
RAT

Remote Autonomous Transporter

(formerly CAT, "Coche Autónomo Teledirigido")

A remotely controlled robot that travels to a specified location while avoiding obstacles.

Daniel Azemar	1390451
Jialuo Chen	1390203
Sergi Pous	1390450
Adrià Rico	1391488

PROJECT SPRINT #2.

DATE: 03 May 2017

Table of Contents

Project description	3
Electronic components	4
Scheme	6
Extra components and 3D pieces	9
Algorithms	17
Initial objectives and fulfilled objectives	21
Annex: RAT WAITER	22
Annex: pictures and videos of the development of RAT	24

RAT

A remotely controlled robot that travels to a specified location while avoiding obstacles.

Note: amendments from sprint 4 are written in blue.

Project description

The sole purpose of RAT is to transport an object from one point to another. To accomplish this, RAT will wait until it remotely receives a coordinate to go to, then it'll travel there while avoiding any obstacle in the way. It'll also periodically send its current position to the remote controller, so that the robot can be localized at any time. [Alternatively, RAT will also be able to be manually controlled.](#)

Below is a detailed description on how we plan to carry out these functionalities:

- Movement:
RAT will be able to move using two DC motors (with wheels attached) controlled by a motor driver board. This setup makes it possible for RAT to move forwards and backwards with an adjustable speed. Rotation will be accomplished by activating only one of the two motors.
- Remotely sending and receiving data:
We'll use two transceivers controlled by two Arduino UNO boards. One of them will be located in the robot, and the other will be connected to the computer used to send the coordinates and receive the current robot position.
- Route tracing:
When RAT receives an objective coordinate, it'll get its current position and orientation using a GPS and a magnetometer respectively. It'll then trace a

straight line from its current position to the objective coordinate, calculate the direction, and rotate accordingly. When it determines it's facing the correct direction, it'll start moving forward until an obstacle is found or the objective coordinates have been reached. Periodical checks will be made to make sure it's not going offtrack.

- Obstacle avoiding:
RAT will be able to detect nearby obstacles using a sonar. Once an obstacle is detected, it'll try to overcome it and then resume its original path. We still don't know a lot about how to code this, but we've been promised we're going to be taught.
- Manual remote control:
RAT will be able to be remotely controlled via a Telegram bot with a simple button interface. The computer running the bot must also be connected via USB to the Arduino used to send signals to RAT. Alternatively, it'll also be possible to manually control RAT directly from the computer.

Note on the limitations of the distance traveled:

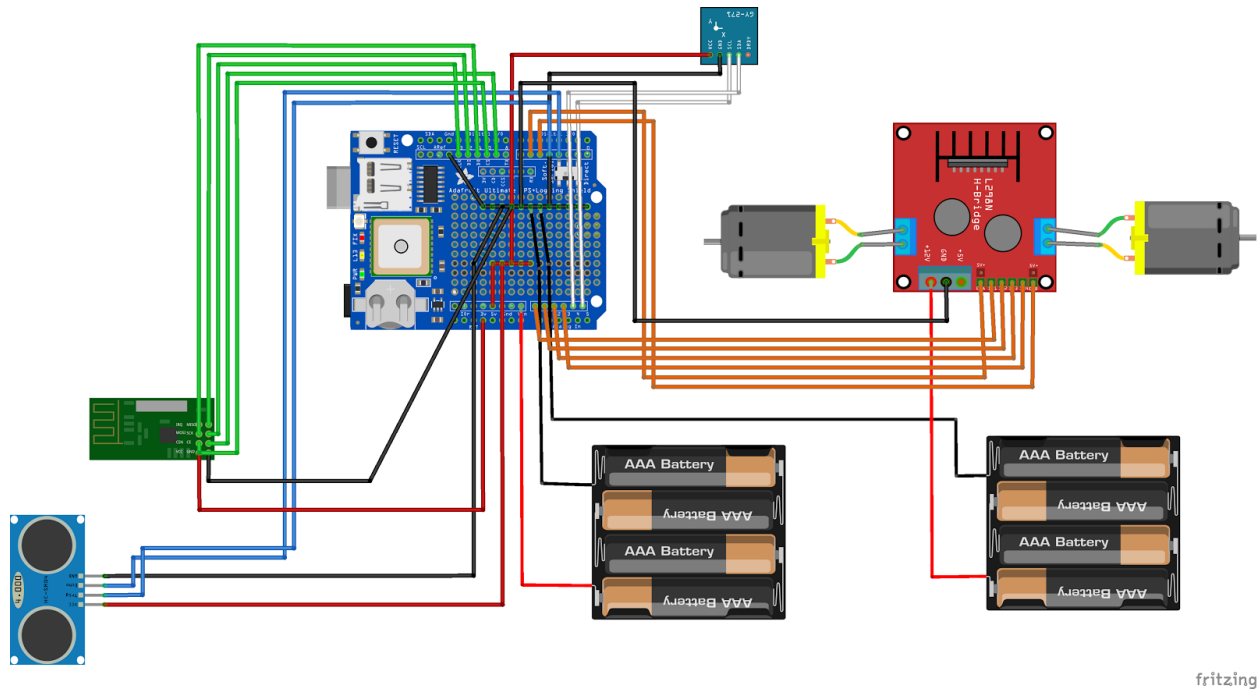
The precision of the GPS used is around 3m, so RAT's duty is to land inside an area of 3m around the specified point. Specifying points too close to RAT would make it The transceiver used has a range of about 100m, so making RAT go towards points farther than that leads to losing all kind of communication with it. It could still, in principle, autonomously get to its objective, and then receive new orders from a different computer nearby.

Electronic components

- Arduino UNO (2x) (one of them is of our own, it'll be used to send data from the computer to the robot using the transceiver)
- nRF24L01P+ (Transceiver) (2x)
- L293D (Motor Driver Board)
- HC-SR04 (Sonar)
- Adafruit Ultimate GPS Logger Shield
- DC motor (2x)
- GY-271 (Magnetometer)

Scheme

RAT



Notes:

- All red wires go to Vcc (5V for all the components except the transceiver which goes to the 3.3V pin).
- All black wires go to GND.
- All the AAA batteries shown in the scheme are 1.2V AA batteries in actuality.
- The GPS shield uses digital pins 0, 1 (for the direct connection), 7 and 8 (for the soft serial connection).

Connections:

Sonar (blue)

Trigger Pin -> Digital Pin 3 (PWM)

Echo Pin -> Digital Pin 4

Motor Driver (orange)

ENA -> Digital Pin 5 (PWM)

ENB -> Digital Pin 6 (PWM)

IN1 -> Analog Pin A0 (Digital Pin 15)

IN2 -> Analog Pin A1 (Digital Pin 15)

IN3 -> Analog Pin A2 (Digital Pin 16)

IN4 -> Analog Pin A3 (Digital Pin 17)

Transceiver (slave, green)

CE Pin -> Digital Pin 9 (PWM)

CSN Pin -> Digital Pin 10 (PWM)

SCK Pin -> Digital Pin 13

MOSI Pin -> Digital Pin 11 (PWM)

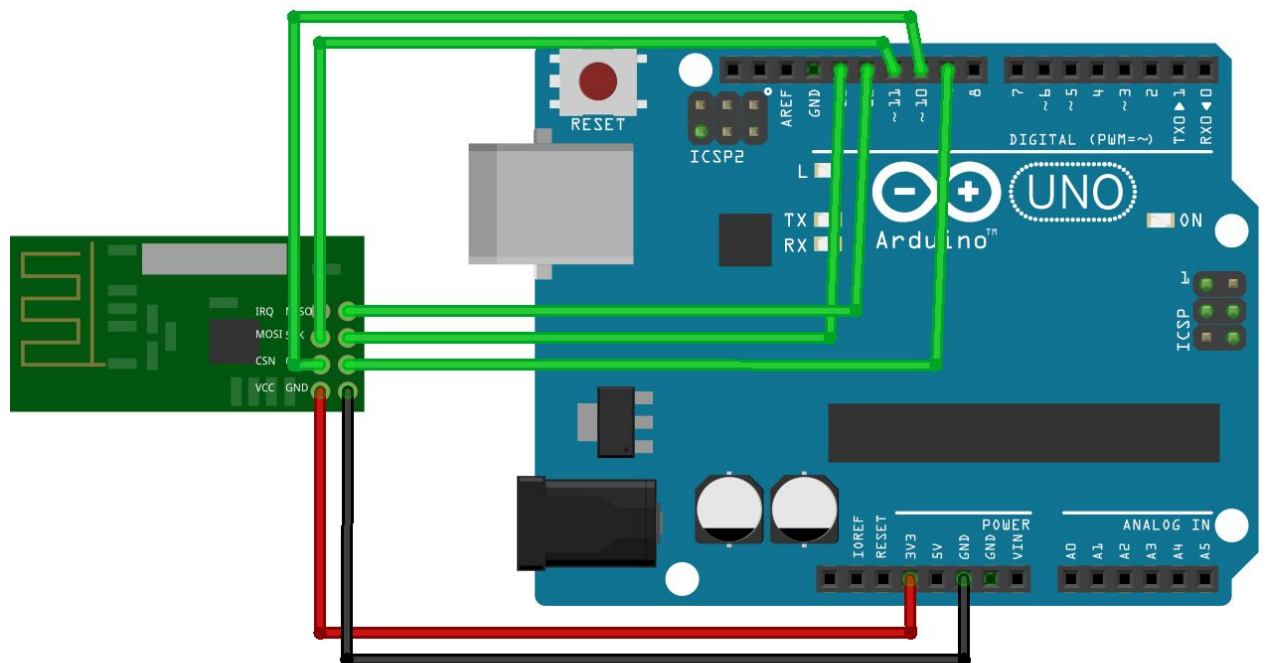
MISO Pin -> Digital Pin 12

Magnetometer (white)

SCL Pin -> Analog Pin A5 (I2C Clock)

SDA Pin -> Analog Pin A4 (I2C Data)

Master (computer)



fritzing

Connections:

USB alimentation

Transceiver (master, green)

CE Pin -> Digital Pin 9 (PWM)

CSN Pin -> Digital Pin 10 (PWM)

SCK Pin -> Digital Pin 13

MOSI Pin -> Digital Pin 11 (PWM)

MISO Pin -> Digital Pin 12

Extra components and 3D pieces

- Wheel for DC motor (2x)
They'll be connected to the motors to allow RAT to move on ground.
- Caster wheel
A support wheel which will be placed on the back of the robot.
- 4 AA Battery Holder (2x)
Used to conveniently hold 4 AA batteries in series so that they produce 5V of voltage.
- 1.2V AA Rechargeable Battery (8x)
4 of them are to power the Arduino board, the other 4 to power the motors.

List of 3D pieces:

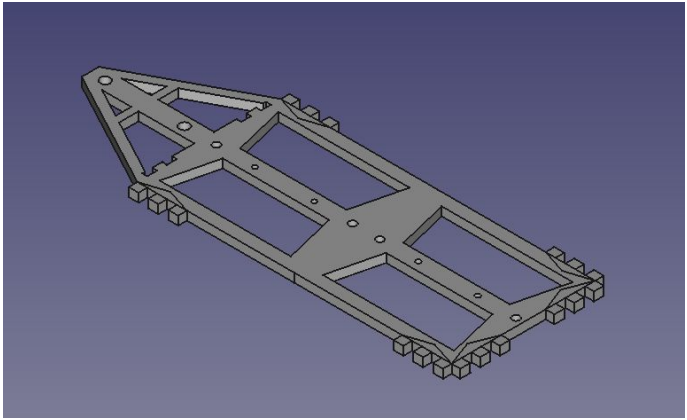
V1 problems and announcement of V2:

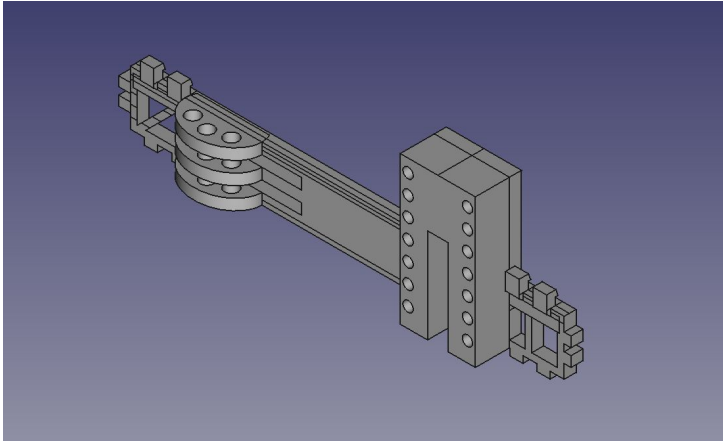
After having seen the errors that V1 had, we are encouraged by a new design and a second print, this time with the aim to be more consistent.

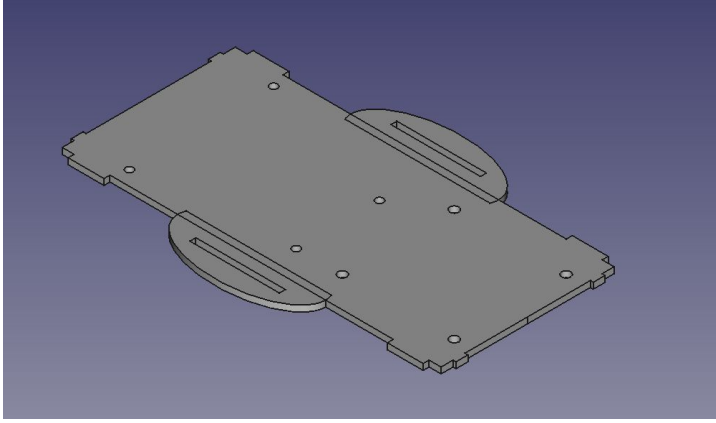
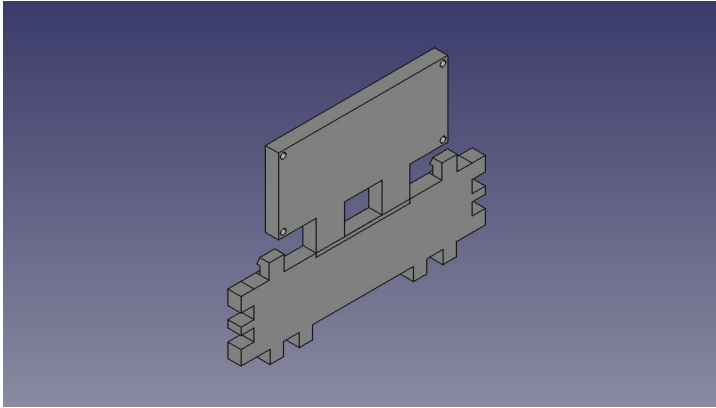
The main purpose of this reprint isn't to fix the errors that we've detected, but to generate fewer pieces (for example, the box holding the batteries is now only one piece, not 5 like before, one per wall).

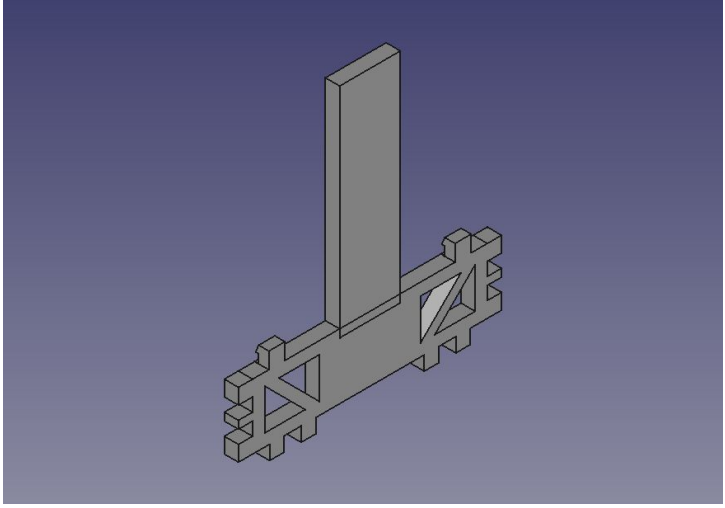
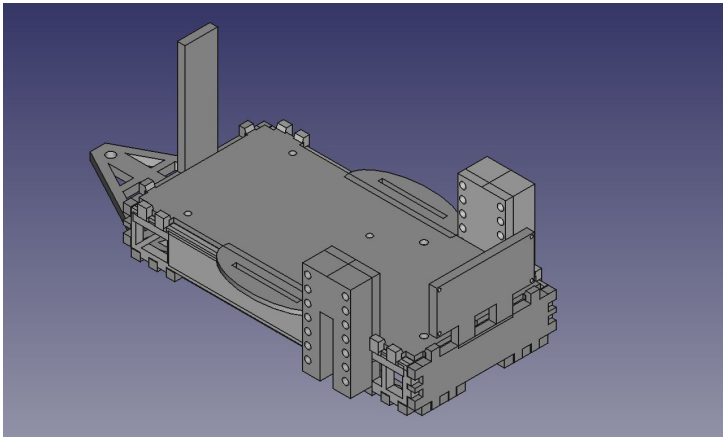
We think that'll fix many problems when joining the pieces (we've struggled a lot with V1) and it also makes the modelling is easier.

Below, we'll show the errors in V1 within the table of pieces of the last sprint:

No.	Description	Image
1	Base of the robot. Here we'll attach both the Battery Holder and the Caster Wheel.	

	<p>Errors detected:</p> <ol style="list-style-type: none"> 1. We put one hole of the caster wheel in the wrong place. We had to drill it. 2. We realized that maybe we needed 10 batteries instead of 8 to supply enough power to the Arduino and all sensors. We had to expand the base. 	
2	<p>Laterals of the robot.</p> <p>Here we'll attach the motors.</p> <p>...and a mystery piece, coming up next sprint!</p>	
	<p>Errors detected:</p> <ol style="list-style-type: none"> 1. The batteries didn't have the holes to pass the wires in the correct place. Had to drill it. 2. We missed a protrusion in the motors so we had to melt the piece to make them fit. 3. The "fit-in" piece to hold the cover wasn't printed precisely enough; the cover did not fit. 	

3	<p>Cover of the robot.</p> <p>Here we'll attach both the Arduino UNO board and the Motor Driver Board.</p>	
	<p>Errors detected:</p> <ol style="list-style-type: none"> 1. The holes where the chips and boards had to fit were not precisely placed. We had to struggle to pass the screws through the board's holes and as a result they were under a lot of pressure due to the tilt of the screws. 	
4	<p>Front of the robot.</p> <p>Here we'll attach the Sonar.</p>	
	<p>Errors detected:</p> <ol style="list-style-type: none"> 1. The holes where the sonar had to fit in were so close to the borders that the printer skipped the thin wall that was separating them. The sonar is now hold by a rubber band. 	

5	<p>Back of the robot.</p> <p>Here we'll attach the Transceiver.</p>	
	<p>Errors detected:</p> <ol style="list-style-type: none"> 1. The transceiver holder is way too small. 	
ALL	<p>Overall view of the structure.</p>	
	<p>Errors detected:</p> <ol style="list-style-type: none"> 1. The batteries in the inside were too tall. The cover couldn't fit at all. 	

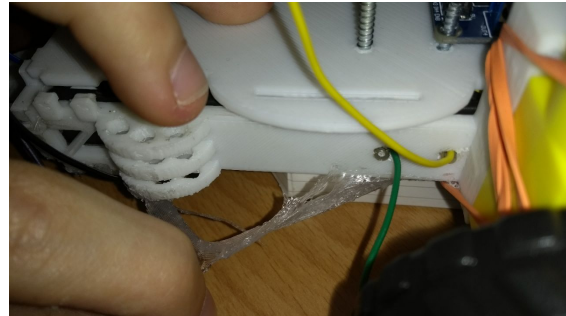
Note:

We'll join all the pieces together by sticking their respective hooks and ledges with superglue. However, the cover will use Snap-fit to make it easy to put it in and out, since the Arduino Board and the batteries are inside the robot, and we'll want to have easy access to them.

For future works!

If you're gonna use silicone to stick the pieces together,

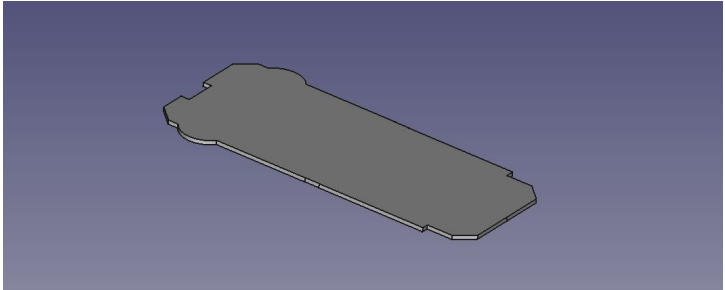
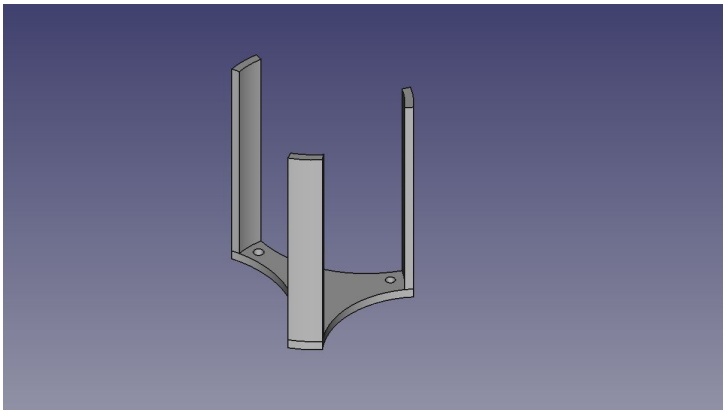
DO NOT USE **KITCHEN** SILICONE! The results are AWFUL.



V2 pieces:

With all that we've mentioned before, here are the new pieces that will hopefully fix all of those problems at once.

No.	Description	Image
1	<p>Base of the robot.</p> <p>Here we'll attach both the Battery Holder and the Caster Wheel, and sonar, and transceiver, and motors.</p> <p>...and a mystery piece, coming up next page!</p>	

	<p>Errors detected:</p> <ol style="list-style-type: none"> Well, the truth is that this piece is already printed. And it has errors... We wanted to print it all in one piece so that we didn't have to join the pieces (see the sticking problems we had before). The problem came with the "fit-in" piece to hold the cover: Just bending it a little was enough to make it break. 	
2	<p>Cover of the robot.</p> <p>Here we'll attach both the Arduino UNO board and the Motor Driver Board.</p>	
3.	<p>And... Ladies and Gentlemen: With all of you... the MYSTERY PIECE.</p> <p>What is its purpose? What was it made for? Will it be useful for the robot at all? SEE RAT WAITER!</p>	

Algorithms

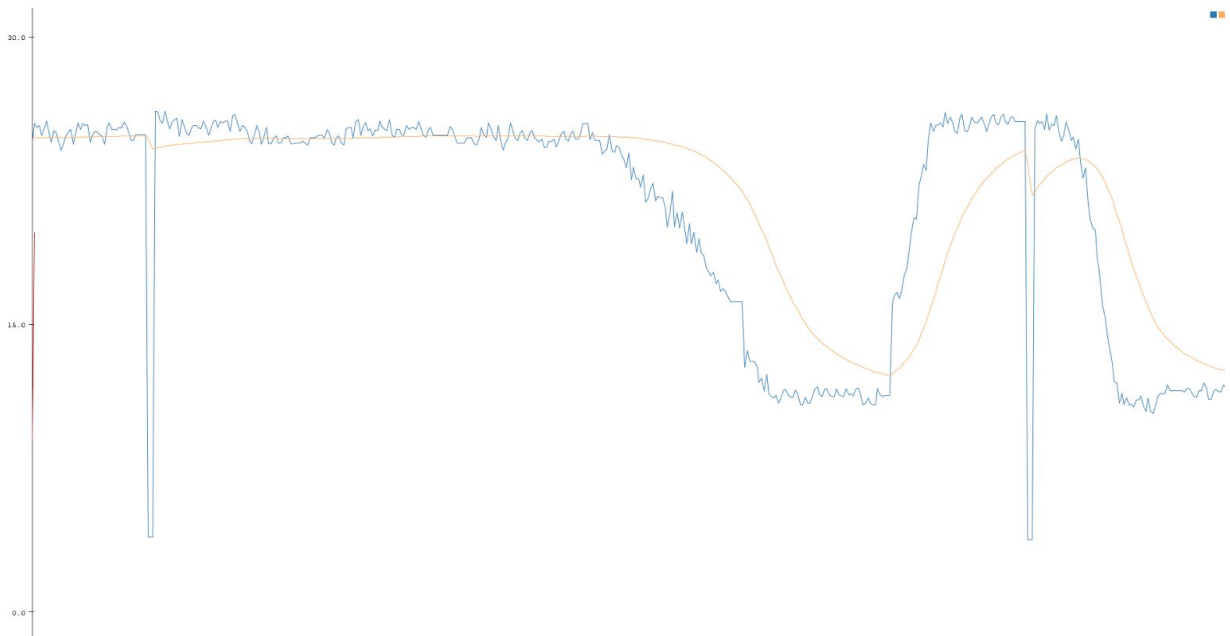
Code on RAT

Sonar

The sonar sends a high-frequency sound wave, then it starts a timer. When the sound wave bounces off an object and returns to the device, the timer stops and the time passed is sent to the Arduino board.

To transform this time to distance, we have divided it by 2 (because the sound travels forward, then backwards), converted it to seconds and multiplied it by the speed of sound.

Because the sonar signal can be quite unstable, we applied a Kalman filter to it.



A graph showing the actual distance obtained by the magnetometer (blue) and the Kalman filtering (orange). As you can see, Kalman helps to avoid the sudden peaks towards 0 (which are caused by the GPS field taking up all the resources during relocation)

GPS

The GPS shield automatically gives us the current location of RAT, thanks to the official library.

Transceiver

We used a very simple protocol to transmit information from the computer to RAT.

First of all, to ensure that no messages are lost, every time we send one we wait for an acknowledgement, and if none is received, we send the message again.

If the first byte received by RAT is not 255, it's interpreted as a manual control function, which could be one of the following:

Code	Meaning
0	Rotate left forward
1	Go forward
2	Rotate right forward
3	Stop
4	Rotate left backwards
5	Go backwards
6	Turn right backwards
7	Increase speed
8	Decrease speed

If the first byte received is 255, it means the computer's about to send a location.

RAT will wait for 4 more bytes to arrive and cast them to a float representing the latitude, then it'll wait for another 4 bytes and cast them to a float representing the longitude. It'll then enter the automatic mode.

RAT

Magnetometer

The magnetometer gives us the intensity of the magnetic field on each axis.

Since we're only interested in rotation around the axis perpendicular to the ground, given how the magnetometer is placed on the robot, we can completely ignore the Y axis, and calculate the angle of the magnetic field by doing the $\arctan2(Z_axis, X_axis)$.

Motors

We control the motors through the motor board. We send the speed of each motor through the ENA and ENB pins. We control each motor using its two designed pins. Here's a table showing how we rotate each motor the desired direction:

Desired effect	Pin 1	Pin 2
Move forwards	LOW	HIGH
Move backwards	HIGH	LOW
Stop	LOW	LOW

Then, to move RAT, we combine the two motors' rotation. Here's a table of the rotation of each motor to make RAT move how we desire:

Desired effect	Left motor rotation	Right motor rotation
Go forward	Forward	Forward
Turn left forward	Stopped	Forward
Turn right forward	Forward	Stopped
Go backwards	Backwards	Backwards
Turn left backwards	Stopped	Backwards
Turn right backwards	Backwards	Stopped

Rotate clockwise about itself	Forward	Backwards
Rotate anti-clockwise about itself	Backwards	Forward
Stop	Stopped	Stopped

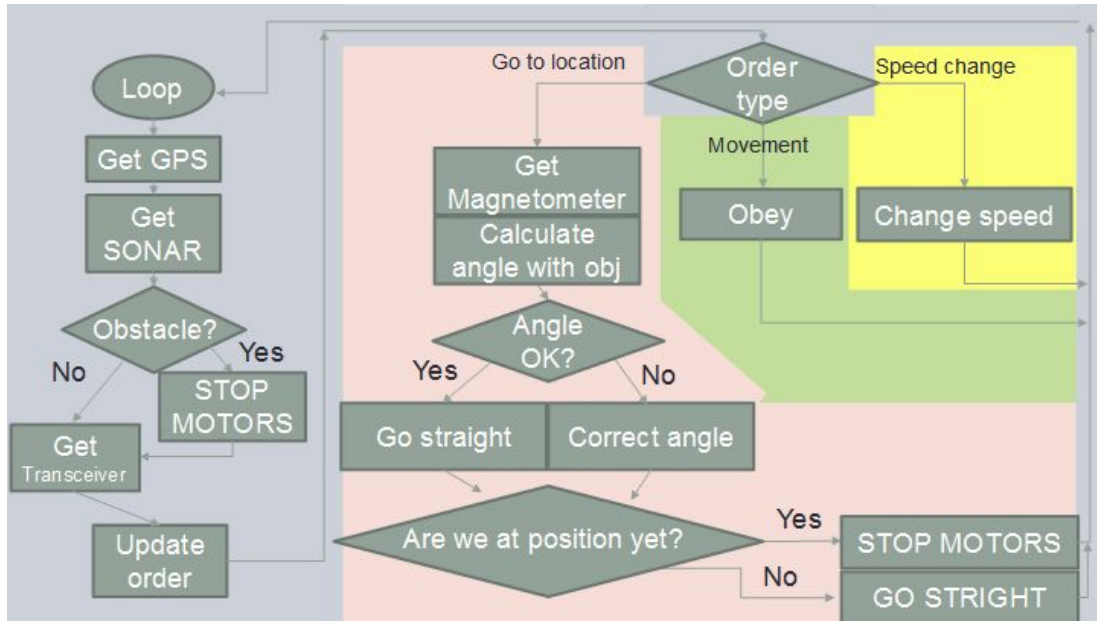
Automatic mode

When RAT enters automatic mode, some calculations must be made to decide where to point to go to the desired location.

First, the current position and orientation must be obtained (as previously described). Then, the objective angle is calculated using $\arctan2(\text{current_longitude} - \text{objective_longitude}, \text{current_latitude} - \text{objective_latitude})$. The order here is important, because we calibrated the magnetometer so that 0° correspond to looking straight to the north, which is the latitude axis. Thus, this axis would correspond to the standard X axis on the cartesian plane, and longitude would correspond to the Y axis, so $\arctan2(Y,X)$ corresponds to the previous expression.

Once the objective angle is known, we calculate which way is faster, if rotating to the left or to the right, and RAT rotates until a certain threshold is met, then it starts moving forwards towards the target coordinates until it arrives, periodically checking if it's on the right course and recalculating its objective angle given its actual position.

Below is an overview of the code flow



Code on the computer (master)

Telegram bot

The computer must be running a Telegram bot server for remote control via Telegram to be possible. The program accepts some input from the telegram API and parses it. If the message received is a location, it sends the magic 255 code (previously mentioned) to the Arduino board via the serial port (USB), and then 8 bytes corresponding to 2 floats which represent the longitude and latitude. If the command received is a manual control action, its code is sent directly via the serial port.

Arduino

The code running on the Arduino connected to the computer is pretty simple. It just listens to the serial port, and whenever it receives a byte, it forwards it to RAT via the transceiver. If the transceiver were to not acknowledge a byte, it is sent again. All other data processing (such as interpreting the coordinates) is done on the RAT code.

Initial objectives and fulfilled objectives

No.	Description	Fulfilled?
1	Remote control via computer	Yes, but you have to send to commands manually through Python, we didn't develop a GUI.
2	Remote control via Telegram bot	Yes, completely functional.
3	Obstacle avoiding	Kind of. We stop and wait until the obstacle is not there anymore. The original objective was to go around them.
4	Autonomous movement towards a specific coordinate	Kind of. RAT correctly turns and points to the correct direction (with an error of $\pm 10^\circ$), but it can get lost if there are interferences in the magnetic field. Also, because the motors are not powerful enough, it sometimes can't manage to turn, never gets to the objective angle and stops indefinitely.

Annex: RAT WAITER

MYSTERY PIECE

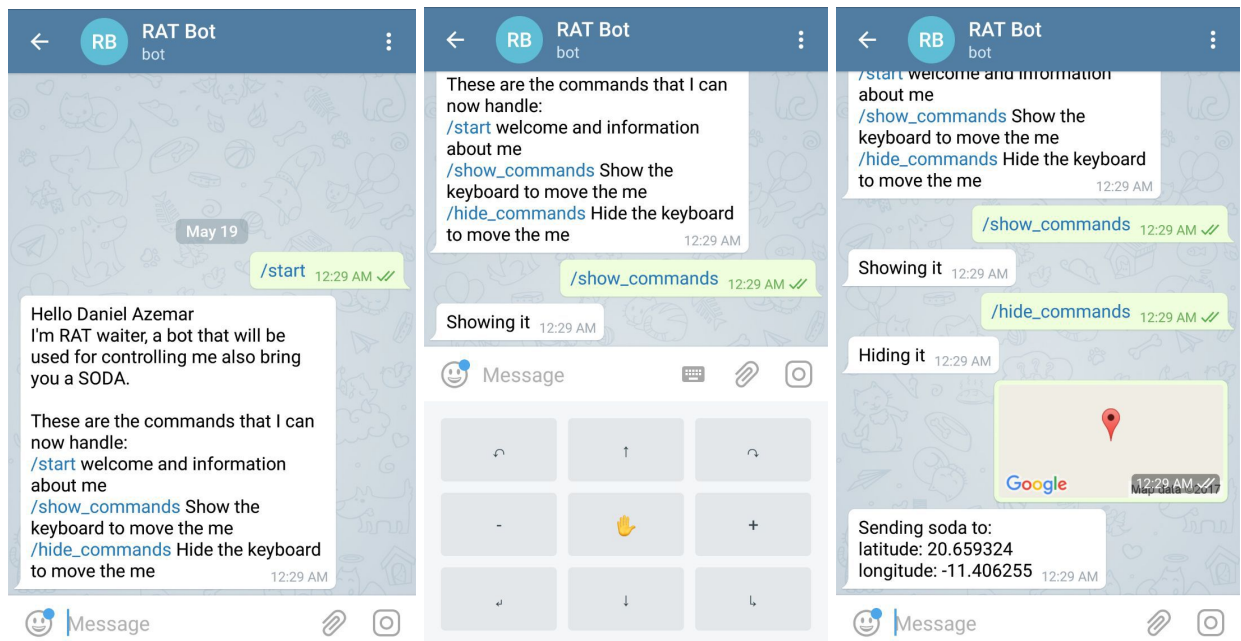
Presentation:

Have you ever thought it's too damn work going down stairs for a soda when you are thirsty at your office correcting exams?

Isn't it tedious meet students at the corridor when you just want to go to the coffee for a beer?

WORRY NO MORE!

INTRODUCING **RAT WAITER**

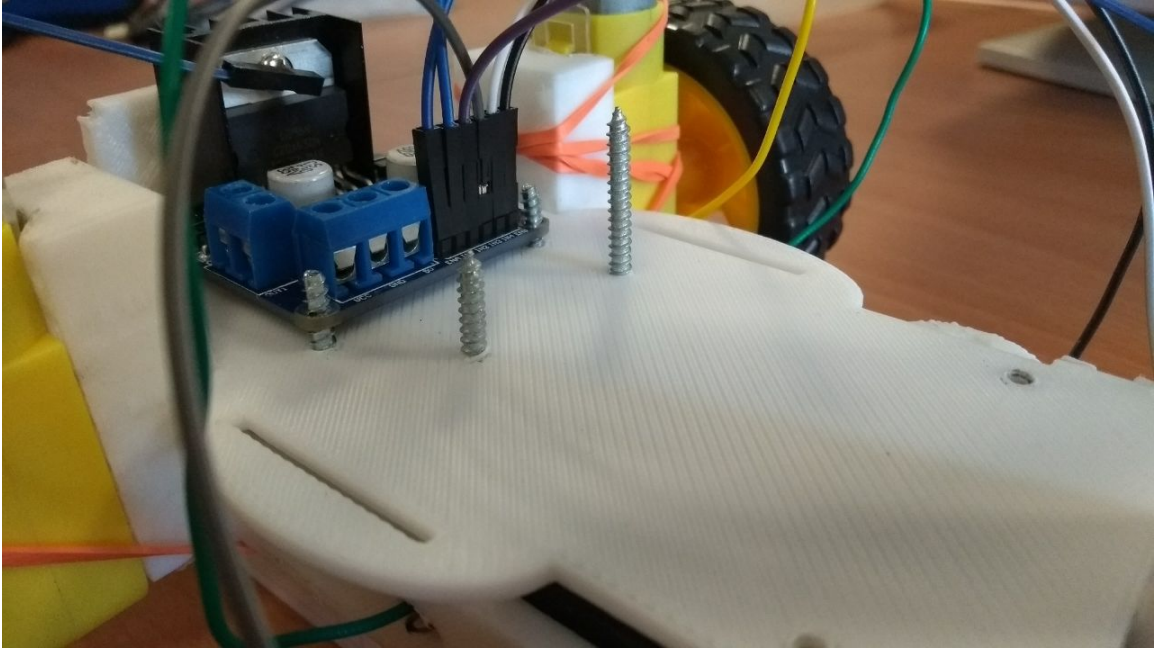


With the new bot we can control RAT in two ways: Manual or Autonomous.

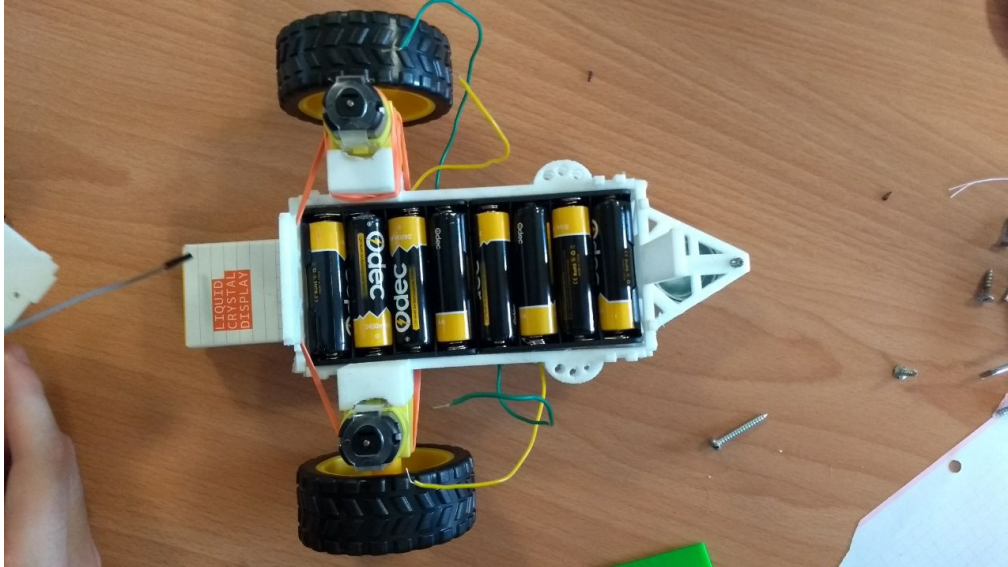
RAT

Annex: pictures and videos of the development of RAT

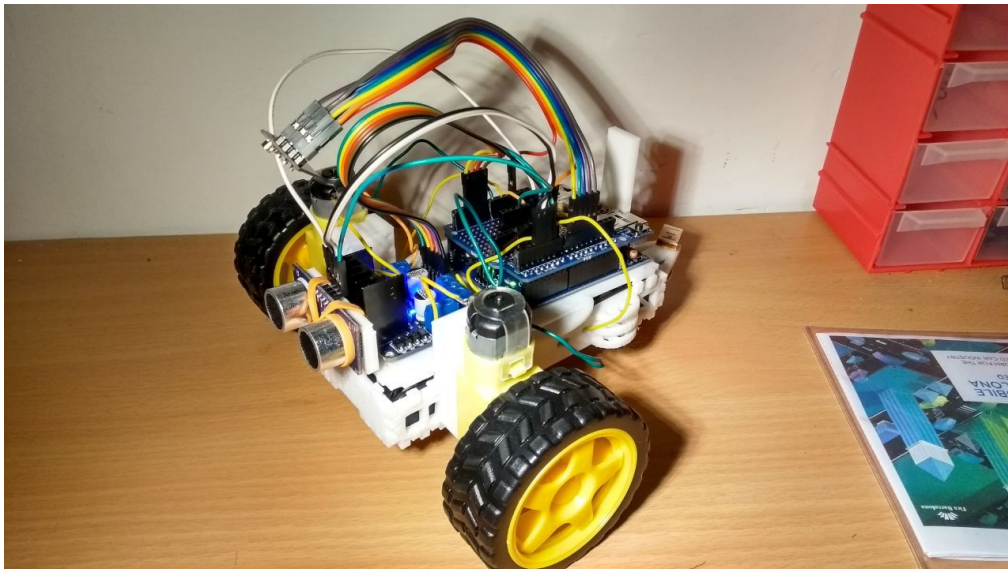
Here we'll show some pictures of our RAT.



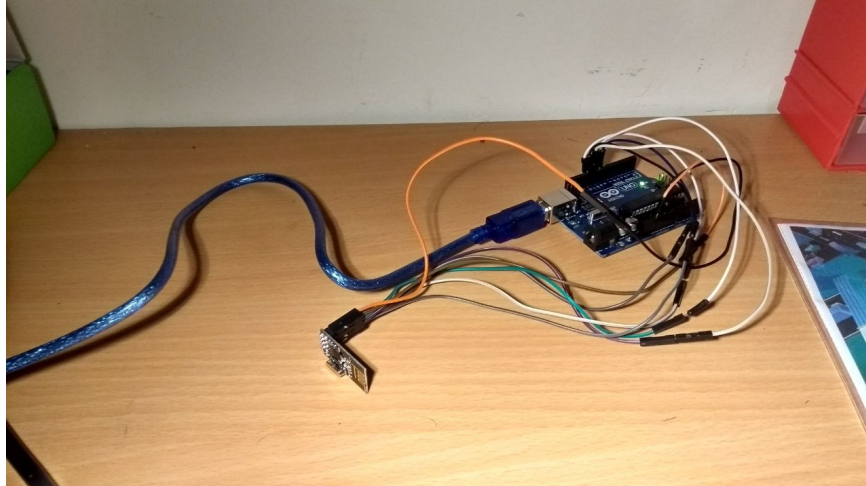
The problems with the screws. It's not appreciable yet, but the chips could eventually break if they diverge too much.



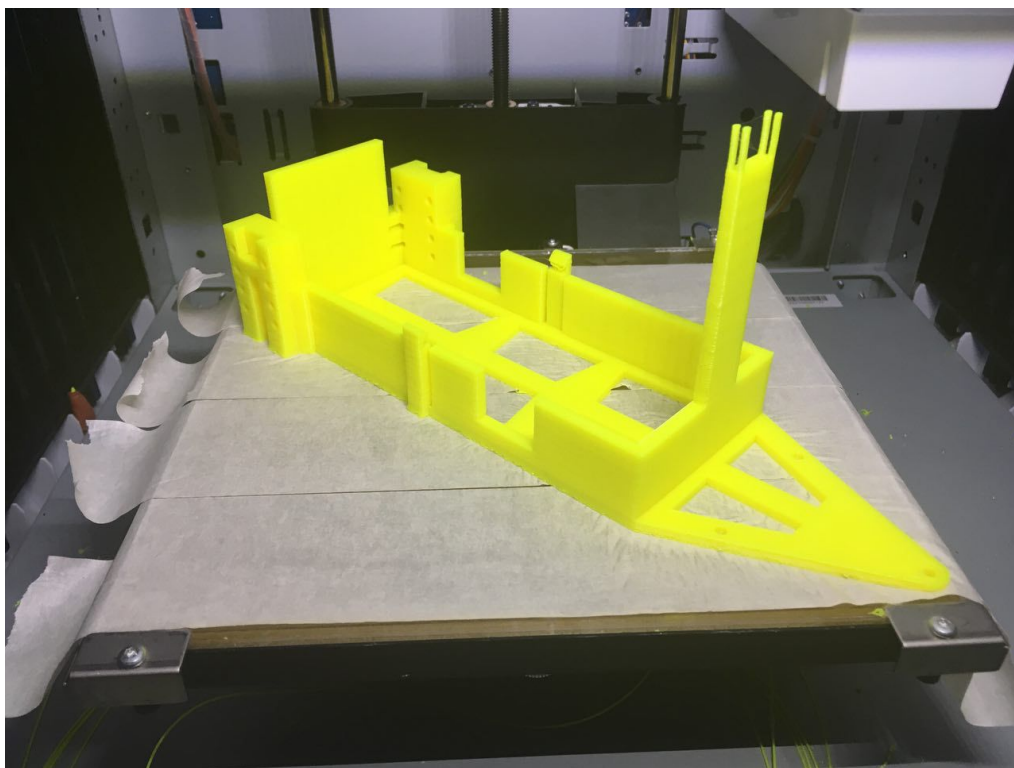
The batteries of the robot. They fit perfectly!



Our robot once (almost) fully assembled. The magnetometer just came yesterday and we've not been able to test it!

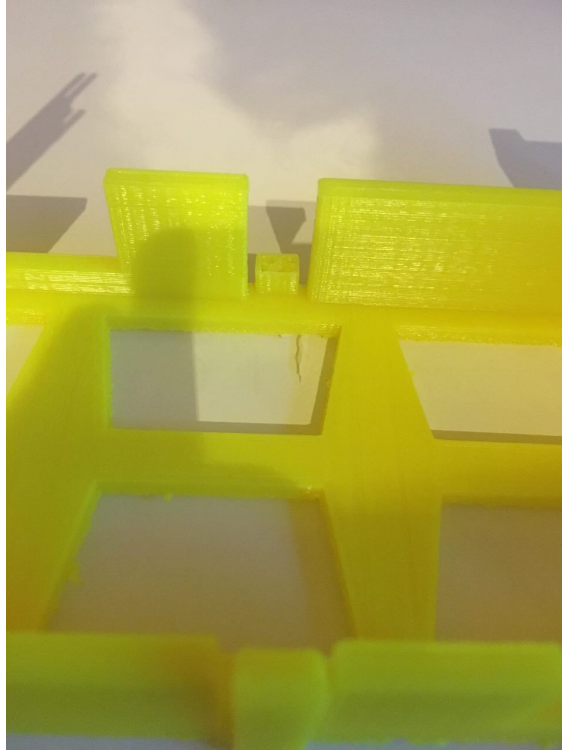


And here is the other transceiver, which is connected to the computer running the Telegram bot.

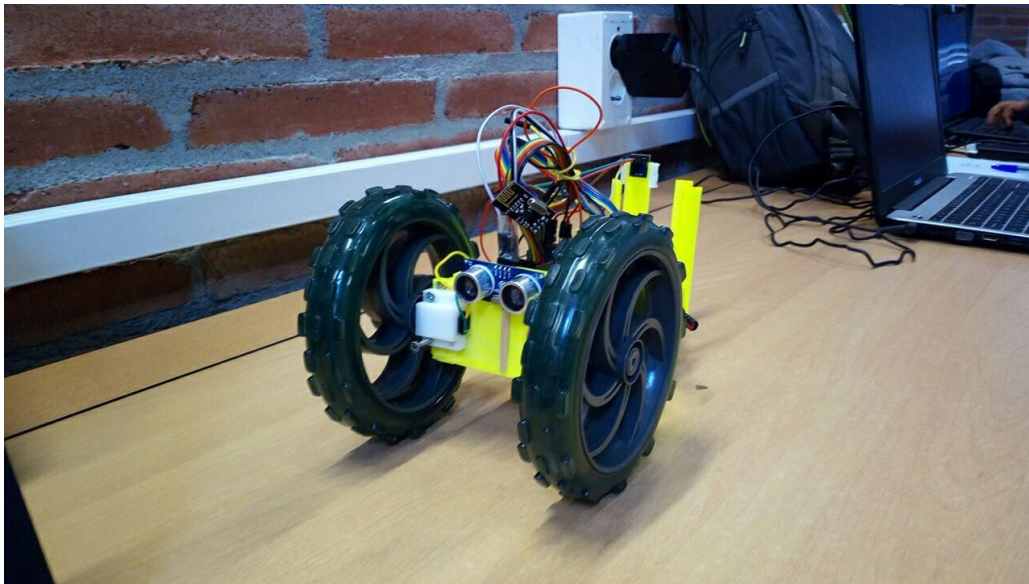


Here is the V2 base piece just coming out from the oven.

RAT



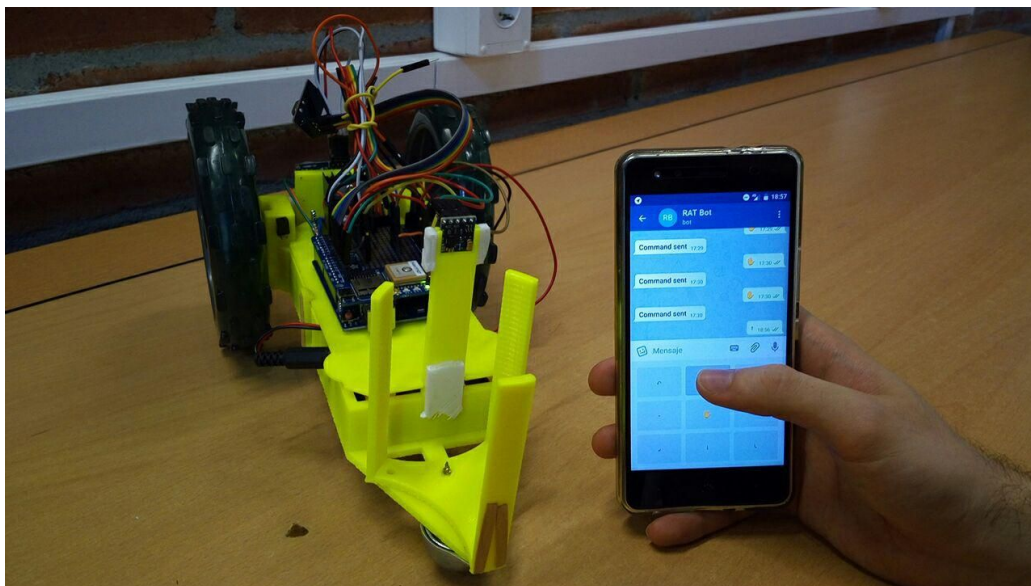
And here is the broken part, just coming out from the oven.



Final prototype with brand new wheels and motors, and 3D pieces.



Here it is carrying a water bottle.



And here it is controlled by mobile phone.

RAT

Here are some videos demonstrating what RAT is capable of doing at the moment:

First communication test with Telegram bot: <https://youtu.be/EcW7nljUPmk>

Remote controlling + obstacle detecting: <https://youtu.be/hFyf0Mygbd4>

Autonomous controlling with GPS and Magnetometer:

<https://youtu.be/UaKTnJVkuZE>

References

This project has been inspired by the following Internet projects:

<http://www.instructables.com/id/Arduino-Powered-Autonomous-Vehicle/>

<http://www.robotshop.com/letsmakerobots/fundamentals-a-gps-guided-vehicle>