



Modular mobile robot for localization experiments

SEMESTER PROJECT

Author :
Ahaggach Yassine

Supervisors :
Frederike Dümbgen
Miranda Kreković
Michalina Pacholska
Mihailo Kolundžija

October 8, 2019

MT-RO MA2

Contents

1	Introduction	3
2	List of specifications	4
3	Robot presentation	5
3.1	General Overview	5
3.1.1	Electronical layer	5
3.1.2	Mechanical layer	6
3.1.3	Experimentation setup layer examples	8
3.2	Critical choices and justification	9
3.2.1	Control and computations	9
3.2.2	Locomotion	9
3.2.3	Motor	10
3.3	Differential drive motion inverse kinematics	11
4	Accuracy and precision evaluation	12
4.1	Accuracy and precision of stepper motors	12
4.2	Systematic evaluations	13
4.2.1	Translation motion	13
4.2.2	Rotation and circular motion	14
5	Conclusion	15
6	Annex	16
6.1	Bill of Materials	16
6.2	Designs and Assembly	18
6.3	Software	20
6.3.1	GRBL and Octoprint	20

6.3.2 Custom scripts 23

7 Bibliography 25

1 | Introduction

Academic research is a complex field combining both theory and practice. From elaborating leads and hypotheses to thorough practical experimentation, research is a cyclic process where several attempts are expected before achieving any concrete result. In order to be as efficient as possible, balancing between both practice and theory in terms of material and time resources is of critical importance.

Experimentation usually amounts to data gathering (measurements), processing and evaluation (ground truth comparison). However, due to the uniqueness of each different research subject as well as imposed performance constraints, the setup and protocol can vastly differ.

In light of this context, automation in regards to conducting experiments contributes largely to streamlining and speeding up the research process. Empowering researchers with experimentation platforms grants them the freedom to manage their time and efforts as they see fit and provides them with characterized proven tools to support their work.

This project focuses on localization based experiments (e.g echolocation algorithms) and aims at providing a mobile modular robot. The robot acts as a basis for various experiments with a shared mobile aspect. The idea behind this project stems from the following observations :

- An autonomous modular mobile robot is a solution to the previously outlined challenges regarding research.
- The current state of mobile robots shows few to none, scalable modular mobile robots capable of interfacing with the various setups needed.
- Custom solutions are the expensive but conventional way to go to automate mobile experiments.
- It is often a matter of one or two components that differentiate one mobile experimentation setup from the other .

Currently supporting two experiments (echoSLAM and indoor wifi based localization), the robot's core design in hardware and software revolves around ease of use and modularity. The robot is precise, easily controllable and allows to effortlessly retrieve experimentation data.

With affordable and available materials, anyone with the plans as well as the documentation is capable of easily reproducing and repurposing the robot.

2 | List of specifications

The robot is an autonomous modular mobile platform that run various experiments. Each experimentation setup shares and connects to the same mobile platform and is supported by the structure. Through analysis of the robot's main tasks, the specifications to respect are as follows :

1. **Scalability** : The robot's design is scalable. In terms of mechanical and electronical performance, adding a complex experimentation layer should cause no issue.
2. **Reproducibility** : The robot's specific mechanical parts are easily reproducible through simple popular manufacturing processes (3d printing, laser cutting ...).
3. **Mechanical modularity** : The robot's design allows for additional mechanical parts to be assembled as to support specific experimentation setups.
4. **Load** : The robot must bear the additional load of the various experimentation setups with no impact on the performance.
5. **Exteroception** : The robot is capable of obstacle avoidance.
6. **Resolution of motion** : The robot is intended to operate in regular sized rooms (7-8m width) , the resolution is nominally set to be ≤ 1 [cm].
7. **Precision of motion** : The control of the robot is repeatable, the motion characterization is well defined.
8. **Trajectory following** : The robot is capable of smooth and precise trajectory following.
9. **Interfacing (electronics/software modularity)** : The robot supports most communication protocols and is capable of interfacing with a wide range of components.
10. **Data gathering** : The robot logs all data general and specific to the experiments and provides the user with a compiled archive at the end of each experiment run.
11. **Communication** : The robot's architecture allows for it to easily integrate other systems (camera tracking systems ,database...)
12. **Autonomy** : The robot runs on rechargeable batteries that can hold for several runs of the experiment. The battery also powers the experimentation setup.
13. **Ergonomy** : The robot is user friendly. Running experiments on the robot is automated and needs minimal to no intervention.

3 | Robot presentation

3.1 General Overview

The robot's architecture is based on a layers concept. Lower layers are for electronical / mechanical functions : power supply, control and localization of the robot (generic tasks that are shared among different experiments) while outer layers are specific to the experiment (e.g microphone + speaker for echo based experiments).

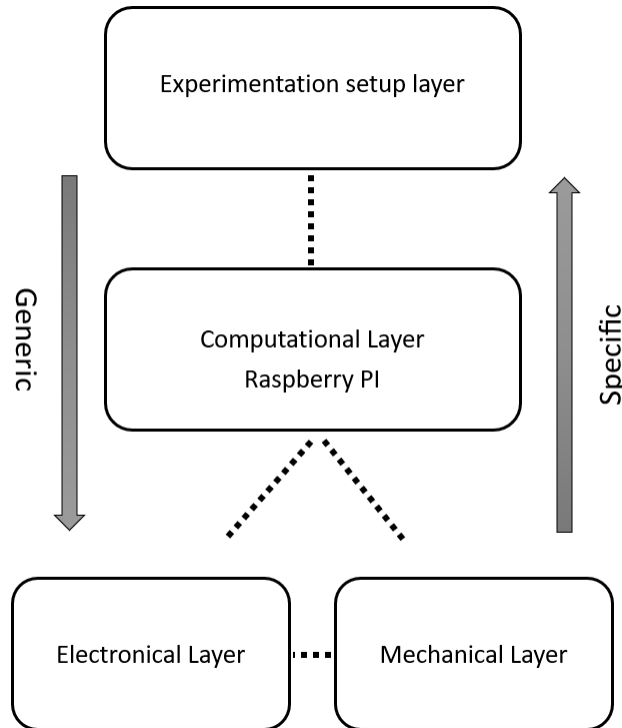


Figure 3.1: Global architecture

Each layer is designed independently with the specifications in mind. The idea is to achieve the most scalable, inter-operable system. Each layer can be replaced, modified and upgraded independently and still maintain normal functioning of the robot.

3.1.1 Electronical layer

The following diagram shows the electronical layer in detail :

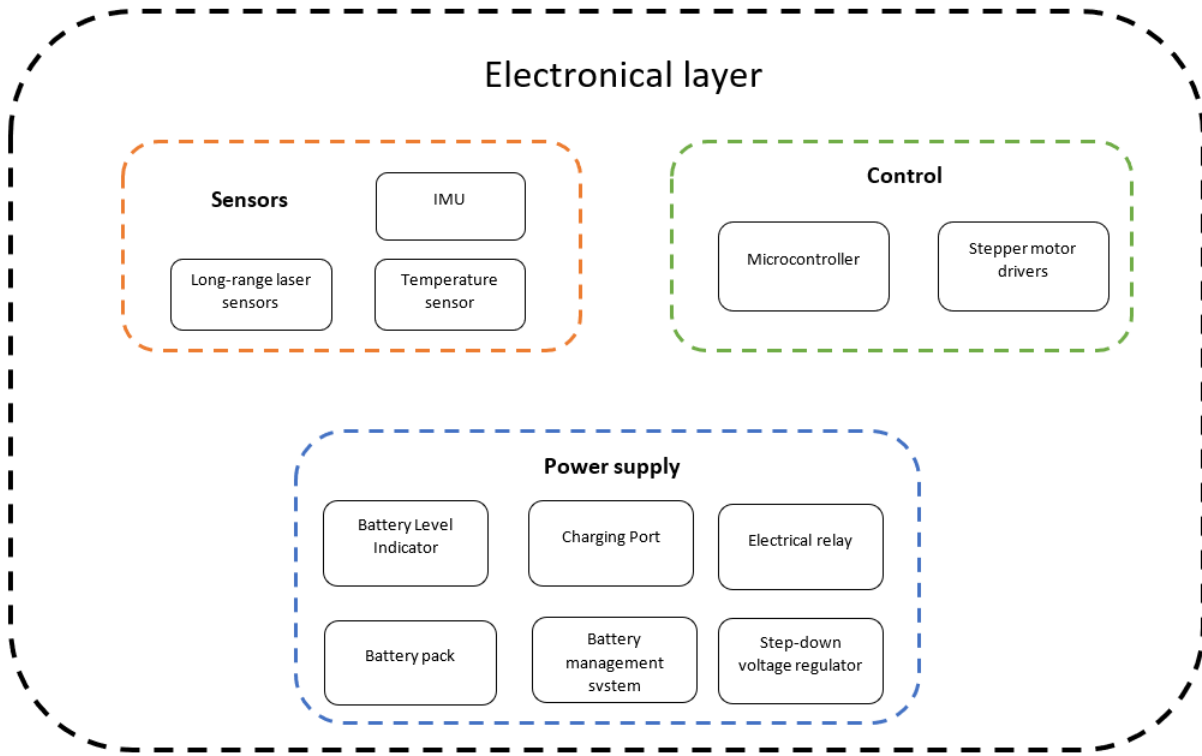


Figure 3.2: Electronical layer

The electronical layer is comprised of three main modules :

- **Power supply :** this module is in charge of providing power to the whole system. The solution is based on a rechargeable battery pack with battery management system and level indicator. Components from the outer layer are either low consumption components (typ. 5V) which are powered through the step down voltage regulator, or high consumption components (typ. 12V) which are directly powered from the battery pack and turned on or off through the electrical relay. This configuration allows for stable and efficient operation power wise.
- **Sensors :** this module provides the general data of the mobile robot. Long-range sensors are used for distance and proximity sensing. The IMU is constituted of a gyroscope, an accelerometer and a magnetometer. The temperature sensor gives ambient temperature readings. These information are not necessarily relevant to the experiment but could provide insight if needed on the external conditions.
- **Control :** In order to control the stepper motors, drivers are needed. The drivers are circuitry capable of commanding stepper motors. The drivers are in turn controlled through a microcontroller.

3.1.2 Mechanical layer

The mechanical layer is made of two main modules :

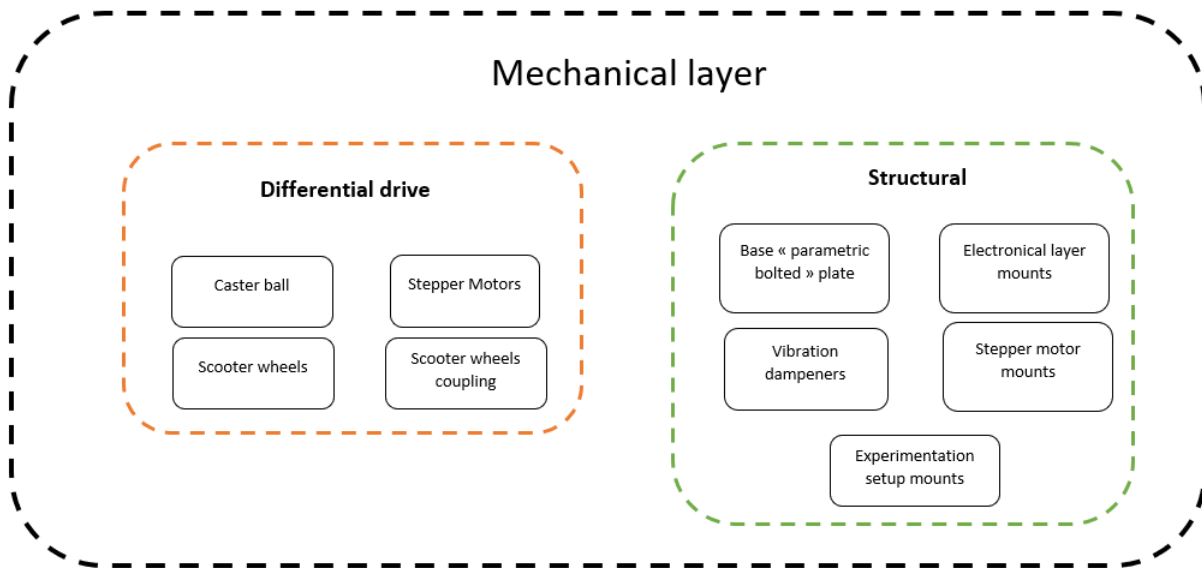


Figure 3.3: Mechanical layer

- **Differential drive :** The robot is a differential wheeled robot. Two scooter wheels coupled to stepper motors drive the robot. A caster ball is used as free rolling ball to provide stability to the system and offload the motors.
- **Structural module :** The mechanical modularity of the robot is represented through the base "parametric bolted" plate which is basically a plate with a fixed number of holes distributed in a parametric way. The holes allow to assemble to the plate various mounts (e.g raspberry pi mount, stepper motor mounts) connecting thus "mechanically" the various components of the robot. Vibration dampeners are also used to reduce stepper motor vibrations.

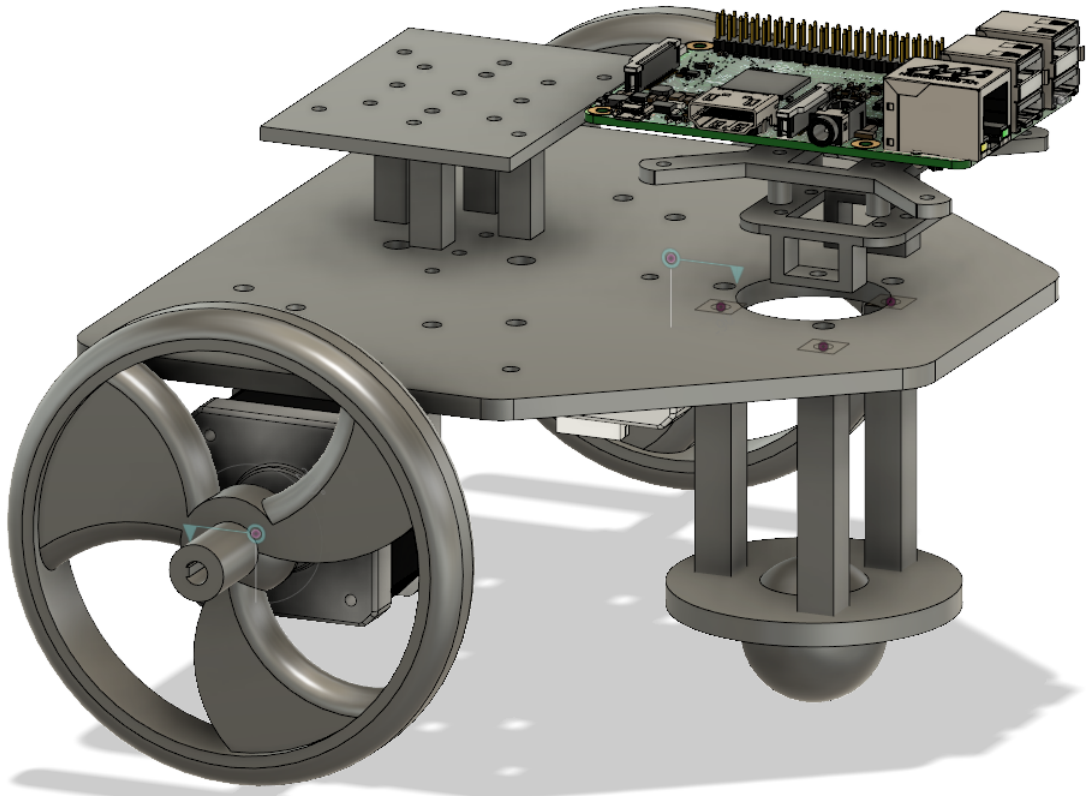


Figure 3.4: Overall view of the mobile robot

3.1.3 Experimentation setup layer examples

The following experimentation layer shows an example of what can currently be interfaced with the robot :

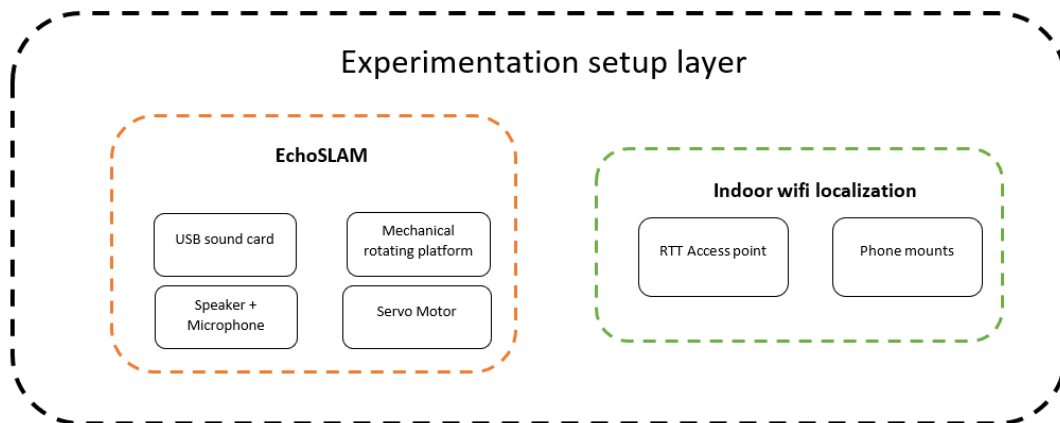


Figure 3.5: Example experimentation setup layer

3.2 Critical choices and justification

3.2.1 Control and computations

	Interfacing (components)	Autonomy	Communication (other systems)	Real-time applications	Justification
Raspberry Pi	++	+	+++	+	<ul style="list-style-type: none"> + Running on Linux and supporting python, the Raspberry PI is a champion in terms of interfacing with other systems. Also, WIFI + Bluetooth. + All popular communication protocols are supported (USB, I2C, UART...) + Can run webservers, can be controlled remotely through SSH. + High power consumption - No analog inputs/outputs , really poor real-time critical hardware signal generation (PWM for driver control is not always in sync)
Microcontroller	+++	+++	+	+++	<ul style="list-style-type: none"> + Analog input/output supported. Most communication protocols supported. Good performance on real-time hardware signal generation. + Low power consumption - Harder to interface with other systems (no linux, no wifi ...)
<p style="text-align: center;">Both to complement each other</p> <p>The raspberry pi is the ideal candidate for building an inter-operable system. With all the IO and communication protocols it supports, one can easily connect additional components. Linux and Python running natively allows to interface with other systems. All these features come at the cost of However, controlling the stepper motor drivers require generating PWM signals with specific timings (real-time constraints). Although capable of generating such signals, the raspberry pi is bad at such low-level functionalities. The microcontroller is there to compensate for this aspect and will be in charge of controlling the motor drivers.</p>					

Figure 3.6: Raspberry Pi and microcontroller comparison

3.2.2 Locomotion

Locomotion model choice is critical for precise trajectory following. Two models were considered, implemented and evaluated.

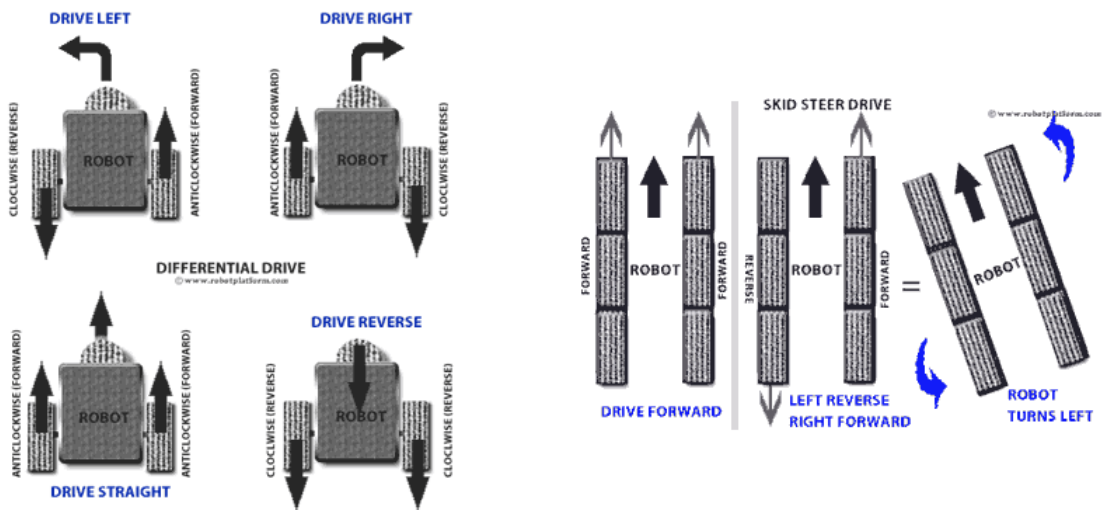


Figure 3.7: Differential drive , skid steer locomotion models [4]

	Load	Resolution	Precision	Ease of control	Justification
Differential Drive	++	+++	+++	++	<ul style="list-style-type: none"> + Resolution and precision of the motor are replicated by the differential drive locomotion - Relies on caster wheel for stability o Only three degrees of freedom (one redundant) , can either move forward, rotate in position, or move in circular motion.
Skid-Steer Drive	+++	++	+	+	<ul style="list-style-type: none"> + Adherence is greatly increased, supported load as well - Loss in resolution and precision when rotating in position due to uncertainty relation to friction, - Vibrations when rotating - Resolution and precision of the motor are not replicated by the skid-steer locomotion, uncertainty of friction impacts resolution and precision
Differential Drive Differential drive locomotion is easy to implement. Rotation resolution and precision is replica of the motor resolution and precision. The caster wheel ensures additional stability. Any trajectory needs to be broken down as combination of linear translation, rotation and circular motion.					

Figure 3.8: Direct drive vs skid steer locomotion

3.2.3 Motor

Motor choice directly impacts resolution and precision. Both stepper motors and DC motors were experimented with.

	Efficiency	Load	Resolution	Precision	Ease of control	Justification
Stepper Motor	+	+++	+++	+++	+	<ul style="list-style-type: none"> + Excellent torque characteristics at low speed + Excellent angular resolution (typ. 1.8°), more resolution and smoother movement can even be achieved using micro stepping. + Discrete motion (steps) + High precision and ability to hold position flawlessly + Reliable internal feedback (if not skipping steps) - Noisy - Important vibrations at low speeds - Control can be hard, control signals timings are important. - Inefficient
DC motor	+++	++	N/A	+	+++	<ul style="list-style-type: none"> + Efficient + Easy to control - Require external feedback - Continuous motion only => low precision
Stepper motors Trajectory following with high resolution and precision is at the core of the mobile robot. As we operate the robot at rather low speeds, stepper motors achieve superior resolution, precision and torque characteristics at the cost of high current consumption.						

Figure 3.9: Stepper motor vs DC motor

3.3 Differential drive motion inverse kinematics

The following diagram shows a differential wheeled robot with its main parameters : **Wheel_Radius** and **Axis_Radius**

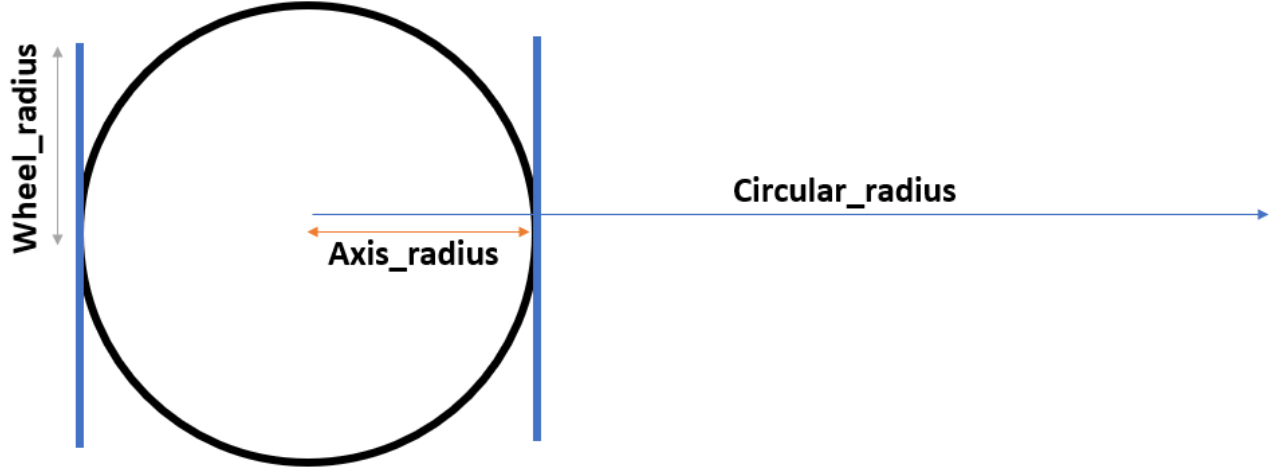


Figure 3.10: Differential wheeled robot diagram

Using a standard stepper motor with : **n steps/rev** and direct coupling we can control the number of steps for : s_l and s_r for left and right wheel respectively. In order to control the motion

- **Number of steps per mm (constant) s :**
$$s = \frac{n}{2\pi * Wheel_radius}$$
- **For a desired translation in mm t :**
$$s_l = s_r = n * t$$
- **For a desired angular rotation in deg θ :** $s_l = -s_r = n * 2\pi * axis_radius * \frac{\theta}{360}$ and $s_l > 0$ for clockwise rotation
- **For a desired circular motion of given radius $Circular_radius$ and given travelled distance t :**

$$s_l = \frac{t}{n} * (1 + \frac{Axis_radius}{Circular_radius})$$

$$s_r = \frac{2t}{n} - s_l$$

4 | Accuracy and precision evaluation

4.1 Accuracy and precision of stepper motors

The choice of stepper motors benefits both resolution and accuracy. The working principle behind stepper motors is illustrated in the following figure

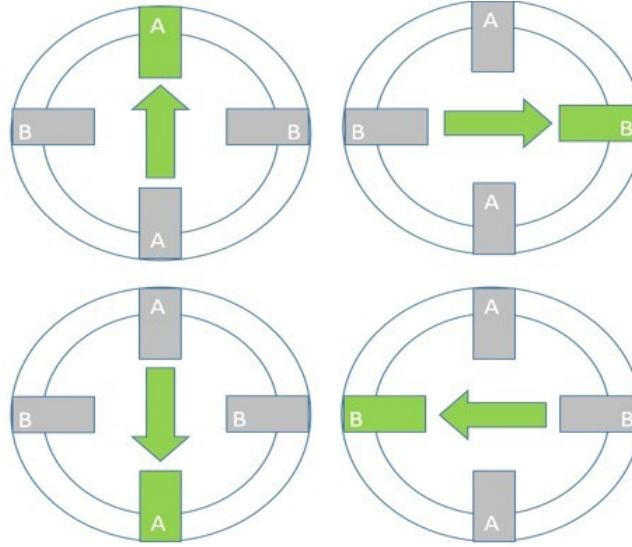


Figure 4.1: Stepper Motor 1-Phase on full rotation [1]

By energizing a coil (or couple of coils), a magnetic field of known direction is generated. The rotor (green arrow in the picture) aligns itself with the generated magnetic field. And thus in discrete steps, we can rotate the rotor by continuously energizing the following coil in sequence. A position can be held accurately by stopping the sequence at any given time and keeping the corresponding coil energized. The currently used stepper motor is a NEMA17 with 200 steps per revolution (corresp. 1.8° resolution). Therefore, in theory, **inaccuracy is only due to the discrete nature of movement.**

In order to increase the resolution and achieve smoother movement, microstepping is used. Microstepping is illustrated in the following figure :

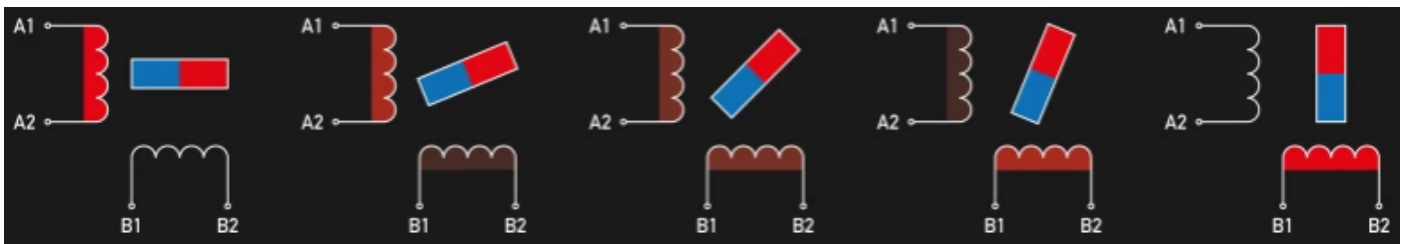


Figure 4.2: Symbolic example of quarter-stepping in a bipolar stepper motor [3]

The resolution is limited by the number of steps per revolution the motor has which is defined in turn by the number of discrete magnetic field directions the rotor is able to align itself with. By energizing several coils with different voltages at the same time we are able to create additional intermediate stable magnetic field directions thus increasing the resolution. This comes at the cost of effective torque which decreases as the intermediate magnetic field will never be as strong as the natural one (aligned with coil). If the power to the steppers is cut, additional inaccuracy can be introduced because of microstepping. The robot is more prone to lose the position of the rotor in these weaker intermediate steps.

4.2 Systematic evaluations

In order to assess the accuracy and precision of the robot, the various type of motions are evaluated. The observed motion under several runs of translation, rotation and circular motion is compared to the given command. Since we are using a **1/16 microstepping factor**, a stepper with **200 steps/rev** (1.8° per step) and a scooter wheel of **nominal diameter 100 mm** we have a resolution $\Delta d = \pi * 100 * \frac{1.8}{360} * \frac{1}{16} = 0.1[mm]$. As such, and due to the stepper motor technology, accuracy and precision are not expected to change over different distances travelled by the robot.

4.2.1 Translation motion

To evaluate translation, the robot is tasked to move forward by 500 mm and backwards by 500 mm for 20 runs, the results are as follows :

<i>Forward 500mm</i>					
50.2	50.5	50.4	51	50.2	
50.3	50.1	50.5	50.4	50.1	
50.4	50.6	50.5	50.2	50.2	
50	50	50.1	50	50.2	
<i>Mean</i>	50.295				
<i>STD</i>	0.248				
<i>Backward 500mm</i>					
	50.1	50.5	50	50	51
	50.4	50.4	50.4	50.4	50
	50	50.4	50.2	50.4	50.2
	51	50	50.4	50.1	50.2
<i>Mean</i>	50.305				
<i>STD</i>	0.294				

Figure 4.3: Evaluation of robot's performance for linear translation

We can observe really low STD and good average. The system is precise and accurate for linear translations. The obtained error could stem from the measuring technique. The evaluation of translation is critical as it is directly linked to the distance travelled by each wheel. The differential drive locomotion model has free rotation (no skid) therefore the distance travelled by each wheel is deterministic for good characterization of angle and rotation center.

4.2.2 Rotation and circular motion

Several runs have been made for rotation and circular motion of radius R . These runs have shown impeccable precision for angular rotation (given correct calibration). Rotation center shifting has proven itself hard to evaluate due to the fact that the shift is so little $< 1[\text{mm}]$ that any actual measurement would rather show inaccuracy in measurement technique over inaccuracy of motion.

5 | Conclusion

Globally, the project has been really challenging, requiring several iterations before reaching satisfying results. The robot stayed true to its original purpose : a modular DIY mobile platform for running experiments. Some key points justifying the robot meeting the specifications set :

- By running 3d printing firmware and using a microcontroller to control the stepper drivers, synchronization between both wheels is ensured.
- The robot benefits largely from the precision and accuracy stepper motors have to offer.
- The several holes on the mounting plate are more than enough to assemble any additional mounts and components.
- The raspberry pi hosts a user friendly interface for easy control and configuration of the robot.

Some points could however still be improved :

- The robot could benefit from higher grade and capacity batteries than the current 18650 li-ion ones.
- The scooter wheels start to show some wear, potentially leading to loss in resolution and accuracy.
- The caster ball shows wear as well, need for cleaning and lubricating to ensure optimal functioning.
- The differential drive locomotion model is not holonomic and constrains the user to break down parametric trajectories to linear translations, in position rotations and radius based turns.

6 | Annex

6.1 Bill of Materials






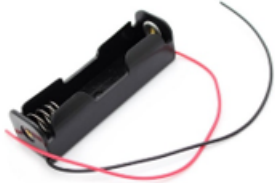
Nomination	Thumbnail	Qty	Type	Description
Raspberry Pi 3 B+		1	Electronics	Main computational unit of the robot
16 GB SD Card		1	Electronics	Needed storage for Raspberry Pi system
Arduino Nano		1	Electronics	Microcontroller for stepper driver control
DRV8825		2	Electronics	Stepper motor driver
18650 Lithium Ion Battery		3	Electronics	12V Battery pack
18650 battery holder		3	Accessory	Battery holder

Figure 6.1: BOM 1/3







Nomination	Thumbnail	Qty	Type	Description
3S Battery management system		1	Electronics	Battery management system circuit
XL4015 Step down buck converter		1	Electronics	Step down converter to drop the 12V from PSU to steady 5V for raspberry pi and Arduino <u>nano</u>
DC 5.5mm barrel jack		1	Accessory	Charging port
3S 12.6V Lithium Battery Capacity Indicator Module		1	Electronics	Battery display indicator
GY-VL53L0XV2 TOF laser sensor		2	Electronics	Distance laser sensor
Caster Ball (24mm inner ball diameter)		1	Mechanical	Caster wall for third support

Figure 6.2: BOM 2/3

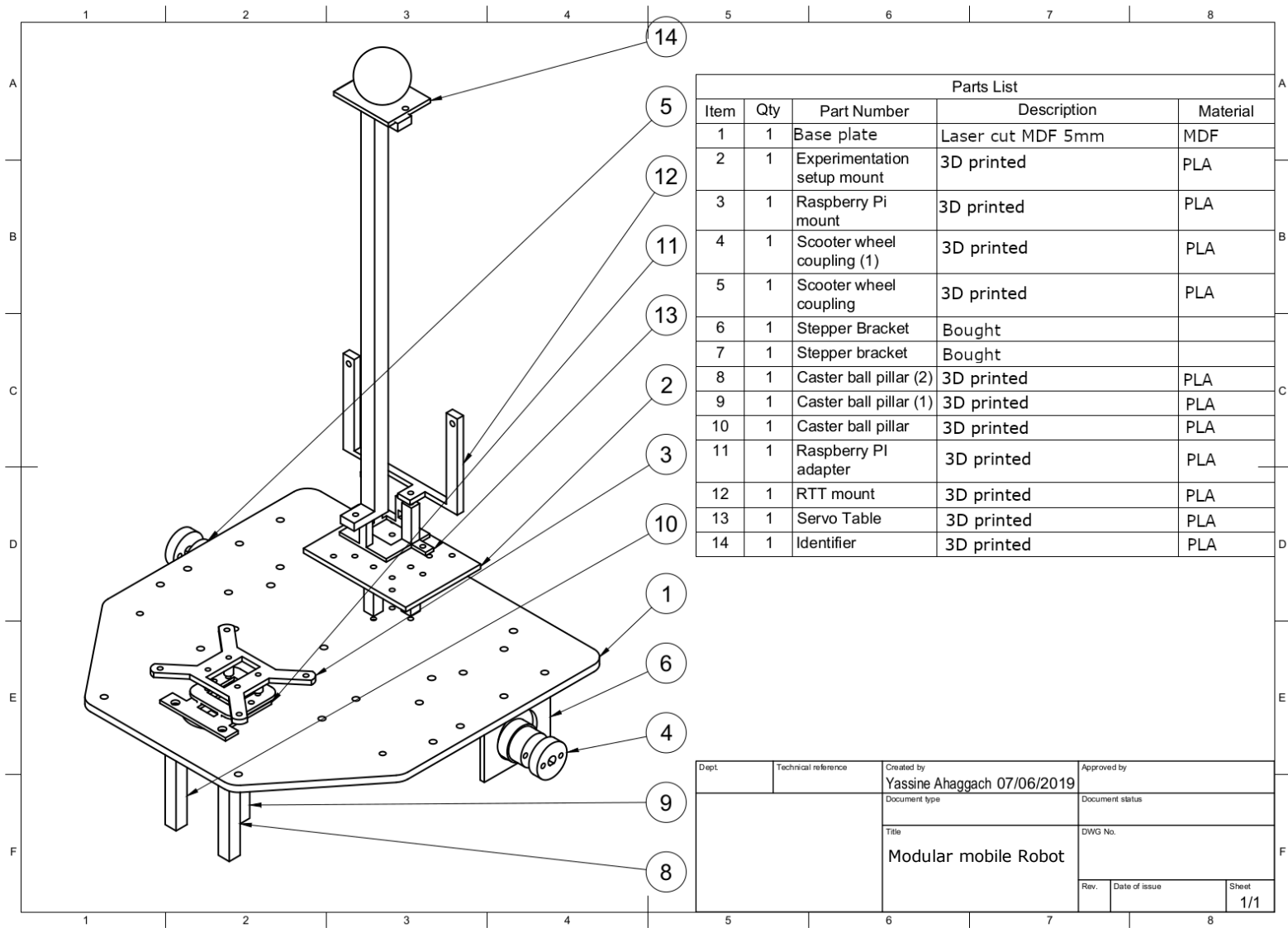
Nomination	Thumbnail	Qty	Type	Description
Scooter Wheels (100mm diameter)		2	Mechanical	Scooter Wheels as robot's wheels
NEMA17 1.8° 1.2A stepper motor		2	Mechanical	Stepper motor
Nema17 Stepper Motor Right Angle bracket		2	Mechanical	Right Angle bracket for stepper motors

Figure 6.3: BOM 3/3

6.2 Designs and Assembly

Nomination	Qty	Use
M2 nuts	5	Used to fasten raspberry pi and battery pack to base
M3 nuts	30	Used to fasten most experiment mounts to the base
M4 nuts	20	Used to fasten some of the experiment mounts to the base
M4 safety nuts	8	Used to fasten the motor brackets to the base
M2 x 18 mm screw	1	Used to assemble battery pack to base
M2 x 12 mm screw	4	Used to assemble raspberry pi to mount
M3 x 12 mm screw	20	Used to assemble most mounts to the base
M3 x 8 mm screw	10	Used to assemble some mounts to the base
M4 x 18 mm screw	8	Used to assemble the motor brackets to the base
M4 x 12 mm screw	20	Used to assemble some of the mounts to the base
M4 washers	16	Used with the safety nuts to reduce strain on base at the motor brackets

Figure 6.4: List of needed screws and nuts



6.3 Software

The raspberry pi automatically connects to any wifi with the following characteristics :

SSID : Robot

Password : Robot0101

The user needs to make sure before turning on the robot that such wifi exists, mobile hotspots work as well. Once connected, and the local IP address of the raspberry pi fetched, the user can SSH with the following credentials :

Username : pi

Password : raspberry

A zip file containing all material relative to an experiment (EchoSlam) is generated at the /home/ folder of the raspberry pi. To access it and download it, use FileZilla and initiate a connection as follows :


IP : *RaspberryPILocalAddress*

Port : 22

In case of using a soundcard, the volumes can be adjusted by running **alsamixer** on the terminal, Pressing F6 to choose the soundcard then F5 to show all channels (in and out).

6.3.1 GRBL and Octoprint

The Arduino nano is running GRBL, a 3d printing firmware. In order to achieve ease of use and the best precision possible, this firmware has been proven far superior to custom-made solutions. The pinout of the arduino nano is as follows :

Arduino Nano V 3.0 GRBL Pinout			Pin diagram for Grbl v0.8 and v0.9		
			ATmega 328P		
Pinout Ref					Pinout Ref
D13	Spindle Direction	D13	D12	Spindle Enable	D12
3V3	Not Used	3V3	D11	Limit Z-Axis	D11
VREF	Not Used	VREF	D10	Limit Y-Axis	D10
A0	Reset/ Abort	A0	D9	Limit X-Axis	D9
A1	Feed Hold	A1	D8	Stepper Enable/Disable	D8
A2	Cycle Star/ Resume	A2	D7	Direction Z Axis	D7
A3	Coolant Enable	A3	D6	Direction Y Axis	D6
A4	(Not Used/ Reserve)	A4	D5	Direction X Axis	D5
A5	Probe	A5	D4	Step Pulse Z Axis	D4
A6	Not Used	A6	D3	Step Pulse Y Axis	D3
A7	Not Used	A7	D2	Step Pulse X Axis	D2
		5V	GND		
		RST	RST		
		GND	RX1		
		VIN	TX1		

With the traditional layout: (NOTE: The probe A5 pin is only available in Grbl v0.9.)

Figure 6.5: Arduino nano GRBL pinout

The firmware believes the left wheel is the **X axis** and the right wheel is the **Y axis** and thus pins : D8,D6,D5,D2,D3 are connected to the corresponding pins on each of the stepper drivers. The following picture illustrates the pinout for each stepper driver :

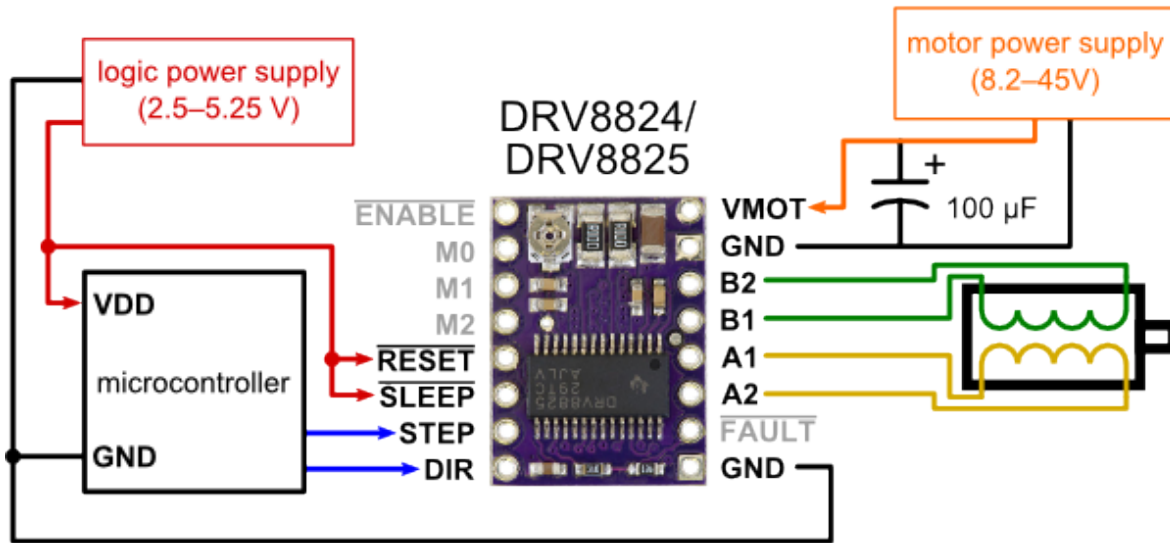


Figure 6.6: DRV8825 stepper driver pinout[6]

The raspberry pi pinout is as follows :

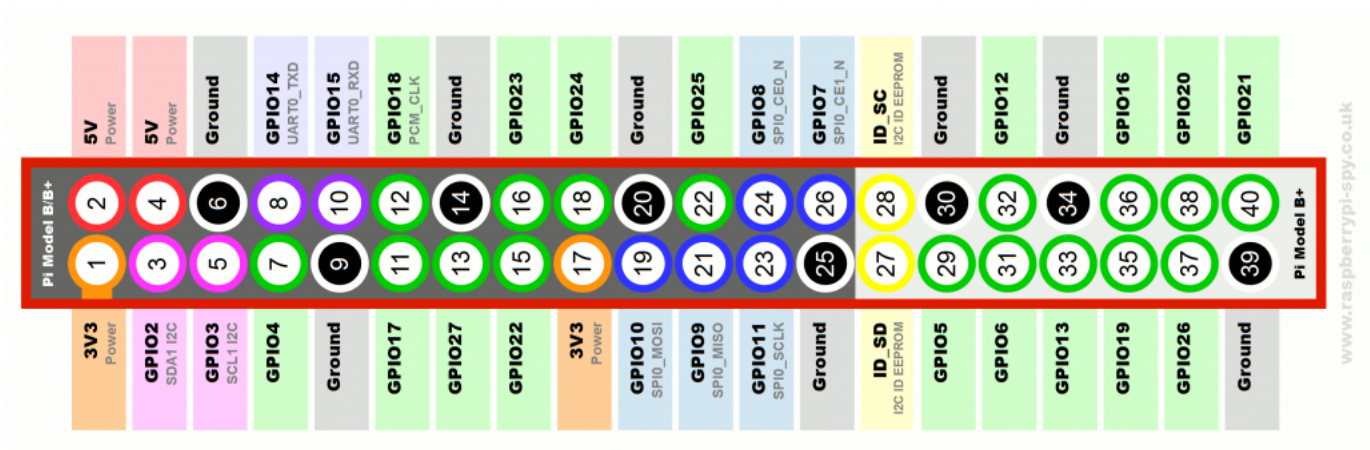


Figure 6.7: Raspberry Pi pinout[5]

The important pins to keep in mind :

- I2c GPIOs : used for communication with the laser sensors.
- GPIOs 26 and 21 : connected to the XSHUNT pins of laser sensors left and right respectively.
- GPIO 20 : used for servo motor PWM signal generation.
- GPIO 16 : used for external relay control.

The arduino Nano is connected to the raspberry pi via USB. Once the SSH connection established the user enters the following command in the terminal

`/OctoPrint/venv/bin/octoprint serve`

The command starts the Octoprint webserver and it can now be accessed from any computer or phone at the following address : `https://*RaspberryPILocalAddress*:5000`

Now that the server is started, one needs to configure some parameters relative to the robot :

By opening the the Octoprint webpage we see :

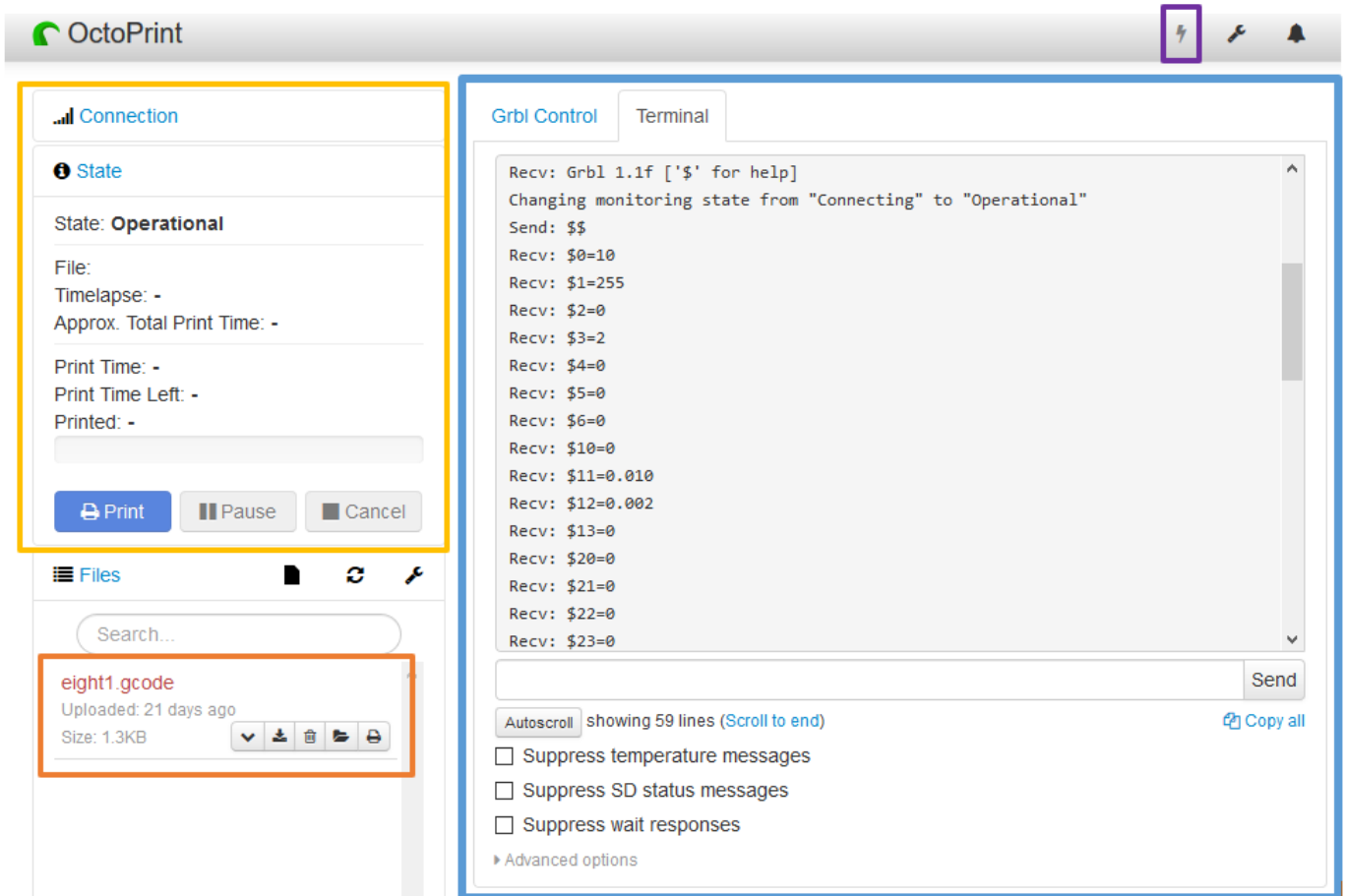


Figure 6.8: Octoprint web interface

- In **Yellow** : the general state of the robot, Print corresponds to starting the movement. Pause and Cancel buttons work as intended
- In **Blue** : the communication terminal. Allows to set some important parameters
- In **Red** : the uploaded gcode files (corresponds to the trajectories)
- In **Purple** : the button to command the electrical relay for the experimentation setup layer

Before controlling the robot, some important general parameters need to be set. By sending `$$` to the microcontroller through the communication terminal, the value of global variables is displayed (max speed, acceleration, steps per mm). To change a parameter, one needs to send `$(ID)=VALUE`, with `ID` being the parameter's ID and `VALUE` the desired value. From the full list of parameters^[2], one needs to know :

- ***ID*=1** corresponds to the delay in milliseconds after a movement before the motors are turned off. Setting this value to 255 ensures the motors are always on (highest precision)
- ***ID*=100 and *ID*=101** correspond to the number of steps per mm for the X axis and Y axis respectively. Refer to the inverse kinematic to compute this parameter. This value can be **calibrated** by commanding a set distance and observing the travelled distance. The new number of steps s' can be computed with the following formula : $s' = s * (\delta x + 1)$ with s being the old number of steps per mm and δx the relative error on the travelled distance.
- ***ID*=110 and *ID*=111** correspond to the max speed in mm/min for the X axis and Y axis (right and left wheels)
- ***ID*=120 and *ID*=121** correspond to the acceleration rate in mm/s^2 for the X axis and Y axis (right and left wheels)

In order to move the robot, Gcode needs to be executed. Gcode is a simple text file containing successive movement commands interpreted by the micro controller. Two main commands to know :

- **G91 X*DIST1* Y*DIST2*** commands the right wheel to relatively move by *DIST1* [mm] and left wheel to move by *DIST2* [mm]. Refer to the inverse kinematics to understand what distances are needed for what type of motion
- **G92 X0 Y0** resets the internal coordinates in terms of travelled X and Y distances. This command is only necessary if one uses G90 which is absolute positioning and not G91 relative positioning

Once the gcode file ready and saved as for instance "trajectory.gcode", one can simply drag and drop the file to the Octoprint interface and press print to start the trajectory following.

- For the indoor wifi localization experiment, do not forget to turn on the RTT with the corresponding switch 6.3.1
- For the echoSLAM experiment, the Gcode file is slightly modified to include a custom command. Upon interpreting the custom command, the microcontroller stops the motors and the raspberry pi runs the experiment before allowing the microcontroller to resume the trajectory following.

A script to generate g-code files from x,y cartesian coordinates is provided. The script differentiates between both experiments and adds or not the custom instruction for the echoSLAM experiment.

6.3.2 Custom scripts

With Octoprint being the new interface, only one script is needed : "gcode.py". This script parses and translates a text file into gcode. The text file is a succession of lines with the following format **i,angle,dist,x,y**. With i the index of the point (or measurement) , $angle$ the angle of in position rotation, $dist$ the distance to travel, x the distance to be travelled by the left wheel, y the distance to be travelled by the left wheel.

- For the indoor wifi localization experiment, simply put 0 on the angle and dist values and input the x and y distances travelled by both wheels directly.
- For the echoSLAM experiment, simply ignore the last two elements of the formatted line

To run the script simply call `python3 gcode.py *EXP_ID* *NUM_ORIEN* *MIN_FREQ* *MAX_FREQ* *DURATION*`

- `*EXP_ID*` being the experimentation ID , 0 for the EchoSlam ,1 for the indoor wifi localization
- `*NUM_ORIEN*`, `*MIN_FREQ*`, `*MAX_FREQ*`, `*DURATION*` concerns only the EchoSlam correspond respectively to the number of orientations of the speaker on each sound experiment, the minimum frequency of the sweep, the max frequency of the sweep and duration of the sweep.
- Ignore all arguments that do not concern the current experiment.

7 | Bibliography

- [1] RS Components. Stepper motors and drives, what is full step, half step and microstepping? <https://www.rs-online.com/designspark/stepper-motors-and-drives-what-is-full-step-half-step-and-microstepping>, Consulted on the 7th June 2019.
- [2] GNEA. Grbl v1.1 configuration. <https://github.com/gnea/grbl/wiki/Grbl-v1.1-Configuration>, Consulted on the 7th June 2019.
- [3] Hackaday. How accurate is microstepping really? <https://hackaday.com/2016/08/29/how-accurate-is-microstepping-really/>, Consulted on the 7th June 2019.
- [4] Robot Platform. Wheel control theory. http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html, Consulted on the 7th June 2019.
- [5] Polulu. Drv8825 stepper motor driver carrier, high current. <https://www.pololu.com/product/2133>, Consulted on the 7th June 2019.
- [6] Raspberry Spy. Simple guide to the raspberry pi gpio header. <https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/>, Consulted on the 7th June 2019.