

3D Printed articulated arm robot with IoT capabilities

2018

A design study into printing off a robotic arm by means of 3D printing,
with actuation and IoT capabilities



By Floris Hemmeler,
Remi Vinel, Andrei Opris
and Joachim Mosseng

EPS programme | Politehnica University





Acknowledgements

The 3D printed robotic arm has been fully assembled and made working. The way this was achieved was by working at the Politehnica University for 4 months with the help of tutors and each other. The laboratory which has been shared with us has been open almost every day, allowing us to work whenever we wanted to, and for this we are grateful. We would like to thank Professor Diana Popescu for all the help and knowledge she has given us regarding this project. And likewise we have had equally a lot of support from Professor Radu Constantin Parpala.

Project summary

In this paper can be read the progress of the EPS project: 3D printed articulated arm robot with IoT capabilities. It all started with a meeting with the tutors that led to the start of the project and a few meetings between the tutors and the group, after which all the requirements from the tutors were brought to the table. These were noted down and put on the first page of a soon-to-be big report on the manufacturing of a small, 3D printed Robotic Arm. These meetings occurred a few times, so as to make sure that the demands/requirements were clear. The group has been divided so that each person puts in his own input. For example, the 3D printing was done mostly all together, same as with the report. However, each person has written about a different subject so that everyone can work together at the same time.

The product for the mid-term evaluation consists of a fully assembled 3D robotic arm with all the printed parts and connections. There has also been made a report on everything important about 3D printing and robots. The programming itself has been done for the final product, and can also be read in this report.

Concluded, it became clear that this project was very doable within the time span of 4 months. The 3D printed parts all seemed to be of decent to great quality and have good

resemblance to the original CAD files. The robot is fully assembled with a newly added base to give it more support. The actuators (motors) function very well and the robot is able to grab objects from the ground.



Figure 0-1: The finished robotic arm, 2018

Table of contents

Acknowledgements	1
Project summary	3
Table of contents.....	4
Chapter 1: Introduction.....	6
Chapter 2: Project management	7
Chapter 3: Research	10
3.1 Robotics	11
3.1.1 Robot architecture	11
3.1.2 Kinematics	13
3.1.4 Robotic Degrees of Freedom.....	15
3.2 3D printing.....	16
3.2.1. How it works / Generals	16
3.2.2 Cura	18
3.2.3 Table with contents.....	20
3.3 IoT and programming.....	21
3.3.1 Internet of Things - IoT.....	21
3.3.2 Concept/principles	22
3.3.3 IoT Platforms - Blynk & PTC-Thingworx.....	23
Chapter 4: Design and assembly	24
4.1.1 Table with different type of robots	25
4.1.2 Choosing the right model for the project.....	26
4.2 3D printing process.....	27
4.3 Robot assembly	28
Chapter 5: Actuation, Control and Internet of Things capabilities	33
5.1 Robot electrical design (actuation and control).....	34
5.1.1 Bill of materials.....	34
5.1.2 Arduino development board – connection with the shield	35
5.1.3 CNC Shield – drivers, motors and wiring diagram.....	36
5.1.4 Software – Arduino IDE, GRBL library, Universal G-code sender	41
5.2 Development of Internet of Things	44
5.2.1 Sensors - a part of IoT.....	44



5.2.2 Connecting sensors to Arduino - wiring, coding, data	46
5.2.3 Internet of Things - solution, IoT platform & Blynk app.....	51
Conclusions.....	58
Personal experience	59
Bibliography.....	61
Attachments	62
A, the print time table	62

Chapter 1: Introduction

This report is the work produced by four Erasmus students, participants in the EPS program. The participants are Andrei Opris from Romania, Rémi Vinel from France, Floris Hemmelder from the Netherlands and Joachim Mosseng from Norway. The EPS program is a positive way to meet the future demands of the ongoing globalization that is taking place in our world today. The world is getting smaller, and international companies can hugely benefit from the experience gained by exchange students. They will also perhaps look favorably upon the experience gained while abroad, in terms of knowledge about other cultures and the ability to adapt to foreign customs. The EPS program allows different engineering fields to meet, work together, form bonds and exchange knowledge from each other's fields, not to mention, improve their English skills. The group is made up of three mechanical engineering students and one civil engineer student. It is also likely to believe that exchange students will find themselves more comfortable in "foreign" environments after studying abroad. They are more likely to gain an enhanced interest in global issues as well as a broader general knowledge. In this regard, it's worth to mention the ongoing work with creating a green future for our planet. This problem cannot be solved by countries itself, but has to be part of an extensive international coordinated program where national borders must be ignored.

The group has been assigned the task to build a robotic arm. The arm has been 3D printed, and there has been made a custom part which is a big and heavy round base on which the arm is located. The arm can move and it has IoT capabilities.

The robotic arm is a well-known machine in the robotic field that most people are familiar with. The robot arm is used in a wide range of production as well as in the medical field. This robot arm will be cheaper, smaller and relatively simple compared to other arms on the market. This will nonetheless serve as a favorable level for this project, as the students in this team are all doing this for the first time.

Chapter 2: Project management

- **Creating something new**

A pre-designed robotic arm has been printed and assembled in this project. The printing material however might differ from what material the arms' owner used. Then the motors with all its components were chosen and assembled based on the required lift strength of the arm, and how it would fit within the assembly. What's new with this robot is that it will go through some structural adjustments and have IoT capabilities.

- **Defined objectives that address to a specific need**

- It needs to be fully 3D printed
- It has to be assembled before the mid-term presentation
- It has to have IoT capabilities
- It must have 4 degrees of freedom (the three basic ones plus one extra gripper)
- Make an improvement of the printed arm, like a custom part

- **The target group identified**

-Technical high schools and universities

Having a usable 3D printed robotic arm in a university is a nice way of showing a prime example of robotics and mechatronics. Students can take a look at the way the robot works, for example how it moves and how the motors make the gears turn. It can serve as a showcase model for engineering students, so that they in turn can make their own version of a robot or modify this existing model.

-Retail market

If production of the robot can be made cheap and affordable (let's say in the 100-200 euro range), people might buy them for whatever application they want. For example, a lazy owner of goldfish who doesn't want to feed (or forgets to feed) his fish every day. With the use of a programmable robotic arm it is possible to let the robot grab a can of fish food and feed the fish on a set time every day, so the owner doesn't have to.

-Factories

Both big and small factories with fully or semi-automated assembly lines use robotic arms and other types of robots. These small, plastic robots might just be what certain companies want in their assembly line. It is 3D printable, and fairly cheap and easy to produce. The only thing one needs is a 3D printer, material to print with and the actuation with wires, boards etc.

- **Clearly defined managerial responsibilities**

- Plan or schedule all the activities
- Make sure it's clear to the other team members
- Be in contact with the supervisors or teachers
- Making sure tasks are divided equally and fairly
- Making sure those tasks will be fulfilled before the agreed upon date

- **Start date and end date**

Start date: March 7th 2018

Mid-term: May 11th 2018

End date: June 6th 2018

- **Specified budget and resources**

To do this project certain resources are necessary. The required resources are the following:

- 3D printer big enough to print all the parts of the robotic arm
- Material for the printer
- The entire list of materials needed or order list, with all the motors actuators etc.
- A workspace with electricity for our personal laptops and for the 3D printer.

- **The chart**

To efficiently plan the project it was necessary to create a chart. In this chart can be read at what duration of time which attribute has been worked on. This can be seen in the GANTT chart down below. The main thing which took the most amount of time was the writing of the final report, which basically means putting the entire process on paper.

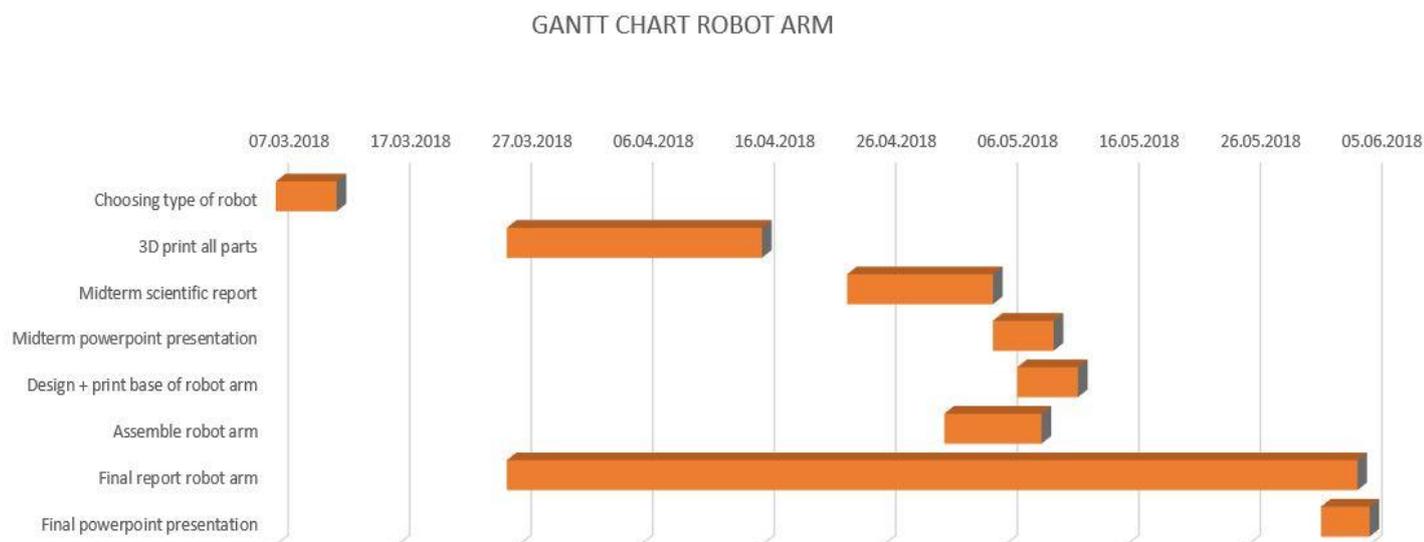


Table 2-1: GANTT Chart, 2018



Chapter 3: Research

3.1 Robotics

The first thing to be researched was robotics. The project is about making a 3D printed robotic arm, so it makes sense to start off with doing research about how robots work in general and how a robotic arm works in terms of robotics.

3.1.1 Robot architecture

There are many different types of robots. To reach the designated point, robots use different coordinate systems:

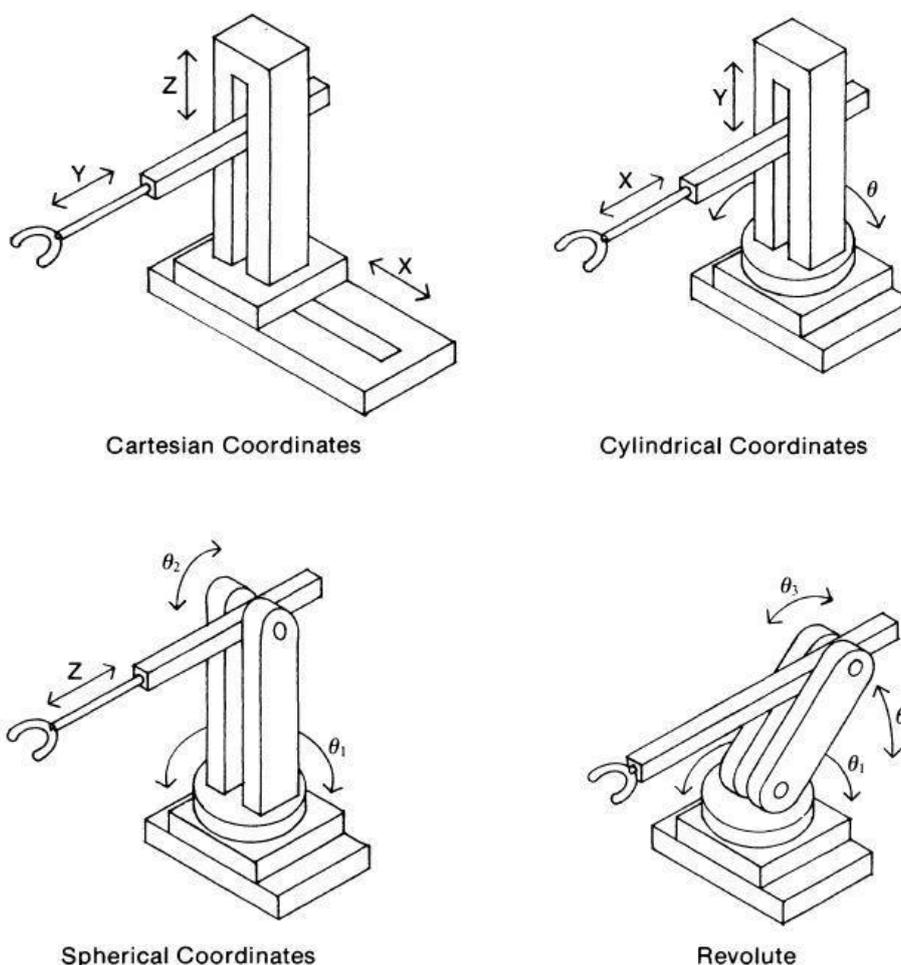


Figure 3-1: Robotic kinematics, (Poole, 1989)

The Cartesian robots are made up of three linear joints. It is the easiest to control, has the highest repeatability and has a good payload capability. However, they do require a larger work envelope than most other robots (Poole, 1989).

The cylindrical robot has a rotational joint at the bottom in addition to two linear joints. This makes the robot work envelope cylindrically shaped. They are especially a good option for pick-and-place jobs, especially suited for transferring materials from conveyor belts. This type of robot is also easy to program and has a good payload. It is generally incapable of reaching the floor, which is a disadvantage (Poole, 1989).

The spherical robots have one linear joint combined with two rotational joints. They are suitable for heavy lifting, small spaces and have a good reach. A disadvantage is the short vertical range. SCARA (Selective Compliance Assembly *Robot Arm*) robots are examples of this class (Poole, 1989).

The revolute robot has all rotational joints and makes it the most flexible. Because of this, it's the most used robot. The revolute robot can have a parallel linkage between the base and the upper arm. This design allows the motors to be placed at the base rather than having the motor in the elbow point, which saves some weight. This design allows the robot to have a compact design, and therefore requires less space in the work area. The end-effector may be moved more quickly and accurately because of the low weight in the elbow point. Most of these robots have articulated motion, which means that joint movement is linked together, limiting orientation changes in the end-effectors during robot motion. Because of the need for articulated motion, the computer controlling gets more difficult. A disadvantage with the revolute robot is the limited capacity with the arm fully stretched out (Poole, 1989).

3.1.2 Kinematics

“Kinematics, which is the English version of the French word *cinématique* from the Greek *κίνημα* (movement), is a branch of science that analyzes motion with no attention to what causes the motion.” (Jazar, 2006)

We distinguish between forward and inverse kinematics. Forward kinematics can tell you where the exact position of the end-effector would be given the certain angles provided to the robot actuators. This way of steering a robot is favorable when we know where the end-effector should be positioned and oriented, as this is much easier. However, when we don't know the exact place where the end-effector should be positioned, inverse kinematics comes in. It is much more difficult to use the end position information to calculate the resultant joint angles needed. The calculation needed for this is called an inverse kinematic problem. Rotation and transformation matrices may be used to solve these problems with DOF mechanisms. Solving the problems may take some time, which makes the robots ability respond in real time difficult. However, if the robot merely repeats the same motion indefinitely, the equations need to be solved only once. The values can be stored in a lookup table for future use (Poole, 1989). Robots with multiple degrees of mobility makes things more difficult in inverse kinematics. There will always be several ways to position an arm to do a certain job, which allows multiple solutions available for the inverse kinematic problem. This can be compared with a human hand picking up an object. There are almost infinitely ways of positions your arm, wrist, fingers etc. The different positions of doing this will all do the job, but are not all equally desirable. They are different in the way they offer the right flexibility, load-carrying ability, speed, accuracy or other factors. The computation time needed will increase quite a lot with the increase in number of joints (Poole, 1989). If we look at the forward kinematics, the calculations are much easier. To illustrate this, we can look at a single plane two DOF case:

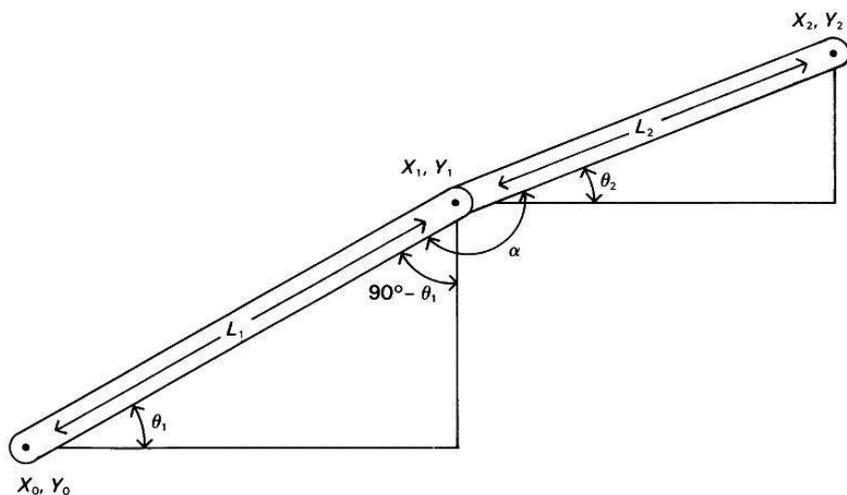


Figure 3-2 Positions of the Robotic Arm, (Poole, 1989)

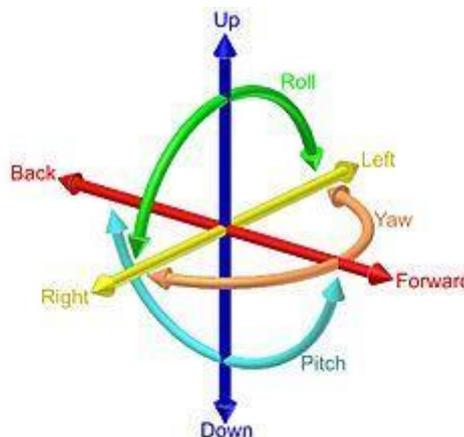
The positions of X_2 and Y_2 can easily be found by the equations:

$$X_2 = X_0 + L_1 \cos \theta_1 + L_2 \cos \theta_2$$

$$Y_2 = Y_0 + L_1 \sin \theta_1 + L_2 \sin \theta_2$$

3.1.4 Robotic Degrees of Freedom

The six degrees of freedom are forward/back, up/down, left/right, yaw, pitch, roll.



Degrees of freedom are not only applicable on robots, but are applicable on all moving objects. For now we will only look on the robotic aspects of degrees of freedom (DOF) and how they are defined. The first three DOF, the main

Figure 3-3 Degrees of Freedom, from Wikipedia

DOF, are the movements of the tool center

point TCP of the gripper on a robot arm. These movements (when looking at an arm) are in left-right, back-forward and up-down direction. However, if the arm wants to grab an object, there need to be more DOF in order to reach all positions in a workspace. These three DOF are often located at or near the TCP of the robot, and can be compared to the human wrist in terms of mobility. Depending on the demands of the robot to be designed for this project can the DOF of the robot be defined. Disregarding the need for a complete wrist, there need to be four DOF. The main DOF and one extra to roll the gripper. These DOF all have their own joint, which is powered by a small motor (actuator). (Poole, 1989)

3.2 3D printing

3.2.1. How it works / Generals

As mentioned above, the robot arm has been 3D printed as much as possible. 3D printing is a process of making three dimensional solid objects from a digital file. The printer is making the solid objects by putting layer by layer of the material, following the approach set up by the digital file. Each of the layers can be seen as a thinly sliced horizontal cross-section of the final object. You could say that 3D printing is the opposite of the traditional subtractive manufacturing which is cutting out pieces rather than leaving spaces empty that serves the same purpose. By using this method, it allows you to save a lot of material. (Thomas Campbell, October 2011)

Here is the principle of how the printers works:

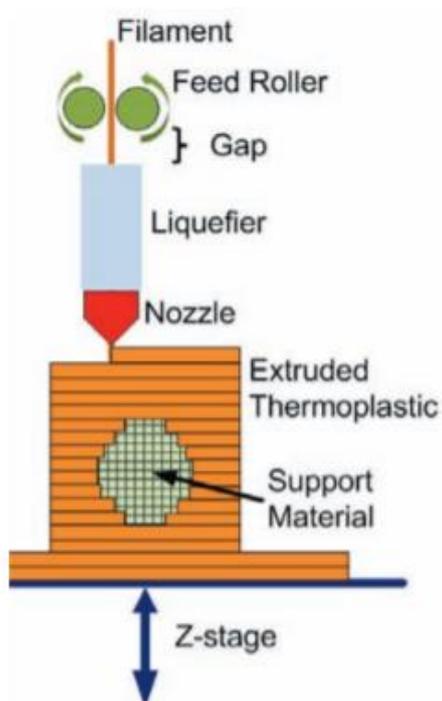


Figure 3-4: Fused Filament Fabrication, (Thomas Campbell, October 2011)

The team has mainly been using two printers, see Table 3-1: The two used printers.

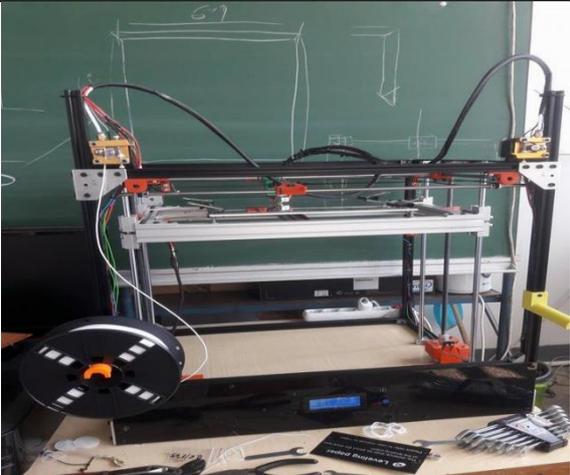
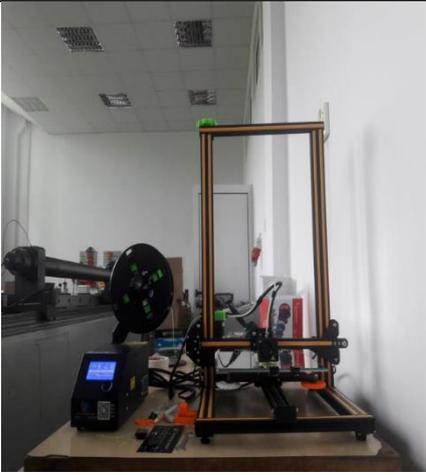
Custom-Made Printer	Creality CR-10
	
<p><i>Figure 3-0-1: Custom-Made printer, 2018</i></p>	<p><i>Figure 3-0-2: Creality CR-10, 2018</i></p>
<p>Nozzle Diameter: 0.6 mm</p>	<p>Nozzle Diameter: 0.4 mm</p>
<p>Filament Diameter : 1.75 mm</p>	<p>Filament Diameter: 1.75 mm</p>
<p>Bed Temperature : 75°C</p>	<p>Bed Temperature: 70°C</p>
<p>Extruder Temperature: 205°C</p>	<p>Extruder Temperature: 200°C</p>
<p>Material Type: PLA Filament</p>	<p>Material Type: PLA Filament</p>
<p>Printing Surface: Glass</p>	<p>Printing Surface: Glass</p>

Table 3-1: The two used printers, 2018

3.2.2 Cura

Cura is an open source 3D printer slicing application. Cura permits to transform the STL file which represent the shape of the part in g-codes which can be read by the 3D printer. More precisely, the g-code is download on a SD card and then this SD card will be connected to the 3D printer to transfer the g-code to the 3D printer. Cura also permits to configure some parameters, for examples the material used, the layer height, the infill, the heating temperature et cetera.

- **STL file**

“STL (an abbreviation of "stereolithography") is a file format native to the stereolithography CAD software. This file format is supported by many other software packages; it is widely used for rapid prototyping, 3D printing and computer-aided manufacturing. STL files describe only the surface geometry of a three-dimensional object without any representation of colour, texture or other common CAD model attributes.” (Wikipedia)

(Chakravorty, 2017)

- **Parameters**

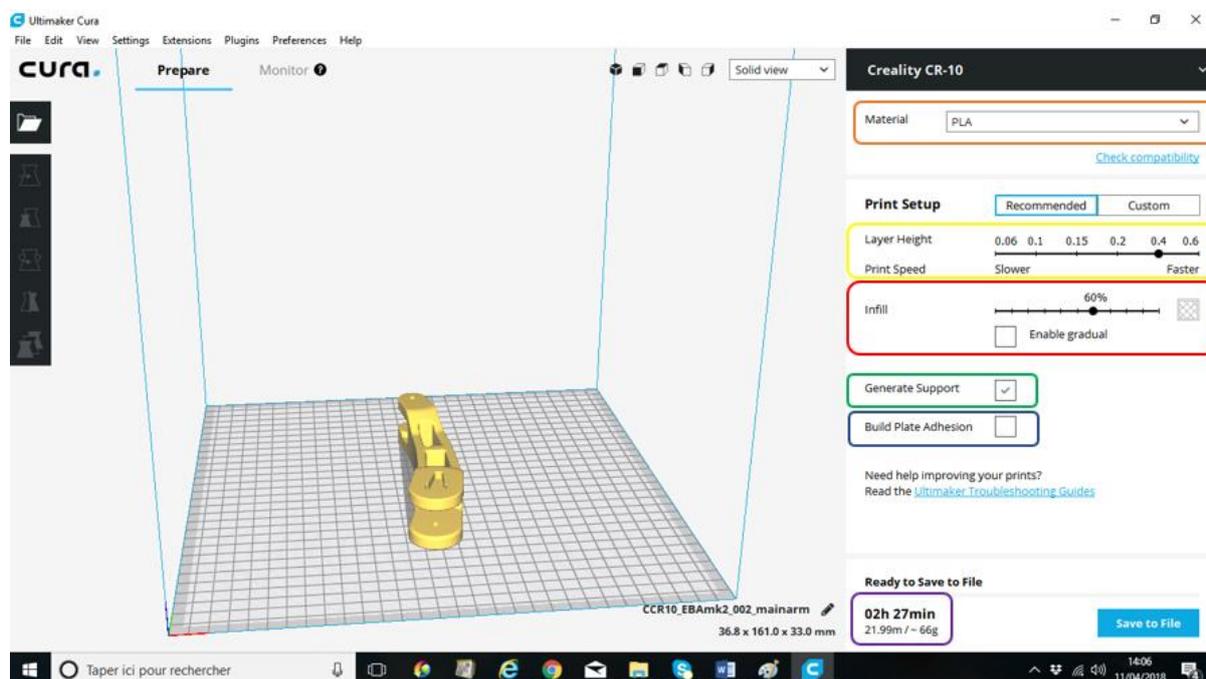


Figure 3-4: Cura parameters screenshot, 2018

First one must indicate to Cura which printer to use, in this case it's the Creality CR-10.

Then on the right, Cura proposes the main parameters which can be modified by:

-Layer Height: This determines the height per layer.

-Infill: This determines the percentage of infill.

-Generate Support: This will add a flat area around or under your object which is easy to cut off afterwards. It is necessary so as to maintain structural integrity and making printing possible.

-Build Plate Adhesion: this generates a structure to support parts of the model which have overhangs. Without these structures, such parts would collapse during printing.

Other parameters are more about the flow of the machine like movement speed, flow through the nozzle and wall thickness.

All these parameters have consequences on the printing time. Increasing the layer height, the printing time decreases, but so does the structural integrity. Decreasing the layer height will on the other hand increase printing time and will also increase the structural integrity. The same consequences apply for the infill; more infill will increase the printing time. Considering all these parameters, Cura calculates the printing time, the weight of the material used, and the length of the printing filament used. Cura can also print several parts at the same time, as long as they all fit on the printing bed.

3.2.3 Table with contents

In this part of the document one can find all the necessary information of the parts from the robot arm regarding 3D printing.

Before printing could start, it was necessary to determine the time it will take to print a part. Some parts take a long time to print, for example 6 hours, and some parts can be made within 10 minutes. The table with all this information can be found in the attachments. Here all parts are listed by name, a picture, file size, print time, amount and if it was already printed or not.

Determining how long each part takes is important, because you can't print all the big parts in one day. So to make the most out of the time available in the lab, it became clear that the best solution is to print the big parts one per day, with maybe some few smaller parts. Printing the big parts first and the smaller ones later gave a big advantage. When a big part has been printed, after waiting about 4-5 hours, it can be evaluated to see if the quality is good enough. Otherwise it has to be printed again, until the quality is good. The full list of parts and their printing times can be found in Appendix A.

File name	File picture	File size	Print time	Number of pieces	Printed
Base_Arm		421 kB	6 h	1	Yes

Table 3-2 Printing time per part, 2018

3.3 IoT and programming

3.3.1 Internet of Things - IoT

IoT or Internet of Things is a new concept which is massively increasing nowadays. This concept is about connecting physical objects to the internet and offering them the capability of communicating or exchanging data remotely. It is a relatively new idea and a lot of companies are investing in IoT development because of its huge potential. The devices which can be connected to IoT are from all domains of activity – cars, watches, smartphones, sensors, actuators, microcontrollers, medical equipment etc. People often refer to these objects as “things”. Basically, a “thing” is a physical device which has an IP address assigned to it and has the ability to communicate with other “things” over the internet infrastructure. Consumers seem to be very satisfied with their devices connected to IoT platforms and the unique features that they benefit of.

Although, IoT is becoming more and more popular and accessible among people, it is not the safest environment. Cyber-attacks and cyber-criminality (or generically – hacking) has a crucial impact on IoT because of the steady growth in the number of users. At the moment, it is hard for this industry to offer a secure and safe place since everything is developing so fast.

According to different sources of documentation about IoT available on the web, it is said that IoT consists of 3 principles called “the three C’s”. The three C’s of IoT are: Control, Connection and Cost saving.

- Control: First, the devices work on their own and are isolated. They can go through life on their own and interaction is necessitated directly between the user and the device.
- Connected: The next level is a device that can communicate with other devices.
- Cooperation-Coordination or cost saving: Finally, analysis of data from the edge to reveal new business, engineering or scientific insights.

(Haboud, 2016)

3.3.2 Concept/principles

As a general idea, IoT can be split into some general terms which can offer a better understanding about the concept. The main elements that has to be mentioned here are: sensors, gateways, cloud servers and mobile/desktop applications.

Sensors can transmit a big collection of information (distance, vibration, temperature, luminosity etc.) and they represent physical objects based on electrical circuits or devices. The gateways represent a link between devices and internet – an exchange of data over the internet platform. Cloud is a server where the information is stored, analyzed and sent further. A way of displaying the captured information is with mobile and desktop application, where the use can take advantage of the graphical interface to read collected data and send commands.

This path of information produces a chain where the collected data is moving along a channel (internet) and it goes back and forth between the user and the end-effector (a sensor, a motor, a device meant to collect and interpret data).

(Alluwar, 2017)

3.3.3 IoT Platforms - Blynk & PTC-Thingworx

When it comes to Arduino and IoT solutions, there are a lot of paid and free platforms for developing an application. A research was carried out in order to find the most suitable IoT platform with the best attributes and the research was based on finding the perfect platform, both for the user and the developers (our team). Platforms such as Thinger.io, Blynk.cc, MyDevice-Cayenne, Thingworx, ThingsBoard etc. are accessible for free for an unlimited or a limited period of time and these platforms are basically cloud platforms ready to store, exchange and expound data. Also, each platform offers the possibility of creating an application or a web-page with a graphical interface where the collected information can be presented with the help of different tools.

It was hard to choose the platform but the convenient solution was Blynk due to its capability of developing an Android or IOS application to show the results directly to phone using their application. Basically, Blynk app offers a unique chance for the developers to create an application for phones inside their main application. Blynk recommends itself as “The most popular mobile app for the IoT. Works with anything: ESP8266, Arduino, Raspberry Pi, SparkFun and others. Blynk will get you online and ready for the **Internet Of Your Things.**” according to their official website. (Blynk, n.d.)

Thingworx is a very interesting IoT platform and it offers solutions for industrial equipment but also for small projects. Thingworx platform comes with something called “IoT University” desired to help people learning this new concept. It has very useful courses and tutorials for free. (PTC, n.d.)

Thingworx is a very good source of documentation for beginners and it was very useful while developing this project. This platform offers a 1-month free membership and access to their servers and application. Even if the platform helps the newcomers to start with the fundamentals of the Internet of Things, in fact it is an “Industrial IoT” service - according to their official website. (PTC, n.d.)

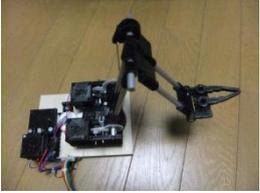
Overall, the background of this service runs on the same concept like any other IoT platform or cloud, but it is more complex and dedicated more for industrial purposes. It offers more stability and in-depth tools for programming and syncing any device to Internet of Things, but the graphical user interface is not so friendly - both for user and developer.



Chapter 4: Design and assembly

4.1.1 Table with different type of robots

At the beginning of the project, the team had to find a robot arm on the internet to use in the project. 4 of the arms the team found interesting were put in a table to organize the information.

MODEL	LINK	INFO ROBOT	ACTUATION	FILES
	https://www.thingiverse.com/thing:541997	3 degrees of freedom. Gripper. Power source: 4 A-size or AA-size batteries. 3V. Plastic and aluminum parts.	5 Mabuchi RE-140 motors.	STL
	https://www.thingiverse.com/thing:2520572	4 degrees of freedom. Gripper. 12 V 3A power. Plastic parts.	3 stepper motors. 1 servo motor (gripper)	STL
	https://www.thingiverse.com/thing:1015238	3 degrees of freedom. Gripper. Plastic parts.	3 Tower Pro MG90S servos 1 SG90 servo (gripper)	STL
	http://www.instructables.com/id/Robot-Arm-UStepper/	4 degrees of freedom. Gripper. Plastic and aluminum parts.	3 NEMA 17 stepper motors. 1 Micro servo (gripper)	STL

--	--	--	--	--

Table 4-1: Different models, 2018

4.1.2 Choosing the right model for the project

The robot arm had to meet these criteria:

- Pieces no longer than about 200 mm (3D printer can't handle bigger parts than this)
- Mostly printable parts (except for motors, screws etc.)
- Closed chain link
- Stepper motors

The team also had a few preferences such as a good design and the robot not being too small. Two of the robots in the table have a frame made up of steel and plastic, which does not fully comply with the criteria “mostly printable parts”. The one that stood out was the robot arm “MK2 plus” which is the second from the top robot arm in table 4-1. This robot has a fairly good size, mostly printable parts, well designed and stepper motors. The choice to go for this robot was easy for the team as this is the most suitable robot arm for the project.

4.2 3D printing process

After all the parts were 3D printed, it became obvious that some of the parts were not accurate enough. Small parts that were meant to fit together, simply did not fit without having to cut out plastic in certain places. This was solved by using another more accurate printer with a smaller nozzle. The parts that needed to be of higher quality were now printed again with a 0.4 mm nozzle instead of 0.6 mm nozzle. This gave better results, but on the other hand, it took longer time to print. This printer used green plastic, which is why the robot consists of both white and green parts.

The most difficult parts to print were the ten bearing balls for the bearing at the base of the robot (figure 4-1). These balls have a diameter of 5 mm, and the spherical shape must be very accurate in order to serve as a moving part in a bearing system. After printing the bearing balls on both printers, the results were far from satisfying as the surface of the balls were not accurate enough. The team also had access to a third 3D printer that was using ABS material. This printer gave a much better result, but the bearing balls still did not have the desired perfect spherical shape at the surface. By polishing each of the ten balls with sandpaper, the result became slightly better, and when assembling the bearing, the result showed to be satisfying.



Figure 4-1: Bearing, 2018

4.3 Robot assembly

The assembly of the robot met a few problems that needed to be addressed as mentioned above. Once all the parts were printed with satisfying quality, the team had to figure out what type of screws/bolts to use. At first, the team did not have any screws or bolts so the robot was first assembled with some cables that were lying around in the workshop. After seeing all the parts together fitted with the cables, the team came up with a solution with steel rods in 3 mm and 4 mm diameter with threads only at the ends. At the ends, nuts were going to make sure it remained together. The team made several trips to shops such as Brico Depot both in and outside of Bucharest to find this type of rods but it was nowhere to be found.

When all the printed parts were done and laid next to each other it was possible to assemble the robot. The arm has a lot of joints, because it's a closed chain working robot. These joints have holes going through them, which require a rod or pin with a mechanism that secures it in its position. It is important that these joints move without too much friction. The less friction there is, the easier it is for the motor to lift objects fluidly. The decision has been made to go with metal bolts and nuts, which were slightly smaller in diameter than the holes were so as to lower the friction. The ends were secured with nuts so that these bolts don't fall out of their joints. The best option was using metal bolts which only have thread on the end of the bolt. This way no thread is touching the inside of the joint, so the amount of friction is minimal, see figure below.

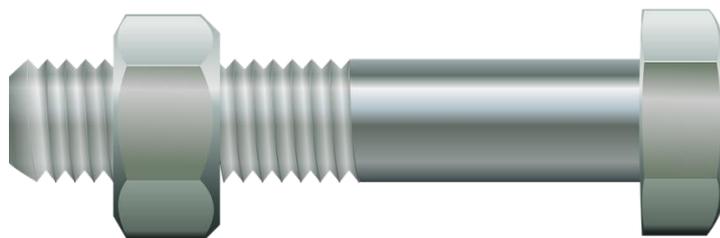


Figure 4-2: Bolt with threads only at the end, Google images, 2018

However, for the dimensions this robot has, there are no existing bolts long and thin enough so that no thread is touching the inside while still leaving thread on the outside for a nut. Below can be seen that the hole on the left has been filled with a bolt with a nut on the end. This bolt has thread all along its length.

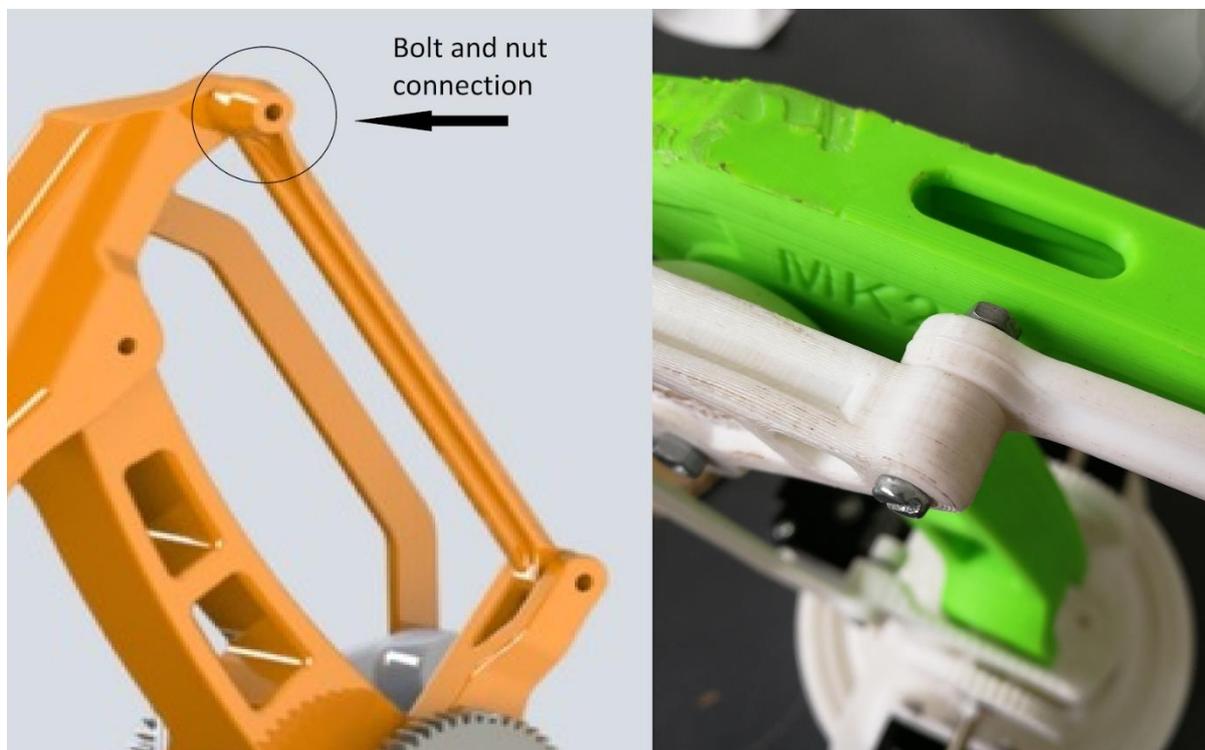


Figure 4-3: Bolt and nut connection, 2018

The base is made up of five parts:

- 2 bearings holders
- Main base
- Base-low
- Base-low gear

To have a better understanding of the difference between the parts, please see the attachments at the end of the report: “A, the print time table”.



Figure 4-4: Base, 2018

All the components of the base have been assembled in the following steps:

First, a nut was inserted in a hole in the main base. Then two screws allowed to fix the base-low with the main base.

After that, one bearing holder was placed under the base-low and the other bearing holder on the base-low gear, as well as the 10 bearing balls between the two bearings holders.



Figure 4-5: Base, 2018

Finally, the base-low and the base-low gear as well as all the parts between them, have been fixed thanks to a socket screw and the nut that is mentioned before.



Figure 4-6: Base, 2018

When the assembly was finished, the base wasn't stable because of the bolt sticking outside at the bottom as seen on figure 4-6. The team decided to design a stand to put the base on it. This stand has been designed on CATIA V5. The stand is by far the biggest part of the robot.

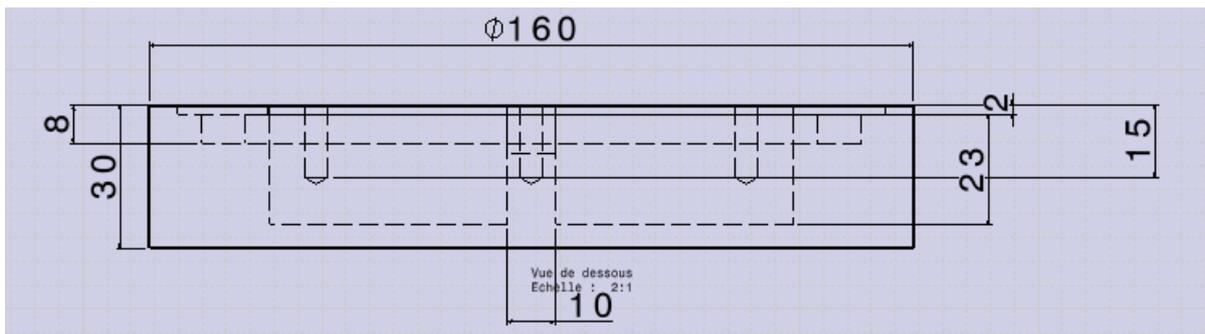


Figure 4-7: Base, 2018

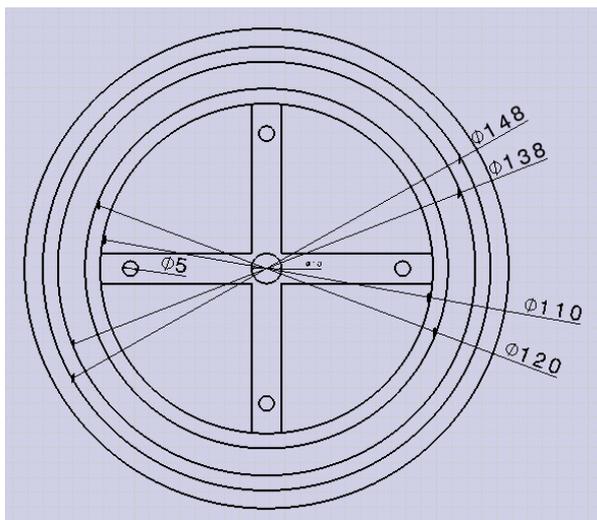


Figure 4-8: Base, 2018

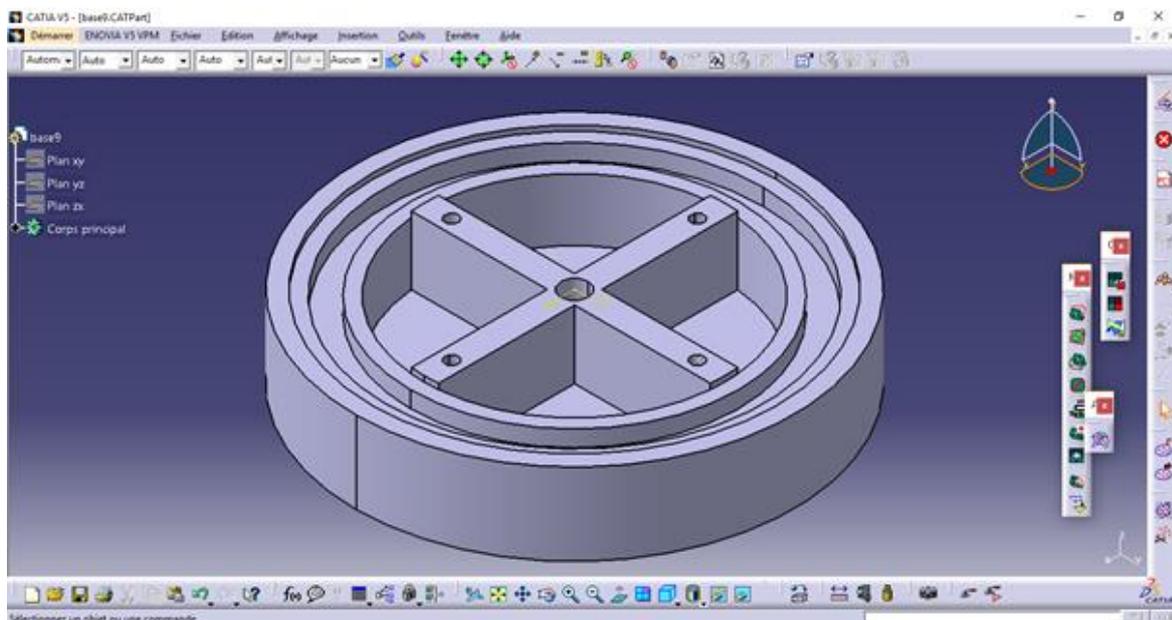


Figure 4-9: The CATIA base, 2018

As shown in the figure above, the stand has four quarter-circle holes. These holes will serve as chambers to put some heavy material in if needed when the arm is operating, for example sand. Indeed, this made the robot arm heavier and much more stable.



Chapter 5: Actuation, Control and Internet of Things capabilities

5.1 Robot electrical design (actuation and control)

5.1.1 Bill of materials

After researching and choosing the most suitable robotic arm for the project, the next step was to create a bill of materials containing all the necessary parts in order to actuate the robot.

No.	Piece	Quantity	Total Price (RON) VAT included
1.	Arduino Mega 2560 Development Board	1	43
2.	Arduino CNC Shield	1	48,8
3.	Arduino D1 WIFI WeMos Shield	1	49,14
4.	Servo Motor SG90	1	10,88
5.	Stepper Motor NEMA 17	3	150
6.	Power Supply 12V 15a	1	64
7.	Male – Female Wires	40	13,5
8.	Male – Male Jumper Wires	30	9
9.	Power Cord for Power Supply	1	8
10.	Crocodile Wires	6	4
11.	A4998 motor driver	3	29.1
12.	Stepper motor cable	3	Free
TOTAL			429,42 RON ~ 92.70 EURO

Table 5-1: Bill of materials, 2018*

*The bill of materials was created by researching different possibilities of actuating a robot. The result also represents a good relation between quality and price of the chosen items.

Moreover, the team improved and changed the bill of materials attached to the project found on the internet. Different items were chosen to create a better prototype and to improve the way of the movement of the initial robotic arm. Some work was put down to contact a lot of Romanian companies that could provide the needed parts for the robot. Everything was ordered and delivered from <https://ardushop.ro/> - an online electronics shop with better prices comparing to other similar companies/retailers.

5.1.2 Arduino development board – connection with the shield

Arduino represents an open source environment (hardware and software) which designs boards and microcontrollers for building and controlling digital devices.

On the market, at the moment, there are several models and types of Arduino boards suitable to be integrated in the project.

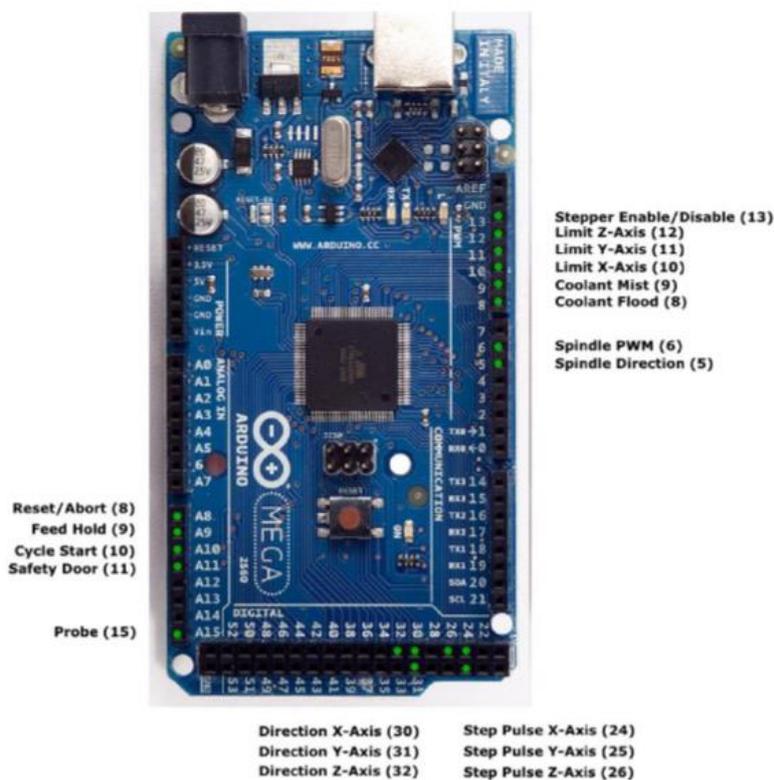
According to the bill of materials, Arduino Mega2560 was chosen as main board for the robotic arm. Most of the robotic arms found on the internet in the research phase were using Arduino UNO, a less powerful board.

Mega2560 was chosen over UNO because Mega2560 is more powerful, it has a better memory and response time, it has more pins for wiring (54 input/output pins on Mega2560 comparing to 16 input/output pins on UNO) and it can also be used for more complex projects in the future.

By choosing Mega2560, the team also solved a compatibility issue between the board and the CNC Shield because the shield was especially made for UNO boards and it has no direct connectivity with Mega2560. This issue was solved by wiring necessary pins from board to the shield – this was also a good starting point for learning the basics about Arduino and electronics.

This part also included a research on finding a wiring scheme or diagram between Mega2560 and the CNC Shield. After finding the scheme below (figure 5-1), the team managed to wire everything and get a good connection. Each marked pinout with green corresponds with the pins on the CNC shield with the same name. For wiring, male – female connectors were used, and in the end, protection cables for wires management to separate and group the wires was used. This is very helpful to keep track of the different cables when unplugging them.

The wiring and wire management is shown in figure 5-2. Black and yellow tubes were used to group and protect the cables.



Arduino Mega 2560 Pins compatible with the CNC shield

Figure 5-1: Pinout for Mega 2560 retrieved from <https://github.com/qrb/qrb/issues/390>



Figure 5-2: Wire management – photo taken while performing laboratory tests

5.1.3 CNC Shield – drivers, motors and wiring diagram

This subchapter will present information about the CNC Shield and the way everything was connected on it. A conclusive table (table 5-2) with pictures is shown below and it will offer a perspective on what was used for this step.

No.	Component	Quantity	Image
1.	CNC Shield	1	
2.	A4998 Motor Driver	3	
3.	NEMA 17 Stepper Motor	3	
4.	SG90 Servo Motor	1	

Table 5-2 : Components list

The CNC Shield is an Arduino Shield made for powering up and driving motors. It's widely used on home-made 3D printers and CNC machines due to its good performance. This shield was chosen because it is sufficient for this project, and it is reliable and cheap compared to similar devices.

This shield has 4 motor spots, one for each axis (X, Y and Z) and one for duplicating or cloning any of those 3 axis. It requires a 12V to 36V power supply in order for it to work. That is why a 12V power supply with 15A was used – this source is sufficient for the motors and will not burn or damage them.

For the safe running of this shield, it is recommended to use motor drivers for steppers - A4988 motors drivers are safe and they make part of a classical build. These drivers come equipped with small aluminum heat-sinks (to prevent the device from heating too much) and the heat-sinks are easily mounted on the driver by gluing them with double adhesive tape on the top of the driver. This is an easy and not time-consuming process which will boost the safety and the capacity of the drivers.

The next step is to mount the drivers on the CNC shield by plugging the male pins of the drivers into the female pins of the shield. It requires a bit of attention when the plugging is done because there is only one way of correct mounting, incorrect mounting can cause damage to the CNC shield when it is powered up. The best way to connect them is to plug the driver's pins with the same name into the motor spots of the shield with the same name (ex: ENABLE pin of the driver with ENABLE pins of the motor spot). The result can be seen in the picture (figure 5-3) that was taken after the mounting process. Now, the stepper motors can be safely plugged in without any risk.

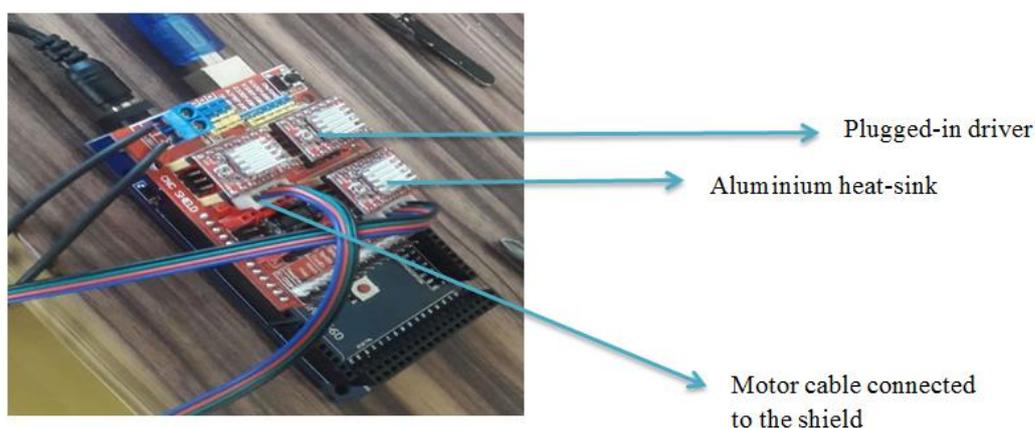


Figure 5-3: Representative photo of our build

In the picture above (figure 5-3), it is possible to see that the drivers are plugged in and the heat-sinks are mounted. The heat-sinks (the small aluminum parts) were working well, as they

feel warm when touching them, meaning that they are transferring heat from the electronics into the external environment (the air).

Now that the drivers have been installed, the next step is to wire the motors to the CNC shield. The stepper motors have been wired with special wires designed for stepper motors right next to the drivers. Also, the same procedure for the servo motor, but the wiring pins/spots are different.

The team built a wiring diagram with a free open-source software called Fritzing (it can be downloaded here for free: <http://fritzing.org>) and a description on what the team did can be seen. The following diagram/scheme (figure 5-4) will provide a strong information of the process.

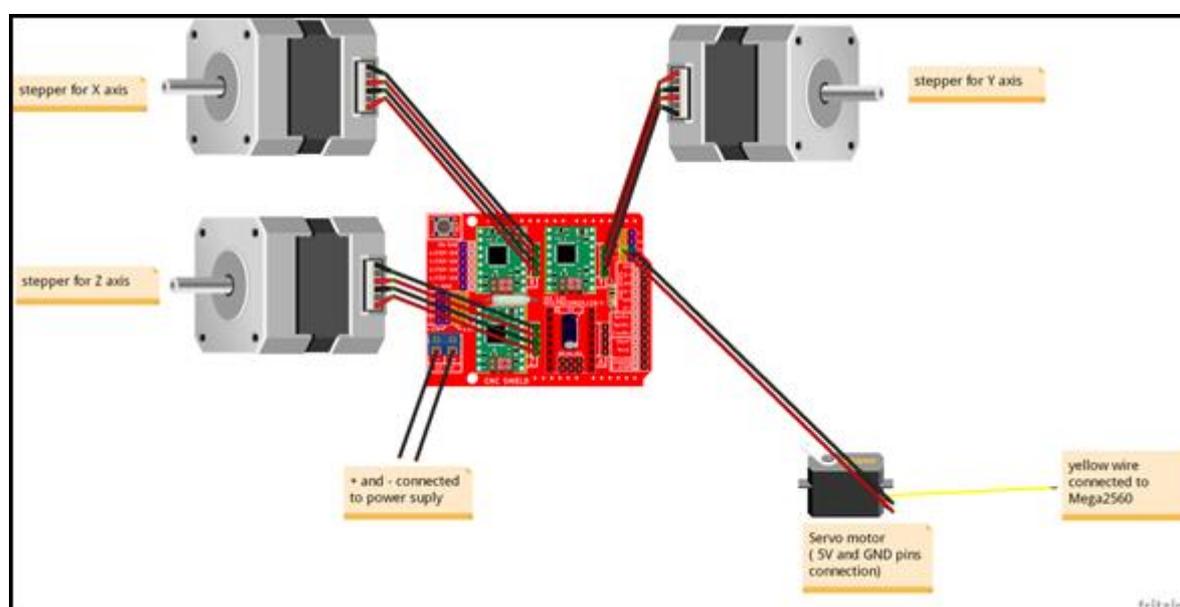


Figure 5-4: Wiring scheme created in Fritzing software, 2018

Figure 5-5 shows that all 3 stepper motors are wired and ready to be tested. The servo motor has 3 cables connected to it. 2 are wired to the CNC shield and 1 is wired on the Mega2560 on pin number 7 (pin 7 is shown in figure 5-1 and it is PWM pin – Pulse Width Modulation pin). It was not possible to connect all 3 cables on the CNC shield because of the compatibility issue between the main board and the shield.

The blue relay on the CNC shield represents the spots where the + and – cables from the power supply had to be wired in order to power everything up. The CNC shield requires an external power supply instead of a charger/USB cable, same as for Mega2650 – this is because of the high current needed to supply the drivers, motors and the shield itself.

After all these steps were made, there was a connection between all the components needed to actuate the robot. Anyway, it was still not possible to move or test the motors without some specific codes or libraries loaded on the main board. Next chapter will contain further information about coding.

Figure 5-5 shows what it looks like after wiring and connecting everything.

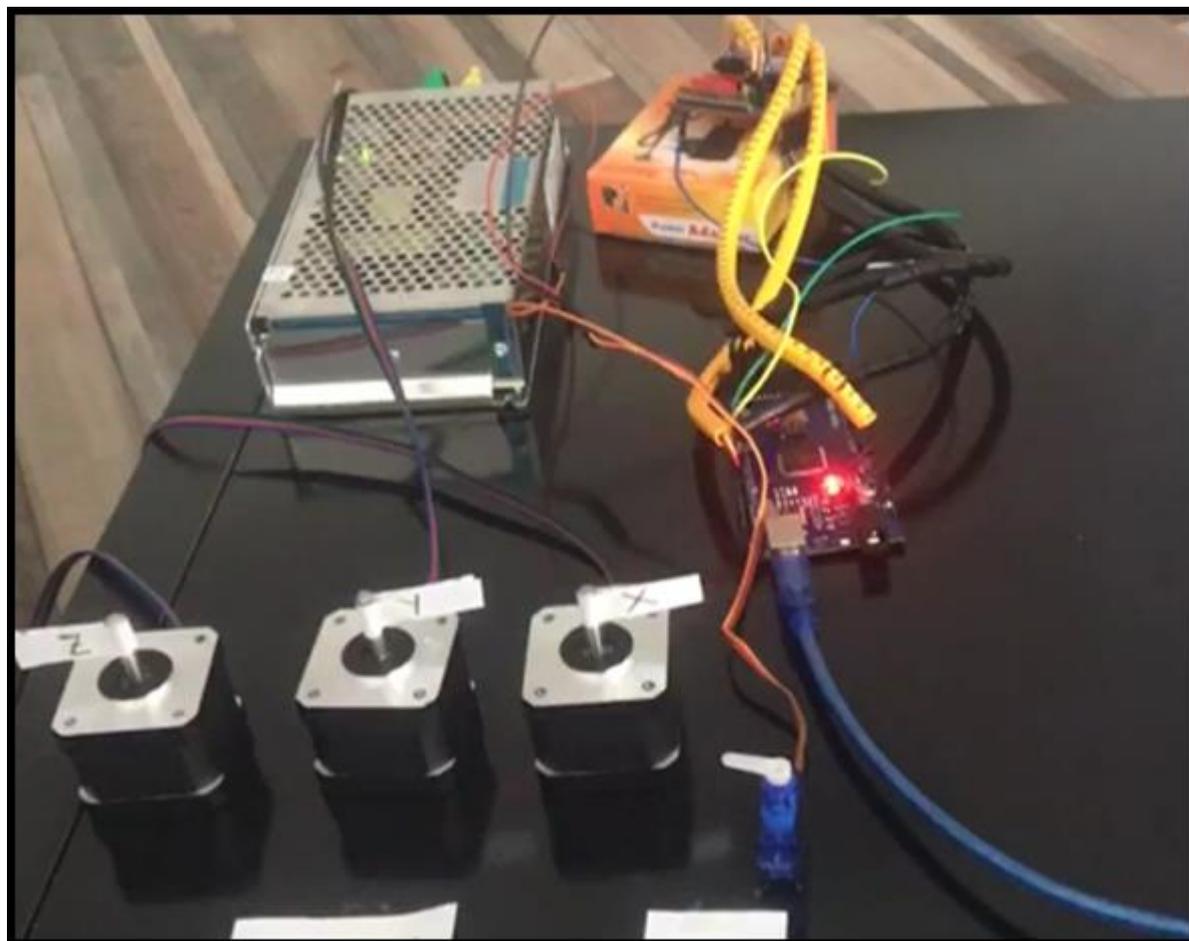


Figure 5-5: Photo of the final build, 2018

5.1.4 Software – Arduino IDE, GRBL library, Universal G-code sender

Now that all the components have been physically connected, there needed to be a digital connection as well, meaning a way to communicate between all the components. This has been done by using Arduino IDE software for coding Arduino boards.

Arduino IDE is an open-source software which grants an easy way to write and upload a code to any Arduino board. In order to use this software, Mega2560 needs to be connected to a PC with a USB cable and then open the software before it can upload the necessary codes and structures. For this project, GRBL library has been used to flash it to the board through Arduino IDE software.

“Grbl is a free, open source, high performance software for controlling the motion of machines that move, that make things, or that make things move, and will run on a straight Arduino. If the maker movement was an industry, Grbl would be the industry standard.” According to github community – retrieved from <https://github.com/gnea/grbl/wiki> (14/04/2018)

This GRBL library will help to communicate and actuate the motors. Below, (figure 5-6) is a screenshot taken from the laptop with the GRBL library right before compiling and uploading it to Mega2560 board.

It can be seen that in this GRBL library, there is included many other libraries such as gcode.h, cpu_map.h, protocol.h etc.

Basically, the GRBL library includes many other libraries in it. It can be stated that this is a good collection of well-chosen libraries, in order to create the perfect environment for actuating the motors.

Installing, building and compiling this library to the main board is a simple process and one of the basics of Arduino IDE software. With this GRBL library it's not necessary to create or write any additional code - only if wanted, because it will enable the necessary things in order to control the motor with another software. In this projects case, the software is Universal G-code Sender which will be presented in the next lines of this subchapter.

```

grblUpload $
#include <config.h>
#include <coolant_control.h>
#include <cpu_map.h>
#include <defaults.h>
#include <eeprom.h>
#include <gcode.h>
#include <grbl.h>
#include <limits.h>
#include <motion_control.h>
#include <nuts_bolts.h>
#include <planner.h>
#include <print.h>
#include <probe.h>
#include <protocol.h>
#include <report.h>
#include <serial.h>
#include <settings.h>
#include <spindle_control.h>
#include <stepper.h>
#include <system.h>

/*****
This sketch compiles and uploads Grbl to your 328p-based Arduino!

To use:
- First make sure you have imported Grbl source code into your Arduino
  IDE. There are details on our Github website on how to do this.
- Select your Arduino Board and Serial Port in the Tools drop-down menu.
  NOTE: Grbl only officially supports 328p-based Arduinos, like the Uno.
  Using other boards will likely not work!
- Then just click 'Upload'. That's it!

```

Figure 5-6 : Screenshot of Arduino IDE running GRBL library

Once uploaded to the board, everything is set to go, but a software to control the motors is still needed. After some research on the internet, it became clear that the best solution compatible with GRBL library was the Universal G-code Sender.

Universal G-code sender is a small application with a graphic user interface that allows the user to control the motors from it. There are buttons for moving the stepper motors on each axis X, Y and Z (buttons for X axis: X+ and X-; buttons for Y axis: Y+ and Y-; buttons for Z axis: Z+ and Z-) and even for controlling the servo motor. An earlier version of this software was not able to move the stepper motor, but in the new releases a stepper motor can be actuated by writing a command line into the command tab: M03 Sx (where x variable can be replaced with a value representing the rotation angle value).

The servo motor is directly connected to the gripper and is opening and closing it in order to grab and release objects. For the maximum opening of the gripper is the command M03 S90 and for fully closing the gripper is the command M03 S10.

Also for the stepper motors, the step size in millimeters can be edited, and it is possible to enable keyboard movement. With the keyboard movement, the stepper motors can be controlled from the keyboard using arrow keys (left arrow for X- , right arrow for X+ , forward arrow for Y+ , backward arrow for Y- , PageUp key for Z+ and PageDown key for Z-). Regarding the servo motor, there is no key bind or button available – the fact that you have to add the main value (the angle) might explain this reason, since a servo and stepper motor are different.

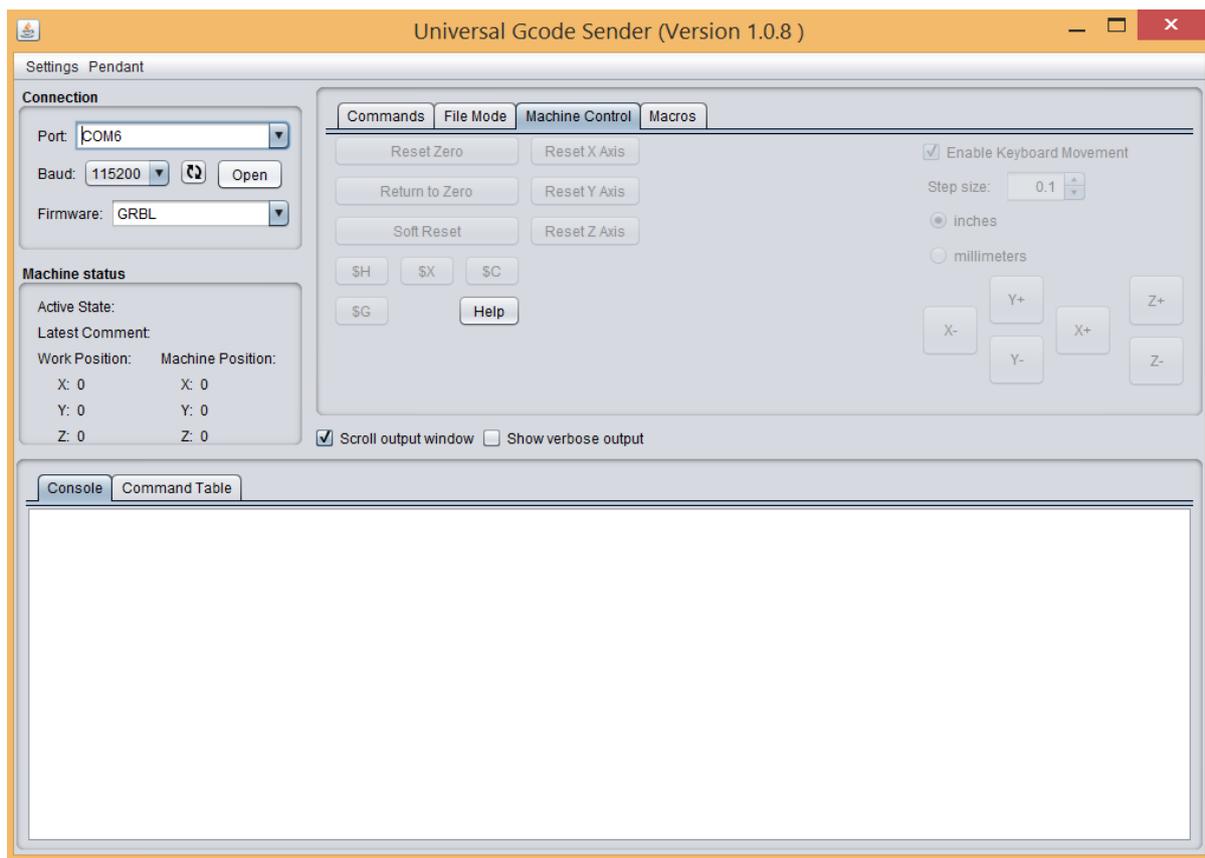


Figure 5-7: Universal G-code Sender software version 1.0.8

Figure 5-7 shows the interface of the application. To start it, it is needed to select the COM Port (USB port where the Arduino board is connected), the firmware (in this project GRBL) and then click the Open button. The COM Port can be found in Windows Device Manager (Windows + X for shortcut path). This way it's possible to control the 4 motors.

5.2 Development of Internet of Things

5.2.1 Sensors - a part of IoT

Sensors are indispensable when talking about Internet of Things and they represent a key in developing and manufacturing IoT applications. On market, there are several types of sensors able to fulfil user's requests and expectations. Since sensors are present everywhere in our lives, Internet of Things help them to be displayed and interpreted with the collected data through some especially created applications.

A similar reasoning was applied to our project – the connectivity of the robotic arm with Internet of Things represented one of the project's purposes. The robotic arm prototype, presented in this project, is manufactured and designed to work in small industrial activities and having an IoT implementation would boost user's experience.

Sensors ready to monitor different aspects and conditions of the working environment were added to the project. It is really important to know more things about the workplace, so you can have a better understanding about the efficiency and reliability of a system.

#	Sensor	Sensor Role	Unit of measurement	Image
1	DHT22	Temperature & Humidity	celsius degrees & a value from 0-100% for humidity	
2	HC-SR04	Distance	cm – for values greater than 1 cm mm – for values less than 1 cm	
3	SW-420	Vibration	a digital output number regarding vibration amplitude - not a standard unit of measurement	

Table 5-3: Sensors table

The table 5-3 above introduces the sensors which are going to be used in the project. There are 3 sensors ready to monitor different data such as temperature, humidity, distance and the vibration.

- **DHT22**: was chosen to monitor and offer information about the temperature and humidity of place where we plan to install/bring our robot. Monitoring these 2 data can be very useful to have our prototype working properly.
- **HC-SR04**: was chosen to measure the distance (in cm) from the gripper to an object which is going to be operated by our arm. It represents a nice and interesting feature for our robot.
- **SW-420**: was chosen in order to monitor and measure the vibration value while using the robot - a constant value of high vibration can cause damage.

The working space is going to be monitored with these 3 sensors and valuable data will be collected and processed with Arduino and Internet of Things. In the bullet list above there can be seen a briefly summary of their role and the intention of supervising the place where the robot will be installed for work.

The temperature & humidity sensor (DHT22) will be placed on the bottom base of the robot, the vibration sensor (SW-420) will be installed on the main arm because there might be the main source of vibration and the distance sensor (HC-SR04) will stay near the gripper so data about the distance remaining until an object is reached can be collected.

This set of data will enable a friendlier experience for the user and a good way to understand what is happening near the user. Moreover, collecting these data is important even in troubleshooting the eventual problems regarding the prototype.

5.2.2 Connecting sensors to Arduino - wiring, coding, data

In this phase another Arduino board will be used for connecting the sensors – the sensors will not be attached the Arduino Mega2560 board from the previous chapter (5.1 Robot electrical Design). The intention was to use another board in order to simplify the things and not to overload Mega2560 and to lower the performance – even if Mega2560 is a strong Arduino board connecting too many devices into it might lead to a faulty execution.

The choice was WeMos D1 R1 board with an integrated ESP8266 module for WiFi connectivity. It is an interesting and powerful board and it represents one of the best choices when it comes about connecting a project to internet of things due to its ESP8266 module. This board is more or less similar to any other Arduino development boards and it works on the same principle.

The WiFi module will enable the feature of connecting the board to an IoT platform and get the results displayed – more detailed documentations will be available in the next subchapter.

A wiring diagram (figure 5-8) and a table (table 5-4) will be presented below and it will offer a better visualization and how to wire every sensor to the WeMos board. The wiring figure was created with Fritzing software.

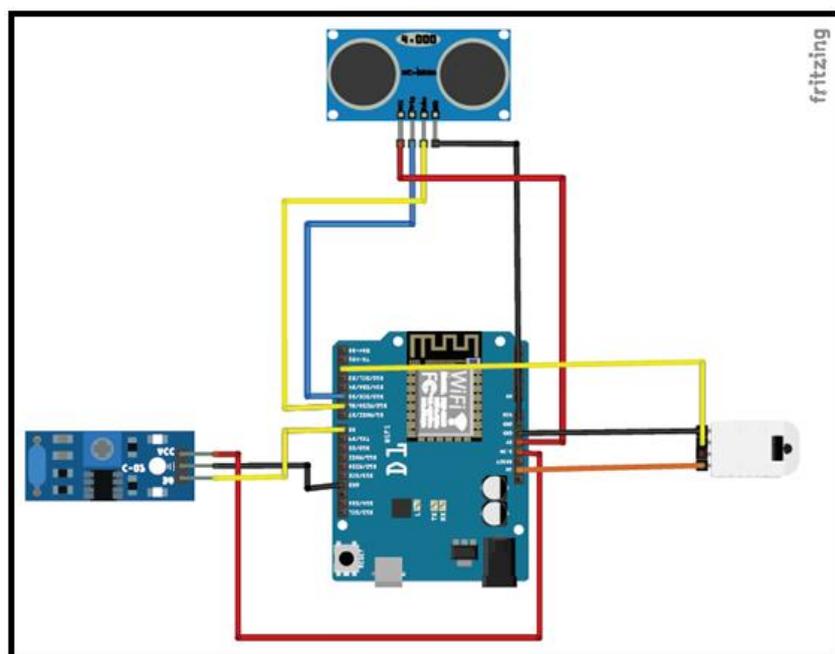


Figure 5-8: Wiring diagram created with Fritzing software

Number	Sensor	Sensor Pin		Board Pin
1	DHT22	GND	->	GND
		OUT	->	D2
		VCC	->	5V
2	SW-420	GND	->	GND
		DO	->	D8
		VCC	->	3.3 V
3	HC-SR04	GND	->	GND
		ECHO	->	D6
		TRIGGER	->	D5
		VCC	->	5V

Table 5-4: Pins connection

The wiring figure and the table represents only a physical connection of our components - the connection was made with male/female wires. In order to display the data captured by the sensors it is required a digital connection which can be achieved with codes and structures.

For the digital connection, Arduino IDE was the main tool responsible at this step. As mentioned in the previous chapter, Arduino IDE was used to compile and upload codes and libraries to MEGA2560 via an USB cable plugged between the computer and the board. In this case the same idea will be used to upload the necessary codes to WeMos board.

Researching on the internet about code examples was a crucial and necessary to find a matching code for the project. Although there was no code created to run all 3 sensors at the same time, codes for running each sensor were available. These basic codes were used as documentation in order to create the final code able to run everything at the same time. Most of the codes were available from Arduino IDE developers and they were already included as examples in the software.

The real challenge was to merge these 3 codes into 1 single code able to run the sensors and to display the data captured. Firstly, a good understanding about C++ and Arduino programming was needed to achieve the goal – documentation was necessary to create and understand some basic knowledge about programming language, structures and set of variables.

The code was created after researching, learning and understanding the basics and it is shown below. It is necessary to mention that this code was genuinely created by the team and it was not available on any online or offline sources.

The final code has a clear and easy way to understand structure and also offers indications and commentaries written in the code with the operator `//` - which is a programming operator that considers everything written on the line after it as a commentary, not as a code line. Also, the code has some special lines written for separating the result displayed by each sensor. The results were easy to read when the code was compiled and built.

The goal of this code was to display the value of 4 variables (temperature, humidity, vibration and distance) so the user can be sure about the work conditions.

A visual representation (table 5-5) regarding the written code is presented below. At first sight, it can be observed how each variable was declared, how every pin was declared and assigned and how the whole structure looks like.

```
#include <SimpleDHT.h>

// for DHT22,
//   VCC: 5V or 3V
//   GND: GND
//   DATA: 2
const int trigPin = D5;
const int echoPin = D6;
long duration;
int distance_cm;
int distance_in;
int vibr_Pin = D8;
int pinDHT22 = D2;
SimpleDHT22 dht22;

void setup() {
  Serial.begin(9600);
  pinMode(vibr_Pin, INPUT);
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}

void loop() {
  // start working...
  Serial.println("=====");
  Serial.println("TEMPERATURE & HUMIDITY");

  // read without samples.
  // @remark We use read2 to get a float data, such as 10.1*C
  // if user doesn't care about the accurate data, use read to get a byte data, such as 10*C.
  float temperature = 0;
  float humidity = 0;
  int err = SimpleDHTerrSuccess;
  if ((err = dht22.read2(pinDHT22, &temperature, &humidity, NULL)) != SimpleDHTerrSuccess)
  {
    Serial.print("Read DHT22 failed, err="); Serial.println(err);delay(2000);
    return; }

  Serial.print("Temperature & Humidity: ");
  Serial.print((float)temperature); Serial.print(" *C, ");
  Serial.print((float)humidity); Serial.println(" RH%");

  // DHT22 sampling rate is 0.5HZ.
  delay(2500);
```

```

Serial.println("=====");
Serial.println("VIBRATION");
long measurement =TP_init();
delay(50);
Serial.print("measurment = ");
Serial.println(measurement);
  Serial.println("=====");
  Serial.println("DISTANCE");
  // Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(500);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(500);
digitalWrite(trigPin, LOW);
/* Reads the echoPin, returns the sound wave
   travel time in microseconds */
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance_cm = duration / 58;
distance_in = duration / 148;
/* Prints the distance on the Serial Monitor
   Serial.print("Distance: ");
   Serial.print(distance_in);
   Serial.println(" inches"); */
Serial.print("Distance: ");
Serial.print(distance_cm);
Serial.println(" centimeters");
/* uncomment the 3 Serial.print lines
   to display measure in inches */

}
long TP_init(){
delay(10);
long measurement=pulseIn (vibr_Pin, HIGH); //wait for the pin to get HIGH and returns
measurement
return measurement;
}

```

Table 5-5 : Code

In order to go further, this code needs to be uploaded to WeMos board and to display the data. The uploading is made by just pushing the Upload button in the Arduino IDE interface

and the result of the running program will be displayed by the Serial Monitor of the software – the Serial Monitor represents a window of the program which shows the results of a code.

To see the result in the monitor and to upload the code, it is a must to have the WeMos board connected to the computer with the USB cable. The result can be seen by running the program and opening the serial monitor (figure 5-9).

The program will display the values for DISTANCE, TEMPERATURE & HUMIDITY and VIBRATION. These variables are updated every 1 second in the serial monitor and the changes between values can be seen by running the program for a longer time. The fluctuation is indicated by the increase or decrease or certain values.

Now, the user has a way to know more things about the working environment but in an offline or local way. This kind of showing the results it is not very efficient since the user has to utilize Arduino IDE all the time and change tabs on his computer between the controlling software of the robot and the serial monitor in order to be in touch with the parameters. The solution will be integrating our code or program to an IoT platform and to find a better way to display the results.

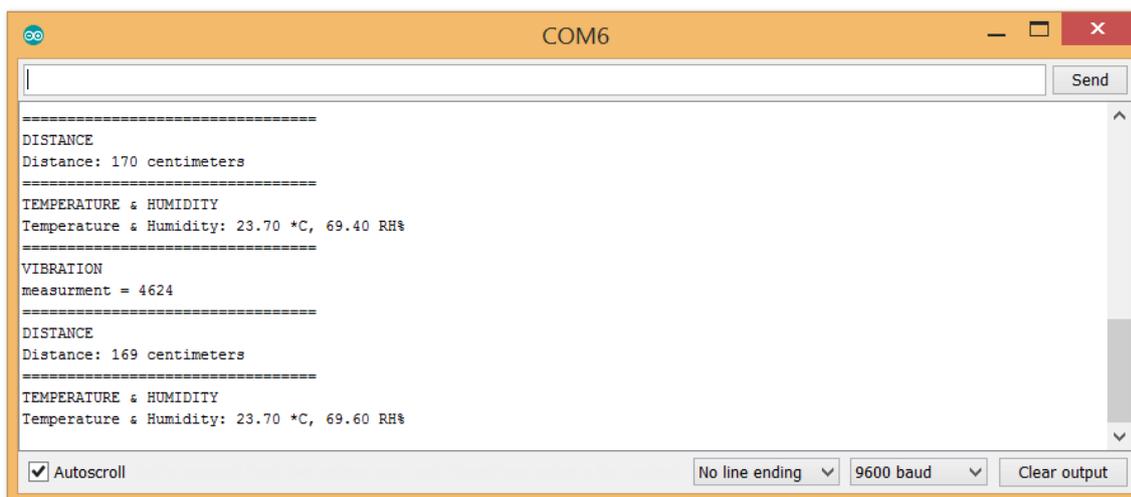


Figure 5-9: Serial monitor

5.2.3 Internet of Things - solution, IoT platform & Blynk app

The aim of this project was to enable IoT capabilities to the robotic arm prototype and to create an interface where to display the connectivity with the internet of things feature.

Blynk comes with their own library for Arduino IDE software and the necessary code lines for ensuring the connecting to their servers. To connect to the Blynk platform, a WiFi hotspot or an active internet connection is needed, and the credentials of the internet connection will be written in the specified code line.

For starting using Blynk it is mandatory to download their phone applications from Android Market, for Android OS users, or from App Store for IOS users. It is a simple process and it can be summed up by downloading and signing up.

They present on their official web page (<http://www.blynk.cc>) a short visual explanation of how everything works behind (figure 5-10). It is a chain of connection between a smartphone, their servers and the digital device that is going to be used. In the first stage the smartphone (where the graphical user interface is loaded and displayed) connects to their servers, the servers access the necessary libraries for the project and then the servers connect to the development board (Arduino, Raspberry PI, ESP8266 etc.). We assume that to the development board are connected some sensors, motors or actuator. Going back on the chain path, data and information collected by our end-effectors will be sent to the development board which will send the information to the servers and the servers will send back everything to the smartphone. In this way the phone will show digitally whatever was registered. In order to sum up and simplify the operation, it can be stated that it is sent a request which will get a response within seconds – everything circulating on an internet path (wired or wireless).

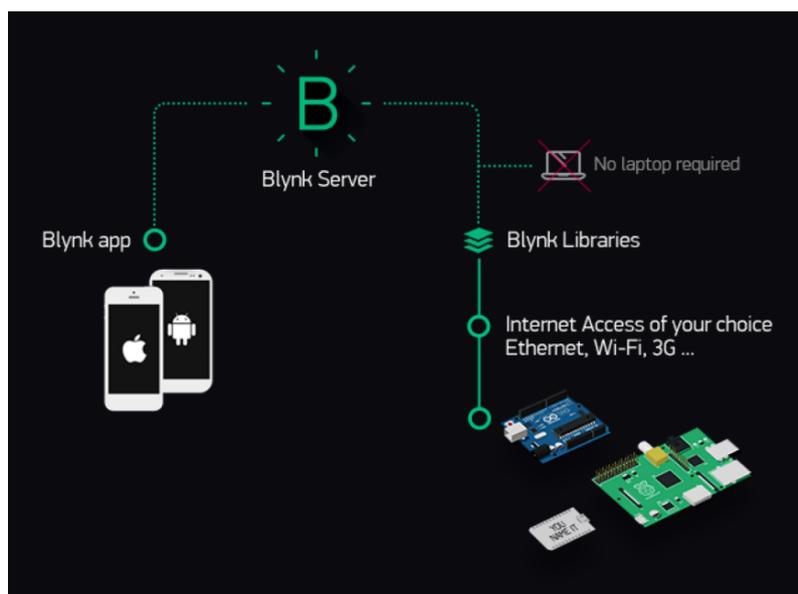


Figure 5-10: Blynk working principle, Photo retracted from <http://blynk.cc> on 15.05.2018

Since the general idea of IoT was understood and how everything works with Blynk, it remains only to ensure a connection between our sensor wired to the WeMos board and Blynk servers.

Firstly, the primordial step is to integrate to the WeMos the necessary code to ensure connectivity to Blynk servers. Once their IDE library is downloaded to our computer, we will easily be remark how to ensure the connectivity. The programming code below (table 5-6) explains the development – code which comes attached with their Arduino IDE library called BLYNK library.

```

/*****

```

```

Blynk is a platform with iOS and Android apps to control
Arduino, Raspberry Pi and the likes over the Internet.

```

```

You can easily build graphic interfaces for all your
projects by simply dragging and dropping widgets.

```

```

Blynk library is licensed under MIT license

```

```

This example code is in public domain.

```

```

*****/

```

```

#define BLYNK_PRINT Serial

```

```

#include <ESP8266WiFi.h>

```

```

#include <BlynkSimpleEsp8266.h>

```

```

// You should get Auth Token in the Blynk App.

```

```

// Go to the Project Settings (nut icon).

```

```

char auth[] = "YourAuthToken";

```

```

// Your WiFi credentials.

```

```

// Set password to "" for open networks.

```

```

char ssid[] = "YourNetworkName";

```

```

char pass[] = "YourPassword";

```

```

void setup()

```

```

{

```

```

  // Debug console

```

```

  Serial.begin(9600);

```

```

  Blynk.begin(auth, ssid, pass);

```

```

}

```

```

void loop()

```

```

{

```

```

  Blynk.run();

```

```

}

```

Table 5-6 : Blynk code

To make everything able to connect we have to change 3 values: ssid[], pass[] and auth[]. As mentioned in the top commentary section of the code, the auth[] value will be generated by the Blynk App installed on a smartphone.

The values for `ssid[]` and `pass[]` will be set in the beginning - these 2 values refers to the WiFi hotspot needed to be created. A smartphone with working mobile data was used in order to create a WiFi hotspot for this project and to complete the code. (for example: `char ssid[] = "AndroidAP7122"`; `char pass[] = "123456789"`; where the SSID means the name of the hotspot and pass is the password).

For the 3rd value, `auth[]` it is necessary to install the app and set our device – in this case ESP8266 WeMos. Once everything is done the app has a field for Auth Tokens and Email The Code button. An email containing the auth code will be send to the email address used for signing up by pressing the button. It remains only to copy and paste into the code the auth code received.

Now everything is ready to go and by completing these 3 values and uploading the code to the development board the connectivity will be made. Since our board is connected to Blynk, it remains only to create the application interface and to write the necessary code to read the sensors – adapting the previous code for 3 sensors to the Blynk platform.

Four “Gauge” widgets will be used to collect and display the data from the sensors. The connection will be made with virtual pins and we will allocate a virtual pin for each sensor in order to collect results. Virtual pins V3, V4, V5 and V7 were used and linked to 4 Gauge widgets because every widget will show the value for Temperature, Humidity, Distance and Vibration.

After adding the widgets and assigning them virtual pins the graphical interface is created. It remains only to modify and adapt the previous code for reading the sensors to Blynk. Blynk comes with their own documentation available at <http://docs.blynk.cc/> where strong information about Blynk structures and codes are provided. The final code was created by following and reading the documentation regarding virtual pins and connectivity through code lines.

There are special structures for the Gauge widgets which is required to be included in the code like in the example below.

Example:

```
BLYNK_WRITE (V5) , Blynk.virtualWrite(V5, distance)
```

The codes will assign to the variable “distance” the virtual pin V5 and the value will be displayed into the widget assigned to pin V5.

Below (table 5-7) it is attached the previous code modified for Blynk. Differences and similitudes between the 2 main codes (previous and Blynk) can be easily identified.

```
#include <SimpleDHT.h>
//#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define TRIGGER D5
#define ECHO D6
int pinDHT22 = D2;
SimpleDHT22 dht22;
int vibr_Pin = D8;

char auth[] = "64f5638f7eba465fb343c1786f53bf16";
char ssid[] = "AndroidAP7122";
char pass[] = "123456789";

BLYNK_WRITE(V5)
{
  int pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable

  // process received value
}
BLYNK_WRITE(V3)
{
  int pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable

  // process received value
}
BLYNK_WRITE(V4)
{
  int pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable

  // process received value
}
BLYNK_WRITE(V7)
{
  int pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable

  // process received value
}

void setup()
{
```

```
// Debug console
Serial.begin(9600);
pinMode(TRIGGER, OUTPUT);
pinMode(ECHO, INPUT);

Blynk.begin(auth, ssid, pass);
}

void loop()
{
  long duration, distance;
  digitalWrite(TRIGGER, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIGGER, LOW);
  duration = pulseIn(ECHO, HIGH);
  distance = (duration/2) / 29.1;
  Blynk.virtualWrite(V5, distance);
  delay(200);
  float timer;
  float temperature = 0;
  float humidity = 0;
  int err = SimpleDHTErrSuccess;
  if ((err = dht22.read2(pinDHT22, &temperature, &humidity, NULL)) != SimpleDHTErrSuccess)
  {
    Serial.print("Read DHT22 failed, err="); Serial.println(err);delay(2000);
    return; }
  Blynk.virtualWrite(V3, temperature);
  Blynk.virtualWrite(V4, humidity);

  Serial.println("VIBRATION");
  long measurement =TP_init();
  delay(50);
  Serial.print("measurment = ");
  Serial.println(measurement);
  Blynk.virtualWrite(V7, measurement);

  Blynk.run();
}
long TP_init(){
```

```
delay(10);  
long measurement=pulseIn (vibr_Pin, HIGH); //wait for the pin to get HIGH and returns  
measurement  
return measurement;  
}
```

Table 5-7 : Final code with Blynk

Connecting the WeMos board to computer, creating the WiFi hotspot and uploading the code above, will enable the IoT feature and the smartphone will display on the screen, inside Blynk application, the values of the sensors. A conclusive image is shown below – a screenshot (figure 5-11) of the app while doing tests.

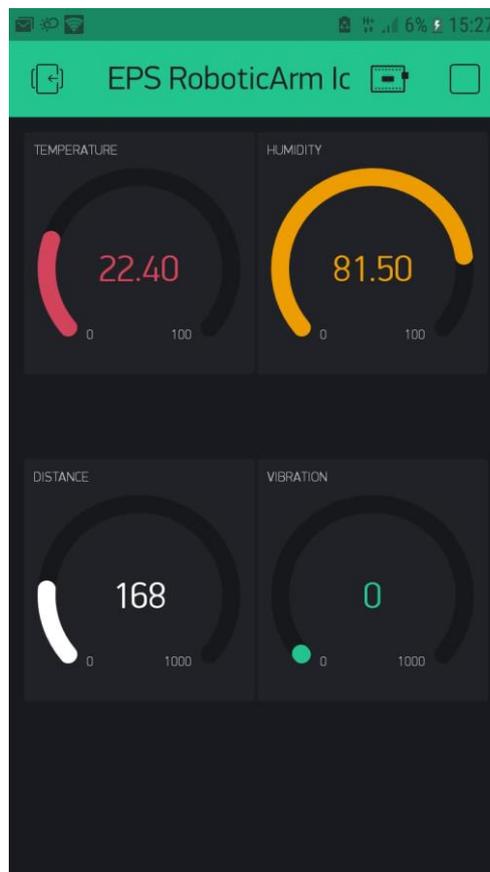


Figure 5-11 : Blynk app running on personal smartphone

Conclusions

In this project, the team had to find a suitable robot model on the internet, 3D print it, assemble it, make some structural adjustments, make it move and integrate it with IoT capabilities.

The team was lucky to find a suitable robot arm on the internet that met all the criteria and a model that all the whole team agreed on. The design was good, and it was an interesting robot to work with.

The 3D printing process showed to be time-consuming and sometimes difficult, but was absolutely possible to overcome by guidance, reading on the topic and by the “try and error” method.

It became clear that we had to do something about the stability of the robot arm. The team decided to design a stand that contributes with some weight and a lot more stability. The team is very satisfied with the solution of this problem.

Enabling IoT capabilities to the prototype was an interesting idea and a real challenge. The robotic arm has been improved and the team changed and upgraded the initial project through these capabilities. According to a case study conducted by Thingworx Academy in 2016, it is said that in 2020 there will be about 50 billion devices connected to Internet of Things. Since the project has IoT capabilities and it makes part of this new era, our team is delighted to have a contribution in reaching the estimated number of IoT devices.

The task of developing the robot arm has sometimes shown to be difficult as most of the team has limited knowledge and experience on the field. The team has had good communication throughout the semester. The work has been shared between us depending on the knowledge each person has on the topic and/or individual interest. It has been an interesting and social semester with lots of challenges and with a steep learning curve. The team is all in all satisfied with the final result of the robot arm as well as the process getting there.

Personal experience

Joachim

My experience in Romania has been a very good one. I have met a lot of interesting people from all over Europe that I have become friends with. Romania has been a very good country to be an exchange student in. It has a lot of beautiful architecture, an amazing nightlife (that I have made a lot use of) and friendly people. It has been fantastic to explore Bucharest's neighboring cities by train while going through an amazing scenery. Visiting Dracula's home, Bran Castle, and go skiing in Sinaia are two experiences that stands out from this semester. The University and its teachers has welcomed us in the best way and have been taking good care of us throughout the semester. I will definitely come back to Bucharest sometime in the future.

Remi

At the beginning, when I arrived here I had some apprehensions about this semester because I didn't know the country and I didn't know anyone. But now I can assure that it was one of the best experience in my life, it permitted to me to meet new people from different country. This thanks to the proximity between each other at the p19 Hostel and the organized parties. But also, thanks to the trips I made, principally in Romania (Timisoara, Constanta, Sinaia ...) and I wanted to say I really appreciated architecture and food of this country. Regarding the university part, the teachers welcomed us very well and supervised during this semester which leads to the good progress of the project.

I think I'll never forget this experience and the people I met, so thank you to giving me the possibility to do this semester.

Floris

This EPS journey has taught me a lot of things. It has been a really nice journey aswell, since ive been able to meet a lot of new people who are all from different origins, meaning profession and country. I have partied a lot with all my Erasmus friends, and made sustainable friendships. The project itself was really educative, since I didn't have any experience with 3D printing or robotics, so that was nice. Now I know how to operate a 3D printer and how to make a robot work. The choice to go abroad to do EPS has been one of the best of my life so far, I'm very satisfied.

Andrei



Working in a multicultural group was a great opportunity for me. This experience helped me a lot to improve some personal and technical skills. The communication was good and we managed to create an interesting project. It was very nice that we could share experiences and ideas and through this we helped each other in a way or another. All in all, the EPS experience was a good experience and I'm glad that I had the opportunity to meet and work with foreign people.

Bibliography

Alluwar, N. (2017, november 21). *What is IoT architecture?* Retrieved from quora.com:
<https://www.quora.com/What-is-IoT-architecture>

Blynk. (n.d.). *How Blynk Works*. Retrieved from blynk.cc: <http://docs.blynk.cc/#getting-started>

Chakravorty, D. (2017, june 13). *STL File Format for 3D printing- Simply Explained*. Retrieved from all3dp: <https://all3dp.com/what-is-stl-file-format-extension-3d-printing/>

Haboud, D. (2016, october 27). *IoT and The Three C's*. Retrieved from altium.com:
<https://resources.altium.com/pcb-design-blog/iot-and-the-three-c-s>

Jazar, P. R. (2006). *Theory of Applied Robotics*. Australia: Springer.

Poole, H. H. (1989). *Fundamentals of Robotics Engineering*. New York, New York 10003: Van Nostrand Reinhold.

PTC. (n.d.). Retrieved from ptc.com: <https://www.ptc.com/en/products/iot>

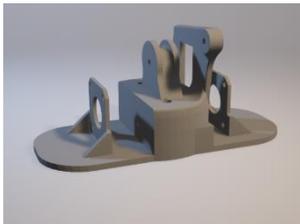
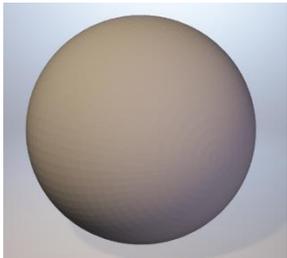
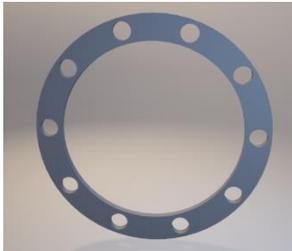
Thomas Campbell, C. W. (October 2011). *Could 3D Printing Change the World?* 1101 15th Street, NW, Washington, DC 20005 (202) 463-7226: Atlantic Council.

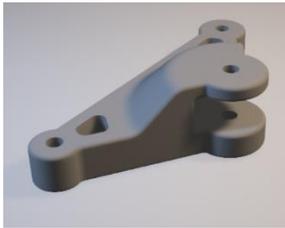
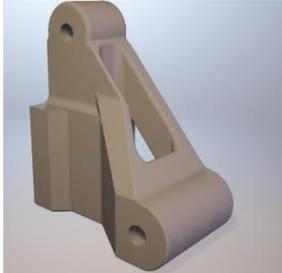
Other useful links that was used for a better understanding of the IoT part:

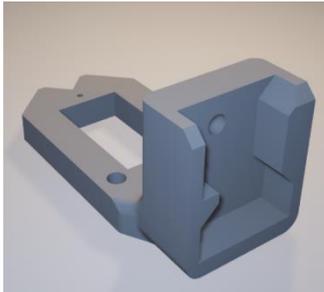
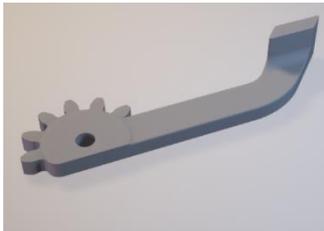
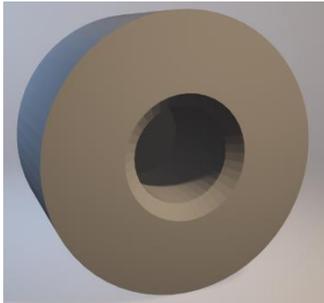
- <https://github.com/grbl/grbl> (GRBL library and documentation)
- <https://community.blynk.cc/t/esp8266-with-3-sensors-and-blynk/25654> ESP8266 with sensors
- Arduino install guide: <http://osoyoo.com/2017/04/07/arduino-uno-cnc-shield-v3-0-a4988/>
- ESP8266: <http://www.instructables.com/id/Esay-IoT-Weather-Station-With-Multiple-Sensors/>
- Thingworx documentation: <https://www.thingworxacademic.com/>
- Internet of Things documentation: <http://iotu.com/>
- Arduino Basics: <https://www.arduino.cc/en/Main/Tutorials>
- Universal G-code Sender: <https://github.com/winder/Universal-G-Code-Sender#>

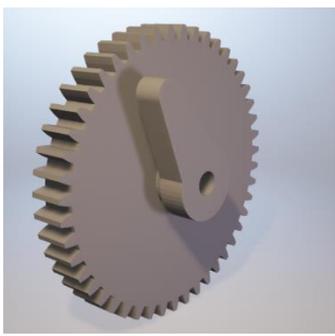
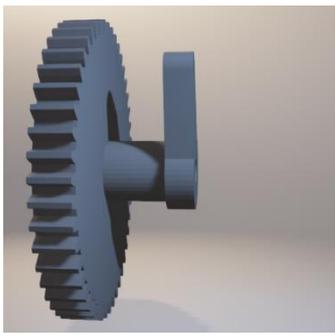
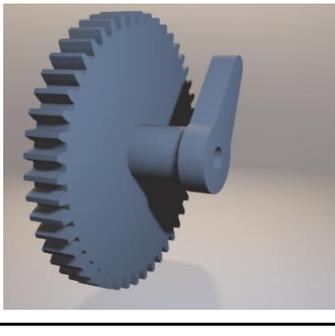
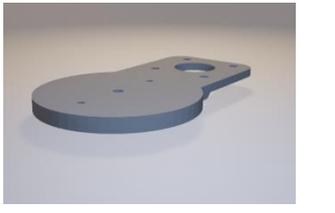
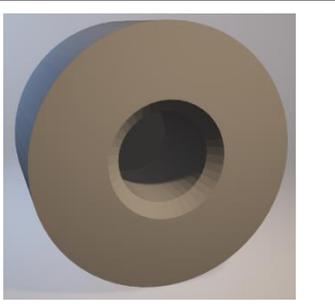
Attachments

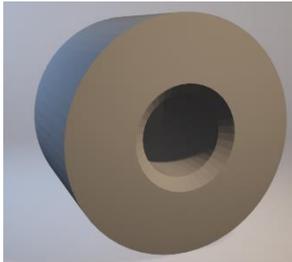
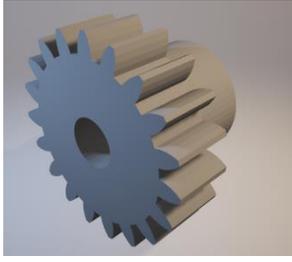
A, the print time table

File name	File picture	File size	Print time	Number of pieces	Printed
Base_Arm		421 kB	6 h	1	Yes
Base_Bearing_Ball		301 kB	10 min	10	Yes
Base_Bearing_Holder		122 kB	11 min	2	yes
EBAmk2_002_mainarm		2.675 kB	1 h 29 min	1	yes
CCR10_EBAmk2_006_horarm_plate		3.370 kB	3 h 30 min	1	yes

EBAmk2_003_varm		1.22 5 kB	19 min	1	yes
EBAmk2_004_link135		800 kB	15 min	1	yes
EBAmk2_005_link135angled		531 kB	15 min	1	yes
EBAmk2_006_horarm__		3.42 6 kB	1 h 25 min	1	Yes
EBAmk2_007_trialink		2.81 5 kB	26 min	1	yes
EBAmk2_008_link147_new		676 kB	16 min	1	yes
EBAmk2_009_trialinkfront		1.33 2 kB	27 min	1	yes

EBAmk2_014_claw_base		348 kB	48 min	1	yes
EBAmk2_015_claw_finger_dx		486 kB	11 min	1	yes
EBAmk2_016_claw_gear_drive		142 kB	9 min	1	yes
EBAmk2_017_claw_finger_sx		421 kB	11 min	1	Yes
EBAmk2_018_claw_gear_driven		122 kB	9 min	1	yes
finalholder		27 kB	4 min	2	yes

gear1		89 kB	1 h	1	yes
gear2		88 kB	1 h 7 min	1	yes
left_gear		88 kB	1 h 7 min	1	yes
Robot_Base_Low		262 kB	1 h 38 min	1	yes
Robot_Base_Low_Gear		731 kB	2 h 46 min	1	yes
screw_holder1		27 kB	4 min	1	yes

screw_holder2		27 kB	4 min	1	yes
Stepper_Gear		90 kB	17 min	3	yes