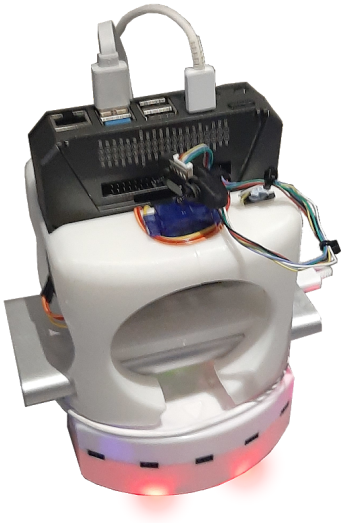
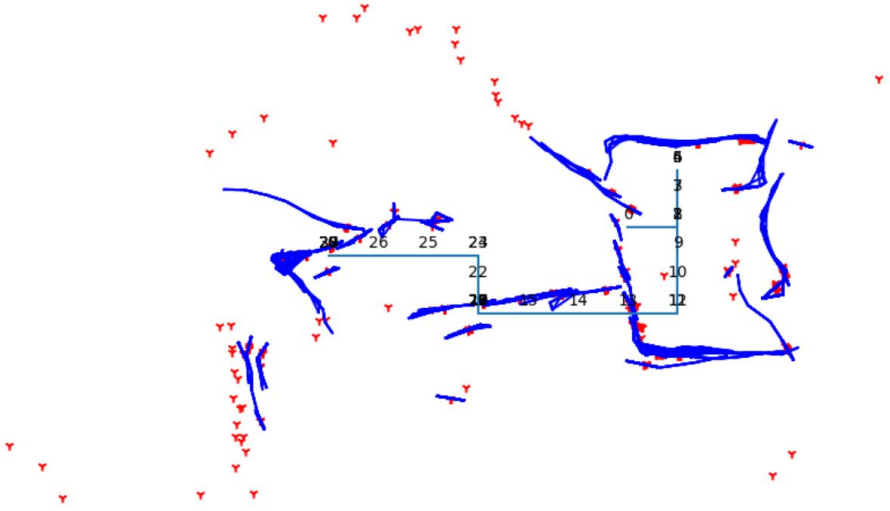
# AUTONOMOUS VEHICLE PROJECT

## Autonomous Mapping with Wifi Signal Intensity Measurement (AMSIM)





| | |
|---|---|
| Module Name: | **Autonomous Vehicles** |
| Module Number: | **TRAN71010** |
| Title of Assignment: | **Module Assignment** |
| Authors: | **GRAS Liam**<br>**FAYOLLE Jérémy**<br>**CHASSERY Léo**<br>**ANTHOINE-MILHOMME Timothée** |

# Abstract

This report aims to synthesize the conducted research in the workframe of the Autonomous Vehicles Module at Staffordshire University. In a concern of improving the Wifi covering in buildings, the following project aims to create wifi strength heat maps using a roaming autonomous vehicle. This system would allow the creation of better design of wifi covering by highlighting dead zones or routers too close to each other.

The robot is powered with a Raspberry Pi 4 (RBP) above a Thymio II robot for the motion and use of sensors and buttons. The robot manages to navigate and harvest data. We 3D printed a support hosting the RBP, a battery and servo motor. On top of the servo motor we placed a laser (Time of Flight sensor).

The goal is to place the robot in an unknown room where it will start to move autonomously after the press of the startup button. The robot will therefore avoid obstacles in front of itself and move accordingly. Along its path the robot will create a physical map of the room using the laser and odometry for positioning. Meanwhile the robot harvested the wifi intensity to create a heat map of the wifi on top of the previous map.

The objective has been partially achieved, the system is able to autonomously retrieve data and process them to create a map. However, the system is not optimized enough regarding positioning and map making.

Nevertheless the robot is showing promising results and could be easily improved by following the various stated tracks.

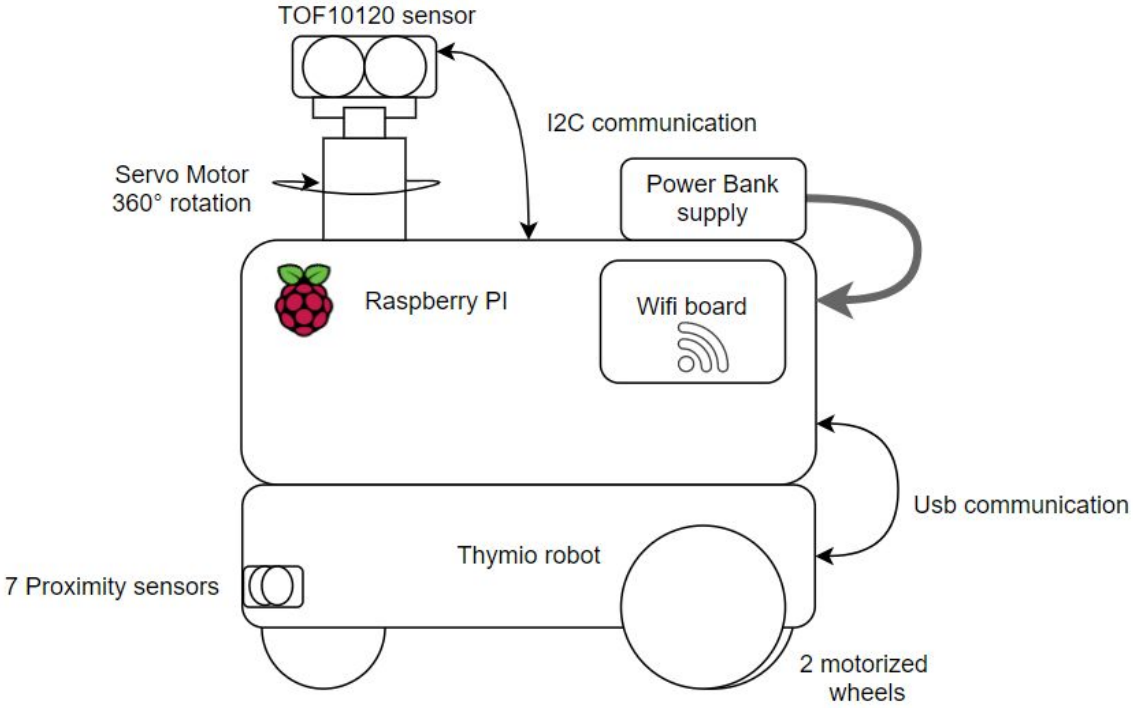
# Table of content

# Introduction

During the first semester of the Msc Robotics & Smart Technologies part of the Autonomous Vehicles module, we were tasked to create an artifact of an autonomous vehicle. The project was supervised by Deby Roberts and realised using the facilities of Staffordshire University and our personal houses because of the current sanitary crisis. The guidelines were voluntarily open in order to allow the most freedom of creation in the project.

We worked on an autonomous robot, able to move around, avoiding obstacles, while mapping certain scalar values for mapping wifi intensity as well as mapping its surroundings. This robot could potentially be used in offices, houses (land robot), mountains or cities (drone robot) in order to see in which zones the signal is not well received. Telecommunication companies could find it interesting for a better coverage, and offices would figure out where to put the IT sector for instance.

The goals would then be to build an efficient “smart” robot as an engineering team, capable of mapping the environment while avoiding obstacles. We generally looked forward to having more than a simulation to provide for this project, working on a real robot interacting with its surroundings.



# Part I: Conception of the vehicle

Originally, we thought about using mBots for our project as we were all familiar with the Arduino environment. However, with the second lockdown and the inability to borrow them from the university. We then decided to use the material we personally own which is composed of  a Thymio II powered with Raspberry Pi 4.

## Thymio II

The Thymio is an educational development robot created by EPFL based on differential drive steering. This robot can be programmed through the Thymio Suite software allowing different levels of coding from VPL (images for kids) to real coding in the Aseba language.

Sum up of its specification:

- 9 Infrared sensors
- 5 Touch Buttons
- 1 Three-axis accelerometer
- 1 Thermometer
- 1 Microphone
- 1 Infrared sensor (receiver for remote control)
- 1 Wireless Module

## Raspberry Pi 4

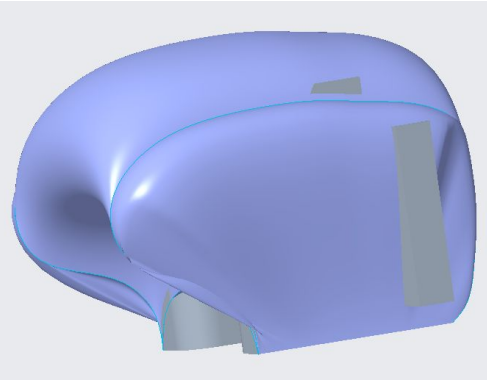
The Raspberry Pi 4 is a single-board computer powered by an ARM Cortex-A72 processor. It is very popular among tinkerers all around the world, and is even used by professionals for prototyping. Thanks to this popularity, a lot of resources are available online to learn about a great variety of applications.

We chose the Raspberry Pi 4 for our project because:

- It can comfortably handle the computing requirements of the project
- It has wifi connectivity
- It has GPIO pins for component connection
- It is portable and can be powered by a regular power bank

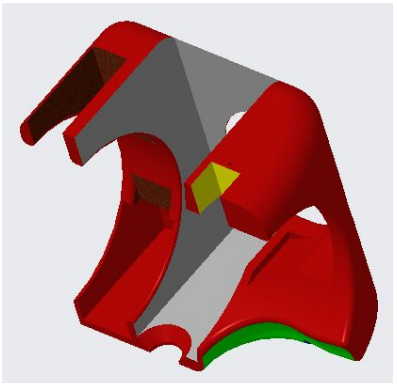
# 3D Modeling

We used the software creo 7 with the knowledge acquired in the module Engineering Design and Manufacture with Patel Bahvesh to design the pieces of our robot. We needed to create a support placed on top of Thymio II. This support can host the battery, the raspberry 4, the servo motor and the laser. The models were then printed in ABS+ using the UpBox+ in the Mellor building.

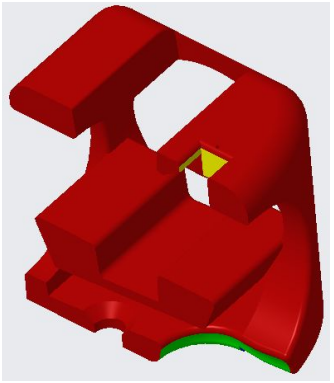
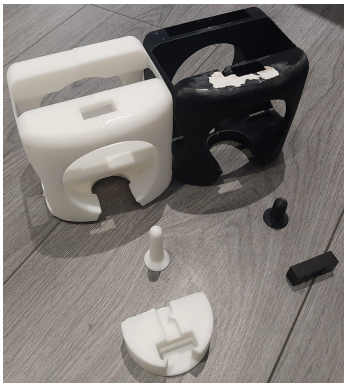




We did multiple versions and talked with the issue of each model with Pete Smith. Our original model had two major flows; it would have taken 48h and 300g of plastic to print. We improved our model and obtained a second version that took 23h to print and 150g of plastic. This modification divided by two the major cost which is very interessant. We splitted the model in two to ease the printing and removed a lot of material inside.




We discovered a physical problem when we tested the second version, the balance of the robot wasn't optimal therefore the robot couldn't move properly. We adjusted it moving the position of the battery and the raspberry pi. We also discovered some fitting problems. We corrected them to ease the assembly. We modified the emplacement of the servo-motor and of the laser to obtain our third and final version that was acceptable.

## Time of flight sensor

As its name suggests the used TOF10120 sensor relies on the Time Of Flight technologies, providing us with the distance between the robot and an obstacle in mm.


After requesting the data, the sensor returns a 2 bytes array containing the distance to the object in mm. It has a range of 10 cm to 1.80m with ± 4mm accuracy.
Although this module supports both UART and I2C, we decided to use I2C as it is faster. We need quick response as the sensor is mounted on the servo motor to scan its environment.

## Servo Motor

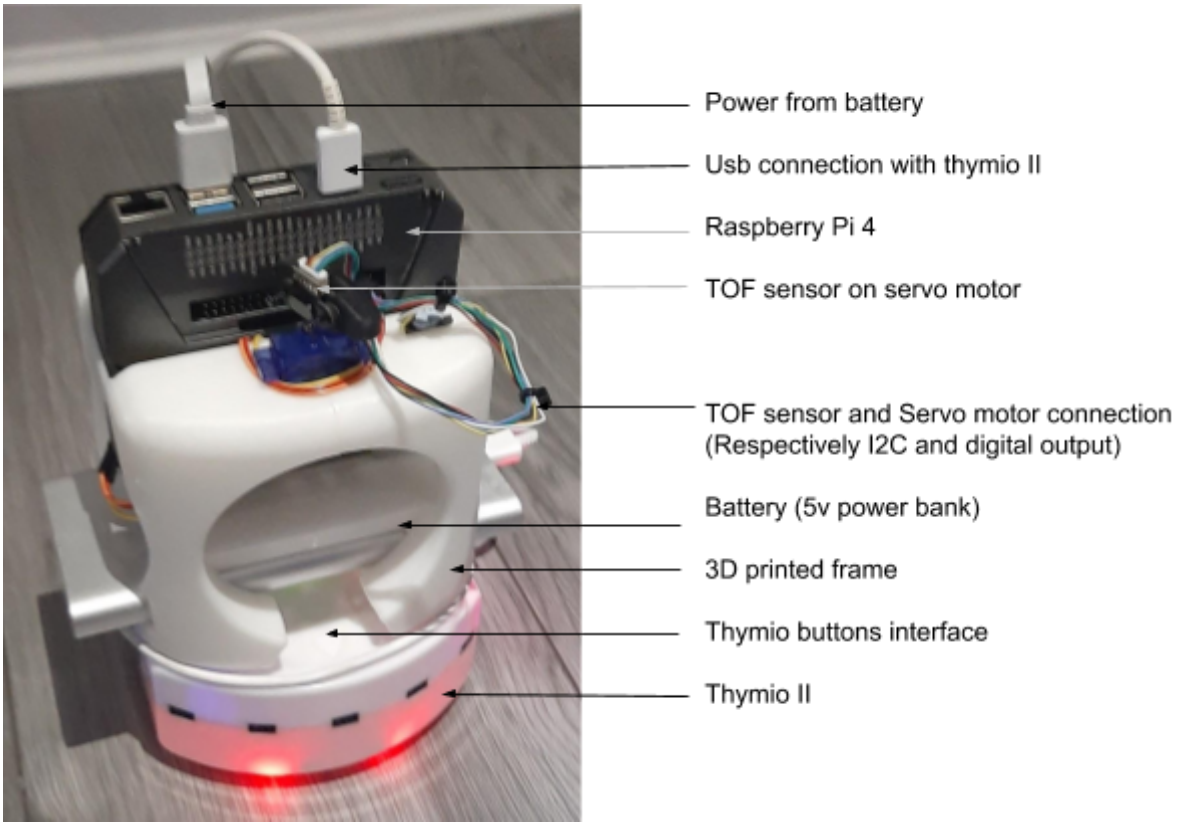
A servo (slave in latin) is a motor capable of precise angular control. It allows the robot to know where he is “looking” at.

## Wiring

We have different types of cables needed to connect the different parts of the robot, the servo motor and the laser are connected with simple jumper cables. Then we have a USB-C to USB between the raspberry and the battery. Finally there is a Micro-USB to USB to connect the Raspberry to the Thymio.

## Assembly

To assemble the last version of our robot we needed to glue the structural parts. Insert the raspberry and the battery, plug them all together using micro-usb and usb-c cables. Then we added the servo-motor screwed to the support. Finally the laser is placed on its support on top of the servo-motor. We plugged the servo-motor and the laser using jumper cables to the raspberry pi 4.




After the first assembly we noticed the box was slipping a bit off the Thymio, it wasn't worth reprinting an entire box for this little problem so we decided to fix the problem with blue tack. We also realized that cable management was quite an issue so we fixed it with some zip ties.

## Ways of improving

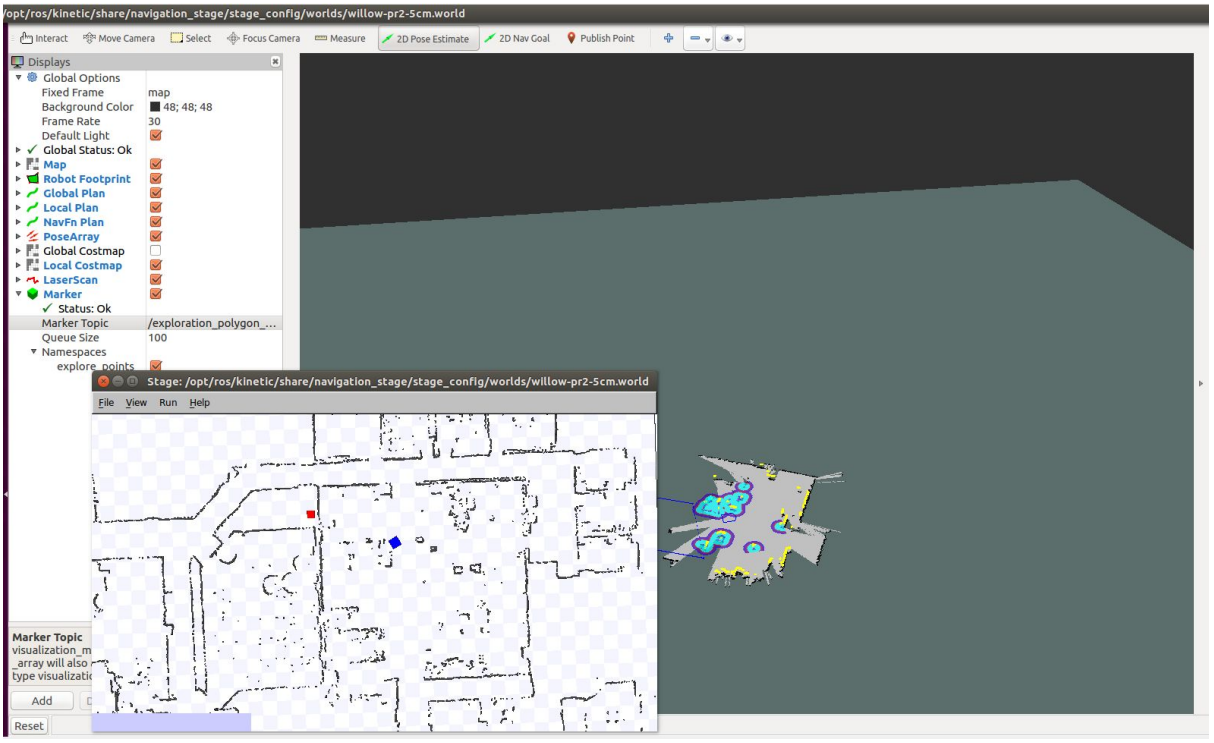
We could improve the assembly of our robot in many ways. The first would be to redesign again the 3D designed part including the problems of slipping and cable management. Then we could use some better hardware like a laser with a bigger range for example as ours is constrained with only 1.80m along with a better step motor rotating on 360°. If we do this last option it would involve redesigning the box to put the laser above every element of the robot to complete a full rotation. The battery emplacement could be redesigned to account for the cable getting stuck in the wall and thus slowing the vehicle down and messing up the mapping.

A larger modification would be to use a better robot than the Thymio II as this is already a 10 year old educational robot. We could use the mBot or build our own if we had some more time, this could be a very interesting project.

# Part II: Navigation

## Simulation Approach to Navigation

Before trying to navigate our robot in the real world we decided we could use a simulation of the navigation to understand the stakes. The ROS software is an amazing tool to simulate some robot movement in space. This software only runs on a Ubuntu interface so we had to install a virtual machine. This software is really powerful but takes a lot of computing power. Running it on a virtual machine is not the most effective way to run ROS but in our study case it was good enough. Then we explored some different ROS packages to find the one most suited to our project. We decided that the package frontier exploration was perfect to try to have an idea of simulating our robot. We then followed the instructions to create a simulation, and explored the interface which was really helpful in the choice we made concerning the creation of our own model of navigation.



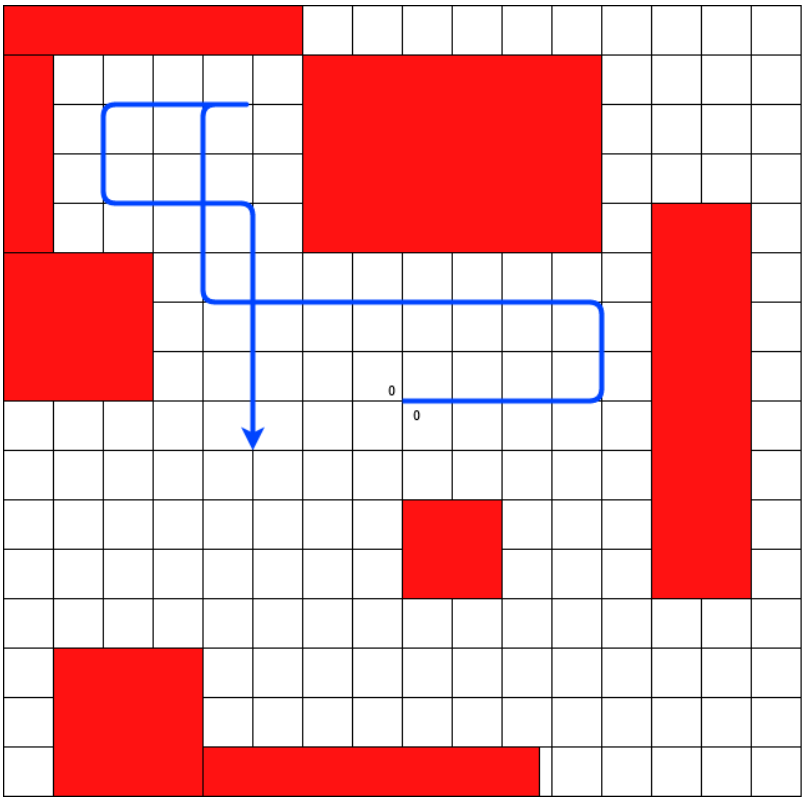

We can see on the image that the robot is detecting the edge of the wall and is following it. This is called frontier exploration. This is one of the basic exploration navigation but in our case it's already difficult to implement and we can only simulate it. This could be a whole project to implement it in our Thymio and it would be very interesting.

# Our Approach to Navigation

Our main goal regarding navigation was for the robot to be able to roam around a given space autonomously while avoiding obstacles and scanning its environment.

Our first approach to making our robot move around a room on its own, was to instruct it to move forward and to appropriately change direction when detecting an obstacle using the array of 5 front facing infrared sensors on the Thymio.
This approach made for a smooth navigation of the robot as the clever positioning of the infrared sensors allowed us to adjust motor strengths based on the relative position of obstacles.
Unfortunately, given the hardware and the scope of the project, this approach was incompatible with the actual function of the robot which is the environment mapping. Indeed, in order to map its environment accurately the robot needs to know its position and orientation any time it recovers data, and although this problem can be solved with a continuous type of movement using means we will cover further in this report, we chose to change our approach to better fit the scale of the project.

We settled on a discontinuous approach to navigating where the robot discretizes the space around it into a grid that it follows for its movement. The basic concept is that since the robot only moves a given distance forward at a time and only turns 90° right or left, it can keep in mind those movements to figure its position and orientation out.



## Odometry

In order to achieve precise movement and rotation, we decided to use odometry to control the wheels of the robot. The techniques consist of measuring the size of the wheel and the speed to obtain the distance travelled. To achieve this, we created different classes in the odometry_thymio class. The first function connect_thymio allows us to connect to the robot. Thymio is not working with python language which makes this a little bit tricky. We need to create a network using the language of thymio called Aseba. Then we execute our code in python while sending the request to thymio in it's own language.

Then we have the movement functions. Here we input a distance in mm and the raspberry communicates to the Thymio to move forward, wait for the time corresponding to the distance in cm then communicate to stop the wheel. For the turn_left and turn_right this is quite similar. The only difference is that one wheel is turning in an opposite direction to make the robot turn on itself.

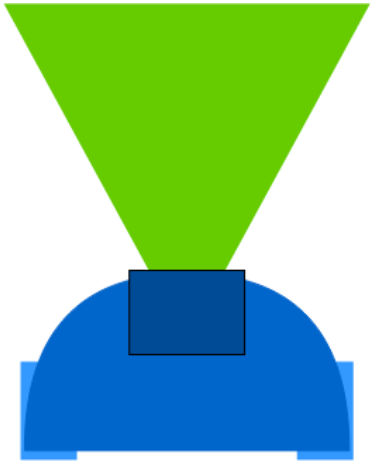
## Obstacle detection

In order to move around without damaging itself or getting stuck, the robot must perform some obstacle avoidance. One of the constraints of our approach is that the robot can't check for obstacles while it is moving, fortunately since it is only moving one given length at a time it only needs to check for obstacles directly in the next "tile" in front of it before moving to ensure a safe path.

To detect obstacles we originally tried to use the array of infrared sensors on the Thymio but 2 major issues made it impractical:

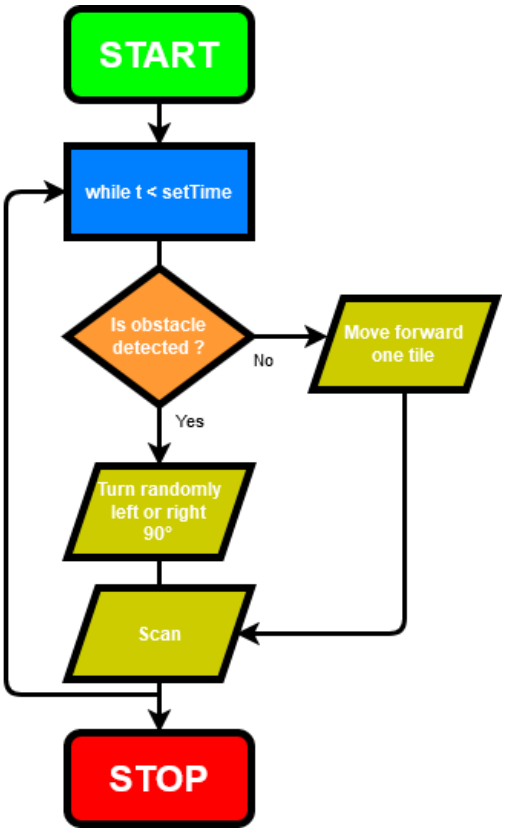
- Firstly, the infrared sensors are very sensitive to the type and reflectivity of the material it tries to detect, for example they would detect a white object from around 7cm distance but black object only from 3cm. It makes the robot very unpredictable and constrained us into moving forward only small steps at a time which would have been impractical and would have lengthened the mapping process considerably.
- Secondly, the interfacing with the Thymio introduced massive delays that rendered the avoidance of obstacles impossible.

To avoid those issues, we chose to use the data gathered while scanning the environment with the TOF sensor. The TOF sensor is mounted onto a servo motor and when a custom made function (sweepMod) is called it gathers distances corresponding to a range of angles, those values are used for the mapping process but we figured that they could be used to detect obstacles too.
By taking the values corresponding to the front of the robot we can give it a sort of field of view which can be used to trigger the obstacle detection.



## Navigation Algorithm

Once we have determined in which way the robot should move, how does it move and how does it detect obstacles, we need to create the logic for its movement.




Since the focus of our project is not autonomous navigation, we decided to stick to a basic algorithm. The concept is that the robot changes direction if it encounters an obstacle.

Once the program is started, the robot waits for an input before starting the main logic loop, at which point it enters a while loop that is active until the exit condition has been met. In this loop, at every iteration, the robot checks if there is an obstacle in the “tile” right in front of it using the data from scan performed in the previous iteration, depending on the result of the check is either move forward a given length or rotate 90° all the while keeping in mind its position and rotation, finally it performs a scan of the environment before stating over.

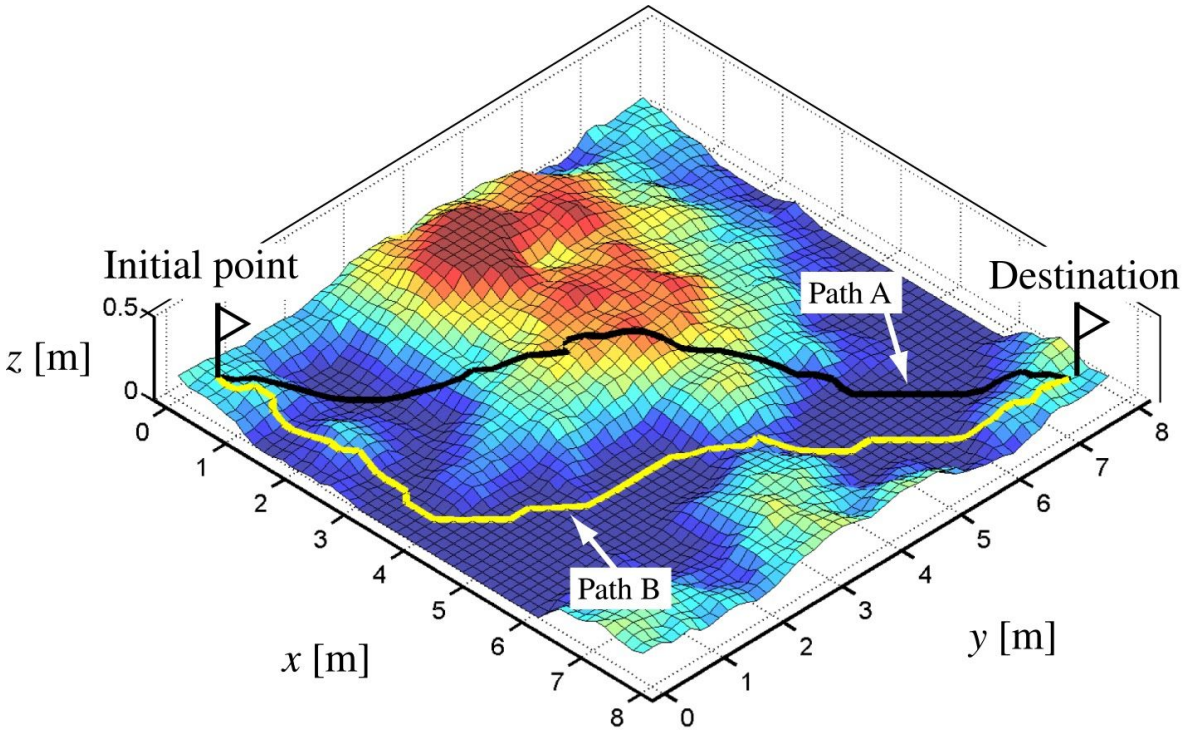
The exit condition we chose is a simple time limit.

## Ways of improving

We could improve navigation in many ways, one of the first modifications we could do would be to record the data while moving. This would involve rethinking the way we harvest data and the way we move.

The odometry is a very environment dependent measure, without encoders on the motors it can be altered easily and is not entirely reliable. There is an accelerometer (quite imprecise) in the Thymio, if we could couple it with a gyroscope we could more accurately control the position and direction of the robot. The GPS solution isn’t really interesting in our case study because the scale isn’t precise enough except if we use some extremely precise (and expensive ) new GPS system. The accelerometer+gyroscope solution could be implemented as the main way of controlling the position or just as a corrector process this could be a very interesting way to explore 2D forward and backward kinematics.

We could also try to develop a SLAM (Simultaneous Localization And Mapping) algorithm for our robot. This process could be a very interesting way to develop a complex algorithm but could be a project on it’s own. Doing the map in real time could be a really interesting feature for our robot. It would require precise coordinate acquisition and fast data harvesting, this would be quite challenging but really rewarding.




The final and most complex improvement to the navigation we could do would be to create an algorithm of path planning to efficiently discover the room. This could be very interesting because it would create the link between this module and the Artificial Intelligence module. Creating this kind of algorithm is quite complex as it requires a really deep understanding of forward and backward kinematic alongside a solid knowledge of differential drive steering. It also requires a really important coding knowledge but overall could be an excellent master thesis topic.

# Part III: Environment mapping

The main focus of the project is to map the environment in which the robot is navigating. We chose to map both objects and signal strength of a Wifi hotspot in a designated area. To do so the robot needs to navigate through the area harvesting data on its environment.

## Wifi signal strength acquisition

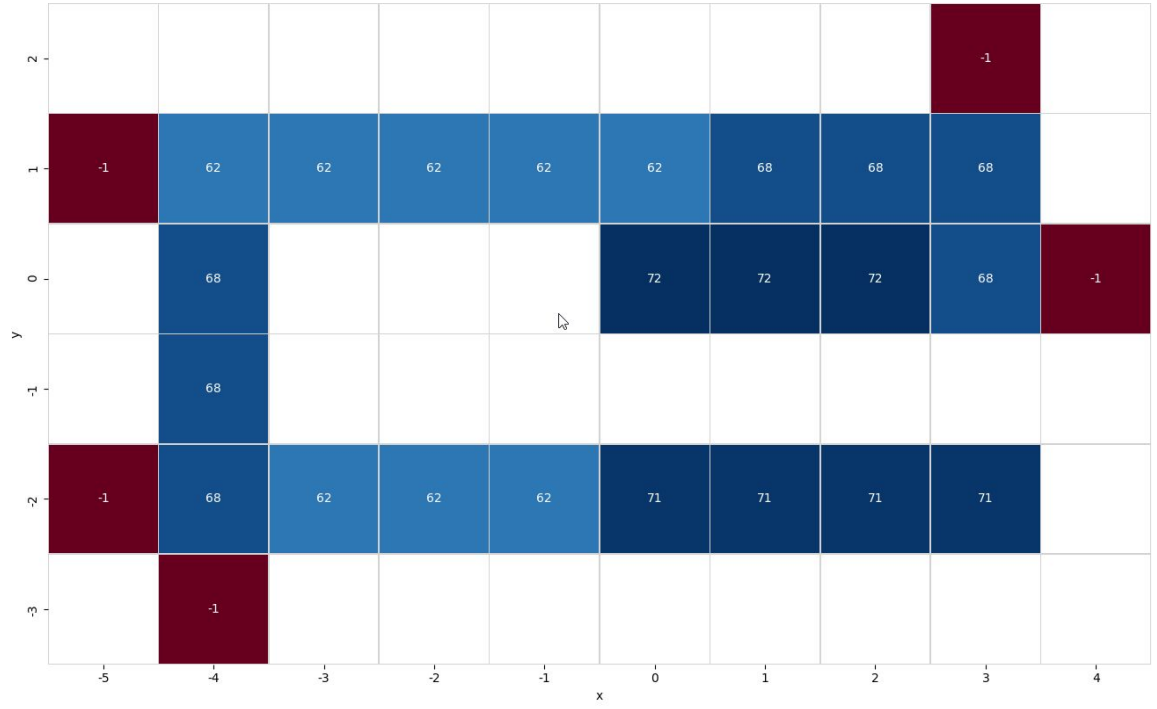
As our raspberry pi runs linux, we can use the subprocess library to check the output of the iwlist process which returns a list of the wireless networks with various details including the signal strength. Using some parsing, we can isolate the name (SSID) of the networks and their corresponding signal strength, then we look for the network name corresponding to our target and retrieve the signal strength.

We discovered that this process could be somewhat unreliable as, from time to time, it outputs a signal strength value of 0 for around 30s when it shouldn't. To work around this issue we created a function that waits until it gets a nonzero value. Unfortunately this means that from time to time the robot stops and waits for a wifi signal.

The robot scans for wifi each time it moves or turns. But, since it would not be able to scan the wifi where the obstacles are: when it encounters an obstacle, it defines the signal strength value of the space in front of it (where the obstacle is) as -1 so that we are able to see where the obstacles are on the heatmap.

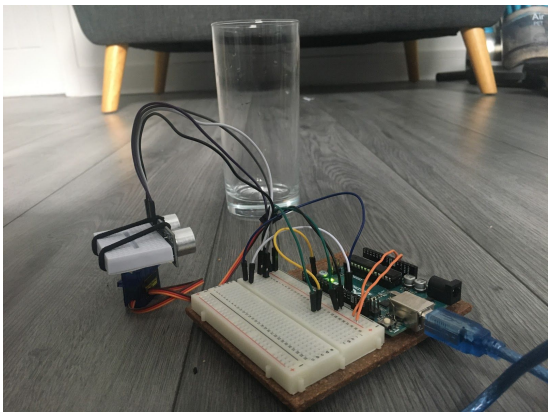
## Wifi heatmap

Creating a heatmap is a matter of reorganizing the data into a convenient dataset using the pandas library while making sure to average or discard any redundant values. Then using the seaborn library we can create a heatmap.

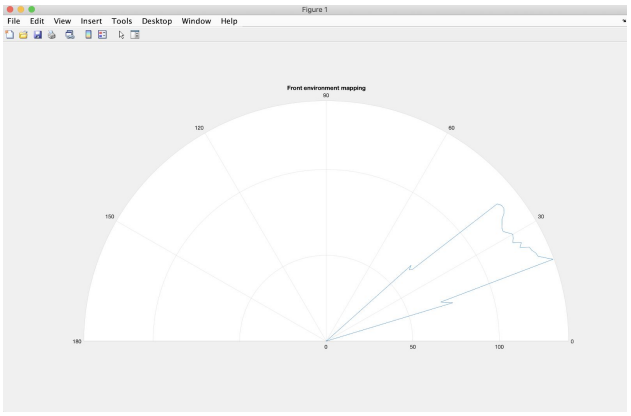


# Environment mapping

We had the chance to have an ultrasonic sensor, a servo, as well as an Arduino Uno Board. Since the university provides a free student Matlab (Mathworks) Licence, we managed to map the surroundings by sweeping the servo a full 180° and back in order to have an average of a distance for each degree in front of our robot.



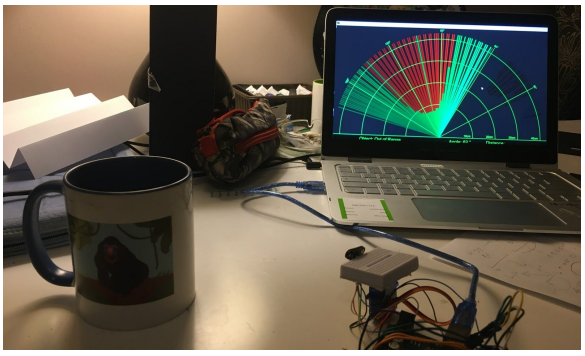
*Servo, HC-SR04, Arduino Uno and a glass*




*Environment Mapping In Matlab*

We initially thought that this could be an awesome cheap artefact for our robot for environment mapping. However, it's rather slow (as we sweep degree per degree) and quite far from our initial goal, being the signal intensity heat mapping.

As we can see on the polar plot, the values of the distance given by the ultrasonic sensor are rather inaccurate because it is a cheap sensor. We decided to buy a laser range sensor TOF-10120 for more precise and accurate results. We found some libraries online on live-radars to have a quicker view of the surroundings.



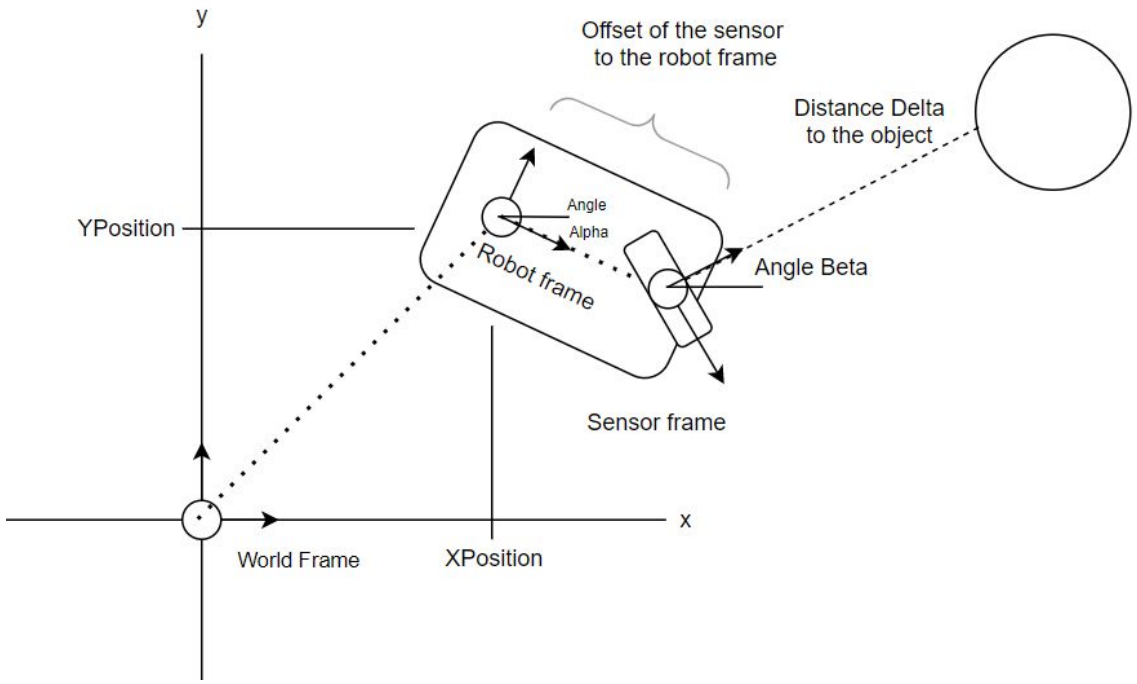
*Live Sweeping Environment Mapping*

Instead of using multiple maps, we will sweep the TOF sensor at every stop and create a scatter plot, cloud of points projected from his current position to see where the physical obstacles are.

# Map creation

A representation of the unknown space where the robot is roaming is one of the primary goals. To proceed this operation the robot makes pauses where it scans the surroundings using the time of flight sensor with a servo motor.
This operation gives the distance of the environment relative to the robot, those distances needed to be reported to the world frame to create relevant maps. To do so we needed the position of the objects with respect to the robot and then of the robot with respect to the world.

The system can be modeled as follow:




Through its journey the robot is collecting data as a CSV file. It contains the records of the position of the robot, its angle relative to the world frame, and the distance of the objects in front of it.
The specific frame we used for the data storage is represented below:

| X position | Y position | Angle Alpha | N value of Distances | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 868.0 | 765.5 | 675.0 | 609.0 | 554.0 | 533.0 |
| 150 | 0 | 0 | 296.0 | 506.5 | 449.0 | 415.5 | 378.5 | 365.0 |
| 300 | 0 | 0 | 172.5 | 162.5 | 238.0 | 230.0 | 209.5 | 199.5 |
| 300 | 0 | 1.5707963 | 153.5 | 149.5 | 265.5 | 230.5 | 214.5 | 206.5 |

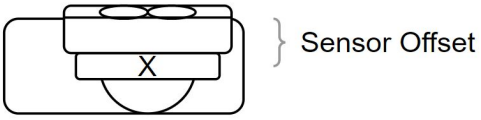
This file is then processed on a different program.
The X, Y and Alpha position is determined by the robot using odometry. As it is a relatively poor way of positioning, the system has been simplified. The robot is only moving along a grid following the X or Y axis with a constant step length. Alpha can thus only take n*(PI/2) values.

The Beta angle is not explicitly saved on the file but is determined by the number of measures and the spanned Angle by the sweep.

```
span = 90# Total angle range
Nstep = 16 #Number of taken values in sweep
b = (span/2) - (((idx-3)*span)/(Nstep-1))
```

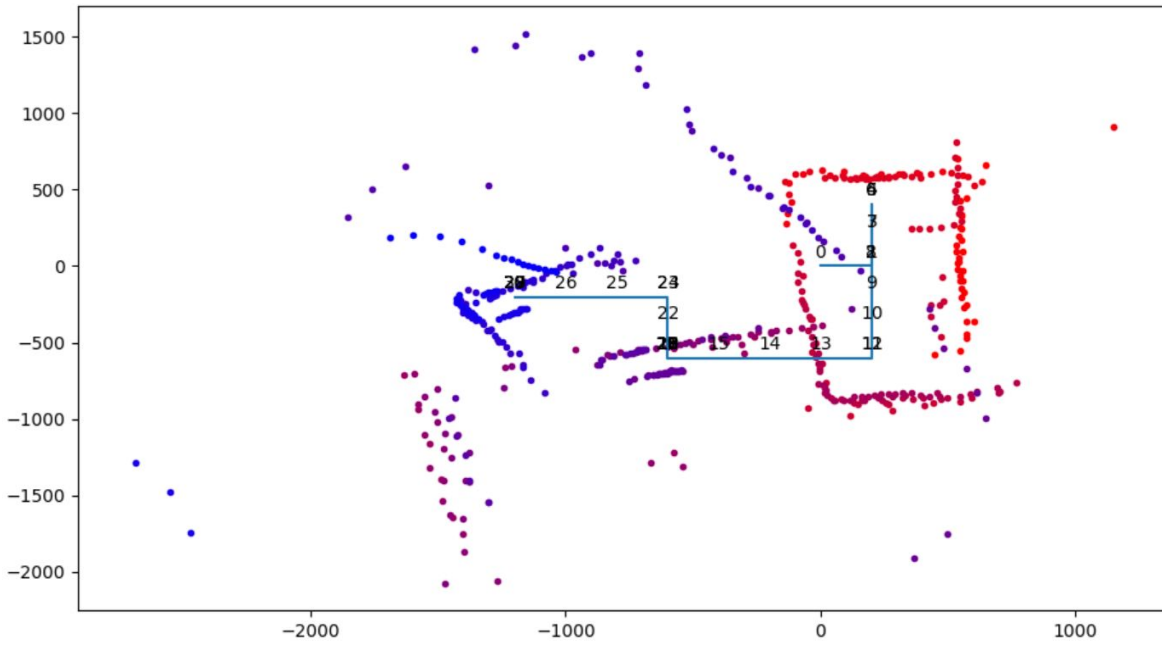
The system is 2D and the obstacles positions rely on the self determined robot position, thus the program is directly computing the value of the obstacle position in the world frame. using the following equations:

$$Xobject = Xposition + CoreOffset * cos(Alpha) + (delta + SensorOffset) * cos(beta + alpha)$$
$$Yobject = Yposition + CoreOffset * sin(Alpha) + (delta + SensorOffset) * sin(beta + alpha)$$

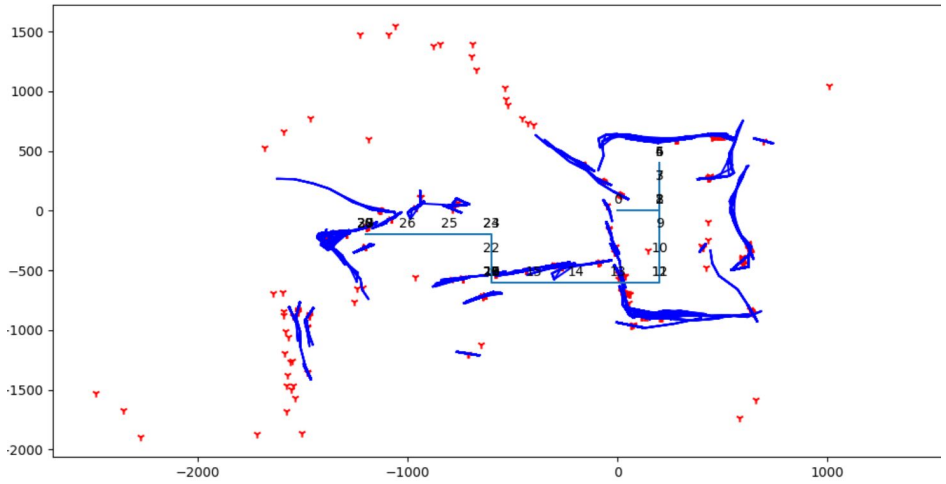



The Sensor offset being the distance between the sensor and the servo axis.

Without further computing this is the plot obtained. It shows the various measures along with the paths of the robot (The blue line).



To obtain a better vision of the environment, another map making program is sorting the data, and creating walls between close points.




## Ways of improving

To improve the map making, there are two paths to explore.

- The first one concerns the robot and the ability to measure precisely its environment. To do so, better sensors could be used, regarding accuracy and efficiency. Sensors like lidar would allow to produce a great amount of data, giving more points to render the surroundings. Those measurements need also to be properly attributed in the world frame, requiring good positioning of the robot. In addition to odometry, an accelerometer and encoder wheel would improve the accuracy of the position measurement. At the moment the position precision is quite poor due to slipping and other perturbation not considered.
- The second path would be a better data processing algorithm. We would think of detecting incoherent values, guessing the shapes of the environment, and delete the non static objects. The possibilities are infinite and will depend with the needed application.

Navigation algorithm is also closely related to map making efficiency, and could increase its efficiency with better space roaming.

Also, the measurement of other values than wifi strength could be imagined, and the same map could be realised using other sensors. The mapping with measurement would be relevant in hazardous areas like nuclear plant accidents, where the hazard could be measured before sending humans.

# Conclusion

Although the robot does not complete its purpose with ideal and precise results, it allowed us to discover the various possibilities of robot conception and to adapt with what we had in our hands. A different approach is needed while designing an autonomous robot. Once the autonomous vehicle starts it has to react in adequacy with our analogical world and find the solutions by itself.

The conception of this robot allowed us to acquire knowledge on sensors and their reliability, 3D design and printing, autonomous navigation, kinematics, etc... Each iteration of our project made us understand the previous mistakes.

This project was made in a short period of time, but thanks to the continuous implication and interest of the whole team, the robot has been brought to an advanced stage. Regarding the specific work frame due to the health crisis, we managed to establish methods to optimise our work. This implied the creation of a shared folder online, and a daily communication to exchange on the next tasks, faced issues or achieved goals.

We have precise ideas to improve various parts of the conception which would allow us to get closer to the first objectives.

Because of the pandemic, we sometimes had to have remote meetings, mostly during lockdown 2. However, we still managed to meet at each other's places every now and then to run the robot. This project was awesome because of the freedom of assignment, we could decide whatever we wanted to present to the teacher.

To ease the access to all the resources of the project we decided to create a Git Hub. All the codes are available at: https://github.com/TimMilhomme/wifiHeatMapRobot. To present the robot we also released a video available on Youtube:https://youtu.be/JUGqk_iCWYM.

# Bibliography

## Hardware

Thymio specifications:
https://www.thymio.org/fr/

Control Thymio From Raspberry:
http://wiki.thymio.org/en:thymioraspyexample

## Navigation

Ros frontier exploration
http://wiki.ros.org/frontier_exploration

## Mapping

Mapping Your Surroundings Using MATLAB and Arduino:
https://uk.mathworks.com/videos/mapping-your-surroundings-using-matlab-and-arduino-121311.html

Arduino Radar Project:
https://howtomechatronics.com/projects/arduino-radar-project/

Wifi acquisition:
https://www.geeksforgeeks.org/how-to-find-available-wifi-networks-using-python/

Heatmap in python:
https://www.statology.org/heatmap-python/