE CP2102 USB TO TTL CONVERTER ..................................................... 9
2.3.- VOLTAGE REGULATOR LM7805 ........................................................................... 9
2.4.- STEPPER MOTOR 28BYJ-48................................................................................ 10
2.5.- ACCELEROMETER MMA8451Q-1 ......................................................................... 12
  2.5.1.- ORIENTATION DETECTION AN4068 ............................................................. 13
2.6.- TEMPERATURE SENSOR DS18B20 ..................................................................... 13
2.7.- RELAY G2RL-472571 ......................................................................................... 14
  2.7.1.- MOSFET BS170 ............................................................................................ 15

**CHAPTER 3.- I2C COMMUNICATION** ........................................................................ 17

3.1.- ADVANTAGES AND DISADVANTAGES ............................................................... 17
3.2.- I2C PROTOCOL .................................................................................................. 18

**CHAPTER 4.- CIRCUIT ASSEMBLY AND MIKROC PROGRAM** ................................. 20

4.1.- PROCESS ............................................................................................................. 21
  4.1.1.- LEDS ............................................................................................................. 21
  4.1.2.- STEPPER MOTOR ......................................................................................... 22
  4.1.3.- TEMPERATURE SENSOR ............................................................................. 26
  4.1.4.- TEMPERATURE CONTROL ......................................................................... 30
  4.1.5.- ACCELEROMETER ....................................................................................... 31

**CHAPTER 5.- CURRENT MEASUREMENT** ................................................................. 33

5.1.- MICROCONTROLLER ......................................................................................... 33
  5.1.1.- RESISTOR ................................................................................................... 34
5.2.- AMPLIFIER ....................................................................................................... 35
5.3.- VOLTMETER ..................................................................................................... 36
5.4.- PROGRAM ......................................................................................................... 36
CHAPTER 6.- RESULTS AND CONCLUSIONS ........................................................................... 38
CHAPTER 7.- APPENDIX ................................................................................................. 41
CHAPTER 8.- FIGURES ................................................................................................. 50
CHAPTER 9.- REFERENCES ............................................................................................ 51
CHAPTER 1. INTRODUCTION

Nowadays, electronics is a clear element present in our lives, to the point of not paying attention. The alarm the radio, the clock, telephone and television, computers and other machines are some devices of electronics. However, Electronics in Medicine has slip by unnoticed for most of our society even though it is present from our birth to our death through elements such as incubators (with temperature and humidity control), radiographs and scanners or automatic blood pressure meters in pharmacies.

Biomedical engineering is a new discipline. It could be defined as the application of the principles of engineering to the life sciences. It is a mix between the engineering design capacity and the analysis tools of mathematics, physics and chemistry to solve problems in medicine, biology, biotechnology, pharmacy, etc.

It is a halfway between Telecommunications Engineering, Electronic Engineering and Computer Engineering and Life Science. Unlike other engineering, this one has a clear orientation towards the research and development of new techniques and products in the field of Biomedicine.

A brief mention should be made of the difference between microelectronics and nanotechnology. The first one is the application of electronic engineering to components and circuits of every small dimensions, microscopic and even molecular level to produce devices and electronic equipment of small dimensions but highly functional, while the second one is a field of applied sciences that is dedicated to control and manipulation at the level of atoms and molecules.

Figure 0. Microelectronics [1]
This Bachelor Final Project goal is make a board to study protozoa. One of the types of protozoa that exist are the ciliates. This project is worked with "Colpoda", which is a genus of ciliates. The survival of these protozoa depends on weather conditions, like temperatura, and other external factors, like movement, that will be used throughout the project.

![Figure 1. Colpoda (Protozoan).](image)

The diploma work has a part of circuit assembly powered by a 9V battery and another part of programming. The circuit will be built on a board which will be connected with the PC using a converter (USB to TTL) and therefore be able to carry out the previously mentioned program. To carry out the circuit assembly that studies the protozoa, the board will be used. In it there will found the different elements and there will be needed a program to make each of them work.

To make this board, the most important element is the microcontroller. A stepper motor is needed and an accelerometer is used to measure its acceleration. Furthermore, the motor produces a movement to make the broth where these protozoa grow, to shake. It will also be used a temperature sensor and a hair dryer. The first one measure room temperatura and if it is more than 30 degrees the hair dryer will be turn off, but if the room temperatura is less than 28 degrees, the hair dryer will be turn on. This part of the circuit will be conected with a relay that it works like a switch controlled by an electrical circuit, where a coil and an electromagnet activate to be of one or more contacts that allows opening or closing other independent electric circuits.
CHAPTER 2. DEVICES

2.1.- PIC18F25K22

The microcontroller is the brain of the project. It has to be programmed to control inputs and outputs necessary for the operation of the rest of the devices. It must to receive values of the relay control, temperatura sensor and stepper motor. That is why the microcontroller is very important in the project. University offered me the best one to ensure that the circuit behave in the correct way.

This microcontroller supports from 2.3V to 5.5V to operate and its characteristics can be seen on the PIC datasheet. In this case, the PIC that has been used in the project, has 28 pins. However, to program it and connect the different inputs and outputs in their pins. Next, a figure with the distribution of all these pins can be seen.

As not all pins are used but then a scheme will be show to know which pins are been used in this project. Furthermore, On the right column of the Figure 3 is a little legend with the connection of the elements that are been used.

![Figure 2. PIC18F25K22 with all pins. [3]](image-url)
This microchip has seventeen analog channels, two I2C ports, two UART and seven timers, and five CCP modules, which can be configured to use the PWM. The basic scheme of the circuit, that has been done with Eagle program, can be seen in the Figure 4.
As it can seen in the figure, the basic circuit is formed by the 9V battery, the voltage regulator and the clock as main elements. It can also seen some ceramic capacitors and a electrolytic capacitor, a pair of resistors and the necessary voltage sources or virtual grounds. Finally, it can seen the switch, which will be operated with the pressure of its button.

2.2. MODULE CP2102 USB TO TTL CONVERTER

The converter CP2102 allows the microcontrolled and the PC to communicate using the USB protocol in a simple way.

To use this board as a programmer is used the next connections:

<table>
<thead>
<tr>
<th>CP2102</th>
<th>PIC18F25K22</th>
</tr>
</thead>
<tbody>
<tr>
<td>3V3</td>
<td></td>
</tr>
<tr>
<td>TXD</td>
<td>Rx</td>
</tr>
<tr>
<td>RXD</td>
<td>Tx</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>+5V</td>
<td>Vcc</td>
</tr>
</tbody>
</table>
2.3.- VOLTAGE REGULATOR LM7805

Our microcontrolled only permits voltage between 2.3V to 5.5V operation. During the project execution is used a battery of 9V, that is why it needed a voltage regulator, which can be seen in the Figure 5, to convert that voltage in 5V and get the correct operation of the PIC.

![Voltage regulator](image)

**Figure 5. Voltage regulator.[4]**

2.4.- STEPPER MOTOR 28BYJ-48

Stepper motors are direct current motors that moves in small steps. Some coils are organized in “phases”, that is, in groups. To get the motor rotate, step by step, each of the phases will receive energy. With these motors, precise position and speed control can be got. For this reason, stepper motors are the engine of choice for many precision motion control application.

There are a wide variety of stepper types. In the project, a unipolar stepper motor, which can be seen in the Figure 6, is used.

![Stepper motor](image)

**Figure 6. Stepper motor.**
The unipolar stepper motor that is used during the work, has 5 wires. This type of motor is very common when the unipolar motors are small. All of wires of the coil are tied together in a wire, which is the fifth one. It can only be operated as a unipolar motor.

![Internal connection diagram](image1)

**Figure 7. Internal connection. [5]**

In this work, the function of the stepper motor is to move a flat base where the broth with the protozoa will be. To get this movement, the stepper motor is connected with the microcontroller thanks to the board with the integrated ULN2003.

![Board with ULN2003](image2)

**Figure 8. Board with ULN2003**

First, the motor was turned only in one direction, for this, it was programmed four leds, which are between the motor and the microcontroller. The program made these leds turn on and turn off consecutively. The faster leds will change its state, the
faster the motor will rotate. On the contrary, the slower the LEDs change, the less velocity the motor will have.

Once the project was advanced, a couple of modifications were made to the program so that the motor would turn a little towards each side.

### 2.5.- ACCELEROMETER MMA8451Q-1

The accelerometer is a sensor that is used to measure the acceleration forces. It measures in meters per square second (m/s²) or in force – G (g). A single G-force for us on planet earth equals 9.8 m/s². These sensors are useful to measure vibrations and movements of a system. An accelerometer has low power, it is feed with 5 volts or less and it produces a current of the order of mA or uA.

Accelerometers can have an analog interface, a digital interface or PWM. In the project, accelerometer with a digital interface will be used. It can communicate through SPI or I2C. These have more functionality and are less susceptible to the noise than analog accelerometers.

MMA8451Q-1 digital accelerometer has been used. It communicates with the I2C communication protocol, which it will be talked in the next chapter. This MMA8451Q-1 is a smart, three-axis, capacitive, micromachined accelerometer with 14 bits of resolution.

![Accelerometer](image.png)

*Figure 9. Accelerometer.*
This device is used in the project to measure the acceleration of the stepper motor and to control that this is the correct one so that the content that is in the base does not fall. To achieve this, the accelerometer has a embedded orientation detection which will be discussed below.

2.5.1.- ORIENTATION DETECTION AN4068

The accelerometer has a sensor that detects the orientation. In the next figure, coordinate system can be seen, where each axis indicates a direction in the plane.

It is known that gravity on Earth is 9.8 metres per square second and that it goes down, however, it must be mentioned, that in the program, the Z axis of the orientation detector is negative when it is facing up.

![Coordinate system](image)

Figure 10. Coordinate system. [6]

Positive Z axis -> g force when the device is upside down, as this can be happening when the device is on its side, like it is in our project, for example.

2.6.- TEMPERATURE SENSOR DS18B20

The temperature sensor is a simple device. Its function is to measure the room temperature from -55 ° C to + 125 ° C (-67 ° F to + 257 ° F) despite its small size, but not all range is used in the project. Furthermore, its voltage range is from -0.5V to +6.0V. Its main feature is that it uses OneWire communication, which is a special protocol that allows send and receive data using a single wire.
Regarding the structure, the pin configuration of the sensor is:

![Figure 11. Connections of the temperature sensor. [7]](image)

Seeing the figure shown and knowing a little bit about the connection of each pin [b], it can be said that it is a simple enough element as it is said before.

**2.7.- RELAY G2RL-472571**

A relay is a switch which controls (open and close) circuits electromechanically. This device has one coil inside and it is used where it is necessary to control a circuit by a separate low-power signal.

If a small current passes through the coil, the electromagnetic switch acquires magnetic force and attracts the needle, then it gets turn on.

In the project the relay is used to turn on and turn off a hairdryer. This one is connected between the above mentioned relay and the input, and it will depends on the temperature detected by the mosfet briefly explained in the previous point.

It has an internal connection [c] and it needs some external elements to operate. The relay needs an auxiliary electric conector to can connect it with the hairdryer, and a diode between terminals 1 and 8 (it has to be careful with the placement of the diode and put it in the same direction as the current so that it does not get damaged).
2.7.1.- MOSFET BS170

Mosfet means “Metal Oxide Semiconductor Field Effect Transistor”. It is used for signal switching and amplification and theses devices are divided in two types, channel N and channel P. The mosfet BS170 that it is used in the project, is the first one. The main advantage of the mosfet transistor is that it uses low power to carry out its purpose and it loses very little energy, which makes it very used in the electronics of microcontrolled specially.

Its package is similar to the temperature sensor, the mosfet has three legs too:

- Drain (D)
- Gate (G)
- Source (S)

![Figure 12. Mosfet. [8]](image)

Between Drain and Source terminals the current passes when Gate is activated by means of voltage. The current when the transistor is activated enters through S and exits through D but G has to have a minimal voltage.

When using the Mosfet BS170 terminal D has been connected to pin 8 of the relay, which it is explained in the following point. Between G and S terminals a resistor of value 10 kΩ is placed. In turn, G is connected to the PIC, to receive and send data and S is connected to the virtual ground (GND). This element is necessary to avoid a short circuit.

In the next figures, the circuit of relay it can be seen. When the current flows through the inductor of the first circuit, the second one will open its switch to operate the hair dryer since the points X and Y of the diagram are connected with the points X and Y of the second image.
Finally, to be able to assemble the circuit, it was necessary to solder the board, with tin, all necessary elements and once done, connect the hairdryer and the supply to verify its correct operation.
CHAPTER 3. I2C COMMUNICATION

There are several communication protocols that could have been used to communicate, but it was chosen I2C because of the advantages that it presents for this situation. I^2C means “Inter-Integrated Circuit” and it is a serial communication bus widely used for communication between microcontrollers and their peripherals in integrated systems. In this section is explained why it is chosen and how it is work to understand it.

3.1. - ADVANTAGES AND DISADVANTAGES

It is important to mention some points that should be known in order to ensure the correct choice of using I2C communication.

On the one hand, some of the advantages that can be found are:

- It only uses two communication wires, one to send data (SDA) and the other one to the clock signal (SCL).
- Supports multiple masters and multiple slaves, making the appropriate connections. Furthermore, if necessary, it is easy to add new slaves to the bus and it is possible to have more than one master.
- It is a widely known and used protocol.

On the other hand, there are some disadvantages:

- It is half-duplex, one to send data an the other one to receive them but it does not allow do so at the same time.
- It is slower than the SPI protocol, but for this application it is not needed to really high communication speed.

Once analyzed the protocol and knowing what are the pros and cons of using this type of communication, advantages are more important than the disadvantages, and therefore, in the next point, it will explain how this protocol works.
3.2. - I2C PROTOCOL

To understand the I2C protocol for programming is necessary to learn the basic of this protocol, so a brief explanation is given. First of all, the devices of a bus can be classified as masters and slaves, since they have only one address for each one. The master starts the data transfer and generates the clock signal, the slave waits for a master to communicate with its to generate the transfer. There can be more than one master but, in this project not used slaves, only used a master.

Only two wires are needed, the clock signal (SCL) and the data line (SDA). Both wires need pull-up resistances and it can be seen in the scheme connection of the Figure 15.

![Scheme connection](image)

Figure 15. Scheme connection. [9]

This protocol uses half-duplex communication, that is, it uses the same wire to send and receive data but these two options cannot be done at the same time. An example of this communication is the communication between humans because while one of them talks, the other one listens, and the other way around. They cannot talk and listen at the same time. A brief difference between half-duplex and full-duplex is showed in the Figure 16.
All I2C bus addresses are 7 bits or 10 bits, in this case, it will work with 7. When this address is sent, 8 bits are always sent. The eighth bit is used to inform the slave if the master is writing (0) or reading (1) from it.

The master generates the condition of START. Each word is placed on the SDA and the first word transferred contains the address of the selected slave. Then, the master read the status of the SDA line, if it is 0, the transfer process continues, but if it is 1, the address circuit does not allow communication and the master generates a stop bit to release the I2C bus. This is very important in the protocol and it is called ACK (acknowledge). Finally, the master generates STOP condition. SDA an SCL lines are raised.
CHAPTER 4. CIRCUIT ASSEMBLY AND MIKROC PROGRAM

In this chapter, the assembly circuit is explained step by step, as well as the program that controls the activity of each device.

Once all elements that are going to be used through the project are within reach, the circuit will be assembled and programmed by parts but first the basic circuit was drawn on a paper to see the main connections of the microcontroller and be able to connect the other elements from that.

![Basic Circuit](image)

**Figure 17. Basic circuit.**

Furthermore, to control the elements of the project the program MikroC has been used. There are a lot of C programs on the market but it is easy to learn because it has a very simple structure and it also has a large number of library functions.

MikroC has been developed by MikroElektronika and during the project, the doubts have been solved with the help of the book [11] that the mentor has given me to understand the program or with some of the examples that had been explained.

Next, the process will be explained, from the placement of the elements on the board to its implementation.
4.1. - PROCESS

Once the basic circuit is achieved and the connections of the pic are known, the assembly of each element are done one by one and programmed with the program MikroC. The general purpose and way of procedure of the program will be explained.

![Circuit assembly](image1.png)

Figure 18. Circuit assembly.

4.1.1. - LEDS

The first element connected to the basic circuit was two red leds how it can be seen in the Figure 19. These leds will turn on and turn for 1 second and this brief program served as a initial contact.

```
PORTB = 0b00001111; //LEDs ARE ON
delay_ms(1000);      //1 SEC DELAY
PORTB = 0b00000000; //LEDs ARE OFF
delay_ms(1000);      //1 SEC DELAY
```

Figure 19. LEDs program.

Leds will work alternately, while one of them will be off (0), the other one will be on (1), and the other way around. It will serve later to facilitate the programming of the stepper motor.
4.1.2.- STEPPER MOTOR

Then, the stepper motor was assembled in the board. The wires of the integrated ULN2003 are connected to the channels (RA1, RA2, RA3, RA4) of the microcontroller and the stepper motor is connected to the integrated. In the Figure 20 the program about it can be seen.

```c
void stepper(int x) {
  int x;
  for (x=0; x<xw; x++) {
    switch (x) {
      case 0:
        PORTA = 0b00001000;
        delay_ms(3); 
        break;
      case 1:
        PORTA = 0b00001100;
        delay_ms(3);
        break;
      case 2:
        PORTA = 0b00001000;
        delay_ms(3);
        break;
      case 3:
        PORTA = 0b00001100;
        delay_ms(3);
        break;
      case 4:
        PORTA = 0b00000100;
        delay_ms(3);
        break;
      case 5:
        PORTA = 0b00000110;
        delay_ms(3);
        break;
      case 6:
        PORTA = 0b00000010;
        delay_ms(3);
        break;
      case 7:
        PORTA = 0b00001000;
        delay_ms(3);
        break;
      default:
        PORTA = 0b00000000;
        delay_ms(3);
        break;
    }
  }
}
```

Figure 20. Stepper motor program.
As it can be seen, a “switch” has been used with eight different cases. In each of them, the LEDs, that are on the same board as the integrated, are on or off depending on the case. These follow a stepped series, from the LED of pin RA4 to the LED of pin RA1. Once the series is finished, all LEDs will be off (the pins will set to zero) and the “switch” will start again.

This program was the first one that was made for the stepper motor, but it was only possible to turn it in one direction. From which was progressing in the project, a series of modifications were made in the programming to achieve rotated a bit in both directions to get this way, move the red plate where would be the protozoa liquid.

To understand the final program of the stepper motor a flow chart is shown in the next figure.
```c
void stepper(int xw, int zw) {
    int z;
    int x;
    int c;
    while(1)
    {
        c=0;
        for (x=0;x<xw; x++)
        {
            switch(c)
            {
                case 0:
                    PORTA = 0b000010000;
                    delay_ms(3) ;
                    c=c+1;
                    break;
                case 1:
                    PORTA = 0b000110000;
                    delay_ms(3) ;
                    c=c+1;
                    break;
                case 2:
                    PORTA = 0b000010000;
                    delay_ms(3) ;
                    c=c+1;
                    break;
                case 3:
                    PORTA = 0b000001100;
                    delay_ms(3) ;
                    c=c+1;
                    break;
                case 4:
                    PORTA = 0b0000001100;
                    delay_ms(3) ;
                    c=c+1;
                    break;
                case 5:
                    PORTA = 0b0000000110;
                    delay_ms(3) ;
                    c=c+1;
                    break;
                case 6:
                    PORTA = 0b00000000110;
                    delay_ms(3) ;
                    c=c+1;
                    break;
                case 7:
                    PORTA = 0b000000000110;
                    delay_ms(3) ;
                    c=0;
            }
        }
    }
}
```
break;
}
}
c=0;
for (z=0; z<kx; z++)
{
    switch(c)
    {
    case 0:
        PORTA = 0b000010010;
        delay_ms(3);
        c=c+1;
        break;
    case 1:
        PORTA = 0b000000010;
        delay_ms(3);
        c=c+1;
        break;
    case 2:
        PORTA = 0b000000110;
        delay_ms(3);
        c=c+1;
        break;
    case 3:
        PORTA = 0b000001000;
        delay_ms(3);
        c=c+1;
        break;
    case 4:
        PORTA = 0b000011000;
        delay_ms(3);
        c=c+1;
        break;
    case 5:
        PORTA = 0b000001000;
        delay_ms(3);
        c=c+1;
        break;
    case 6:
        PORTA = 0b000110000;
        delay_ms(3);
        c=c+1;
        break;
    case 7:
        PORTA = 0b000100000;
        delay_ms(3);
        c=0;
        break;
The second program of the stepper motor is not more than an extension of the first one. So that it turns in the opposite direction to that which was had, the leds will be turned on in the same way, but in the opposite direction.

Eight is the minimum value that the motor step can acquire because it has eight bits, but in this project one thousand steps are taken to get that the motor can turn a considerable angle.

### 4.1.3.- TEMPERATURE SENSOR

The next program to explain will be the temperature device. This sensor will capture the room temperature, as it has been explained in the previous points. For the operation of this element a resistor has been needed, in addition to the obligatory supply connections.

And for this operation, it will use a code a bit longer and more complicated than the one used for the stepper motor.
In the first part of the code all variables that are going to be used throughout the program are defined, and then, all pins are configured as digital inputs and digital outputs.

In order to verify that the conversion that is made is going to be well done, it is checked with the last line of the code that is seen in the previous figure. The temperature has a value of 256 because using the conversion, it has to multiply sixteen by the value of the degrees, which in this case has been used also sixteen degrees:

\[ 16 \text{ degrees} \cdot 16 = 256 \]
In the second part of the program, the conversion of the temperature values is carried out. First the sensor checks if the temperature is negative, so that from there the correct conversion can be made.

In the next array we find 8 bits. Bit 7 will define the sign of the temperature, 1 if it is negative and 0 if it is positive. The next two bits (6 and 5) will belong to the tens and units respectively, of the integer value. Bit 4 will serve to separate the integer number of decimals and bits 3, 2, 1 and 0 will be the decimals of the temperature. Below we can see an example:
In this case, the value of the temperature would be +23.8260 degrees.
In the third part of the program, the temperature conversion will be performed as in part two, but within an infinite loop so that the result is displayed every certain period of time. Finally, the value obtained with the conversion already made will be shown on the PC screen.

### 4.1.4.- TEMPERATURE CONTROL

As explained in the previous point, this sensor measures the room temperature. This temperature will cause a hairdryer, that is connected to the circuit board through the relay, turns on or turns off.

The hairdryer will turn on or turn off automatically thanks to the code that has been implemented. In this project, when the temperature is lower than 28 degrees, it will turn on to heat the air and if it is greater than 30 degrees, it will turn off to let the air cool down.
As it can be seen in the figure of the temperature control, the last two pins of port B (RB6 and RB7) will be used to connect the temperature sensor to the accelerometer and the results will can be seen in the screen of the computer. As mentioned in the temperature check of the sensor, the value 16, typical of the ASCII conversion, is multiplied by the temperature limit values (28 and 30 degrees). Therefore, at the time of programming, the values of 448 and 480 respectively will be used.

4.1.5.- ACCELEROMETER

In this part, instead of putting all the program of accelerometer, only will be shown the most important parts since this code is much longer and complex than the previous ones. In the Figure 27 is the flowchart that allows to make an idea of how the entire program works.

```c
// ---------------------------------------- CONTROL
void control (int tempera) { 
  if (tempera < 448) {  // convert 28° -> 28*16 = 448
    PORTB = 0b11000000;
  }
  if (tempera > 480)  // convert 30° -> 30*16 = 480
    PORTB = 0b00000000;
}
```

Figure 26. Temperature control program.

Figure 27. Accelerometer flow chart.
In order to understand this program, the operation of the I2C protocol has been explained in the previous chapter, this form of communication is the most important part of the realization of the program.

First, all variables are declared, all inputs and outputs are configured and communications UART1 and I2C are started to start the program. The next step is to initialize the MMA accelerometer using the "if" condition structure. In the event that the device is not recognized, it will return to the beginning of the program and that error should be corrected. Otherwise, the operation of the MMA would proceed. Once it has been recognized, the channels RB6 and RB7 of the pic would be turned on and would operate the accelerometer thanks to the connections of said pins with it. The next and last step is the entrance to an infinite loop, where the calculation and the conversion of the ax, ay and az coordinates of the space where the accelerometer is located, to be shown on the screen. These values will be displayed automatically every certain period of time.
CHAPTER 5. CURRENT MEASUREMENT

Once the main project was finished controlling the microcontroller for protozoa shaker, it was extended. The last objective was base on obtaining the microcontroller current consumption. To achieve this point, the microcontroller, the amplifier and the voltmeter were used.

Below, a diagram of the connections needed to measure the current can be seen.

![Diagram of connections]

**Figure 28. Final circuit diagram.**

In the next points a brief explanation of the current measurement is shown since the complete one is in the complementary report [12].

5.1.- MICROCONTROLLER

The microcontroller that has been used is the same that in the rest of the project. Its current measurement range is between the values of 1µA and 100mA, therefore, the best current would be current of 10mA and it has served to put the most appropriate resistance.
Although at the beginning there was another resistor of $100\Omega$ instead of the microcontroller, has the necessary measures for the work were carried out with the second one. In this part, the amperemeter was placed between the points A and B to measure the current because as the elements are in serial, the current, that flow through them, is the same.

5.1.1. - RESISTOR

The resistor has $10\Omega \pm 0.1\%$ of tolerance, it means that the value of the resistor is very accurate since the margin of error will be $10\Omega \pm 0.01\Omega$. As this value is very exact due to tolerance, as it has been said before, the voltage represents the current following the Ohm’s law ($U = R \cdot I$).

This resistor has $1/4W$ of power and if the current of the simple circuit is compared between the experimental and the theoretical parts, it can be seen that there is a small difference in its values because one part of the current is lost due to the power.

- Theoretical current.

$$I = \frac{U}{R} = 0.9A$$
Experimental current.

\[ I = 0.57 \, A \]

5.2.- AMPLIFIER

For the amplifier it was necessary to choose the correct one. It started with three possible options:

- LM741 amplifier.
- LM358 amplifier.
- Rail to Rail amplifier.

The correct option is chosen according to the most appropriate characteristics of each of them to measure the current, in this case, the second one was used.

The resistor of 10Ω is \( R_{\text{sense}} \) and it is used to calculate the value of \( R_{\text{load}} \). First use the left part of the amplifier circuit, disconnecting the rest. Taking as data the 5V supply, the current in that part of the circuit, 10mA, the value of \( R_{\text{sense}} \) and using the Kirchoff law, the solution of the resistor is 490Ω.
The next step is to calculate the value of the resistors R1 and R2 using all the circuit but now the input voltage of the amplifier is 0.1V and the output voltage is 5V, so its gain is 50. The value of R2 is supposed, $50k\Omega$ and therefore, the value of R1 is $1k\Omega$.

Then, with all theoretical values, the amplifier was assembled on the board following the connections in its datasheet [g].

But the output voltage of the amplifier cannot provide 5V, but it can provide less. To correct this value, the value of the input voltage had to be reduced by half, instead of 0.1 V, it would be 0.05V To reduce this voltage, it is necessary to reduce the current that is circulating through the resistors and this is achieved by increasing the value of them. Now, the new output voltage is 2.5 V, and with an equation, the new $R_{load} = 980\Omega$ as it can see in the Project.

5.3.- VOLTMETER

The voltmeter will be the second microcontroller, also the PIC18F25K22. The output of the amplifier will be the input of this microcontroller. The input goes to an analog channel, in this project it is RA7. Moreover, this voltmeter does the current measurement and experiment with speed of measurement.

5.4.- PROGRAM

Once the circuit is assembled, and the measurements are similar to those needed, the program is done in MikroC. The most important part is shown below:
In this part of the program all the calculations are developed. In the first paragraph AD conversion is used to convert the output volts of the amplifier (between 0V and 5V) to the ASCII code, where the maximum value is \(2^{10} - 1 = 1023\) corresponding to 5V. In the second paragraph the current measurement is made, and depending on the voltage value it will give a value of the current. Then, both the AD voltage and the current are shown by the PC screen and it enter in an infinite loop so that this result is automatically displayed every 2 seconds.
CHAPTER 6. RESULTS AND CONCLUSIONS

The first element to operate was the stepper motor. Enough precision had to be used to move the container in the right angle but without throwing the liquid inside.

Figure 32. Base of the protozoa broth.

In conjunction with the stepper motor, the accelerometer was configured. It would be responsible for providing the coordinates in space (x, y and z). As it can see in the images shown below, in the left figure the accelerometer would place at rest on the table, so most of its coordinates are zero or close to zero (except on the z axis due to the gravity). In the right figure, the coordinates correspond to the board in a position in space, in this case, it was placed inclined.

Figure 33. Results of the accelerometer.
In the previous figure each picture has three columns. The left one is the “ax” coordinate, the column that is in the middle is the “ay” coordinate and the right one is the “az” coordinate of the axis.

Another element that had to be programmed and whose results can also be seen, is the temperature sensor. The value is displayed on the screen after the conversion. The first values that can be seen, belong to the ambient temperature, at which time the hairdryer would turn on to be less than 28 degrees. When the hairdryer was turned on, it would cause that temperature to increase its value, as can be seen in the subsequent values, until reaching a higher value of 30 degrees. At this time, the hairdryer would turn off and the temperature would decrease.

The objective of this function is that the ambient temperature is between a range of values more or less constant, in this case, between 28 and 30 degrees.

After this project, the program of the shaker’s control is completed, so it is possible to get the movement of the protozoa broth with the stepper motor at a suitable temperature with the temperature sensor and the hairdryer.

The second part of the project, once the previous elements operate correctly, the result of the current measurement can be seen. As shown in Figure 35, the values
of the AD voltage converted with the ASCII code and the measurement of the current are approximate to the theoretical value.

As conclusion to the project, it must be highlighted that the objectives fixed at the beginning of it have been successfully achieved.

However, in this study, some limitations can be seen, such as the accuracy of the elements. This project taken to professional research laboratories would require improved elements and therefore more expensive.

Although following in the range in which we move, some improvement could be made with the introduction of more functions, such as humidity control, which would serve the protozoa to improve their development capacity.

After this brief conclusion, this project is concluded.
### Table about the functions of each pin.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0</td>
<td>AN1 C12IN0-</td>
</tr>
<tr>
<td>RA1</td>
<td>AN1 C12IN1-</td>
</tr>
<tr>
<td>RA2</td>
<td>AN2 C1IN-</td>
</tr>
<tr>
<td>RA3</td>
<td>AN2 C1IN+</td>
</tr>
<tr>
<td>RA4</td>
<td>C1OUT SRQ</td>
</tr>
<tr>
<td>RA5</td>
<td>AN4 C2OUT</td>
</tr>
<tr>
<td>RA6</td>
<td></td>
</tr>
<tr>
<td>RA7</td>
<td></td>
</tr>
<tr>
<td>RB0</td>
<td>AN12 SKI</td>
</tr>
<tr>
<td>RB1</td>
<td>AN16 C12IN3-</td>
</tr>
<tr>
<td>RB2</td>
<td>AN18 CTED1</td>
</tr>
<tr>
<td>RB3</td>
<td>AN18 C12IN2-</td>
</tr>
<tr>
<td>RB4</td>
<td>AN19 P1D</td>
</tr>
<tr>
<td>RB5</td>
<td>AN19 CTED2</td>
</tr>
<tr>
<td>RB6</td>
<td></td>
</tr>
<tr>
<td>RB7</td>
<td></td>
</tr>
<tr>
<td>RC0</td>
<td></td>
</tr>
<tr>
<td>RC1</td>
<td></td>
</tr>
<tr>
<td>RC2</td>
<td>AN14 CTPLS</td>
</tr>
<tr>
<td>RC3</td>
<td>AN15</td>
</tr>
<tr>
<td>RC4</td>
<td>AN16</td>
</tr>
<tr>
<td>RC5</td>
<td>AN17</td>
</tr>
<tr>
<td>RC6</td>
<td>AN18</td>
</tr>
<tr>
<td>RC7</td>
<td>AN19</td>
</tr>
<tr>
<td>RE3</td>
<td></td>
</tr>
<tr>
<td>Vss</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
1. CCP2/P2A multiplexed in fuses.
2. TCKI multiplexed in fuses.
4. CCP3/P3A multiplexed in fuses.
[b] Pin description about the temperature sensor.

### Pin Description

<table>
<thead>
<tr>
<th>PIN</th>
<th>SO</th>
<th>µSOP</th>
<th>TO-92</th>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 6, 7, 8</td>
<td>2, 3, 5, 6, 7</td>
<td>—</td>
<td>N.C.</td>
<td>No Connection</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>3</td>
<td>VDD</td>
<td>Optional VDD. VDD must be grounded for operation in parasite power mode.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>DQ</td>
<td>Data Input/Output. Open-drain 1-Wire interface pin. Also provides power to the device when used in parasite power mode (see the Powering the DS18B20 section.)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>GND</td>
<td>Ground</td>
<td></td>
</tr>
</tbody>
</table>

[c] Internal connection and electrical data of the relay.

<table>
<thead>
<tr>
<th>Model</th>
<th>Contact form</th>
<th>Coil ratings</th>
<th>Contact ratings</th>
<th>Number of test operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>G2RL-1A</td>
<td>SPST-NO (1a)</td>
<td>3 to 48 VDC</td>
<td>12 A, 250 VAC (General Use) 40°C</td>
<td>100,000</td>
</tr>
<tr>
<td>G2RL-1</td>
<td>SPST-NO (1a)</td>
<td>3 to 48 VDC</td>
<td>12 A, 24 VDC (Resistive) 40°C</td>
<td>50,000</td>
</tr>
<tr>
<td>G2RL-1A-E (HA)</td>
<td>SPST-NO (1a)</td>
<td>3 to 48 VDC</td>
<td>16 A, 250 VAC (General Use) 40°C</td>
<td>100,000</td>
</tr>
<tr>
<td>G2RL-1E (HA)</td>
<td>SPST-NO (1a)</td>
<td>3 to 48 VDC</td>
<td>16 A, 24 VDC (Resistive) 40°C</td>
<td>50,000</td>
</tr>
<tr>
<td>G2RL-1A-E-ASl</td>
<td>SPST-NO (1a)</td>
<td>3 to 48 VDC</td>
<td>16 A, 250 VAC (Resistive) 85°C</td>
<td>30,000</td>
</tr>
<tr>
<td>G2RL-1A-E-CV</td>
<td>SPST-NO (1a)</td>
<td>3 to 48 VDC</td>
<td>TV-3 40°C</td>
<td>25,000</td>
</tr>
<tr>
<td>G2RL-1A-H</td>
<td>SPST-NO (1a)</td>
<td>3 to 48 VDC</td>
<td>10 A, 250 VAC (General Use) 40°C</td>
<td>50,000</td>
</tr>
<tr>
<td>G2RL-1H</td>
<td>SPST-NO (1a)</td>
<td>3 to 48 VDC</td>
<td>10 A, 24 VDC (Resistive) 40°C</td>
<td>50,000</td>
</tr>
<tr>
<td>G2RL-2A (HA)</td>
<td>DPST-NO (2a)</td>
<td>3 to 48 VDC</td>
<td>8 A, 277 VAC (General Use) 40°C</td>
<td>100,000</td>
</tr>
<tr>
<td>G2RL-2 (HA)</td>
<td>DPST-NO (2a)</td>
<td>3 to 48 VDC</td>
<td>8 A, 30 VDC (Resistive) 40°C</td>
<td>100,000</td>
</tr>
</tbody>
</table>
[d] Base where the broth of the protozoa are.

[e] Total program of the accelerometer.

```c
#define NMA8481_ADDRESS 0x1D // prednastavljeno na Adafruit NMA8481

#define NMA8481_CTRL_REG1 0x2A
#define NMA8481_CTRL_REG2 0x2B
#define NMA8481_XYT_DATA_CFG 0x0E
#define NMA8481_STATUS 0x00

 /* NMA8481 3-axis accelerometer control register bit masks */
#define NMA8481_ACTIVE_BIT 0x01 // postavi bit ACTIVE
#define NMA8481_STANDBY_BIT_MASK 0xF0 // zbrisi bit ACTIVE -> STANDBY
#define NMA8481_F_READ_BIT 0x02 // inicializira 8 bitno brzine sensorja
#define NMA8481_RST_BIT 0x40 // resetira pospeškometer
#define NMA8481_XYT_new_BIT 0x08 // resetira pospeškometer

 /* External 3-axis accelerometer data register addresses */
#define NMA8481_OUT_X_MSB 0x01
#define NMA8481_OUT_X_LSB 0x02
#define NMA8481_OUT_Y_MSB 0x03
#define NMA8481_OUT_Y_LSB 0x04
#define NMA8481_OUT_Z_MSB 0x05
#define NMA8481_OUT_Z_LSB 0x06

char txt[7]; // prednastena celja // number of variable for ASCII print

void NMA8481_I2C_Write(unsigned short address, unsigned short data)
{
    // the function is entered in the given register (address) NMA8481, one byte of data (data),
    // void NMA8481_I2C_Read(unsigned short address, unsigned short data)

    // addresses NMA8481 / Adafruit = 0x1D, -> 7 bit I2C title = 3A (1 x left-hand shift + 0)
    // I2C_Start(); // issue I2C start signal
    // I2C_Wr(0x3A); // send byte via I2C (device address + F)
    // I2C_Wr(address); // send byte (address of the location)
    // I2C_Wr(data); // send data (data to be written)
    // I2C_Stop(); // issue I2C stop signal
}
```
The function reads from the specified MMA8451 (one) address, one byte of data.

```c
short MMA8451_I2C_ReqRead(unsigned short address) {
    short tmp = 0;
    address = (address & 0x1F) | 0x40; // 7 bit I2C address + 3A (1 x move left + 0)
    I2C1_Start(); // issue I2C start signal
    I2C1_Wr(0x3B); // send byte via I2C (device address + R)
    I2C1_Wr(address); // read byte in data ADDRESS

    address = (address & 0x1F) | 0x41; // 7 bit I2C address + 3B (1 x move left + 1)
    I2C1_Repeat_S(String); // issue I2C signal repeated start
    I2C1_Wr(0x3B); // send byte (device address + R)
    tmp = I2C1_Rd(); // Read the 1 byte of data (no acknowledge) !! reads without
                     // ACK because it is 1st and last byte
    I2C1_Stop(); // issue I2C stop signal
    return tmp;
}
```

// The function checks if the MMA8451 accelerometer on the I2C is responding
// after writing to the accelerometer, the keypad returns ACK

```c
char get_ack_from_MMA8451() {
    char status;
    // address MMA8451/Adfruit = 0x1D, -> 7 bit I2C address = 3A (1 x move left 0)
    I2C1_Start();
    status = I2C1_Wr(0x3A); // Status = 0 if got an ACK
    I2C1_Stop();

    if (status == 0) {
        return(1); // ACK accepted, accelerometer responds
    } else {
        return(0); // ACK no accepted, accelerometer NY / does not respond
    }
}
```

```c
void MMA8451_Reset() {
    // MMA8451 is programmed by placing the RST bit in the CTRL_REG2 register at 1
    // The RST bit in the CTRL_REG2 register can also be set (exception) in the STANDBY state
    // MMA8451 datas, p. 46 - under Table 44: RST bit is used to activate the software reset.
    // The reset mechanism can be enabled IN STANDBY AND ACTIVE MODE
    // When the reset bit is enabled, all registers are reset and are LOADED WITH DEFAULT VALUES ->
    // setting +/- 2g in +/-
    MMA8451_I2C_RegWrite(MMA8451_CTRL_REG2, MMA8451_RST_BIT); // Reset the MMA8451 with CTRL_REG2
    delay_ms(100); // wait a bit for a reset
}
```
char uspesno;
    // address MMA8451/Adafruit = 0x1D, -> 7 bit I2C naslov = 3A (1 x move left + 0)
    uspesno = get_ack_from_MMA8451();
    if ( uspesno == 1 )
    {
        // with the program RESET MMA8451 we set the default preset and STANDBY state
        MMA8451_Reset(); //perform a RESET accelerometer MMA8451
        // sets 8-bit read acceleration (F_READ = fast read)
        // and go to the ACTIVE state
        MMA8451_I2C_RegWrite(MMA8451_CTRL_REG1, MMA8451_ACTIVE_BIT | MMA8451_F_READ_BIT);
        return(1);
    }
    else
    {
        return(0); // If the RESET accelerometer is unsuccessful, END and return 0
    }
}

// go to STANDBY
void MMA8451_Standby()
{
    // Read the current MMA8451_CTRL_REG1 registry value.
    // Put the sensor in the STANDBY state by setting the ACTIVE = 0 bit
    vpisano = MMA8451_I2C_RegRead(MMA8451_CTRL_REG1); // read the current state of CTRL_REG1
    MMA8451_I2C_RegWrite(MMA8451_CTRL_REG1, vpisano & MMA8451_STANDBY_BIT_MASK); // delete ACTIVE bit
}

// go into the state of ACTIVE
void MMA8451_Active()
{
    // Read the current MMA8451_CTRL_REG1 registry value.
    // Put the sensor in the ACTIVE state by setting the ACTIVE = 1 bit
    vpisano = MMA8451_I2C_RegRead(MMA8451_CTRL_REG1); // read the current state of CTRL_REG1
    MMA8451_I2C_RegWrite(MMA8451_CTRL_REG1, vpisano | MMA8451_ACTIVE_BIT ); // set ACTIVE bit
}

// the function reads from the given MMA8451, 9 consecutive BYTES, i.e. 8-bit acceleration ax, ay and az
// The MMA8451 has a auto-linking function for the address -> so we can read sequential data
char MMA8451_I2C_FastRead_Axis(int ax, int ay, int az )
{
    int ax, ay, az;
    short tmp[3];
    int i;
    char new_data;
    // are new data waiting for reading?
    // -------------------------------
    // my reading is so slow (1 second) that all be placed on 1 in the STATUS register !!
    // -------------------------------
    new_data = MMA8451_I2C_RegRead(MMA8451_STATUS); // I probably read ISS
```c
new_data = new_data & MMA8451_AIXZ_new_BIT;
if(new_data == MMA8451_AIXZ_new_BIT) // no novi podatki, bit je postavljen
{
    new_data = 1;
    // I read byte byte of data, READING ON DELAYS
    ax = MMA8451_I2C_RegRead(MMA8451_OUT_X_MSB);
    ay = MMA8451_I2C_RegRead(MMA8451_OUT_X_LSB);
    az = MMA8451_I2C_RegRead(MMA8451_OUT_Z_MSB);
}
else
{
    new_data = 0;
}
// overwrite output variables
//
ax = tmp[0];
ay = tmp[1];
az = tmp[2];
//
return (new_data); // return 1 = read new accelerations, 0 not new accelerations
```

**MAIN PROGRAM**

```c
------------ main program ------------

int main() {
    char uspesno;
    int accel_X;
    int accel_Y;
    int accel_Z;
    // initialization of PORTS - LEDs
    ANSELB = 0x00; // HBO do RB7 are digital
    TRISB = 0x00; // HBO in RB7 are outputs
    PORTB = 0; // all outputs to 0
    PORTC initialization - USB to UART and I2C are on PORTC
    ANSEL = 0; // digital USB to UART, I2C communication
    TRIS = 0x18; // Output: Pin17 - RC6 / Txl UART, Pin18 - RC7 / Rxl UART
    // all others are digital outputs */
    // initialization of UART1 for PC prints
    UART1_Init(115200);
    delay_ms(100);
    // initialization I2C
    I2C1_Init(100000);
    delay_ms(100);

    // initialization MMA8451 !!
    // successfully - MMA8451_Init();
    if(MMA8451_Init())
    {
        // flashing LED marks the start of the program
        PORTB.B7 = 1; // turns on LED on RB7
        PORTB.B6 = 1; // turns on LED on RB6
        delay_ms(500);
        PORTB.B7 = 0; // turns on LED on RB7
        PORTB.B6 = 0; // turns off the LED on RB6
        delay_ms(500);
        PORTB.B7 = 1; // turns on LED on RB7
        PORTB.B6 = 1; // turns on LED on RB6
        delay_ms(500);
        PORTB.B7 = 0; // turns on LED on RB7
        PORTB.B6 = 0; // turns off the LED on RB6
    }
```
```c
// carry out measurements of acceleration
while(1)
{
    accel_X = MBA451_I2C_RegRead(MBA451_OUT_X_LSB);
    accel_Y = MBA451_I2C_RegRead(MBA451_OUT_Y_LSB);
    accel_Z = MBA451_I2C_RegRead(MBA451_OUT_Z_LSB);
    // convert the received acceleration to ASCII to print to the PC
    //ByteToStrWithZeros(accel_X,txt);
    IntToStrWithZeros(accel_X,txt);
    UART1_Write_Text(txt);  // izpis ax
    UART1_Write_Text(" ");
    //ByteToStrWithZeros(accel_Y,txt);
    IntToStrWithZeros(accel_Y,txt);
    UART1_Write_Text(txt);  // izpis ay
    UART1_Write_Text(" ");
    //ByteToStrWithZeros(accel_Z,txt);
    IntToStrWithZeros(accel_Z,txt);
    UART1_Write_Text(txt);  // izpis az
    UART1_Write_Text(" ");
    UART1_Write(10);
    UART1_Write(10);  // to a new type
delay_ms(1000);

}  // konec while

else
{
    UART1_Write_Text(" I DO NOT FIND MBA451 accelerometer");
    UART1_Write(13);
    UART1_Write(10);
}
```

// end of the program
### DC Characteristics: RC Run Supply Current

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device Characteristics</th>
<th>Typ Max</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D020</td>
<td>Supply Current (IoD)</td>
<td>3.6 23</td>
<td>µA</td>
<td>VDD = 1.8V (RC_RUN mode, LFINTOSC source)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.9 25</td>
<td>µA +25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.9 ---</td>
<td>µA +60°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0 28</td>
<td>µA +85°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.0 30</td>
<td>µA +125°C</td>
<td></td>
</tr>
<tr>
<td>D021</td>
<td></td>
<td>6.1 25</td>
<td>µA</td>
<td>VDD = 3.0V (RC_RUN mode, LFINTOSC source)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.4 30</td>
<td>µA +25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.6 ---</td>
<td>µA +60°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.7 35</td>
<td>µA +85°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.7 40</td>
<td>µA +125°C</td>
<td></td>
</tr>
<tr>
<td>D022</td>
<td></td>
<td>16 35</td>
<td>µA</td>
<td>VDD = 2.3V (RC_RUN mode, LFINTOSC source)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 35</td>
<td>µA +25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 35</td>
<td>µA +85°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 50</td>
<td>µA +125°C</td>
<td></td>
</tr>
<tr>
<td>D023</td>
<td></td>
<td>16 50</td>
<td>µA</td>
<td>VDD = 3.0V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 50</td>
<td>µA +25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>21 60</td>
<td>µA +85°C</td>
<td></td>
</tr>
<tr>
<td>D024</td>
<td></td>
<td>19 55</td>
<td>µA</td>
<td>VDD = 5.0V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21 55</td>
<td>µA +25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 55</td>
<td>µA +85°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>23 70</td>
<td>µA +125°C</td>
<td></td>
</tr>
<tr>
<td>D025</td>
<td></td>
<td>0.14 0.25 mA</td>
<td>-40°C to +125°C</td>
<td>VDD = 1.8V (RC_RUN mode, MFINTOSC source)</td>
</tr>
<tr>
<td>D026</td>
<td></td>
<td>0.17 0.30 mA</td>
<td>-40°C to +125°C</td>
<td>VDD = 3.0V</td>
</tr>
<tr>
<td>D027</td>
<td></td>
<td>0.18 0.25 mA</td>
<td>-40°C to +125°C</td>
<td>VDD = 2.3V (RC_RUN mode, MFINTOSC source)</td>
</tr>
<tr>
<td>D028</td>
<td></td>
<td>0.20 0.30 mA</td>
<td>-40°C to +125°C</td>
<td>VDD = 3.0V</td>
</tr>
</tbody>
</table>

**Note 1:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

**Test condition:** All Peripheral Module Control bits in PMD0, PMD1 and PMD2 set to '1'.

**All I/O pins set as outputs driven to Vss; MCLR = Vcc; GSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).**
Pin configuration of the LM358 amplifier.
CHAPTER 8. FIGURES

Figure 0. Microelectronics.
Figure 1. Colpoda (Protozoan).
Figure 2. PIC18F25K22 with all pins.
Figure 3. PIC18F25K22 with pins that are been used.
Figure 4. Schematic with the PIC18F25K22.
Figure 5. Voltage regulator.
Figure 6. Stepper motor.
Figure 7. Internal connection.
Figure 8. Board with ULN2003.
Figure 9. Accelerometer.
Figure 10. Coordinate system.
Figure 11. Connections of the temperature sensor.
Figure 12. Mosfet.
Figure 13. Relay circuit diagram.
Figure 14. Relay with the hairdryer.
Figure 15. Scheme connection.
Figure 16. Full-duplex vs. Half-duplex.
Figure 17. Basic circuit.
Figure 18. Circuit assembly.
Figure 19. LEDs program.
Figure 20. Stepper motor program.
Figure 21. Stepper motor flow chart.
Figure 22. Final stepper motor program.
Figure 23. Temperature sensor program (First part).
Figure 24. Temperature sensor program (Second part).
Figure 25. Temperature sensor program (Third part).
Figure 26. Temperature control program.
Figure 27. Accelerometer flow chart.
Figure 28. Final circuit diagram.
Figure 29. Current measurement.
Figure 30. LM358.
Figure 31. Current measurement program.
Figure 32. Base of the protozoa broth.
Figure 33. Results of the accelerometer.
Figure 34. Results of the temperature sensor.
Figure 35. Results of the current measurement.
CHAPTER 9. REFERENCES


[12] Andrea’s Project. (14/06/2018)