

POLITECNICO DI TORINO

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

Corso di Laurea Magistrale in Mechatronic Engineering

Tesi di Laurea Magistrale

**Development of didactic test bench to verify the
reliability of control and protection systems**



**Politecnico
di Torino**

Relatore

Prof. Demichela Micaela

Candidato

Stefan Theodor Tiberius

October 2021

SUMMARY

1	INTRODUCTION	8
1.1	AIM AND STRUCTURE OF THE THESIS	8
1.2	GENERAL OVERVIEW OF INDUSTRIAL CONTROL SYSTEM	9
1.3	STEP RESPONSE	11
1.4	PID CONTROLLER	13
1.5	PLC	15
2	DEFINITION OF THE COMPONENTS OF THE SYSTEM	17
2.1	ELECTRICAL COMPONENTS	17
2.1.1	<i>Arduino UNO</i>	17
2.1.2	<i>Shield for Arduino</i>	20
2.1.3	<i>Wires, resistors, and other components</i>	21
2.1.4	<i>LCD Display</i>	25
2.1.5	<i>Chip DS18B20</i>	29
2.1.6	<i>Sensor level water</i>	30
2.1.7	<i>IR Sensor PNA4602N</i>	31
2.1.8	<i>Relay 4 channel</i>	33
2.2	MECHANICAL AND HYDRAULIC COMPONENTS	34
2.2.1	<i>Micro Pump</i>	34
2.2.2	<i>RS Flow switch FS3</i>	35
2.2.3	<i>Heater Aquarium H-300</i>	36
2.2.4	<i>Reservoirs and silicone tube</i>	37
2.3	3D PRINTER AND SOFTWARE ANNEXED	38
2.3.1	<i>Creality Ender 3</i>	38
2.3.2	<i>Software used</i>	40
2.3.3	<i>Models printed</i>	42
3	PROPOSED SYSTEM	47
3.1	GENERAL SCHEMATIZATION OF THE SYSTEM	47
3.2	FIRST ELECTRONIC CIRCUIT	48
3.3	FINAL CIRCUIT WITHOUT CONTROL ON HEATER AQUARIUM AND PUMP	50
3.4	CODE ANALYSIS	52
3.4.1	<i>Library and definition of variable and Object</i>	52
3.4.2	<i>Void setup</i>	53
3.4.3	<i>Void Loop</i>	53
3.5	COMPLETE SYSTEM WITHOUT CONTROL ON HEATER AQUARIUM AND PUMP	61
3.6	FINAL CIRCUIT WITH CONTROL ON HEATER AQUARIUM AND PUMP	62
3.7	DIFFERENCE AND SIMILARITY OF THE TWO TYPES OF PROJECTS	66
3.8	POSSIBLE INDUSTRIAL APPLICATION	67
3.9	POSSIBLE IMPROVEMENT	68
4	RELIABILITY OF THE SYSTEM	71
4.1	FAILURE RATE	71
4.2	RELIABILITY OF A SYSTEM	72

4.3	MEAN TIME BETWEEN FAILURES	72
4.4	APPLICATION ON THIS THESIS	73
4.4.1	<i>Water level failure analysis</i>	73
4.4.2	<i>Temperature and thermostat failure analysis</i>	74
4.4.3	<i>Flow switches</i>	75
4.4.4	<i>Micro Pump</i>	75
4.4.5	<i>3D printer</i>	75
4.5	CONCLUSION ABOUT RELIABILITY	77
5	CONCLUSION	78
6	BIBLIOGRAFIA	79

I. LIST OF FIGURES

<i>Figure 1: Industrial robot (1)</i>	9
<i>Figure 2: Types of Control system (2)</i>	10
<i>Figure 3: Step response (3)</i>	11
<i>Figure 4: Maximum Overshoot (4)</i>	12
<i>Figure 5: Rise time (5)</i>	12
<i>Figure 6: Closed Loop system</i>	13
<i>Figure 7: PID controller (6)</i>	15
<i>Figure 8: Controllino Mega (7)</i>	16
<i>Figure 9: Arduino UNO board (8)</i>	18
<i>Figure 10: ATmega328 (9)</i>	19
<i>Figure 11: Arduino UNO rev3 schematic circuit (10)</i>	20
<i>Figure 12: Prototyping shield (11)</i>	20
<i>Figure 13: band code of a resistor (12)</i>	21
<i>Figure 14: Picture of some Led similar to the 3 used in this project (13)</i>	22
<i>Figure 15: Breadboard (14)</i>	22
<i>Figure 16: jump wires (15)</i>	23
<i>Figure 17: Potentiometer (16)</i>	23
<i>Figure 18: Electronic terminal block (17)</i>	24
<i>Figure 19: jumper wires of different dimensions (18)</i>	24
<i>Figure 20 table with the function of the LCD's pins (19)</i>	25
<i>Figure 21: LCD Display with 16 pins (20)</i>	26
<i>Figure 22: scheme of connection between LCD and Arduino UNO (21)</i>	27
<i>Figure 23: Display LCD with I2C protocol (22)</i>	27
<i>Figure 24: PCF8574 (23)</i>	28
<i>Figure 25: Example connection between LCD I2C and Arduino (24)</i>	28
<i>Figure 26: DS18B20 sensor (25)</i>	29
<i>Figure 27: block diagram of the DS18B20 sensor (26)</i>	29
<i>Figure 28: water level sensor</i>	30
<i>Figure 29. transformation of the signal (27)</i>	31
<i>Figure 30: IR-Receiver (28)</i>	32
<i>Figure 31: Relay with 4 channels used in this thesis (29)</i>	33
<i>Figure 32:Decdeal 600 L/H (30)</i>	34

Figure 33: Flow Switch (31) 35

Figure 34: FS3 Flow switch (32) 35

Figure 35: Aquarium Heater (33) 36

Figure 36: one of the 3 reservoirs used 37

Figure 37: a flexible silicone tub (34) 37

Figure 38: Ender 3 printer (35) 38

Figure 39: example of SLA printing (36) 39

Figure 40: FFF printing example (37) 40

Figure 41: SolidWorks interface 41

Figure 42: Creality Slicer interface 42

Figure 43: 3D printing procedures 42

Figure 44: First design of the Arduino case 43

Figure 45: LCD case 44

Figure 46: A SolidWorks view of the Arduino Case 44

Figure 47: Creality Slicer view of the Arduino board 45

Figure 48: Printer while printing the case with the parameters 45

Figure 49: nozzles printed by me 46

Figure 50: scheme of the system 47

Figure 51: First Prototype 49

Figure 52: Micropump RS400 180 49

Figure 53: Final Circuit with all the sensor 50

Figure 54: library and definition of some variable 52

Figure 55: Void Setup of the code 53

Figure 56: mapping of the level sensor and main menu 53

Figure 57: Void Loop, warning part 54

Figure 58: Case condition 1 55

Figure 59: Switch case 2 55

Figure 60: third switch case condition and default condition 56

Figure 61: Main menu 57

Figure 62: Water Level 57

Figure 63: Water Temperature 58

Figure 64: Warning mode 58

Figure 65: System in pause 59

Figure 66: Arduino Case..... 60

Figure 67: Complete system..... 61

Figure 68: electrical socket (38)..... 62

Figure 69: Scheme..... 62

Figure 70: Relay connection..... 63

Figure 71: Definition of the variable 64

Figure 72: Void setup of the control configuration 64

Figure 73: if conditions of low temperatures and level 65

Figure 74: If conditions of high temperature and level 65

Figure 75: Matlab code..... 68

Figure 76: Trend of Temperature over Time 69

Figure 77: Arduino code..... 69

Figure 78 Bathtub curve of the failure rate (39)..... 71

Figure 79: Code the calibrate the sensor..... 73

Figure 80: Serial Monitor..... 74

II. LIST OF ACRONYMS

1. CPU: Central Processing Unit
2. DC Direct Current
3. F: Service life
4. f: Probability density of failure
5. FFF: Fused filament fabrication
6. FIT (or λ): Failure rate (Failure per Time)
7. IP: Internet Protocol
8. IR: Infrared
9. LD: Ladder Diagram
10. LED: Light Emitting Diode
11. MTBF: Mean Time Between Failure
12. MTTF: Mean Time To Failure
13. MTTR: Mean Time To Repair
14. NC: Normally Closed
15. NO: Normally Open
16. PD: Proportional and Derivative
17. PI: Proportional Integral
18. PID: Proportional Integral Derivative
19. PLA: Polylactic acid
20. PLC: Programmable Logic Control
21. PWM: Pulse Width Modulation
22. R: Probability of survival
23. RAM: Random Access Memory
24. ROM: Read-Only Memory
25. RTD: Resistive Temperature Detector
26. SLA: stereolithography apparatus
27. T: Temperature

1 INTRODUCTION

1.1 AIM AND STRUCTURE OF THE THESIS

The objective of this thesis is to develop a chemical test bench, using many components, and verify the reliability of this system. This project will be use by students enrolled in Chemical Engineering to develop some didactical experiments. The principal components used to develop this testbench are sensors (used to measure the water level and the temperature), a micropumps, a microcontroller, a flow switch, reservoirs, LED and resistors, IR receiver and lastly a heater for aquarium (used to heat the water contained in the reservoir). After a first part of design, I create then the physical model: a chemical mixing experimental setup that is controlled by Arduino Uno. During the test I required some component, and I designed it by myself using a 3D printer, more specifically I printed some nozzles and an Arduino case. At the end of the thesis, I did some quickly test on it to verify the reliability. All the code written on the IDE of Arduino and some sketch on SolidWorks are shown during the test. This thesis is structured in the following way:

- A general overview of the system and its application: in this part will be analyzed in general the control system used in industrial application and in particular this system.
- An overview of the components used in this thesis.
- The design part and the creation of the system: there will be show the codes, the functioning of the system.
- A theoretical part concerning the concept of reliability and some test.

In the paragraph concerning the design part I'll show a first prototype of the system and I'll explain why I didn't continue that way. Later, I'll focus on the final model illustrates how works all the parts and why I selected some components instead of other. This final model is divided in two, one without the control part of the pump and the heater and ones with the control of these two systems with Arduino by means of a relay (but more complicated).

1.2 GENERAL OVERVIEW OF INDUSTRIAL CONTROL SYSTEM

We can describe the automation as the technology that reduce human intervention in industrial application. In the last year, the number of industrial robots increased as well as all the techniques that helps the human in their job. Nowadays with the fourth industrial revolution, the iteration between human and machines increased and by means of internet of things the communication between machine and computer. Now the robot can transfer the data directly to an ERP software (by means a PLC) and the data can be analyzed directly from computer, for example using a query between the database containing the acquired data and the software used to analyze it (for example Excel). The robots are used more specifically in all the application where the type of job can be dangerous for people as for example painting (where there are toxic fumes) or where it is necessary to have big reliability (for example in welding where a robot is more fast and more precise than humans).

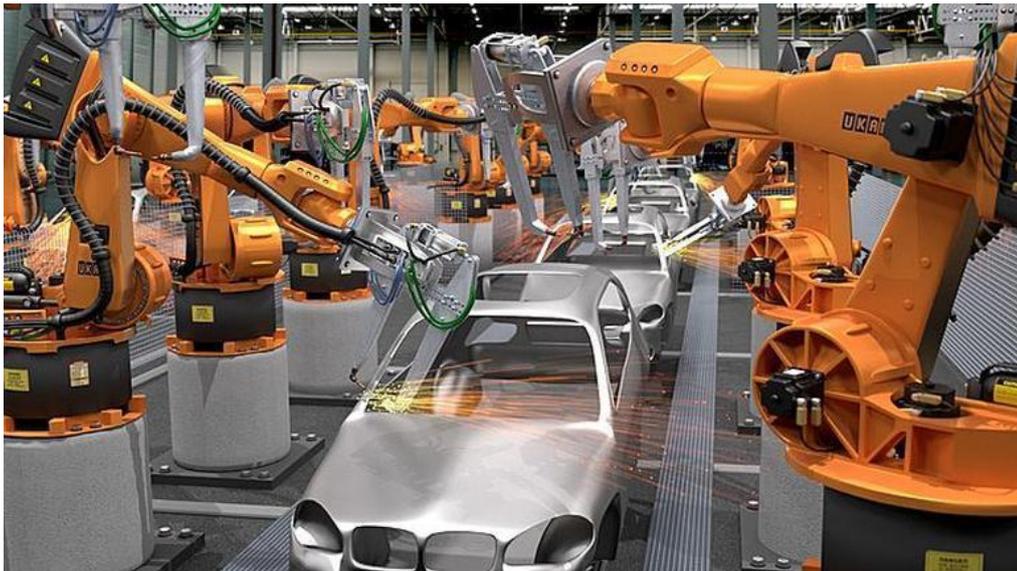
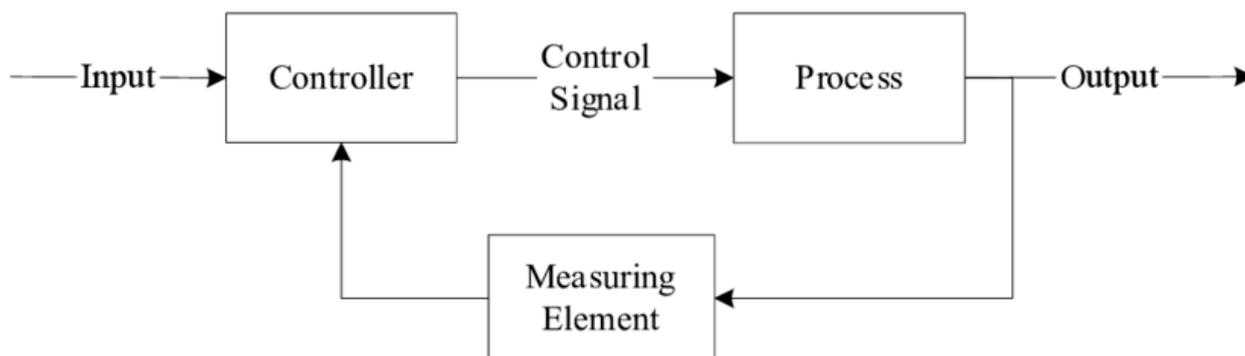


Figure 1: Industrial robot (1)

In order to develop a model capable to give drive our system on required state minimizing delay, overshoot and steady-state error we need a controller. The controller monitors the process variable comparing it with a set point. The difference between these two variables is the error
We can subdivide in 2 macro category the types of control system:



Open Loop System



Closed Loop System

Figure 2: Types of Control system (2)

- **Open-loop** controller is the simplest type of controller because it does not have any sensor and so the controller doesn't know the output variable and if there is any type of disturbances. By means of a mathematical model, once we know how the desired output is, we can create a controller that give to the system a certain type of input in order to reach the desired output. For example, if we want to heat something but we don't have a thermostat we will not know when to switch off (we know that starting from 10 °C in 10 minutes we reach the needed temperature of 15°C, but if someone place a cube of ice in the system it will not reach the desired temperature, because after 10 minutes the system turn off and the temperature will be less than 15 °C). This type of controller is the simplest ones because it doesn't have any type of sensor, so it is light, cheaper and it is more reliable. As disadvantages it is not robust because an interference will vary the output.
- **Closed-loop** controller (or feedback controller): in this case there are sensor that measure the output and sends the data to the controller. The difference between this signal and control signal (that is the output of the controller but also the input of the system) is the error. The aim is to reduce that error because smaller is the error and better is the controller. There are widely used sensor (that measure system's variable), transducers

(that converts some variables into another, the sensor is a type of transducer) a controller (something, like a piston that mechanically move something or close a flow switches ecc. More general a close loop controller measures some Process Variable PV (for example temperature) and compare it with the Set Point SP (the target value, for example 15 °C). There are many types of Closed Loop controller, as for example the On-Off Controller (one of the simplest one) but also the PID (undoubtedly the most adopted in industrial application).

1.2.1.1.1 Step Response

Given as input of the controller a step response there are some important parameters to measure.

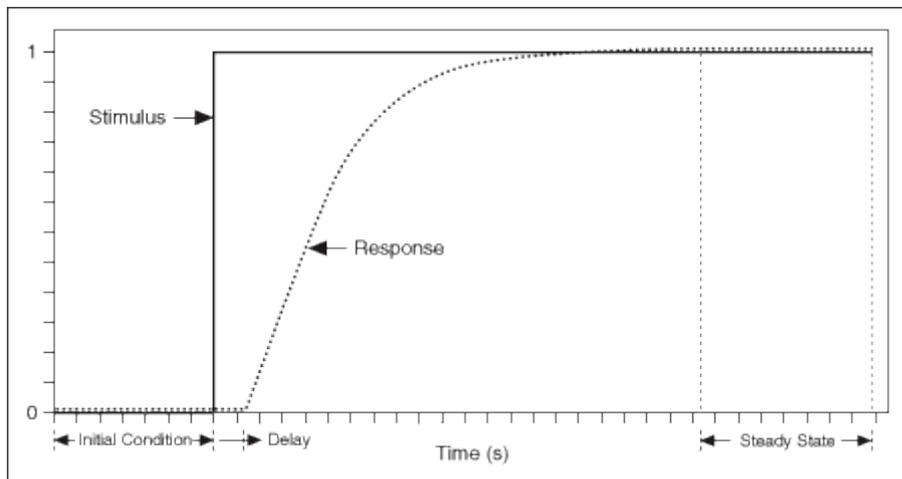


Figure 3: Step response (3)

In the upper picture we can see the input step stimulus and the output response. First of all, as we can see in image, starting from some initial condition, once we apply the step input the system response comes with a certain delay (not immediately) and also the output convergences to our step input (remembering that the input is also our set point, i.e. what we want to obtain) only after a certain time, then we will have the Steady State. There is other two important variable:

- **Maximum overshoot:** is a parameter that tell us how much our response varies from 1, and it is important to have a small maximum overshoot:

$$s_{hat} = \max\left(\frac{y(t)}{y_{sp}}\right) - 1$$

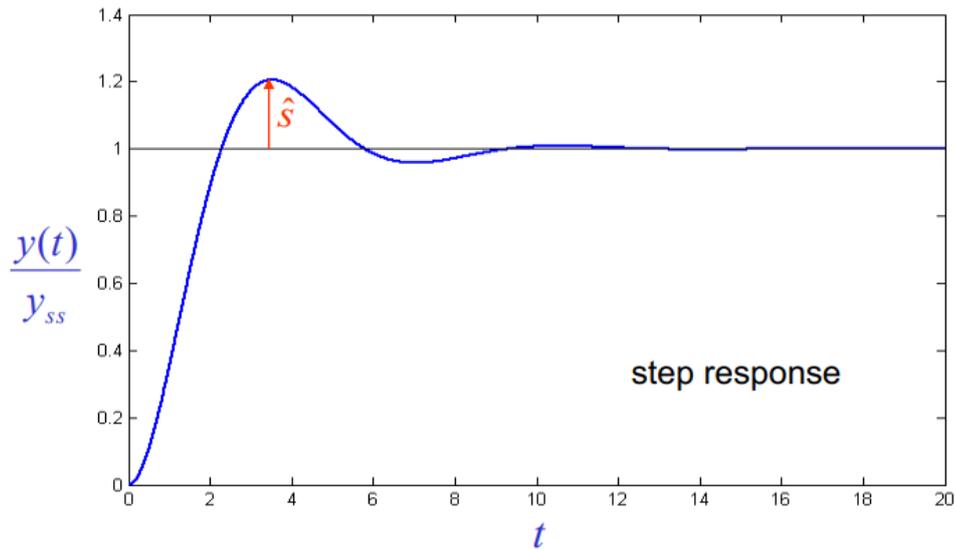


Figure 4: Maximum Overshoot (4)

- **Rise time:** is the time necessary to obtain a certain percentage of the steady state value, in most cases is about 90%. Less is the rise time and better is the control system. It is defined as:

$$t = \frac{y(t)}{y_{sp}} = 0.9 = 90\%$$

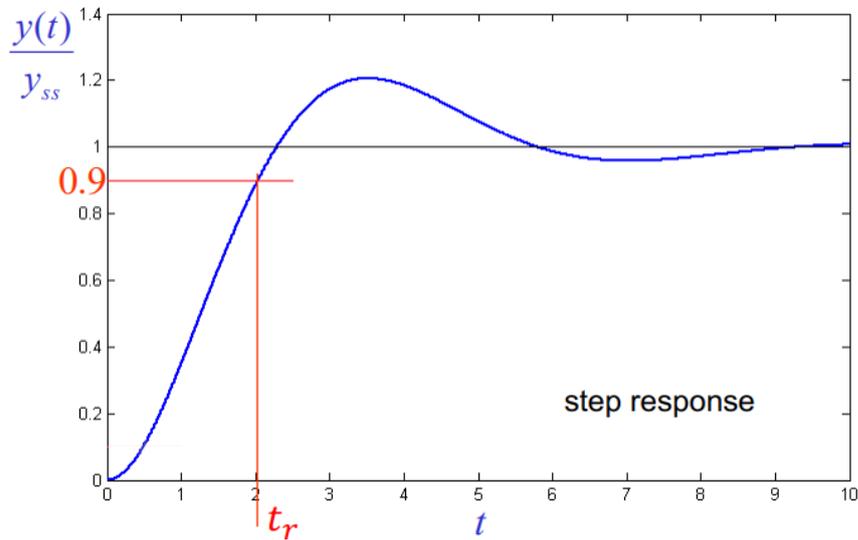


Figure 5: Rise time (5)

1.3 PID CONTROLLER

The widely used type of closed loop controller is the PID controller. There are simple, reliable, efficient, and also cheaper than other solution because they have the best rapport cost/efficiency. It is constituted by a proportional action (P) an integral action (I) and a derivate action (D). There are many combinations of these 3 actions:

- One mode: using only P action or I action.
- Two modes: using PI or PD.
- Three modes: using all 3 obtaining a PID controller.

Starting from the following closed loop system (draw by me):

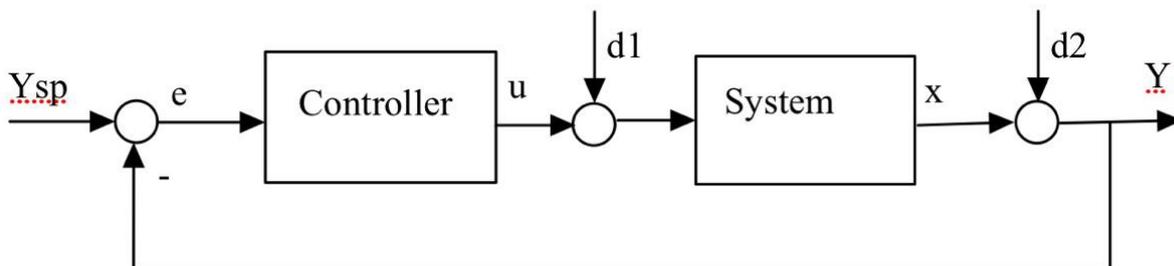


Figure 6: Closed Loop system

Where Y_{sp} is the set point (a step function) entering in our system, e is the error (the difference between the set point variable and Y) and we want to minimize it, u is the control variable (as output of our controller) and then we have $d1$ (an external disturbances modeled as a step function), x is the controlled variable (for example the temperature) and summing up the disturbances $d2$ (due to the sensor) give us the measured variable Y (i.e. the measured variable Y is not the correct controlled variable because it is affected by a disturbances error). Our goal is to converge Y to Y_{sp} and have an error close to zero. As an example, if we want to have $20\text{ }^{\circ}\text{C}$, it will be our Y_{sp} variable, if we measure it with a cheap thermometer there will be an error and we measure a temperature different from the real one $Y=x+d2$. The difference between Y and Y_{sp} is the error that is an input of our controller and as output t activate the heater, the warm entering in our system modifies its temperature and x will change. If we add some ice, there will be our disturbances $d2$. We have:

- **Proportional action:** given a gain K_c we can link together $u(t)$ and $e(t)$ proportionally:

$$u(t) = K_c \cdot e(t)$$

- **Integral action:** it is used because ensures a zero error during the operation. It is equal to:

$$u(t) = \frac{K_c}{T_1} \int_0^t e(r) dr + u(0)$$

The integral action gives a phase displacement of 90° . A controller PI is useful because the integral action combined with the proportional one gives an error equal to zero during steady operation.

- **Derivative action:** it is the opposite of the integral action an anticipation of 0° on the phase:

$$u(t) = K_c \cdot T_d \frac{d_e(t)}{dt}$$

The derivative action is not used alone because the control system will not be low pass filter anymore.

Summing up the 3 different action (proportional, derivative and integrative) we obtain:

$$u(t) = K_c \cdot \left(e(t) + \frac{1}{T_1} \int_0^t e(r) dr + T_d \frac{d_e(t)}{dt} \right)$$

Where:

- K_c is the gain (also called proportional gain).
- T_1 is the integrative time (also called reset time).
- T_d in the derivative time (also called rate time).

The proportional, integrative, and derivative action give to our plant three different transfer function (using the Laplace Transform where s in the Laplace's variable):

Proportional action's transfer function:

$$P(s) = K_c$$

Integrative action's transfer function:

$$I(s) = \frac{K_c}{s \cdot T_I}$$

Derivative action's transfer function:

$$P(s) = \frac{K_c \cdot T_D \cdot s}{1 + s \cdot \frac{T_D}{N}}$$

With, in general, N between 5 and 10. Using the Laplace transform we have the following transfer function of our PID controller:

$$PID(s) = \frac{U(s)}{E(s)} = K_c \left(1 + \frac{1}{s \cdot T_I} + s \cdot T_D \right)$$

The final scheme of the controller that we obtain is the following one:

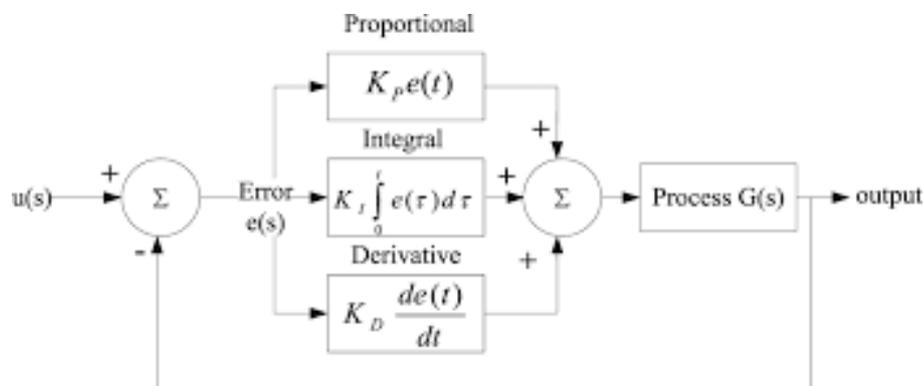


Figure 7: PID controller (6)

1.4 PLC

PLC is a Programmable Logic Controller widely used to control many different types of industrial process. It consists in a digital device that memorize information and elaborates some instruction doing some specific function. There was introduced in 1960's and was used right away because they are programmable, reusable and reliable. The PLC can also be a PID controller, and the PLC

are based on microcontroller, ordinary microprocessor, Digital Signal Processor (DSP) or Field Programmable Gate Array (FPGA). An example of PLC is Controllino MEGA.



Figure 8: Controllino Mega (7)

Since the PLC are more reliable and Robust than a simple board, there are used in industrial application. The concept of reliability is presented in the final chapter, without dwelling too much: the Controllino mega was not used in this thesis because the idea is to have something simple to program (in C or C++) and portable. It is true that Controllino mega is more robust, but since this project is not used in industrial application where the safety is important, the Arduino UNO is good. In fact, as can be seen in the Reliability's paragraph, the problem is not the reliability of the Arduino's board, but of the sensor and since the sensor used are the same the reliability of the complex system, in this application, is almost the same.

2 DEFINITION OF THE COMPONENTS OF THE SYSTEM

2.1 ELECTRICAL COMPONENTS

2.1.1 Arduino UNO

The main component is the board Eleego UNO, that use the same hardware of the board Arduino Uno. Arduino UNO is an open-source electronic device developed by Arduino that use a microcontroller named Atmega328p. This board has 14 digital input-output pins used to interface with circuits or other boards. It also has other 6 analog input-output pins: the analog input pins are used to measure the voltage between 0V and 5V with a precision of 10 bit (i.e. the resolution of the digital measurements of this voltage, in our case 10 bits means a subdivision in 1024 values, in other words a resolution of 4.88mV) used to connect other device as for example potentiometer, sensor of temperature ecc. Moreover, it can also be powered by a 9V battery. It also has some pin used for alimentation (at 5V or 3.3V) and ground. This board can be programmable by means of the Arduino IDE via an USB cable that ca be also be used to power the board. The IDE of Arduino consist in an editor that includes the “syntax highlighting” (a text editor for programming that highlights the text in different colors and fonts based on some syntax rules, as for example the `#include` of libraries in yellow). This Ide does not need to run the program from command line interface, but once the codes is finished it can simply be putted on the board: this can be an advantage because make the prototyping phase easier and faster and the board can run the code without the computer using only a battery, but also a disadvantage because can be ran only a code per time. There is included a library software named Wiring that simplify the input/output operations. In order to create a working file, we need to use 2 functions:

- Void `setup()`: this function is called only once at the start of the program, and it can be used in order to setup something that will not change while the software is running;
- Void `loop()`: this function is repeated continuously (we can use inside this function the function `delay` choosing how many ms repeat the action);

Definition of the components of the system

Turning back on the board: below a picture of Arduino Uno:

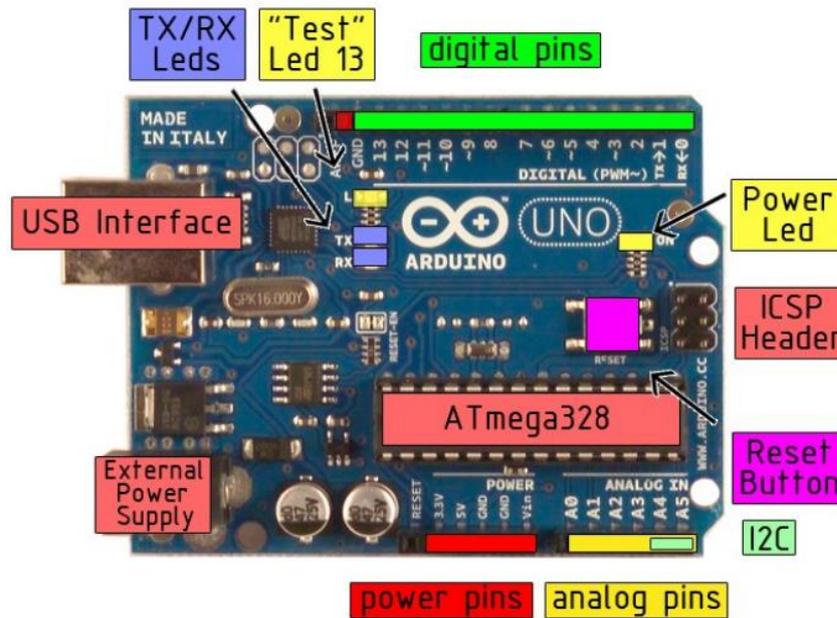


Figure 9: Arduino UNO board (8)

- In Red we have the Power pins, in yellow the analog pins: we can use both the 5V and 3.3V to provide power to modules connected to Arduino, or we can connect a battery to Arduino (in the “External power supply” when the power is not supplied by the USB interface) to provide a voltage till 12V by means the Vin pin. Lastly, we have the GND (ground) pins.
- In yellow we have the Analog pins: those pins can be only used as input, and it can have 1024 difference values (values between the range 0 and 1023) and they read values of Voltage between 0V and 5V and these values can be read 10000 per seconds. Pins A4 and A5 are I2C (Inter Integrated Circuits): they use a serial communication with a Slave and a Master.
- In purple we have the reset button: it can be used to restart the program (it is the same as unplugging and plugging back the boards). When the button is pushed, the led/pin 13 should flash a couple of times.
- In blue the RX e TX pins: two communication system and it is better to doesn't connect anything, because the communication between the computer and the board are coming through these pins.
- In green the digital pins: the board through these digital pins can receive signal as input and can also send information as output. The digital pins can manage only 0 or 1 signals (in few words low signal or high signal) and these pins can be used only when there are 2

Definition of the components of the system

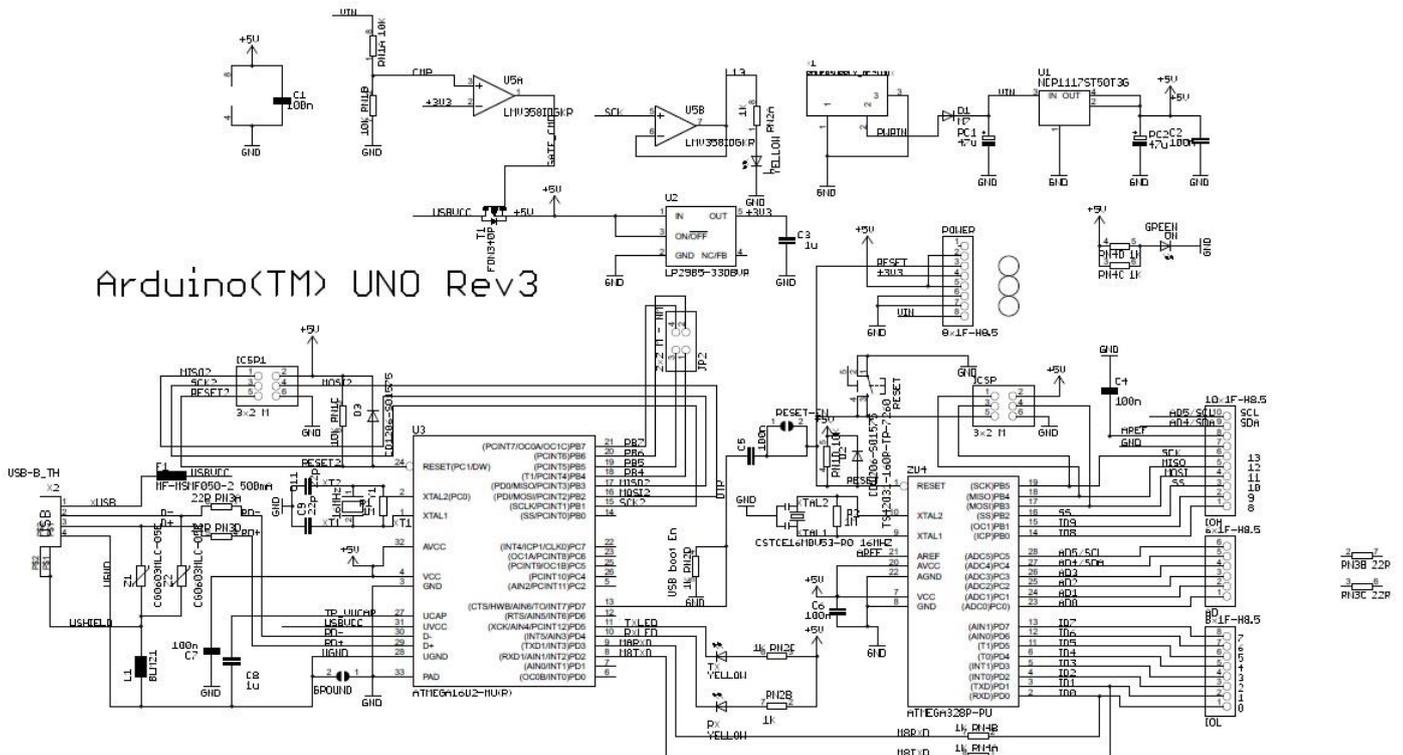
states. In order to avoid these situations, there are also digital pins equipped with PWM (pulse width modulation): these pins are 3,5,6,9,10 and 1. They can send or receive values between 0 and 255 and they can behave as an analog pin. By means of that we can use it to receive IR signal.

About the specific of the board Arduino UNO, has said previously is formed by the ATmega328: an 8-bit Microchip based on RISC design. It has 32 KB flash memory (electronic nonvolatile memory that can be erased and reprogrammed), 1 KB of EEPROM (a memory electrically erasable and programmable read only, no volatile with small erase blocks) and 2 KB SRAM (a volatile type of RAM that can maintain the data for a time theoretically infinite with low time of read and low consumption of power). The Atmega328 is a single-chip microcontroller created by Atmel in 2016 that can work at a maximum operating frequency of 20 MHz and 32 general purpose working register (the register is an accessible location quickly accessible).



Figure 10: ATmega328 (9)

Below we have the entire schematic circuit of the Arduino UNO board:



Definition of the components of the system

Figure 11: Arduino UNO rev3 schematic circuit (10)

Even if this scheme is from the third version of Arduino UNO (rev3), the scheme is not so different from past version and from the Elegoo UNO used by me and since Arduino UNO is open source the hardware and software part is almost the same.

2.1.2 Shield for Arduino

Arduino is widely used for rapid prototyping, and the shields are some adjunctive elements that are connected directly in the upper part of Arduino giving to it some implementation, for example a Wi-Fi connection, Bluetooth connection ecc. Every shield has a specific function that can be used simply connecting it on the board. Some shields are compatible each other and they allow us to connect one upper the others creating a column with many shields and many new options. The shield used in this thesis is the **Prototyping Shield**. It consists in a PCB board with many holes (used for example to weld part of circuit on it), and I connected a mini breadboard on it. Thanks, of that I had less cable, and all the circuit is simpler.

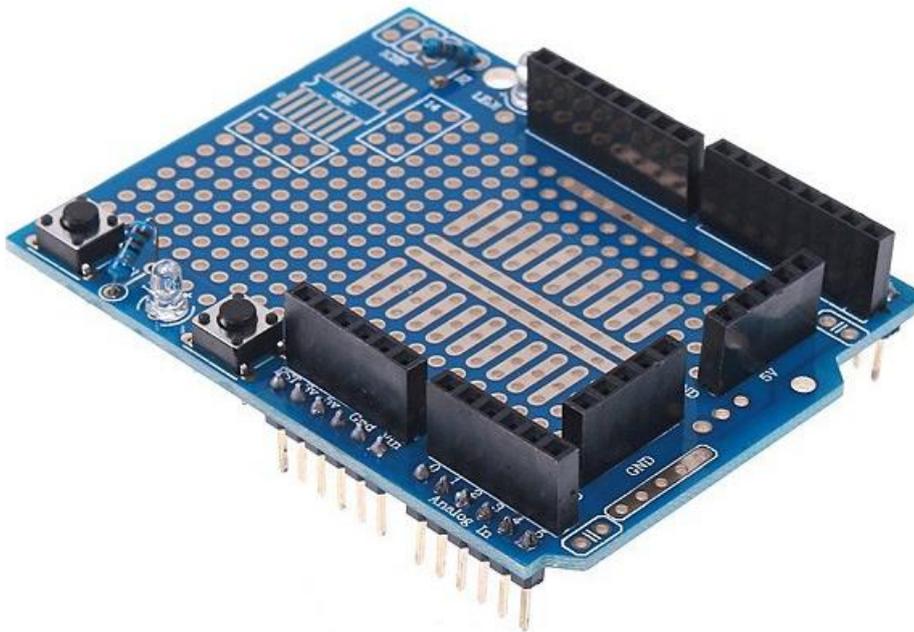


Figure 12: Prototyping shield (11)

In the first prototype I did not use this shield and I used a simple breadboard: as consequence there was too many wires, and all was too complicated.

In this thesis I will use other components showed in the next paragraph.

2.1.3 Wires, resistors, and other components

The other components used for this project are:

- Resistors: used to decrease the values of the currents flowing through our sensors. I use resistors with values of resistance between the range 500-1000Ω. The value of the resistor is chosen according of the “band code” according to the following picture:

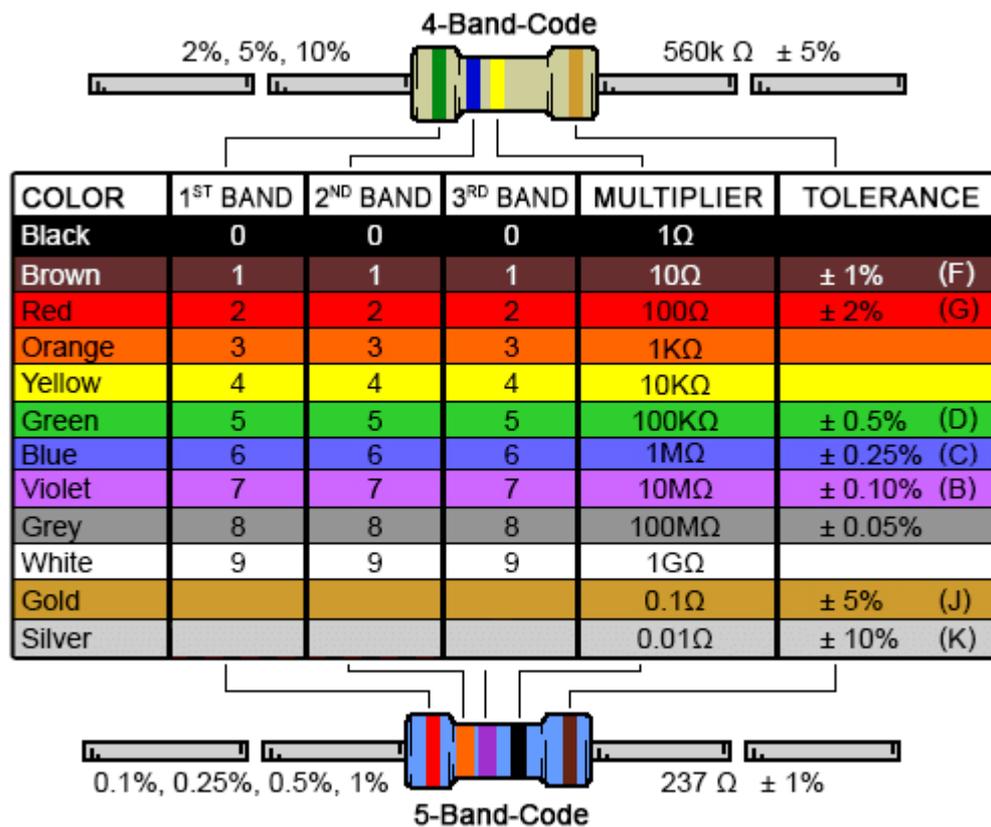


Figure 13: band code of a resistor (12)

The resistor can have more 4,5 or 6 band: we need to multiply the different values given by the corresponding color: 2 (red) 3 (orange) 7 (violet) *1 (black) with a tolerance of 1% (brown). The resistor follows the Ohm’s rule:

$$V = R \cdot I$$

Where r is the resistor, I is the current and V is the electric potential difference.

- Led: used only to give us a look about the situation: of the system is in pause the led is blue, if it is working is green, if it is red the level of water is higher than the highest level

Definition of the components of the system

accepted. The led cannot be connected directly the source of energy at 5V because the led will burn, so we need to use a resistor with a value of resistance equal to

$$R = \frac{V_{tot} - V_{led}}{I_{LED}} = \frac{5 - 2}{0.02} = 150 \Omega$$

Where V_{tot} is the total voltage as output from Arduino (in our case 5V), V_{led} is the voltage across our LED and I_{LED} is the maximum current that can flow inside the LED.

In general, without a datasheet the V_{led} is circa 2V and the maximum current that it can have without breaking it is about 20-40 mA. I choose a resistor of resistance equal 500 Ω in order to feel safer.



Figure 14: Picture of some Led similar to the 3 used in this project (13)

- Breadboard: it is the faster things that can be used in rapid prototyping, because the alternative is the “Stripboard” but on it the components can be only welded, and the connections are permanent.

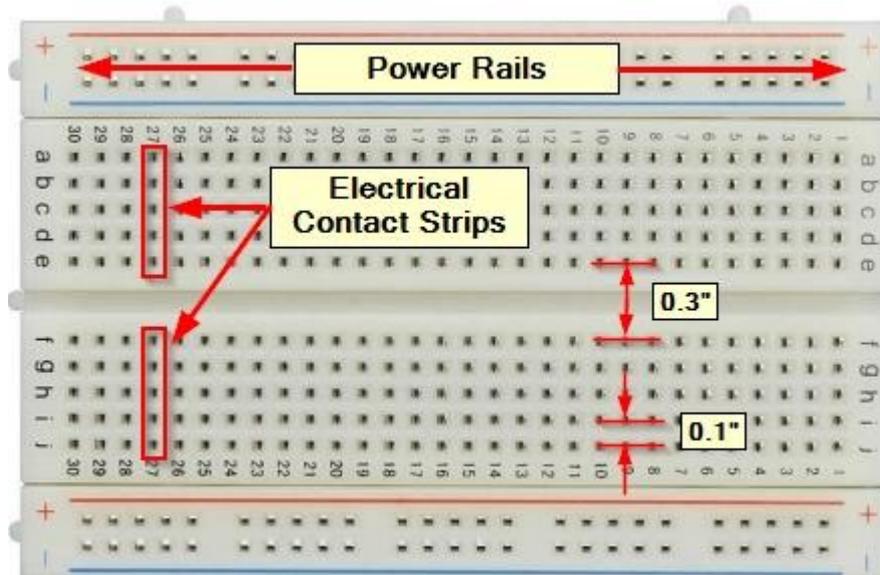


Figure 15: Breadboard (14)

The breadboard is a piece of plastic with a lot of holes where can be plugged various components as for example resistor or cable. These holes are connected electrically:

Definition of the components of the system

The power rails (the voltage in red and the ground in blue) are connected horizontally, the Electrical Contact Strips are connected vertically (as for example all the holes on number 27) but the 2 parts of the Breadboard (the upper and below ones) are not connected.

- Jump wire: it is an electrical wire used to interconnect various components among them or with the Breadboard. It can be of various type: in this project I use the Solid tips that can be F-F, M-F, M-M.



Figure 16: jump wires (15)

- Potentiometer: it is a component having three terminals. It works as a voltage divider and rotating a part similar to a joystick it can vary its resistance and as consequence the voltage. It behaves as a couple of resistors having a total resistance constant, but taken individually their values can change, increasing or decreasing.

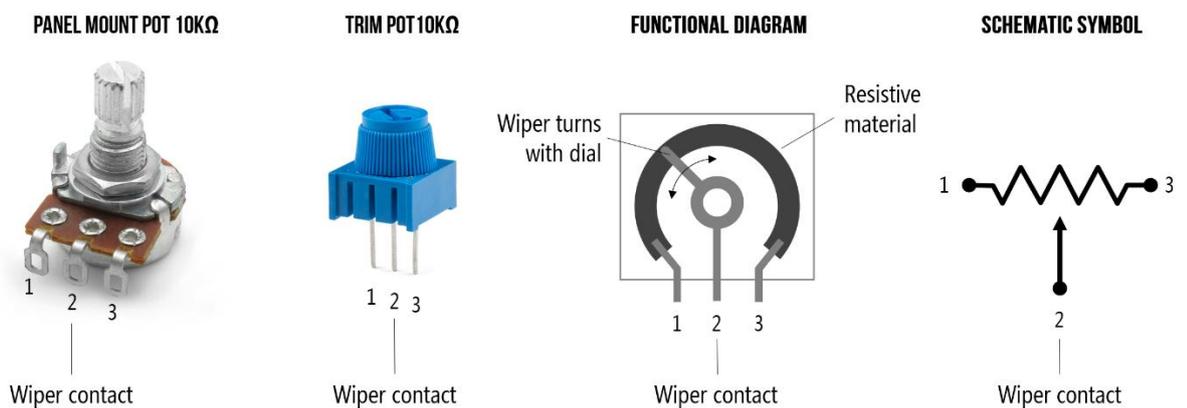


Figure 17: Potentiometer (16)

Rotating the joystick, the length of the semi circumference U_i increase and decrease and as consequence also the resistance. The potentiometer will be necessary used with the LCD.

Definition of the components of the system

- Electronic terminal block: I used it to connect the wires with the jumper F-F or F-M because the wires cannot be connected directly to the breadboard. They are also used in the second project where the temperature and the level of the water are directly controlled by the board: since the heater aquarium and the pump must be connected directly to the socket and to the relay, I use these terminal blocks.



Figure 18: Electronic terminal block (17)

- Jumper of many measures: since the connection between mini breadboard on the prototyping shield and Arduino is very close, I preferred to use some different jumper instead of the classic jump wires (saw before) for some type of connection without tangling the wires too much. This type of connector is not necessary but simplify the system.

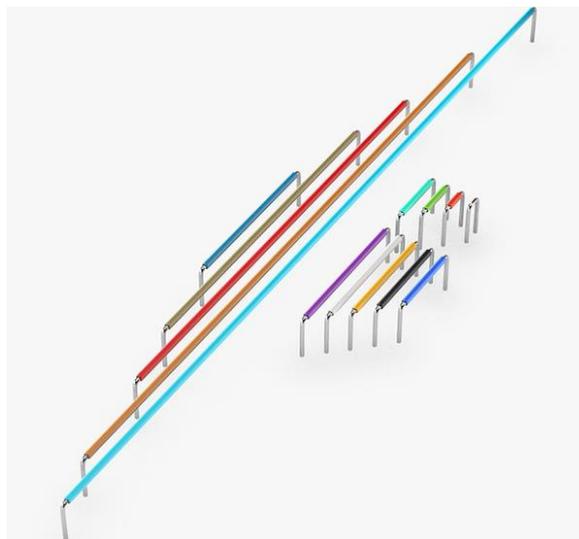


Figure 19: jumper wires of different dimensions (18)

2.1.4 LCD Display

2.1.4.1 Standard LCD Display

In order to display some information, we need an LCD display connected to our Arduino board. The Display used during the first prototype of this project is an LCD 16x2 (two rows x 16 bit displayed, so at maximum I can display only 32 character). This LCD is compatible with the HITACHI driver HD44780 (an application specific integrated circuit used to control Display LCD used only to visualize character). In order to control this display, we need the Library LiquidCrystal.h that allows us to communicate by means 4 or 8 bits. This LCD use 16 pins (there is another one that use the I2C technology and have only 4 pins, but it is not used in this project) that are resumed in the following table:

PIN NO	Symbol	Fuction
1	VSS	GND
2	VDD	+5V
3	V0	Contrast adjustment
4	RS	H/L Register select signal
5	R/W	H/L Read/Write signal
6	E	H/L Enable signal
7	DB0	H/L Data bus line
8	DB1	H/L Data bus line
9	DB2	H/L Data bus line
10	DB3	H/L Data bus line
11	DB4	H/L Data bus line
12	DB5	H/L Data bus line
13	DB6	H/L Data bus line
14	DB7	H/L Data bus line
15	A	+4.2V for LED
16	K	Power supply for BKL(0V)

Figure 20 table with the function of the LCD's pins (19)

Definition of the components of the system

The pins are connected according to:

- Pin 1: connected to the ground (i.e. to the GND pin of Arduino);
- Pin 2: connected to the power supply (i.e. the 5V of Arduino);
- Pin 3: this pin is connected to the potentiometer in order to increase/decrease the contrast of the display because by means of the potentiometer we can vary the voltage applied on pin 3 between 0V and 5V and as consequence the contrast vary.
- Pin 4: it is used to choose the register: if it is equal to 0, we have a command, if it is equal to 1, we have Data.
- Pin 5: it is used to choose the signal read or write: if 0 is write, if 1 is read.
- Pin 6: to allow the writing on register (the register is a small part of memory used to make faster the execution of software, because it provides a faster access to the values used more frequently);
- From Pin 7 to 14 we have the communication (in the case of 4-bit communication the pins 7,8,9 and 10 are no used, but I'll use that because I need a 8 bit operation) and these will be connected to the digital pin;
- Pin 15 and Pin 16: are not necessary in my case, there are respectively the Anode and the Cathode and are used for backlight.

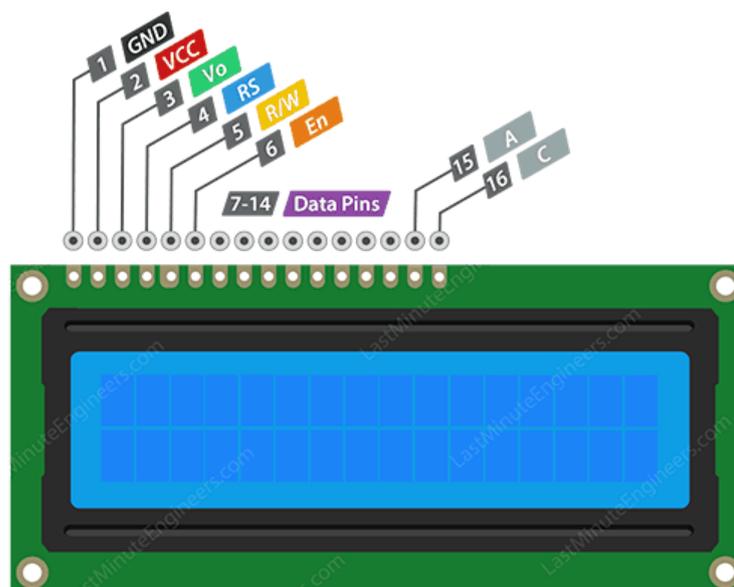


Figure 21: LCD Display with 16 pins (20)

Definition of the components of the system

In the following picture an image as example of how can be connected the LCD to the board. The resistor with the arrow is the potentiometer. The problem with this configuration is that there are many wires and the probability of commit some mistake is very high. Furthermore, if some wires are disconnected from breadboard, many times will be wasted to re-connect it.

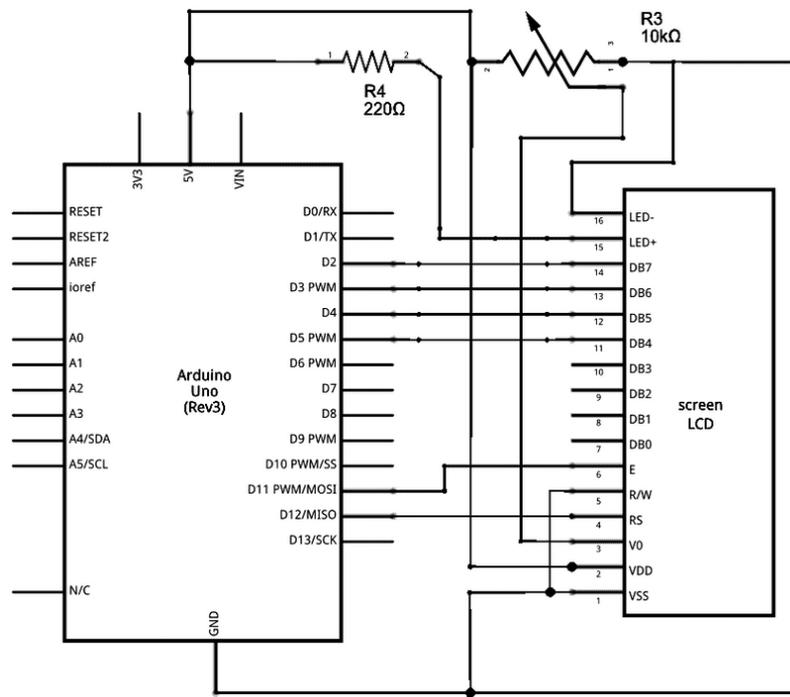


Figure 22: scheme of connection between LCD and Arduino UNO (21)

2.1.4.2 LCD Display with I2C protocol

There is an alternative: using an LCD I2C Display. This type of LCD has an I2C protocol, and it need the 2 pins of Arduino saw before. The good thing about this type of LCD is that we need only 4 wires: 2 will be connected to the 2 pins of Arduino that use the I2C protocol, one will be the alimentation at 5V and the other one is the ground.



Figure 23: Display LCD with I2C protocol (22)

Definition of the components of the system

The display used by me has a part behind it welded by constructor where there are present the pins and also a blue part used to regulate the contrast of the LCD. This part is the PCF8574: a CMOS circuit that provides 8-bit Input-Output port that use the I2C protocol.



Figure 24: PCF8574 (23)

The advantages of using this type of protocol are:

- It uses only 2 lines for transmission, and there will be only 2 pins of Arduino used (with the other LD the pin use was 16 and as consequence to many spaces used. There are obviously need the GND and the Vd (at 3.3V or 5V) pins.
- It can be used by everyone, so many board-constructor implement it.

Below an example of the type of connection and how simpler is the one with I2C protocol.

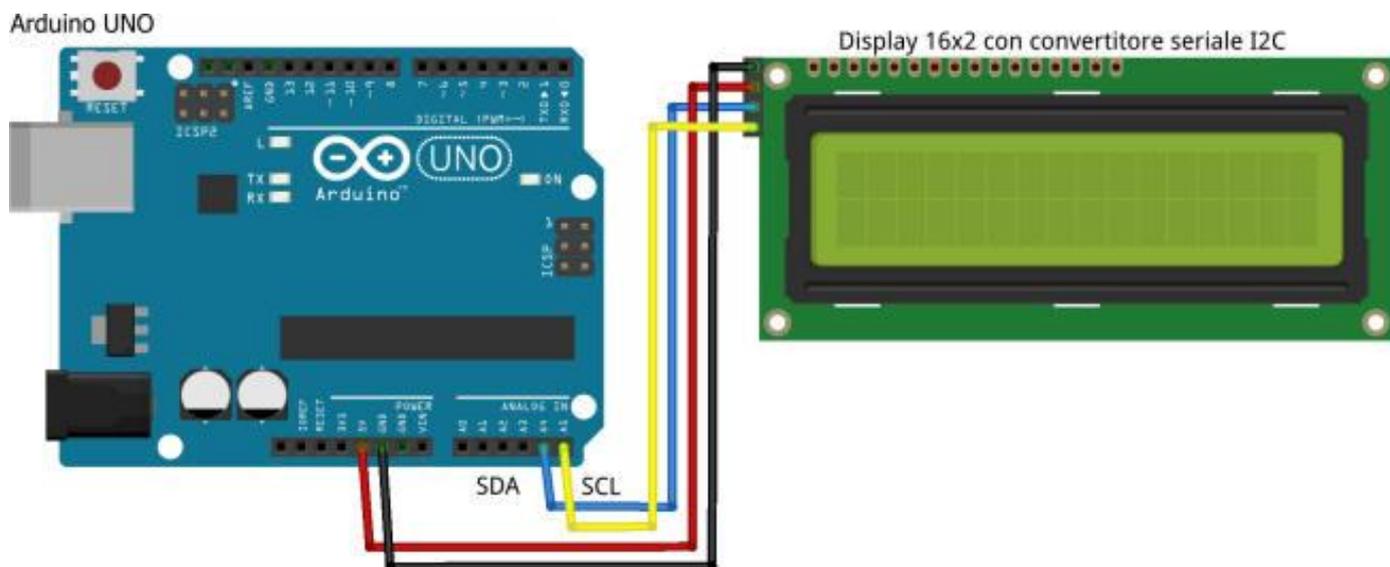


Figure 25: Example connection between LCD I2C and Arduino (24)

2.1.5 Chip DS18B20

It is a waterproof water sensor. It can detect temperature between $-55\text{ }^{\circ}\text{C}$ and $+125\text{ }^{\circ}\text{C}$ with an error of $0.5\text{ }^{\circ}\text{C}$. It has a length of 1m.



Figure 26: DS18B20 sensor (25)

Below we have a block diagram of the DS18B20 sensor.

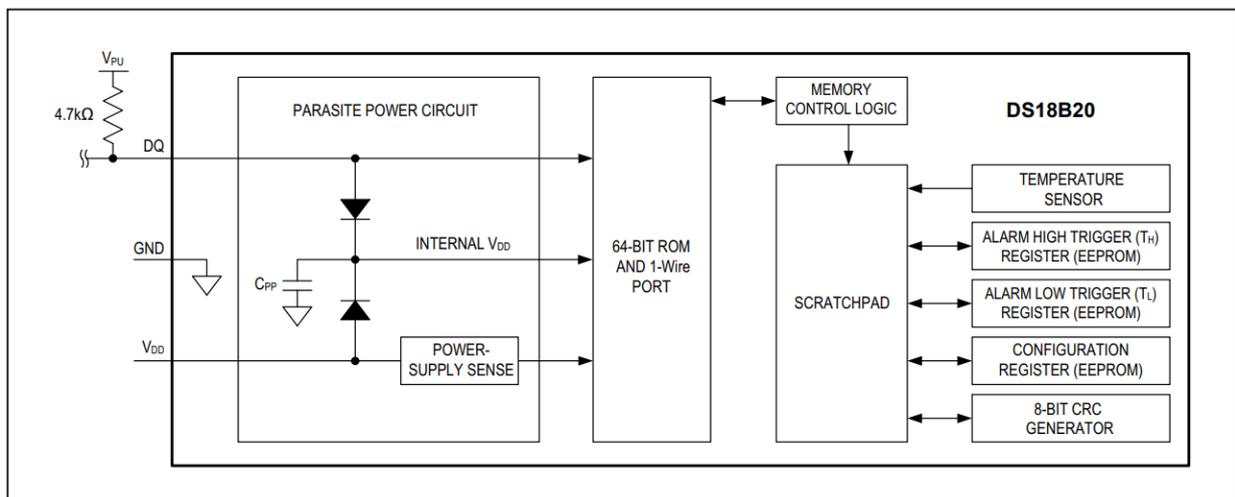


Figure 27: block diagram of the DS18B20 sensor (26)

As we can see is better to add a resistor with a resistance of $4,7\text{ k}\Omega$. In the 64-bit ROM is stored the serial code of the sensor. The temperature measured digitally is stored in the “scratchpad memory”. There are also 2 EEPROM are 2 no volatile memory that stores the temperature measured also if the power turns off. The power enter inside the sensor through the resistor by means the 1-Wire bus (referred as parasite power). Since it is waterproof it will be putted inside the middle reservoir and will measures the temperature that will be displayed and it will be

Definition of the components of the system

compared with the heater aquarium (that have a thermostat inside it, and when it reaches the required temperature it turn-off).

2.1.6 Sensor level water

This sensor is used to measure the level of a liquid and send this information to Arduino.

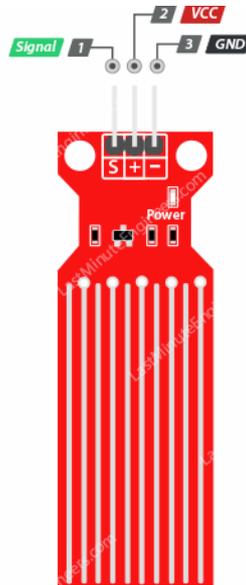


Figure 28: water level sensor

The green pin of the signal must be connected to one of the various analog inputs of Arduino, the red pin must be connected to the power supply with an input voltage in the range 3.3V-5V and lastly the black pin must be connected to the GND of Arduino. Focusing on the sensor we have some traces: one of copper and one of power are alternate respectively and there are not connected when dry, but only when wet. These traces /conductors act a variable resistor whose resistance varies together with the variation of the water level: the resistance is inversely proportional to the height of the water; this is due to the fact that more water result in a better conductivity and that give us a lower resistance. It is very important to calibrate the sensor before use it, as shown in the chapter about reliability. This sensor is used in this thesis to measure the water level of the first tank, and if the level is bigger than 90% it warns the user (in one version) or directly activate the pump (in the other version). As shown in the final chapter, the reliability of this component is very small.

Definition of the components of the system

2.1.7 IR Sensor PNA4602N

First of all, we can talk about the IR LED: a special type of LED made by semiconductor material that emit infrared radiation don't visible by our eyes because it has a wavelength's of 1mm-700mm, and this infrared is captured by an IR receiver. The IR LED is mounted on the remote which send the signal through the IR LED.

Infrared is a type o light so I can't be pass through stuff as for example a wall: it needs a direct line between the LED and the IR receiver. Since IR is emitted by sun or other source of light there will be a lot of noise and our Remote will covert a binary signal (given from the button respectively pushed) and transform it into an electrical signal and then this electrical signal will be converted in light signal by the led. This light will be converted back in binary from the receiver and then passed to our microcontroller. It can avoid the natural noise focusing on 38 kHz (the same frequency sent from the remote) using a band-pass filter, because it is different from the signal from other source (as for example environmental).

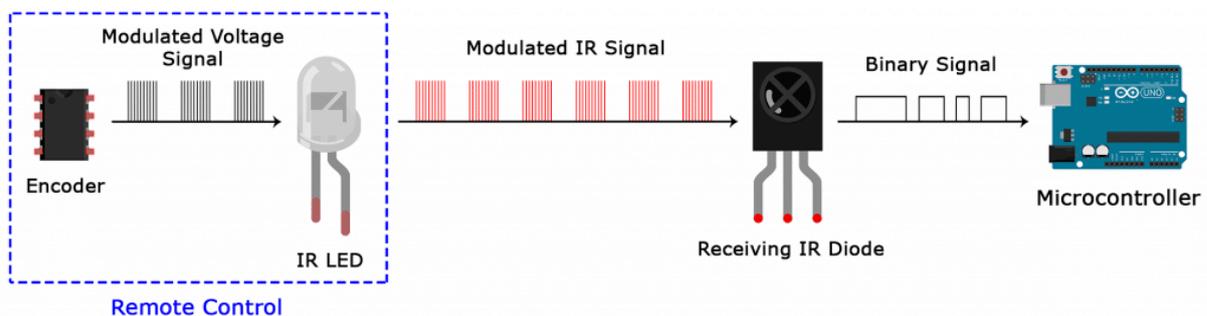


Figure 29. transformation of the signal (27)

The IR receiver is a photodiode, a semiconductor with a junction p-n capable to converts light in an electrical current generated when the photons (due to the infrared light) are absorbed by the photodiode. It has a structure like a diode one. Focusing more on the Elegoo remote, in order to know what button was pushed a hexadecimal code is generated, one for every different button: since I don't know a priori what the corresponding hexadecimal code is, I pushed the buttons and I observed what was the generated code and saving that hexadecimal code I linked the button with some function. Since different remotes have different codes, I can use a different type of remote and see (for example with the serial monitor of Arduino IDE) the corresponding hexadecimal code.

Definition of the components of the system

Focusing on receiver, it has three pins as we can see in the following picture:

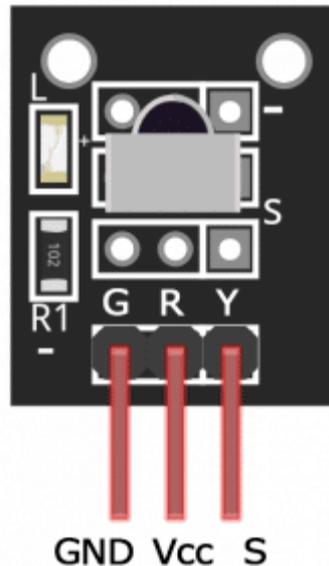


Figure 30: IR-Receiver (28)

The first one is the ground pin, the power supply pin (connected to the 5V pin of Arduino) and then the signal pin that will be connected to the digital pin. The library IRremote.h need to be installed in order to work properly. I choose to use an IR receiver because I want to choose what to display on the LCD:

1. If the button 1 is pushed it displays and if it was in pause it returns on
2. If the button 2 is pushed it displays and if it was in pause it returns on
3. If the button 3 is pushed it go in pause

The idea to use a remote is very good because make the Board independent from the computer: the user can interface with the program (all the function can be chosen by pushing a button, and the information are displayed on the display) only with the Remote and all the system can be contained in a simple case printed 3D with PLA.

2.1.8 Relay 4 channel

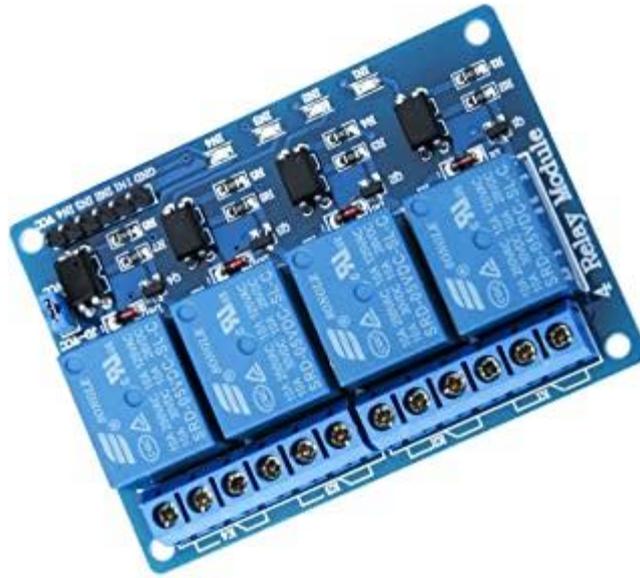


Figure 31: Relay with 4 channels used in this thesis (29)

Since it is not possible to connect the component of Arduino directly to the 220's socket, in order to control it (and to avoid the burning of the board) it is needed a relay. In this thesis I control the heater aquarium and the pump (when the level of the water is bigger than 90% the pump is activated and then turned off when the level is less than 90% and the heater aquarium work in the range 25°C-30°C). In few words, the relay can control a circuit (or more specifically a component) by means of an electro-magnet. The circuit is activated digitally, but to do this is important to connect properly the component: it has 2 configurations:

- NO (Normally Open): the contact with the circuit/component remains open until the relay is not energized.
- NC (Normally Cosed): it is the opposite of the normally open, it remains closed until the relay is not energized.

Relays are used to control an electrical circuit by opening and closing the contacts of another circuit. In both case (Normally Open or Normally Closed) applying electrical current to the contacts of the relay, will change their state.

2.2 MECHANICAL AND HYDRAULIC COMPONENTS

In this thesis will be used also mechanical and hydraulic component necessary to “move” the fluid, heat it and control its flow.

2.2.1 Micro Pump

The first mechanical part used in this thesis is a micropump. Before on focusing on it, I made a small recap of how the pumps generally works. A pump is a device that move fluids, in our case liquid (and more general water) by means of a mechanical action. It takes the electrical energy and converts it in mechanical energy. The micropump used is Decdeal 600L/H: a brushless micropump. There are some advantages using a brushless pump instead of one with brushes:

- It has a higher life duration.
- There are not brushes to substitute: better maintenance.
- High efficiency (about 85-90%).

But there are also some disadvantages:

- Higher cost.



Figure 32:Decdeal 600 L/H (30)

Definition of the components of the system

This pump will be connected at 220V directly to the socket. It pumps the water from the upper reservoir to the middle ones. When the water will have the correct level, the pump will be turned off, and then if the level is upper a certain range it will be turned on. Thanks to the lcd connected to the board there will be a notification about the level in order to turn off the pump.

2.2.2 RS Flow switch FS3

A flow switch is capable to decrease the flow of water if it goes in a certain range. In this project it will be used as a security switch, in other words: if the flow reaches 0.2 l/min (this data is given by the constructor) then will remain constant, and it cannot overcome it.



Figure 33: Flow Switch (31)

Below a scheme of the flow switch:

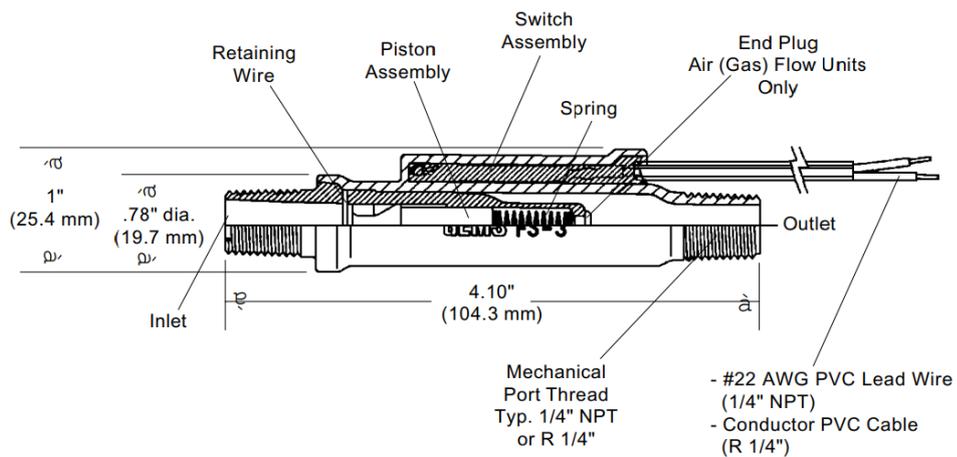


Figure 34: FS3 Flow switch (32)

2.2.3 Heater Aquarium H-300

This heater aquarium will be used to heat the water in the middle reservoir. It is not controlled by Arduino, but it is directly connected to the socket. It converts electricity to heat.



Figure 35: Aquarium Heater (33)

Even if it cannot be controlled directly by the microcontroller, its maximum temperature can be chosen and when it reaches it, the temperature will remain constant (and the water will not overheat). If the water decreases its temperature and became lower than the set one, it turns on. It can be attached to the sides of the reservoir by means of the suction cups.

It is composed by a resistor and a thermostat:

- The resistor is needed to perform the joule heating: the power generated by an electrical conductor is proportional the resistance value of the resistor and the square of the current flowing into it:

$$P = I^2 \cdot R$$

Where P is the power, I is the current and R is the resistance of the resistor inside the heater aquarium. Focusing on a microscopic level, the power dissipated is equal to:

$$P = \int_{\tau} E \cdot J \, d\tau$$

Where τ is the volume, P is the total power dissipated and J is the charge's density.

- The thermostat is used to measure the temperature and control the system as a closed loop control device.

Definition of the components of the system

2.2.4 Reservoirs and silicone tube

The reservoir chosen it three simple plastics transparent one. They have three different dimension and the middle one it has a hole (drilled with a conical tip) and a flexible silicone water tube. It is connected directly to the reservoir with nozzles design by me and printed by a 3D printer.



Figure 36: one of the 3 reservoirs used

The dimension of the three reservoirs is different:

- The smallest one: is used as drain tank.
- The middle one: is the tank where initially there is the eater and the pump and when the level is higher than a certain value it is drained in the bigger one.
- The bigger one: is where the water going on and it is heated by the heater aquarium. There is a waterproof thermometer and the thermostat of the heater aquarium.

The flexible silicone tube as a diameter of 0.5 cm and has a total length of 10m, but I cut it a little piece because the reservoir will stay near one each other.



Figure 37: a flexible silicone tub (34)

2.3 3D PRINTER AND SOFTWARE ANNEXED

Since the nozzles of the pump and of the flow switches have not the same dimension of the silicone tube, I preferred to print some components to connect them. I also wanted to print an Arduino case in order to make the system more compact and avoiding the dispersion of the battery (needed for the alimentation of the board), the board, the lcd and all the wires. Thanks to the case, all these components are together in the same case, making it easier to manage.

2.3.1 Creality Ender 3

The construction of 3D objects by means of a 3D printer is named additive manufacturing and it is a rapid prototyping technique (i.e. a technique used to quickly create an object). In nowadays is more common also in industrial application and for the creation of complex shapes. There are many ways to create the 3D object (before printing it) as for example using a 3d scanner or as in my case a computer-aided design (CAD). The 3D printer used is the Creality Ender 3 and the CAD software is SolidWorks (Student Version). The CAD is used to design curves and lines in 3D and then I use a slice software to transform the .step files of Solidworks in G-code in order to be directly processed by the 3D printer.



Figure 38: Ender 3 printer (35)

I used the 3D printer to print nozzles and a case for microcontroller because I want to have all the wires and components in the most compact way. There are many materials that can be used with a 3d print, but mostly we can separate the 3D printer in 2 categories:

Definition of the components of the system

- SLA, also known as stereolithography apparatus is a resin type of printing that produces object in a layer-by-layer fashion using photochemical processes by means of light (more specifically a laser or an optical technology named DLP) that produces monomers and linking together to form polymers that together make the 3D model. The particularity of this process is that the object is created upside down as we show below. The principal material used in SLA printing is the resin and it is more expensive of the simplest PLA used more frequently. This type of fumes is toxic. Furthermore, is better to use gloves with the piece just finished and then put in another device (that can be sometimes the same printer) to do the polymerization. The working process includes a laser that illuminates the bottom of the tank filled with resin, and once it is lighted it solidifies creating a layer of the final object. Layer after layer the object is lifted (by a platform) and the next layer is solidified. I avoided to use this technology because the cost of the printer is higher than a FFF, the resin is circa 50\$ at kg, and is hard to manage the printed object. This technology is used mostly to print miniatures because the details are higher than a FFF printer. In general, the object printed with this type of printer are smaller because the plate is smaller than a FFF printer.

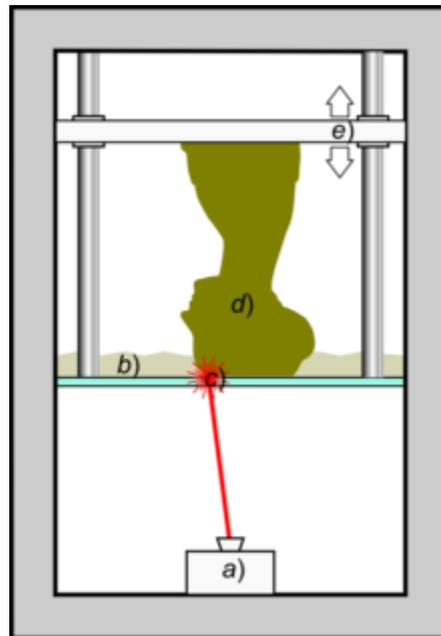


Figure 39: example of SLA printing (36)

- FFF also known as Fused Filament Fabrication: it uses a continuous filament of a thermoplastic material. The shape of the object is defined by the computer. It has 3 axes where it can move, it deposits the material on a 2D plane and then layer by layer it goes up creating the final shape. It has an extruder where the material is printed out as a filament.

Definition of the components of the system

There can be use a wide variety of material, the commonly used are ABS (butadiene styrene) and PLA (polylactic acid) that was used in this project. In order to print properly the plate must be warmed up at 60 °C and the extruder at 200 °C (for PLA). Once is everything set, a moving extruder melts the PLA and deposits it according to the G-code (generated by the slicer software) layer after layer generating the final object. In this project I use a white filament of PLA, because for my purpose it has the best trade-off between cost and efficiency (it also is made from renewable resources, and it also not produce toxic fumes if incinerated). 1 kg of PLA is circa 20\$, it is simple to manage, it is not toxic, and the final result is good because I don't needed detail but only a case to contain the object with some tolerance not restrictive. Furthermore, the case is big, and the less expensive SLA's printer cannot print a case nig like he once needed to me.

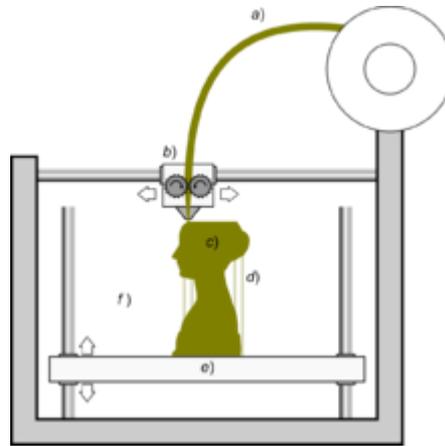


Figure 40: FFF printing example (37)

2.3.2 Software used

For the 3D design of this thesis, I used two software:

- SolidWorks Student Edition: it is a 3D CAD (computer-aided design): software used during 3D design, especially for mechanical draft. Once the 3D model is finished, it can be chosen a type of material and can be calculated its mass, its inertia momentum (and other data used for static calculation). I designed with SolidWorks many object needed during this thesis, and once I finished it, I saved it in .stl format, and opened with a slicer software. It is also useful the information of the object: one the user has been inserted the type of material, the software automatically shows us the mass, density and other useful information.

Definition of the components of the system

SolidWorks is not the unique software used to create 3D object: it can be also used Fusion360.

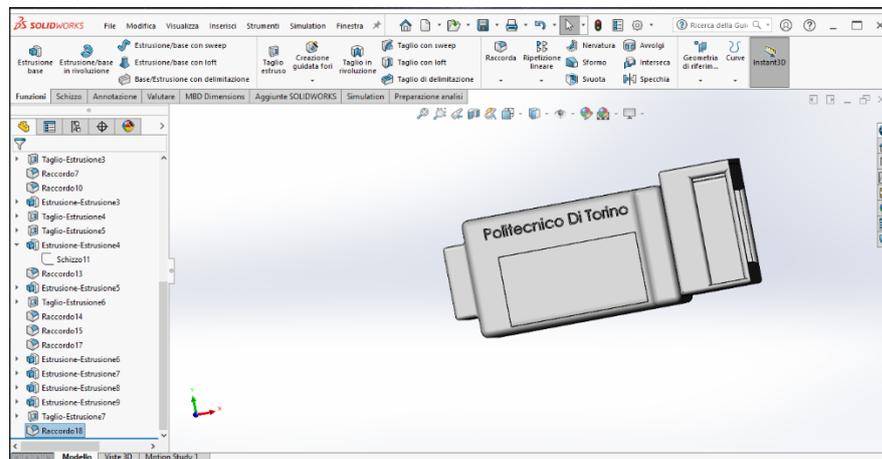


Figure 41: SolidWorks interface

- Creality Slicer: it is a slice software: it takes the .stl files (created with Solidworks) and save it directly in G.-code in order to be correctly read by the printer. The most used software for this type of applications is Cura, but I preferred to use the one given with my 3D printer. There are some options that can be chosen: as for example the type of material, the diameter of the material from extruder, the velocity, if it needs a support and our model can also be scaled by a factor (chosen by the user) in all 3 the axis or only one. Furthermore, you can choose how to print the model, for example printing vertically the model or horizontally. The software divides the object in a stack of layer and together of other information (as for example the good temperatures or bad, any type of additional support ecc,) is saved in G-code and can be transferred directly to the printer with a Memory-Card.

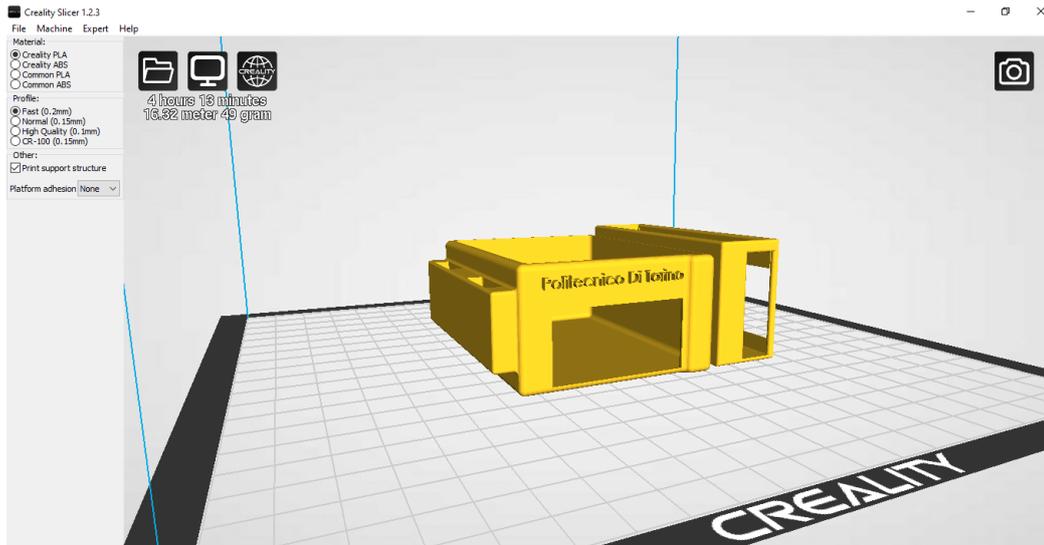


Figure 42: Creality Slicer interface

Below an example of steps in a 3D print (3D CAD then G-code then printing and then the final object):

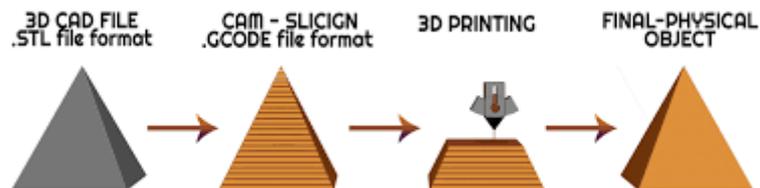


Figure 43: 3D printing procedures

2.3.3 Models printed

In this paragraph will be show what was printed by me, and how. For all the model needed, I measured the dimension, and I designed the case and the nozzles starting from it. The general schema is the following:

1. Measure physically something: for example, I measured the Arduino's board and the LCD display.
2. Design with SolidWorks my model: once I had my measurements values, I designed what I needed.
3. Save the 3D model in the .stl format and open it with Creality slicer, and transform it a G-code easy to be understood by the 3D printer;
4. Set-up the printer: pre-heat the extruder, set the axes, start to print.

Definition of the components of the system

For the purpose of this thesis, I printed a first case (quickly replaced by another) and four nozzles needed to connect the silicone tubes with the flow switches and the pump. As said before the technology used is the FFF printing, using a white coil of PLA. I choose the PLA because it is biodegradable, and it has a low price (about 20 euros for 1kg).

2.3.3.1 *Arduino's Case*

The Arduino case has 2 parts:

1. The middle part contains the Arduino board, the shield and the mini breadboard with all the connection.
2. The left part has a space that contain the battery and another space that can contain the two sensors. The space for the sensor is smaller, but it can be filled also with a clip or something else allowing the connection with the first or middle tank.

This case is useful because protect and board and make the system more compact. I designed a first version that wasn't satisfactory: it could contain only the board and not shield, moreover it didn't had spaces for battery, sensors, and LCD. Due to the bad heating of the bed, the lower part wasn't uniform. Below a picture of the first project:



Figure 44: First design of the Arduino case

I will not focus on this first version because I prefer the final where there is a space for the battery. As said before both the version was designed on SolidWorks following this procedure explained before. There is also another version with a part that can contain the LCD.

Definition of the components of the system

I preferred to print a separate case for the lcd because I want to have it independently as in the following picture.



Figure 45: LCD case

Below the Solidworks screenshot of the case:

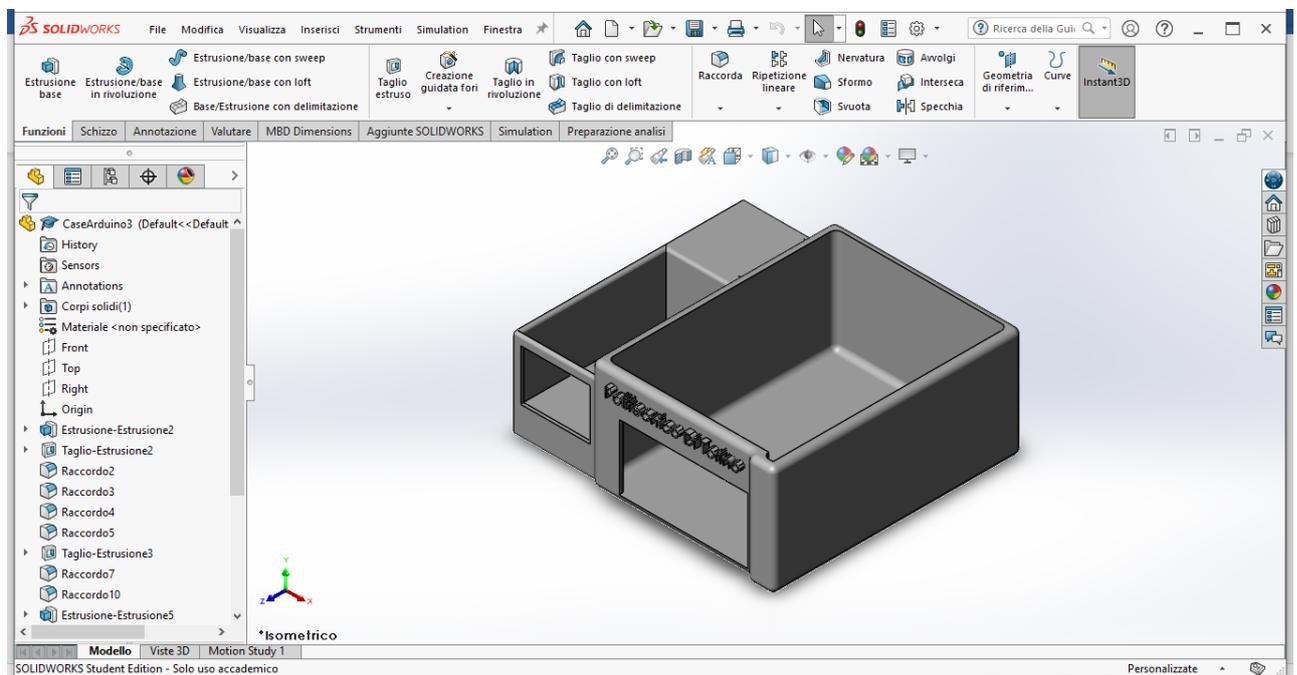


Figure 46: A SolidWorks view of the Arduino Case

I created this model measuring physically the dimension of the board (with the shield and the mini breadboard on it), of the display and of the battery. The space for battery is needed because when the code will be loaded on the board then it will run without a computer, and it will need only an alimentation (for example a battery of 5V). When I finished the model I exported it in a format .stl and I opened it with Creality Slicer:

Definition of the components of the system

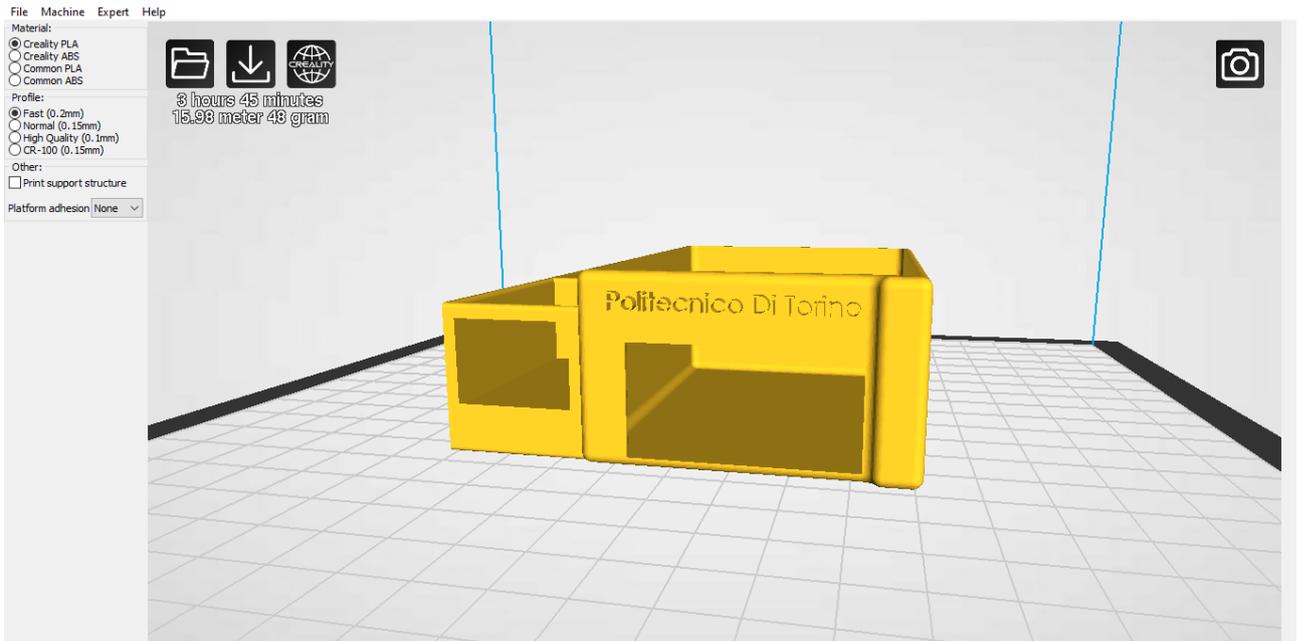


Figure 47: Creality Slicer view of the Arduino board

From the various type of print option, I selected the PLA as material, the profile as faster (because this case doesn't need too much detail), I didn't print a support (because don't needed) and the scale remained 1:1. Below an image of the printer while printing the case:



Figure 48: Printer while printing the case with the parameters

The total time to printing the case is bigger than 4 hours and 41 minutes (as said from the software) but is almost 6 hours. While printing it can be seen from the display some parameters: for example, the actual position of the three axes, the temperature (for PLA printing is good to have

Definition of the components of the system

at 200 °C the extruder and 60°C the bed) and the percentual of printing. Below an image of the case finished. As said in the introduction there are two versions of this project: in the simplest one (the one without the control of the temperature and on the water's level) all the system can be put in the case, and it occupy a small amount of space. Vice versa in the other version is more difficult to have all in the case because there is also the relay, but it can be improved the case designing another with the necessary space for the relay, but there will be more wires and a more complex system.

2.3.3.2 Nozzles

In order to connect the silicone tube with the other components, I designed 4 different nozzles:

- The first one is used to connect the silicone tube with the pump.
- Two of them is to connect the 2 flow switches with the silicon tube (the input and the output of the flow switches has different dimension).
- The last one is to connect the bottom of the middle reservoir with the silicon tube, to drain the liquid.

The dimensions are different, and the nozzles was designed starting from some measurements. Since the design was simple, and the total time of printing (about 25 minutes). The internal diameter of the nozzles (i.e. where the water is flowing) is the same of the silicone tubes, while the external diameter is the same of the nozzles of the pump and of the flows switches. Since the diameter is very small, the water's flow never turns on the flow switches (used only to be safer) and this brings us to the conclusion that the flow switches are not necessary. I report only a final image of the four printed nozzles.



Figure 49: nozzles printed by me

3 PROPOSED SYSTEM

3.1 GENERAL SCHEMATIZATION OF THE SYSTEM

The proposed is illustrated in the scheme below.

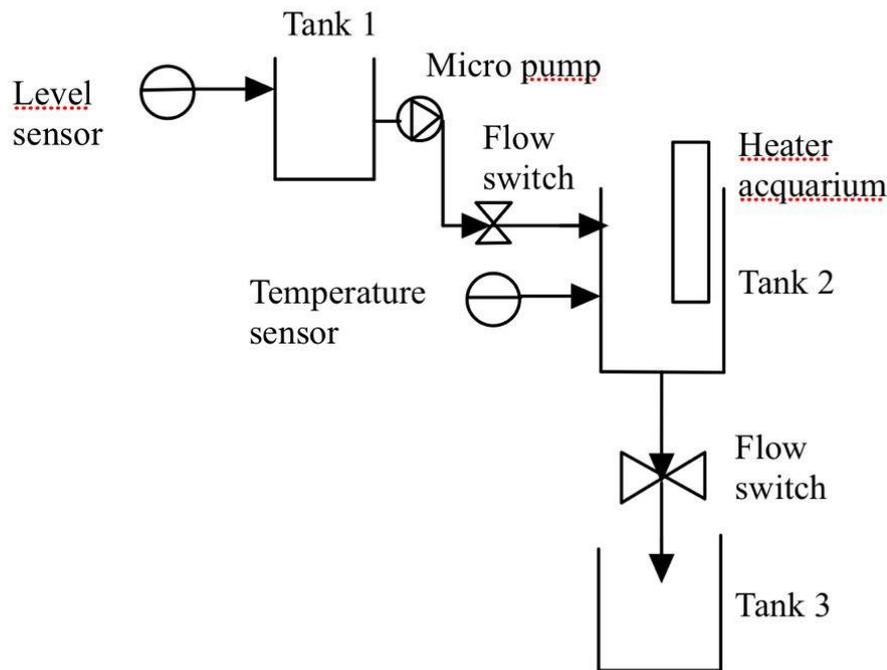


Figure 50: scheme of the system

Focusing on the sensor: the temperature sensor and the level sensor are connected to Arduino by means of a breadboard, the flow switches, the micropump are connected directly to the socket. The values acquired by sensors are displayed on a LCD system, and the data can be read from it by switching by means of a remote. The three tanks are in plastic and with different dimensions. The fluid pass through some silicon tube that are connected directly to the flow switches and the pump. The heater aquarium is in the middle tank, the temperature is selected (directly from the heater) and once it is reached, it will remain constant. The temperature sensor is used to see if everything works fine and to measure the reliability of the system.

The electronic parts with the circuit connected to the boards are shown more detail in the followings paragraphs because during the test I improved the system by choosing different component, saving time when assembling the circuit and saving spaces (in the final version I use less wires making all the system simpler).

3.2 FIRST ELECTRONIC CIRCUIT

This paragraph shows an electronic scheme not used in the final one. I choose to show this version because even if the final results are the same (i.e. the value acquired by the system are the same, the remote works great as well etc.) the system show some limits. The main differences between this version and the final are:

- In the first version I used a standard LCD, with 16 outputs connected to 16 pins of Arduino. The final version has as said before an LCD with a I2C protocol that has only 4 pins, greatly simplifying the number of wires. Furthermore, in the first version the wires cannot be connected directly to the board because it needs a resistor and a potentiometer, so the wires were connected to Arduino through a breadboard increasing the number of wires. This high number of wires doesn't allow to put in a case the LCD because the spaces used is too much.
- In the first version I used a standard breadboard (one with big dimension) that occupy to many spaces. Moreover, since the breadboard is bigger than the Arduino it is very difficult to put together the microcontroller and the breadboard and as consequence they stay far from each other, increasing the difficult to have a compact system. In the final version I used a prototype shield within a mini-board sticked on it. This configuration gives me a more compact system simplifying the system.
- I choose to use some bridges wires with various lengths based on the connections I had to make. This is since some connection doesn't need a long wire because the mini breadboard is very near to the pins of Arduino thank to the shield.

In the new configuration I needed to modify a little bit the code, because led and some pins of the sensor are now connected to different Arduino's pins. Furthermore, with the new LCD I2C the library needed to work are different from the standard LCD. At the end of the thesis will be show also the old code and the final code. A part of this little difference there are not any difference, nor in the code part nor in the conceptual part.

Below there is a picture of the first circuit:

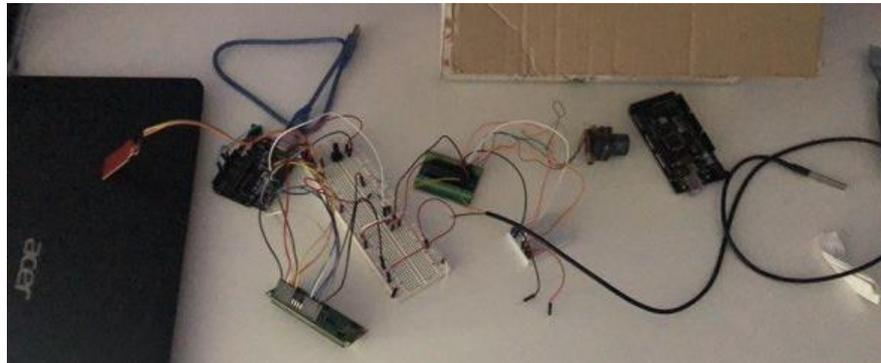


Figure 51: First Prototype

In the upper photo is not present the IR receiver and the LED, but there are more wires than the final project. This first prototype had the standard LCD with 16 pins, and I used a larger breadboard. In the final project, thanks to the new LCD with I2C protocol the number of wires is only 4, as consequence of less wires I preferred to use a smaller breadboard. Furthermore, I didn't use only the standard jumper (all with the same length) but also some "bridge jumper" with various dimension obtaining a cleaner and simpler circuit. Lastly, in the first prototype I used a motor directly connected to Arduino, but since its power wasn't too much (it has an alimentation of 5V-12V) I preferred to use a motor more powerful, but that cannot be directly controlled from Arduino. I remedied using some silicon fluid with a smaller diameter to control the flow better. Below an image of the pump and its datasheet initially used but not included in the final project.



NOTE Micropumps are required to be filled with liquid (primed) before they can operate.

DO NOT run your Micropump dry. Ensure that the pump has been primed before use.

RS Micropump Performance Data	Input Voltage (V)	Current Draw (A)	Power Usage (W)	Pressure (mBar)	Pressure (psi)	Free Flow Rate (ml/min)	Operating Temperature range (C)
M400S	6.0 12.0	0.22 0.60	1.3 7.1	86 380	1.2 5.5	1495 2865	-20 to 120 -20 to 120
M400K	6.0 12.0	0.20 0.48	1.2 5.8	86 380	1.2 5.5	500 1150	-20 to 120 -20 to 120

Figure 52: Micropump RS400 180

3.3 FINAL CIRCUIT WITHOUT CONTROL ON HEATER AQUARIUM AND PUMP

The aim of this paragraph is to show the configuration of the circuit and explain the connection. In this first circuit the pump is controlled manually (when the display of the LCD shows a warning of the water level, the operator turns on the pump) and the heater aquarium use a internal thermostat in order to turn on or turn off the temperature of the water.

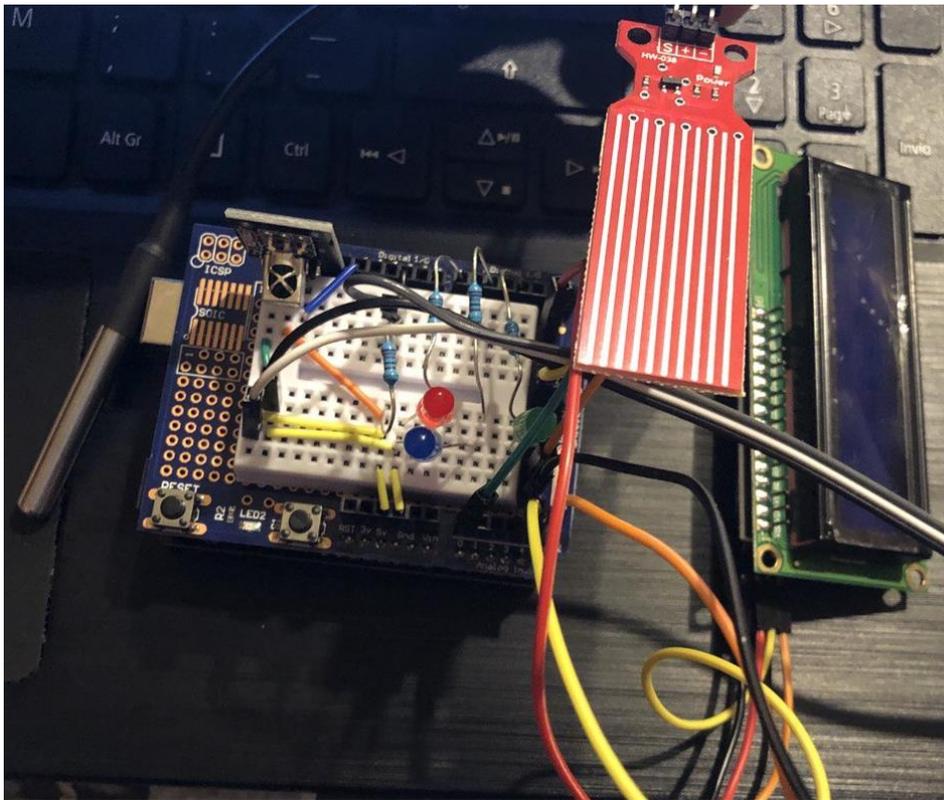


Figure 53: Final Circuit with all the sensor

There are many components connected to Arduino:

- Prototyping shield: the good thing of having this shield on the board is that I have all the component near each other. Moreover, there are many pins for 5V and GND (5 pins of the alimentionation and 5 for the ground). On this shield I glued the mini breadboard (this breadboard is divided in 2 part and the current flow in the vertical pins of each part as explained previously). On this breadboard are connected the IR receiver, water sensor, temperature sensor, led and resistors.
- Water level sensor: this sensor is connected to the analog A0 pin of Arduino and to the 5V and GND pins. It measures the level of the water in the tank, and (with a code shown below) it warns when the water is at 90% of the height of the reservoir.

Proposed System

- Waterproof temperature sensor: it measures the temperature of the liquid. It has 2 pins connected to the 5V and GND pins and one to the digital pin. There is also a resistor to decrease the current flowing into it.
- IR receiver: it receives the infrared sensor from the remote, and with a switch case it shows us the temperature, the water level or put the system in stand-by.
- The LCD show us the information about temperature and water sensor, and I can compare it the real one in order to measure the reliability of the system. It has 4 pins: 2 of them are connected to 5V and GND pins, the other 2 (using the protocol I2C) are connected to the Analog pins A4 and A5 (SDA and SCL pins).
- Three LED of different color: green if everything is fine, blue if the system is in pause, red if the water has exceeded the maximum allowed level. There are needed 3 resistors because without them the current will burn the LEDs.

All these components are compacted thanks to the Arduino case, and it is easy to transport and to use. Once the code is generated and loaded on the board, it will not need any computer connection though the USB cable, but a battery of 9V connected to the power alimentation pin will be enough. Moreover, there will be a part of the case with a space for the battery to make simpler the transportation. In the past paragraph I have focused on the components: in the next paragraph I will focus on the code analysis that is a bit different of the code in the case where a control the circuit. This first project it is better to transport since it no need a computer and the heater aquarium and the pump are not directly connected to Arduino. I want to show both the final result, without the control of the part and with the control part (with both the circuit) because the final user can use the more properly (and safer). In the following paragraph will be show the code analysis: the two variants of the circuit are the same except the part of the control. The differences of the two parts will be analyzed in more detail further on.

3.4 CODE ANALYSIS

3.4.1 Library and definition of variable and Object

First of all, I included the library necessary to make the components work properly:

```
#include <IRremote.h> //Library for remote
#include <Wire.h> // Library for I2C communication
#include <LiquidCrystal_I2C.h> // Library for LCD
#include <OneWire.h> //Library for termometer
#include <DallasTemperature.h> //Library for termometer
// he LCD is connected to the board by the protocol I2C using pins A4 and A5
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 16, 2); //LCD has the port 0x27
IRrecv irrecv(13); //the Receiver is connected to the digital pin13
decode_results res; //I created this object named res

int led1 = 3; //GREEN LED
int led2 = 7; //RED LED
int led3 = 5; //BLUE LED
int livello = 0; //level of the water measured by the sensor
int minlevel = 0; //minimum level that can be reached
int maxlevel = 100; //maximum level that can be reached
int i = 0;
int flag = 0; //useful to enter in the switches or not
//the following object and float variables are needed for the temperature sensor
OneWire oneWire(10);
DallasTemperature temp(&oneWire);
float tc;
```

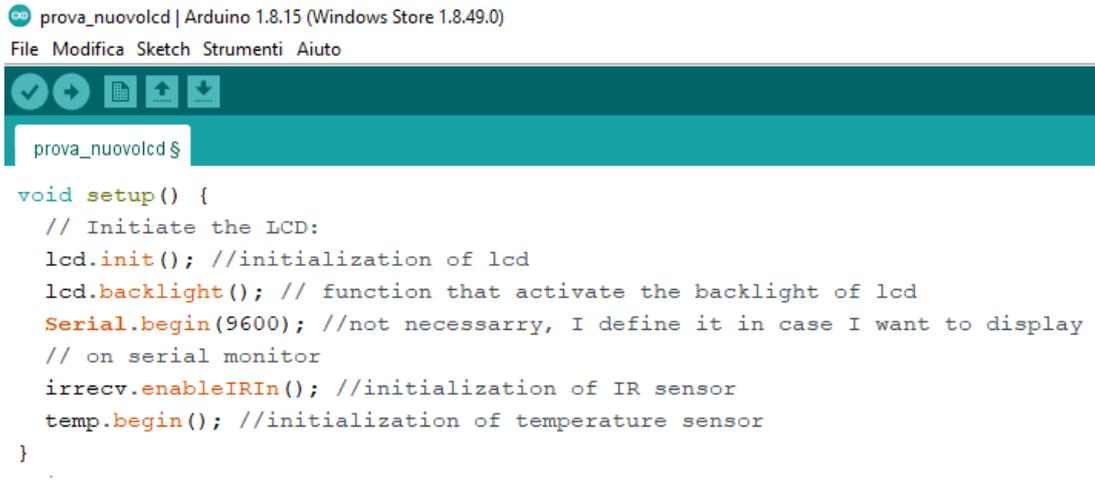
Figure 54: library and definition of some variable

I included the library for the remote, the temperature sensor, the IR Receiver and the LCD with protocol I2C. Then I defined some object. One of the objects defined concerns the LED with protocol I2C, and it has a serial port equal to 0x27 (it is standard for the most of microcontroller). In the same object I also said the number of bits of the display and the number of rows. Then I defined the three LEDs, giving on each a digital pin. The next step is the sensor: I defined a variable named “Livello” where I consider the level of the water measured, and I defined a minimum level and a maximum one (I want to have the level between 0 and 100 because I want a percentage value). Then I defined a variable I and a flag, used to display the main menu or some information relevant based on the pushed button. Lastly, I defined the object for the measurements of the temperature and a float variable used to memorize the temperature in Celsius. All these variables are used in the code to store the information and then display it on the LCD.

All the variables can be modified by the user, as for example the minimum level of temperature, the maximum level of temperature and the percentage of water in the first tank that activate the relay.

3.4.2 Void setup

The void setup run only once when the code is on the board, so I used it to initialize some variable:



```

prova_nuovolcd | Arduino 1.8.15 (Windows Store 1.8.49.0)
File Modifica Sketch Strumenti Aiuto
prova_nuovolcd$
void setup() {
  // Initiate the LCD:
  lcd.init(); //initialization of lcd
  lcd.backlight(); // function that activate the backlight of lcd
  Serial.begin(9600); //not necessary, I define it in case I want to display
  // on serial monitor
  irrecv.enableIRIn(); //initialization of IR sensor
  temp.begin(); //initialization of temperature sensor
}

```

Figure 55: Void Setup of the code

The first line is to initialize the lcd, then I turn the backlight on. The function serial.begin(9600) is used to display the information on the serial monitor: I left it here, but is not necessary because I display the information on the display. In the next step I initialize the Infrared receiver and the temperature sensor

3.4.3 Void Loop

```

void loop() {
  livello = analogRead(0); //from thr analog pin 0 I read the water level
  if (livello < minlevel)
    livello = minlevel; //In case of the water level is less than 0
    // (but never happened) is only to be sure
  if (livello > maxlevel)
    livello = maxlevel; //in case level>100 (sometimes happen

  livello = map(livello, minlevel, maxlevel, 0, 100); //this map the level
  //in order to have equal value between 0 and 100
  temp.requestTemperatures(); //take the temperature
  tc = temp.getTempCByIndex(0); //transform it in Celsius
  if (flag == 0) //flag equal zero means nothing happens: main menu
  {
    lcd.clear(); //creal precedent display
    lcd.setCursor(0, 0); //character one, line one
    lcd.print("1 for Level");
    lcd.setCursor(0, 1); //character one, line two
    lcd.print("2 for Temp.");
  }
}

```

Figure 56: mapping of the level sensor and main menu

First of all, I take the liquid level by means of the sensor through the analog pin A0. To be sure and to be safe, I put some if conditions:

- If the liquid sensor is less than 0 then I take 0 (it never happened).
- If the liquid is bigger than 100, I take 100 (this happens very frequently, sometimes it reach values of 500-600).

Then I map this value with the function map in order to have a percentual between 0% and 100%.

Then I take the temperature with the sensor, and I save the value in Celsius.

Then I use the flag variable defined above equal to 0, so if it is equal to zero, thanks to an if it enters inside this condition and displays the main menu:

1. If the button 1 is pushed it display the level of the liquid.
2. If the button 2 is pushed it display the temperature of the liquid.
3. If the button 3 is pushed it display the system will be in pause (this option is not displayed in the main menu)

Before displaying it, I clear the display: I do that because the main menu is displayed also when the system goes in pause (after the third button is pushed). The cursor is settled in the following way: the first number is the character (so the zero at the left is the first character of the row) and the second number is the number of row (if equal to zero is the first row, if equal to 1 is the second row).

```
if (livello >= 90) //warning about the level: pay attention!
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Attention ");
  lcd.setCursor(0, 1);
  lcd.print(livello);
  digitalWrite( led1, LOW);
  digitalWrite( led2, HIGH); //display the red led because it is a bad thing
  digitalWrite( led3, LOW);
  delay(1000);
  flag = 0; //once the level is less than 90, it turn on main menu
}
if ( irrecv.decode(&res) || flag != 0) //it enter in this flag if it
//takes value from remote of if flag is different from 0 (starting value)
{ Serial.println(res.value);
  irrecv.resume();

  switch ( res.value) //switching in base of the value receiver by the IR receiver
  {
```

Figure 57: Void Loop, warning part

The next step is an if condition that warn us if the water's level is greater than 90%: then will be displayed a message ("Attention") and the current level. It also turns on the red led. The flag will turn zero, because when the level will return less than 90% it is useful to have the main menu. There is a delay command: every 1000ms there will be a refresh of the water level. There is another if condition, and it is satisfied each time the infrared sensor receive a signal or if the flag is different from zero (this second condition is useful because otherwise it doesn't enter inside the switch case, and the values displayed will be not actualized). Then we enter inside the switch case with the first condition:

```
case 16724175: //it enter here if receiver an IR value from button 1 or if it was pushed before
case 1:
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Water Level:"); //measure water level
  lcd.setCursor(0, 1);
  lcd.print(livello);
  lcd.print("%");
  digitalWrite( led1, HIGH); //led green because everithing ok
  digitalWrite( led2, LOW);
  digitalWrite( led3, LOW);
  flag = 1; //in order to not display again the main menu
  res.value = 1;
  delay(500);
  break;
```

Figure 58: Case condition 1

I enter in this condition if the signal receiver by the IR receiver is equal to 16724175 or if it entered before (because I give a value to the signal receiver equal to 1). In this switch case is displayed the water level, and the green led is turn on. There is also a delay of 500ms, i.e. the screen is refreshed every 500ms.

The next switch case is the following one:

```
case 16718055:
case 2:
  lcd.clear();
  lcd.setCursor(0, 0); //0 significa all'inizio della riga, mentre 1 è la riga cioè due perche 0
  //muove il cursore sulla riga sottostante,visto che le righe partono da zero, 1 significa 2
  lcd.print("Temperature:");
  lcd.setCursor(0, 1);
  lcd.print(tc);
  lcd.print(" Celsius");
  digitalWrite( led1, HIGH);
  digitalWrite( led2, LOW);
  digitalWrite( led3, LOW);
  flag = 1;
  res.value = 2;
  delay(500);
  break;
```

Figure 59: Switch case 2

In this case the temperature will be displayed (in Celsius) and the green led is turn on (I turn on the led each time because maybe there was before a warning or the system in pause that turn on the red led, and the blue led respectively. I enter in this case when the signal is equal to 16718055 and 2 (2 if there was pushed the button 2 before in order to refresh the screen every 500ms).

The last witch case condition is the following one:

```
case 16743045: // button n.3 put the sysstem in pause
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Sistema in pausa");
  flag = 0;
  digitalWrite( led1, LOW);
  digitalWrite( led2, LOW);
  digitalWrite( led3, HIGH); //blue led on because it is in pause
  delay(500);
  break;
default:
  delay(500); //if any button is push, there will be a delay of 500ms
  break;
}
}
```

Figure 60: third switch case condition and default condition

In the third and last switch case I turn on the blue led, and I put the system in pause: there won't any measurements. There will be a pause of 500ms and then, since the flag is again zero, the main menu will be showed again. There is also a default case where the signal is 500ms (in the case where another button is pushed). The next images show the different configuration of the LCD that can be chosen according to the button pushed:

- Main menu: where you can choose to see the water level or the temperature.
- Level of the water (expressed in %).
- Temperature in Celsius.
- Warning (if the temperature is bigger than the maximum chosen by the user or if the level of the water is bigger than 35%).
- Pause (after 500ms it turns on main menu).

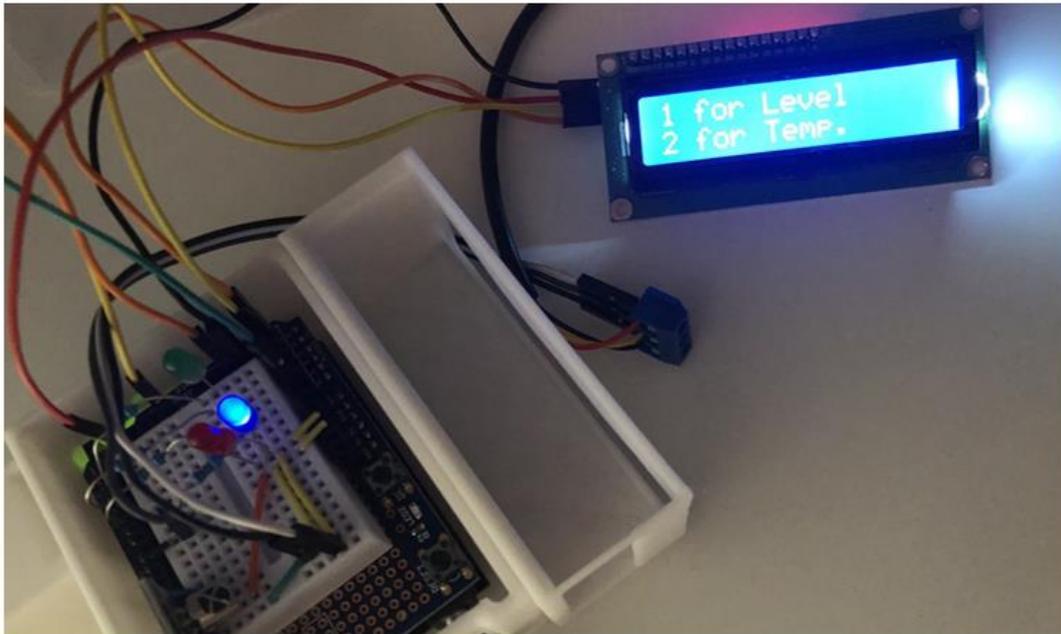


Figure 61: Main menu

The image above shows the main menu and the blue led on.

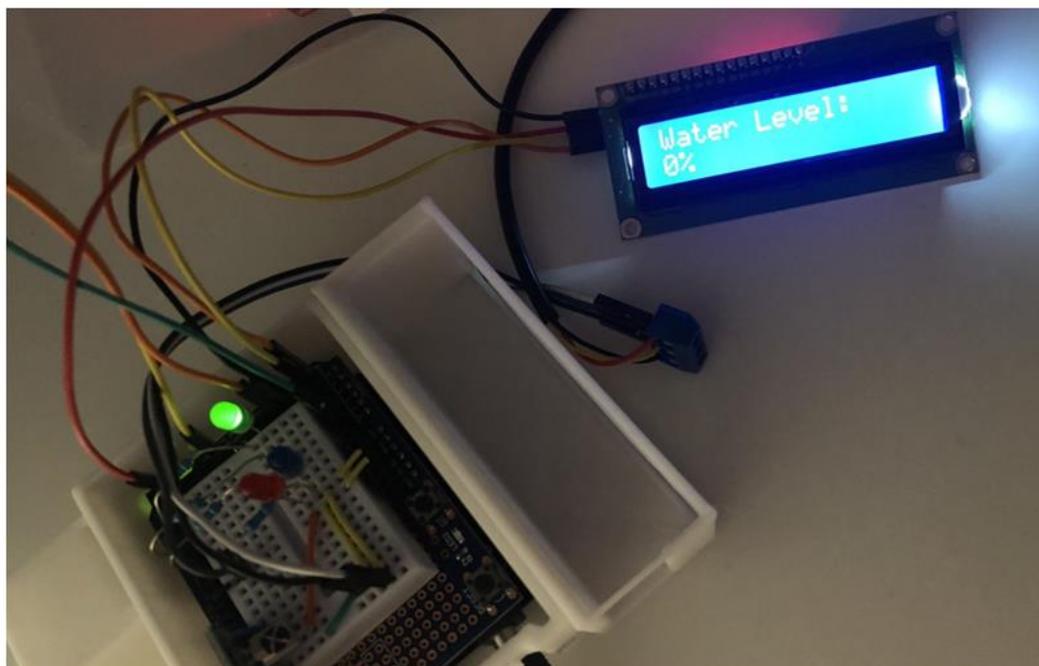


Figure 62: Water Level

The image above shows the mode when the water level is displayed, and the green led on.



Figure 63: Water Temperature

The image above shows the temperature of the water in Celsius.



Figure 64: Warning mode

The image above shows a warning: the led became red and display the water level independently of the previously display, and remain as much as the level is upper than 90%.



Figure 65: System in pause

The last image above shows the system in pause: the led became blue and after 500ms it turns on the first menu when you can choose the mode. To have the system in pause the button number 3 must be pushed. The Led and the message of warning are useful for the project without the automatic control of the heater aquarium and of the pump: the system warns the user when the temperature and the level of the water reaches some value (defined precedentially from the user directly in the code) and the red led blinks (vice versa if everything is fine the green led is turned in).As said before, all the electronic part (i.e. all concerning the Arduino's board, the mini-breadboard, the shield of rapid prototyping, the sensor, the battery etc.) is inside a box of PLA printed by me. Thank of this case I saved a lot of space, and the system appears clearer and simply. Furthermore, once the software is loaded on the board it does not need anymore the connection via USB cable to the computer, but it can need only an alimentation of a battery of 9V.

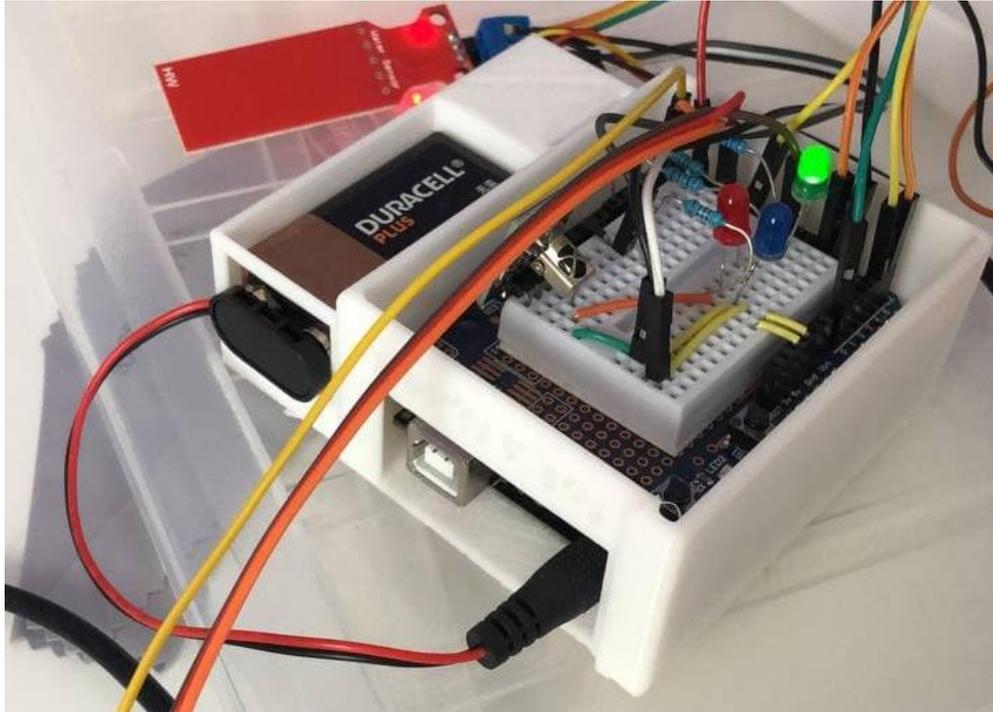


Figure 66: Arduino Case

The case is in 2 parts: the Arduino board with the mini breadboard is in the center part and at the right part there is the battery. This case is convenient to use for the version without the control of the 2 components since the relay occupy a lot of space and there are 2 cable and other more connections.

3.5 COMPLETE SYSTEM WITHOUT CONTROL ON HEATER AQUARIUM AND PUMP

The complete system is composed by 3 parts:

- The electronic part inside the case oriented by me.
- A First reservoir that have a pump connected to a flow switch by means of a silicon tube and a couple of nozzles printed by me. Inside of this reservoir there will be the level sensor.
- A second reservoir which there is the heater aquarium and a silicon tube connected to a flow switch. Inside of this reservoir here will be the temperature sensor.
- A third tank: connected directly on the second by means of a switch flow. It acts as drain tank and there is not any type of sensor on it.

The pump is used to pump the water from the first reservoir to the second and its power can be modified. The flow switches are used to control the water's flow and are permanently connected to the alimentation to make the system safer. Below an image of the system:



Figure 67: Complete system

When the LCD show the warning about the water level, the user can activate the pump and decrease the water level. The heater aquarium turns off when the temperatures reach the requested one (for example 30°C). The temperature is settled up directly on the heater. In the following paragraph will be show the circuit where the temperature and the water level are settled by the user in the code. IN the picture there is also the relay used to control the temperature and the water level.

3.6 FINAL CIRCUIT WITH CONTROL ON HEATER AQUARIUM AND PUMP

In this circuit I added a relay with 4 modules, with 2 of them occupied that can control the heater aquarium and the pump.

WARNING: these two components are connected directly to the 220V, there are illustrated only for didactical purpose but avoid using this configuration if you don't know what you are doing.

In this configuration I used an electrical socket with free cables as in the picture:



Figure 68: electrical socket (38)

I cut the final part of the electrical cable of the heater aquarium and of the pump and by means of a relay I connected it to the relay according to the following scheme (created by me):

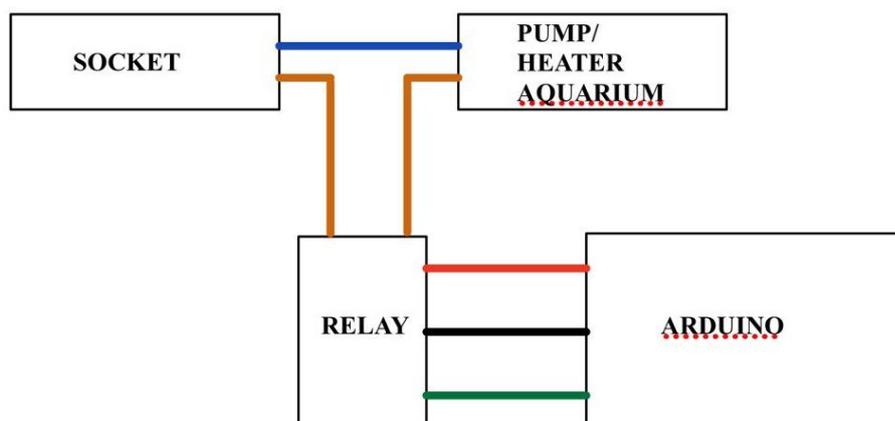


Figure 69: Scheme

Typically, the electrical cable is composed by the phase (the red one) and the neuter (the blue one). I connected 2 the blue cable among them (the blue cable of the pump and of my electrical

cable and the red cables are connected to the Arduino Relay in the configuration normally open as can be seen in the following pictures:

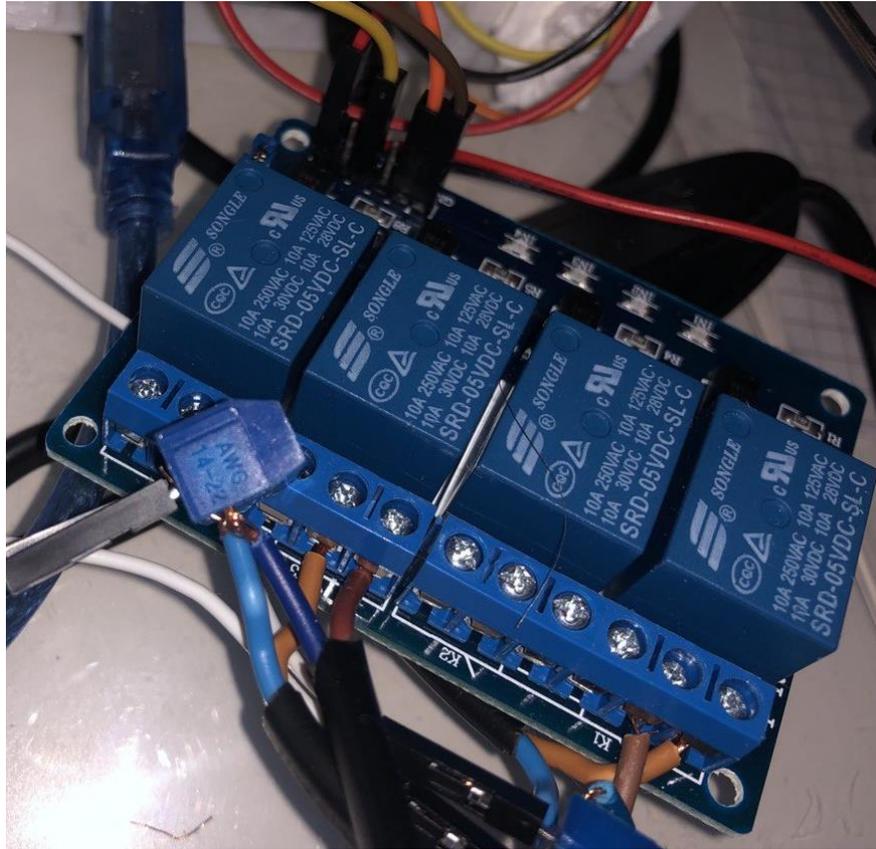


Figure 70: Relay connection

This picture shows the connection with the relay (I connected the pump to the relay number 1 and the heater aquarium to the relay number 3), in the final project I used electrical tape to close the connections and make the project safer. The rest of the circuit is the same as seen before. The code is almost the same, the unique variations are:

- Definition of two variables to control the relay (the relay is connected to Arduino by means two 4 cables, one is connected to the 5V alimentation, ones on the ground GND and the other two are needed to control the relay number 1 and number 3 and are connected respectively to the digital pin number 2 and number 1).
- The initial configuration of the variables, where I switch-up the heater aquarium and the pump (this only to see if they work properly, in fact after a delay of 1000ms chosen by me they will be controlled by the void loop).
- In the void loop, every 1000ms, if the temperature is less than 25 °C the heater turns on, if it is bigger than 30°C it turns off (it is important to notice that the heater has an internal

thermostat that turn off or turn on the heater when the temperature chosen is reached or not) and if the water level in the first tank reaches the 90% of the capacity or not.

The code remained the same. There are some problems in this new configuration since the internal thermostat of the heater goes in contrast with the thermometer waterproof putted by me. This point will be touched better in the next paragraph. First of all, I defined the variable useful for the relay:

- The relay needed for the control of the level of the water is connected to the digital pin number 2.
- The relay needed for the control of the temperature of the water is connected to the digital pin number 1.

```
int led1 = 3; //GREEN LED
int led2 = 7; //RED LED
int led3 = 5; //BLUE LED
int relelev=2; //pin of the rele of level
int reletemp=1; //pin of the rele of temp
int livello = 0; //level of the water measured by the sensor
int minlevel = 0; //minimum level that can be reached
int maxlevel = 500; //maximum level that can be reached
int maxtemp = 30;
int mintemp = 25;
```

Figure 71: Definition of the variable

I initialized the relay's output, and I have both on low (i.e. off) because I connect the heater aquarium and the pump on normally open.

```
void setup() {
  // Initiate the LCD:
  lcd.init(); //initialization of lcd
  lcd.backlight(); // function that activate the backlight of lcd
  Serial.begin(9600); //not necessary, I define it in case I want to display
  // on serial monitor
  irrecv.enableIRIn(); //initialization of IR sensor
  temp.begin(); //initialization of temperature sensor
  pinMode(reletemp, OUTPUT);
  pinMode(relelev, OUTPUT);
  digitalWrite( relelev, LOW); //there are both initialized
  digitalWrite( reletemp, LOW);
}
```

Figure 72: Void setup of the control configuration

Then I defined the if-else condition for the low condition of the temperature of the water and for the low level of the water. In this case I decided that if the level is less than 90%, I turn on the pump and if the temperature is less than 25 °C then I turn on the heater aquarium.

```

    if (livello < 90) //warning about the level: pay attention!
    {

digitalWrite( led1, HIGH); //green led ok because ok
digitalWrite( led2, LOW); // the red led is OFF because it is a bad thing
digitalWrite( led3, LOW);
digitalWrite( relelev, LOW); //turn off the rele when the level is less than 90%
delay(500);
}
    if (tc < mintemp) //warning about the level: pay attention!
    {

digitalWrite( led1, HIGH); //green led ok because ok
digitalWrite( led2, LOW); // the red led is OFF because it is a bad thing
digitalWrite( led3, LOW);
digitalWrite( reletemp, HIGH);
delay(500);
}
}

```

Figure 73: if conditions of low temperatures and level

In the last image below is show how the relay work in the case where the temperature reaches the temperature of 30°C and if surpasses 90% of the level of the water. It is important the fact that the two relays work different: to switch on the relay of the temperature the relay must be turned off (DigitalWrite to low) and the level of the water to be turned on should have DigitalWrite on high (this is due to the fact that they are connected in a different way) (viceversa for the case where water level is 90% and the temperature reaches the maximum defined from the variable).

```

    if (livello >= 90) //warning about the level: pay attention!
    {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Attention ");
        lcd.setCursor(0, 1);
        lcd.print(livello);
        digitalWrite( led1, LOW);
        digitalWrite( led2, HIGH); //display the red led because it is a bad thing
        digitalWrite( led3, LOW);
        digitalWrite( relelev, HIGH);
        delay(500);
        flag = 0; //once the level is less than 90, it turn on main menu
    }
    if (tc >= maxtemp) //warning about the level: pay attention!
    {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Attention ");
        lcd.setCursor(0, 1);
        lcd.print(tc);
        lcd.print(" Celsius");
        digitalWrite( led1, LOW);
        digitalWrite( led2, HIGH); //display the red led because it is a bad thing
        digitalWrite( led3, LOW);
        digitalWrite( reletemp, LOW); //turns off
        delay(500);
        flag = 0; // it turns on main menu
    }
}

```

Figure 74: If conditions of high temperature and level

The if-condition that turns on the pump if the level is higher than 90% and turns-off the heater if the temperature is higher than 30°C it also shows a message of warning on the LCD and turns on the red led when the condition is satisfied.

3.7 DIFFERENCE AND SIMILARITY OF THE TWO TYPES OF PROJECTS

As said before the electrical and mechanical part are the same, the unique variation is the control part of the heater and the pump.

The pro of the project without control system are:

- Simpler and it occupy less space because it hasn't the cable that connect Arduino to the relay and to the component.
- Everything can stay in the case and makes it easily transportable.
- In case of problem, the operator can turn on or turns off the component.
- It is not dangerous!
- This configuration has less problem than the automatized one since is simpler than the automatized (the relays increase the complex of the system).

The cons of the project without control system are:

- Since it is not automatized and fully controlled by Arduino it must be continuously observed by the operator.
- The operator must be focused on the LCD or on the LED in order to see the information needed to turn on or turn off the heater aquarium and the pump.

The pro of the project with control system are:

- It is fully controlled by Arduino, so the operator can focus only on the observation of the data and nothing else.
- There can be used another pump or another type of heater simply removing the existent and replacing with other;

The cons of the project without control system are:

- The system is more complicated and there are more cable and cannot stay everything in a single case.

- The thermostat of the heater interferences with the water-proof thermometer.
- It is more dangerous, so use it with caution!

Most of the program is similar in the two version, the unique variations are the definition of the two variables for the relay and the if-else conditions to activate or turn off the relays. Furthermore, the case can be used also for this version, there is only a problem about the relay because it cannot enter in the case, but it is needed a bigger case or another for the relay. The relay adds also more wires that must be connected to the socket and to the Arduino's board by means the relay. The flow stiches can be avoided because the maximum volume flow rate inside the silicon tube is less than the threshold of the flow switch.

3.8 POSSIBLE INDUSTRIAL APPLICATION

Modifying the microcontroller and using an industrial PLC instead of Arduino Uno, this project can be also used in a industry, for example a chemical one. There is a problem only with the level sensor because it is not reliable, and it is better to use one more robust or for example a float switch. Other consideration can be done on the pump and the heater used: I choose this micropump because the volume of water is very small and also the tank used in this thesis. But it is possible to connect a more powerful pump and a better heater, because the ones used by me was designed to heat a small reservoir of water with some fish inside it (and the maximum temperature that can be reached is about 35°C). There is other type of heater that can be used, as for example a heat exchanger

3.9 POSSIBLE IMPROVEMENT

Matlab can be used to store all the information of temperature (or water level) and see the variation. I report in this paragraph the collection for the temperature, but this code can be done identically for the water level. Following the guide in the bibliography [52] and uploading the file `adiao.pde` on the board, we can build a communication between Arduino and Matlab. This communication is necessary to Matlab to read the serial (I used the 9600) and communicate the information to Matlab. In this case on the Arduino IDE, I write only a code capable to read the information for the sensor of temperature. Then, on Matlab, using Simulink, if wanted, can be build the control part and in this case the code wrote in C/C++ on the Arduino IDE became simpler. Once I insert the code on the board, I start the communication between the board and Matlab using the following function `a=arduino('port')`. The port is the port used by the computer to communicate with the board (in my case I use the COM7. I can read the information of the port directly on Matlab and I can plot all the variation of the temperature, in real time, on Matlab and display it on a graph. In the following screenshot I show the Matlab Code wrote by me:

```
clear all
close all
clc
delete(instrfind({'port'},{'COM7'}));
tempArduino=serial('COM7','BaudRate',9600);
fopen(tempArduino);
title('Temperature read by sensor');
for time=1:100
    ylim([10 40]);
    xlim([0 100]);
    temp=fscanf(tempArduino,'%f'); %convert to double
    hold on
    plot(time,temp,'x');
    drawnow
end
fclose(tempArduino);
delete(tempArduino);
```

Figure 75: Matlab code

First, I define on the script the serial port (9600) used and the port of my computer (COM7). Then, with `fopen`, I read the information. Then I take the first 100 measurements (alternatively I can use a while cycle and read till I disconnect the board, but I preferred to read in a graph all the data) and I plot all my easements obtaining the plot show below. This is important to collect the measure of temperature during the experiment (the same discussion can be done for the water level).

Instead of the variable temp, I can also define a vector and collect my 100 measurements (or if I prefer more) and have all my measurements in this vector (instead of temp I will use temp(i), but first I define the variable temp as temp=zeros(1,100)).

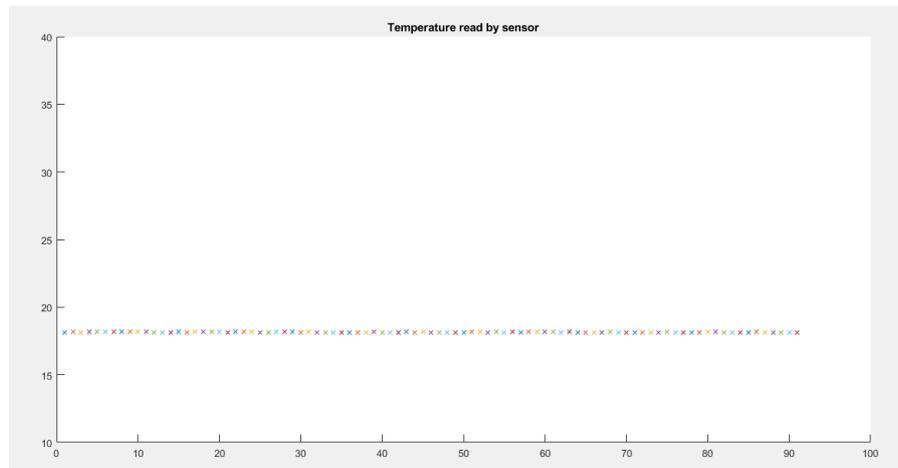


Figure 76: Trend of Temperature over Time

On the abscissa coordinates we have the number of measure (every step are 500ms, but this can be varied from the Arduino code in C++) in total I choose 100 measurements, that are 50 seconds(100*500ms). On the ordinate coordinates I have the measure of the temperature (I choose a maximum of 40 °C). The point in the graph above are real measurements did by me. The same discussion can be done for the water level. There is not necessary to compile a new Arduino code, but for this example I used the following one:

```
FileTempPerMatlab | Arduino 1.8.13
File Modifica Sketch Strumenti Aiuto
FileTempPerMatlab
#include <OneWire.h>
#include <DallasTemperature.h>

OneWire temp(10); //Define the pin used to read the temperature
DallasTemperature sensors(&temp);
void setup()
{
  Serial.begin(9600);
  sensors.begin();
}
void loop()
{
  sensors.requestTemperatures();
  Serial.print(sensors.getTempCByIndex(0)); //Se lee e imprime la temperatura en grados Centigrados
  Serial.println(" °C");
  delay(500); //500ms between two measure
}
```

Figure 77: Arduino code

I did this code to collect data of temperature, but there can be done another identically for the water level. If I define `Serial.begin(9600)` on the whole code did by me, I can use the first code presented using a `Serial.print`. The `serial.print` print out the data collected from the serial port as ASCII code. Arduino as only a serial port, used to communicate with the computer or other device. For the control Part, using Simulink, is necessary to download some library in order to write to the digital pin (low or high) where will be connected the relay. The input variable (temperature or water level) can be taken directly from Matlab from the script seen.

4 RELIABILITY OF THE SYSTEM

The reliability can be defined as the capability of a system to respect its technical specification during the time. A system is more reliable as the probability of having a failure decrease. Since in this thesis is used an Arduino board, I focus on the concept of fault tolerance taken in consideration by the embedded systems. Typically, there are two types of failure:

- Permanent failure: typically, are manufacturing defects or this defect appears as the device ages.
- Transient failures: are for example single event upset (SEU) due to the injection of charge in the substrate of the memory. If found, they can be repaired.

In this project the concept of reliability is applied on the component: as we can be seeing some components are more reliable than other due their architecture, for example the water level sensor is not a reliable component since after few times it barely works. In the next paragraph will be defined some concept about reliability necessary to this thesis.

4.1 FAILURE RATE

We can define the failure rate λ (of a component) as the number of failures every 1 billion of hours.

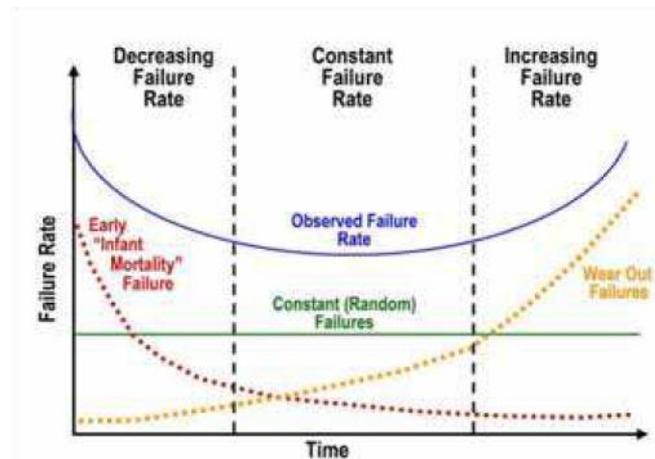


Figure 78 Bathtub curve of the failure rate (39)

Since a system is composed by more component, if a sub-component fails then all the system fails. The failure rate of the system is the sum of the failure rate of all the n -subcomponents of the system.

$$\lambda_{system} = \sum_{i=1}^n \lambda_n$$

4.2 RELIABILITY OF A SYSTEM

At the time equal to zero N components (equal each other) start to work until a time equal to T. If Nb are the components broken, the component remained good are equal to Ng. The reliability is equal to:

$$R(T) = \frac{Ng}{N}$$

I define now the function density of probability to have a failure as:

$$f(t) = \frac{dF(t)}{dt}$$

It is a function that tell us the probability to have a failure of a component between the time t and t+dt (an infinitesimal amount of time). There is a correlation between the failure and the reliability:

$$R(T) = 1 - F(t) = 1 - \int_0^t f(t)dt = \int_t^{\infty} f(t)dt$$

4.3 MEAN TIME BETWEEN FAILURES

It is the predicted variable between failures, and it is used for reparable system. It is the sum of the mean time to failures (MTTF) and the main time to repair (MMTR):

$$MTBF = MTTF + MMTR$$

For system very reliable MTBF is very close to MTTF:

$$MTTF = \int_t^{\infty} t \cdot f(t)dt = \frac{1}{\lambda}$$

In general, the MTBF is used for reparable system (in fact it measures the time between failures) and MMTF measure the time to have a failure that will break it irreparably. I didn't measure this

coefficient in my thesis because there are useful for system that works continuously, but I didn't few tests, this made the coefficient independent from the working time.

4.4 APPLICATION ON THIS THESIS

There was not any type of problem with the Arduino board: I did many tests with this board and every time it works perfectly. The problem are the water level and the heater aquarium because as will be see in the next paragraph there is a conflict between the internal thermostat of the heater and the thermometer.

4.4.1 Water level failure analysis

This component is very fragile: I did only 10-15 test with it, and it stopped working. Furthermore, I need to calibrate it with the following circuit because the values it takes are different from the type of liquid where it is put.

```
#define sensorPin A0

int level = 0; // there we save the water level

void setup() {
  Serial.begin(9600);
}

void loop() {
  int level = analogRead(sensorPin);

  Serial.print("Water level: ");
  Serial.println(level);

  delay(1000);
}
```

Figure 79: Code the calibrate the sensor

Then the serial monitor will show some values: I take the biggest one and I put it inside my code on the board to calibrate the sensor with the function map.



Figure 80: Serial Monitor

The problem consists in the architecture of this component: since it uses some parallel conductors in copper that vary its resistance when the water level change. But the fact that it is inside the water (and maybe it is not completely waterproof) and since it should be calibrating each time, the values it takes are not perfectly realistic. It has a bad reliability and for this reason is better to change this sensor frequently. After 20/30 test I changed the sensor with another. There are better sensors, but more expensive and more complicated that need relay or other device and cannot be simply connect to a board. For a generical purpose is less expensive to buy some level sensor and substitute and calibrate them each time.

4.4.2 Temperature and thermostat failure analysis

As said before there are some interferences in the measurements of the temperature through thermometer and the thermostat of the heater aquarium. In the code, I choose a temperature of 30°C to switch off the heater, and the same temperature is chosen, but the inaccuracy of the thermostat gives some bad behavior: since the thermostat is a device used to control the temperature, at parity of price it results less accurate than the thermometer, and can turn off the heater when the water doesn't have exactly 30 °C. The best way is not to use a waterproof thermometer as used by me but buy a better heater and put inside the water and all the job is did by the internal thermostat. The difference, in our case, is also in the architecture:

- The thermostat, generally, has a simple resistor with a variable resistance; The values of the resistance vary with the temperature (is named thermistor)

- The waterproof thermometer used by me, has a more complex architecture, because it implements different memory, diode, register, trigger and it is more accurate.

For this reason, is very complex to determine the reliability of the system, also because the delta between the temperature measured by the thermostat inside the heater and the thermometer can be different if we put the thermometer near the heater (where there is the resistance that heat up).

4.4.3 Flow switches

The reliability of the flow switches wasn't measured since the water flowing inside them never surpass the critical flow. The reason is the silicone tube: I choose some tube with a small area and the flow is fixed by the radius of the tube itself. The volumetric flow rate is given by:

$$Q = v \cdot A$$

Where v is the flow velocity and A is cross area of the tube where the water is flowing. The maximum volumetric flow rate is dependent of the pump, and it is equal to 600 l/h:

$$Q = v \cdot A = 600 \frac{l}{h} = 0,00016667 \frac{m^3}{s} = 0,1667 \frac{l}{s}$$

But the switches used by me has a maximum flow rate of $0,2 \frac{l}{s}$: i.e., they presence can be avoided since there cannot be overcome the maximum volumetric flow rate.

4.4.4 Micro Pump

The micropump used is very simple: it is activated by the board when the level is higher than 90%. Normally, if not connected to the board, it has only 2 modes: turned on when connected to a socket e turned off otherwise. It is simplicity in the architecture give us a reliable system without any type of remarkable problem. In fact, I didn't observe any type of problem during my test.

4.4.5 3D printer

There are some problems with the PLA used by me. I make 3 different cases till the final satisfactory result because the others they simply broke. This is due to the material used: the PLA

absorb the humidity and it became more fragile and also is harder for the extruder to work with a bad material because the diameter of the filament increment.

4.5 ACCELERATED LIFE TESTING (ALT)

The Accelerated Life Testing consist in the measure of the reliability inducing in the component stress and damaging it faster than if it operates on the field. In this test the device is subjected to shock, vibration, critical temperature and power cycling. It is necessary to define the acceleration factor as:

$$A_F = \frac{T_{FIELD}}{T_{TEST}}$$

It is the ration between the time that the component stay in the field and the time in the test for a particular failure. There are 2 types of ALT:

- Usage Rate Acceleration: in this case is increased the stress applied to the device. The stress can be a torture stress, a shake test or a bake stress. An example for Arduino can be the plug-in and plug out continuously of the board on power supply, because the current entering the board is the same as the current expected in the field, but in the field maybe the power is connected few times per days and in the accelerated test 10 times per minutes.
- Higher Stress Acceleration: in this case the device will have a higher stress than the ones that have on field. For example, make the Arduino board work at temperatures higher than those from the project, or it can receive higher current from power supply.

Starting from the result of this test can be build some model that can be then used to predict the reliability of the component on the field. Applying the accelerated test on my project, I observed that using frequently the water level sensor in water it stops working after few days, this is due to the fact that I put and remove frequently the sensor from water, and it is designed to remain for more time in a liquid. I removed and reconnected Arduino to my computer many times but it doesn't have any type of problem: in fact, I have had this board for two years and I used for many projects without any problem.

4.6 CONCLUSION ABOUT RELIABILITY

To better understand the reliability of the system is important to divide the two system (the controllable one and the not controllable). The non-controllable is more reliable because there is not any interference between thermometer and thermostat. The water level works the same in both configurations, but it is a component with a high failures rate: in case of emergency can be useful to disconnect or connect manually the pump. Using an industrial PLC or a more stable (for example Controllino mega) increase the reliability of the system but without changing the sensors the reliability of the complete system remains the same. It is not necessary to use a PLC in this application because the task required are perfectly did by the Arduino board. The material used for the 3D print can broke or damage the extruder. During this month I teste the board, the sensors and the other component and with except the water sensor all works good.

5 CONCLUSION

The system made during this thesis can be satisfactorily used as didactic test bench. For didactic purpose is strictly advised to use the version without the controlling part, because there are some risks using that version due to the electrical wires and the water of the tank near to the board. In this thesis was used many electronic and mechanical components merged together in order to obtain a complete system able to measure the temperature and the water level given as output an LCD message and a blinking led. There are used various sensor: for example, temperature sensor, level sensor and also an infrared sensor. The electrical part (without the heater aquarium and the pump) are inside a case designed by me and printed with PLA. I did some consideration about reliability on the final system and its component. This system is versatile: using other system and modifying the variable and little part of the code it can be used to control other thing. Furthermore, the control of the temperature and of the water level is not useful only in an industrial mixer, but it can be used also in other application. In conclusion, the final result is satisfactory, it is clear that is not robust as an industrial PLC with better sensor but changing some sensor with better one and using PLC instead of Arduino (for Example Controllino Mega, that can be used with the Arduino IDE and so the code is not needed to be modified) the system will result more stable and robust, and it will be perfect also for industrial application.

6 BIBLIOGRAFIA

1. **ZDNet**. ZDNet. *ZDNet*. [Online] <https://www.zdnet.com/a/hub/i/r/2018/07/07/4a431e8d-6cdf-43f6-a49e-acb5cfe2fd8d/resize/770xauto/26bbda2f6373dc5d7b141e1191279bca/industrial-robot.jpg>.
2. **Medium**. Medium. *Medium*. [Online] https://miro.medium.com/max/730/1*10w0pmiKLroIReN9M271-g.jpeg.
3. **NATIONAL INSTRUMENTS CORP**. NATIONAL INSTRUMENTS CORP. *NI*. [Online] https://zone.ni.com/reference/en-XX/help/372458D-01/lvsysidconcepts/si_step_response/.
4. **Novara, Carlo**. Automatic Control. Torino : s.n., 2019.
5. —. Automatic Control. Torino : s.n., 2019.
6. **Ganesan, Thomas George and V. De Gruyter**. *De Gruyter*. [Online] 2020. <https://www.degruyter.com/document/doi/10.1515/cppm-2020-2001/html>.
7. **Controllino**. Controllino. *Controllino*. [Online] <https://www.controllino.com/product/controllino-mega/>.
8. **Arduino**. Arduino. *Arduino*. [Online] <https://datasheet.octopart.com/A000066-Arduino-datasheet-38879526.pdf>.
9. **Atmel**. Wikipedia. *Wikipedia*. [Online] <https://en.wikipedia.org/wiki/ATmega328>.
10. **Schematic diagram panel**. Schematic diagram panel. *Schematic diagram panel*. [Online] 10 September 2017. <http://scematicdip.blogspot.com/2017/09/arduino-uno-schematic-diagram.html>.
11. **Arduiner**. Arduiner. *Arduiner*. [Online] <https://www.arduiner.com/it/prodotto/arduino-prototyping-prototipo-shield-protoshield-w-mini/>.
12. **DigiKey**. DigiKey. *DigiKey*. [Online] <https://www.digikey.it/-/media/Images/Marketing/Resources/Calculator/resistor-color-chart.png?la=it-IT&ts=4db603f5-4e9b-4759-84b7-21a04d18b1a8>.
13. **Visual LED**. Visual LED. *Visual LED*. [Online] <https://visualled.com/it/glossario/led-dip/>.
14. **Proto Supplies**. Proto Supplies. *Proto Supplies*. [Online] <https://protosupplies.com/guide-to-solderless-breadboards/>.
15. **Wikipedia**. Wikipedia. *Wikipedia*. [Online] https://en.wikipedia.org/wiki/Jump_wire.
16. **Physical Computing**. Physical Computing. *Physical Computing*. [Online] <https://makeabilitylab.github.io/physcomp/electronics/variable-resistors.html>.
17. **RS**. RS. *RS*. [Online] [https://it.rs-online.com/web/p/morsettiere-pcb/1930586/?cm_mmc=IT-PLA-DS3A-_-google-_-CSS_IT_IT_Connettori_Whoop-_- \(IT%3AWhoop!\)%20Accessori%20per%20morsettiere-_-1930586&gclid=CjwKCAjw64eJBhAGEiwABr9o2DdkaYxMKgiSW0WlIMtkulm9amxYxnzI9CJfCtmgIFwNN55URO9Jdh](https://it.rs-online.com/web/p/morsettiere-pcb/1930586/?cm_mmc=IT-PLA-DS3A-_-google-_-CSS_IT_IT_Connettori_Whoop-_- (IT%3AWhoop!)%20Accessori%20per%20morsettiere-_-1930586&gclid=CjwKCAjw64eJBhAGEiwABr9o2DdkaYxMKgiSW0WlIMtkulm9amxYxnzI9CJfCtmgIFwNN55URO9Jdh).

18. **Banggod.** Banggod. *Banggod*. [Online] https://www.banggood.com/Geekcreit-140pcs-U-Shape-Solderless-Breadboard-Jumper-Cable-Dupont-Wire-For-Shield-p-78680.html?utm_source=googleshopping&utm_medium=cpc_organic&gmcCountry=IT&utm_content=minha&utm_campaign=minha-it-en-pc¤cy=EUR&cur_warehouse.
19. **Progetti Arduino.** Progetti Arduino. *Progetti Arduino*. [Online] <https://www.progettiarduino.com/9-arduino-display-lcd.html>.
20. **Last Minute Engineers.** Last Minute Engineers. *Last Minute Engineers*. [Online] <https://lastminuteengineers.com/arduino-1602-character-lcd-tutorial/>.
21. **Arduino.** Arduino. *Arduino*. [Online] <https://www.arduino.cc/en/pmwiki.php?n=Tutorial/LiquidCrystalAutoscroll>.
22. **Tech Maker.** Tech Maker. *Tech Maker*. [Online] <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.techmaker.it%2Fdisplay%2F377-display-lcd-16x2-giallo-verde-retroilluminato-con-interfaccia-i2c&psig=AOvVaw1Wh8wnMiyPa4v1AuyCxb18&ust=1629739877981000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCNiT1KmUxfICF>.
23. **Joom.** Joom. *Joom*. [Online] <https://www.joom.com/it/products/5d78933e8b45130101c739ff>.
24. **Arduino.** Arduino. *Arduino*. [Online] https://create.arduino.cc/projecthub/arduino_uno_guy/i2c-liquid-crystal-displays-5b806c.
25. **Michele Ardito website.** Michele Ardito website. *Michele Ardito website*. [Online] <http://www.micheleardito.info/ma/arduino/how-to-measure-temperature-with-arduino-and-ds18b20-sensor/>.
26. **Maxim Integrated Products, Inc.** Maxim Integrated Products, Inc. *Maxim Integrated Products, Inc.* [Online] 2019. <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>.
27. **Circuit basics.** Circuit basics. *Circuit basics*. [Online] <https://www.circuitbasics.com/arduino-ir-remote-receiver-tutorial/>.
28. **facile, Arduino.** Arduino facile. *Arduino facile*. [Online] <http://www.arduinofacile.it/2020/04/22/controllo-di-un-led-mediante-telecomando-elegoo/>.
29. **Elegoo Relay 4 Channel.** Elegoo. *Elegoo*. [Online] https://cdn.shopify.com/s/files/1/0296/9026/5648/products/elegoo-relay-module-with-optocoupler-8-channel4-channel-dc-5v-arduino-stem-kits-elegoo-shop-238186_360x.jpg?v=1622707650.
30. **Decdeal.** Amazon. *Amazon*. [Online] https://www.amazon.it/Decdeal-Sommergibile-Acquario-Giardini-AC220-240V/dp/B073R9CDR3/ref=asc_df_B073R9CDR3/?tag=googshopit-21&linkCode=df0&hvadid=279834020359&hvpos=&hvnetw=g&hvrnd=12664380086686091712&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmndl=&hvlocint=&.
31. **RS.** RS. *RS*. [Online] <https://it.rs-online.com/web/p/flussimetri-e-indicatori-di-portata/3956978/>.
32. **Gems Sensor.** Gems Sensor. *Gems Sensor*. [Online] https://www.gemssensors.com/docs/default-source/resource-files/instructions/instructions_169175.
33. **Decdeal.** Decdeal. *Decdeal*. [Online] <https://www.decdeal.com/p-h14027-300.html>.

34. **Made in China.** Made in China. *Made in China*. [Online] <https://dongrong-silicone.en.made-in-china.com/product/TviJmbWAvDpK/China-High-Temperature-Silicone-Tubing-Hose-Silicon-Pipe-Silicone-Tube.html>.
35. **Creality.** Creality. *Creality*. [Online] <https://www.creality.com/goods-detail/ender-3-3d-printer>.
36. **Wikipedia.** Wikipedia. *Wikipedia*. [Online] <https://en.wikipedia.org/wiki/Stereolithography>.
37. —. Wikipedia. *Wikipedia*. [Online] https://en.wikipedia.org/wiki/Fused_filament_fabrication.
38. **Arditi.** Arditì. *Arditi*. [Online] https://www.amazon.it/gp/product/B07SPGRT7G/ref=ppx_yo_dt_b_asin_title_o09_s02?ie=UTF8&psc=1.
39. **Raghavan, Cher Ming Tan and Nagarajan.** Simulated Annealing for Mixture Distribution. *Simulated Annealing for Mixture Distribution*. Singapore : I-Tech Education and, 2008.
40. **Ebeling, C.E.** An Introduction to Reliability and Maintainability Engineering. s.l. : Waveland, 2005.
41. **R. Manzini, A. Regattieri.** Manutenzione dei sistemi produttivi.
42. **C. Bonivento, L. Gentili, A. Paoli.** Sistemi di Automazione Industriale. Architetture e Controllo.
43. **Marro, G.** Controlli Automatici .
44. **Pham, Hoang.** Handbook of Reliability Engineering. s.l. : Springer.
45. **Seneviratne, Pradeeka.** Building Arduino PLCs. s.l. : Apress.
46. **Kumar, Yathov Kumar Bala.** *Design of didactic test bench to verify the reliability of components and systems*. Torino : s.n., 2019.
47. **Cal', Michele.** *Automazione di un banco elettropneumatico con*. Torino : s.n., 2020.
48. **Automazione plus.** Che cos'è l'automazione? *Automazione plus*. [Online] https://automazione-plus.it/che-cose-lautomazione_71032/.
49. **Aliverti, Paolo.** *Arduino trucchi e segreti: 120 idee per risolvere ogni problema*. s.l. : Edizioni LSWR.
50. **Calderan, Pier.** *Stampa 3D: il manuale per hobbisti e maker*. s.l. : Apogeo.
51. **Bolzern, Paolo, Scattolini, Riccardo, Schiavoni, Nicola.** *Fondamenti di controlli automatici*. s.l. : McGraw-Hill Education, 2015.
52. <https://michelediluca.altervista.org.arduino-matlab-comunicazione-seriale>. [Online] <https://michelediluca.altervista.org/arduino-matlab-comunicazione-seriale/>.
53. <https://vrelqtroniq.blogspot.com.monitoreo-de-temperatura-con-matlab-y>. [Online] <https://vrelqtroniq.blogspot.com/2018/02/monitoreo-de-temperatura-con-matlab-y.html>.
54. [https://www.ansys.com.Predicting Electronic Parts Failures with Accelerated Life Testing \(ALT\)](https://www.ansys.com.Predicting Electronic Parts Failures with Accelerated Life Testing (ALT)). [Online] <https://www.ansys.com/it-it/blog/electronic-accelerated-life-testing>.

55. <https://en.wikipedia.org>. *Accelerated_life_testing*. [Online]
https://en.wikipedia.org/wiki/Accelerated_life_testing.