Picomouse team project

KNABENHANS Felix, AUMONT Adrien, KACEM Sophia, PEREGO Zacharie,
MORAGA Paco, HUSTAVA Adam

November 5, 2024

1 General overview

The objective of this project is to design an autonomous vehicle (the "mouse") that first explores a labyrinth to find a solution, and then uses that solution to navigate the maze as quickly as possible without crashing. The process involves two phases: exploration and optimized maze-solving.

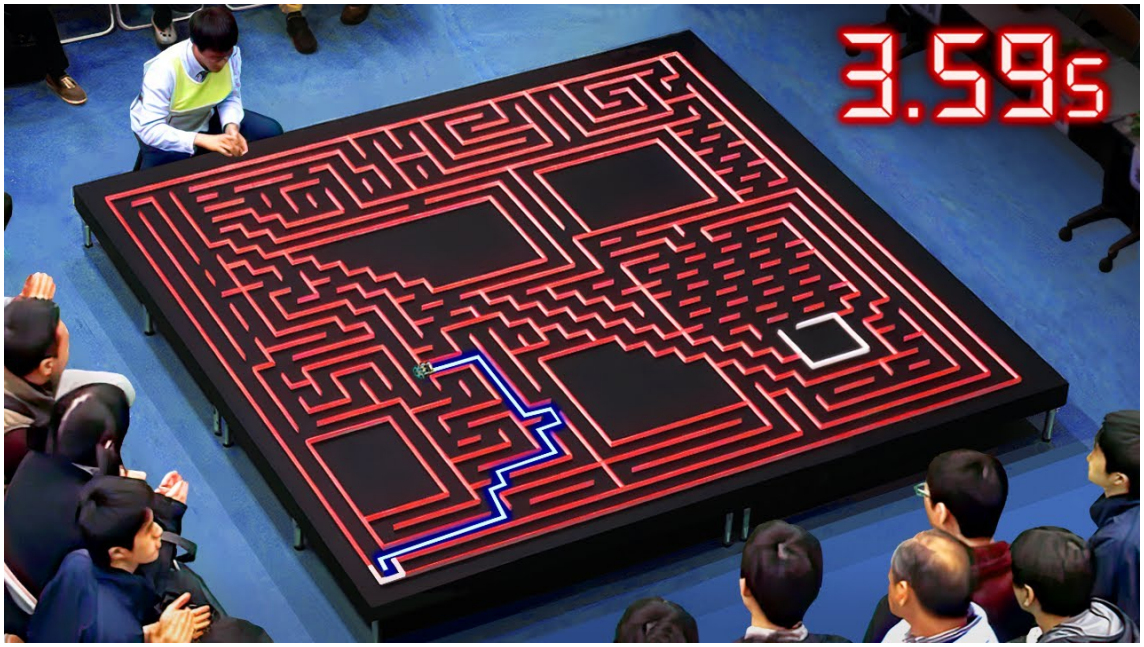


Figure 1: "Veritasium" Youtube miniature representing the japanese IEEE micromouse contest

2 Hardware side

The primary hardware challenges involve situating the mouse within the maze, given that the solution is always located at the center in competition settings. This can be achieved using time of flight sensors to determine the distance of the mouse from the walls. We plan to have 2 different TOF sensors. 1 for long-distance sensing (up to 2 meters) and 4 others for short distance sensing (down to 5mm) which will be used to ensure that the mouse is centered on the path and that it stops at the correct distance from the wall before turns. The 4 sensors would be placed to the side of the mouse to be able to assess it's surrounding and the long distance sensor would be placed straight ahead to help us with a general sense of position 2.

Another challenge is ensuring the mouse makes precise turns. we plan to use gyroscopes to achieve the highest possible degree of angular accuracy. To optimize the mouse's path, an advanced (optional) feature could be to allow turns other than the standard 90 degrees. This would improve efficiency, especially in more complex mazes, and is one of the key technological advancements in the official IEEE Micromouse competition.

Seeing how important the gyro is to our project, we looked for the best fitting gyroscope we could find, the one we chose has minimal drift issues (integrated filters) which will streamline

the coding process for us reducing our workload. And it has a very good refresh rate which will insure precision in the mouse's movements especially if we want to make continuous turns without stopping the mouse at every corner.

Building the mouse is another challenge, fortunately we don't need to design it from scratch there exists numerous designs online which we used as inspiration.

2.1 Classic Micromouse (Maze-Solving)

- Maze Dimensions: 16x16 grid, each cell measuring 18 cm x 18 cm. (approx $9m^2$)
- Path Width: 16 cm between the walls.
- Wall Height: 5 cm.

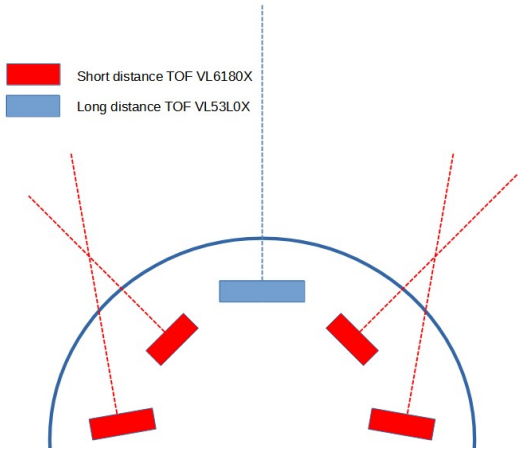


Figure 2: Captors layout

3 Software side

The primary software challenge is fitting everything into the microcontroller, as the mouse must operate autonomously. We plan to use established maze-solving algorithms, such as the Flood Fill algorithm, during the exploration phase. This algorithm allows the mouse to follow an optimistic path toward the center and updates the path as it encounters obstacles. Once the mouse reaches the center, it can refine its path and store it as the solution. Although this approach may not guarantee the shortest path, we can run the algorithm in reverse, from the center to the starting point, to increase the likelihood of finding an optimal path. Another software challenge lies in processing data from the TOF sensors and gyroscopes to ensure accurate turns and avoid frequent collisions. This aspect is less clear at the moment and will likely be the primary focus of troubleshooting during the project. We want to use the Bluetooth capacity of our ESP-32 to help us debug and log the mouse's actions. The idea would be to have the mouse send it current position in the maze, the sensor data and the algorithm state. Optionally we could make this data apparent in real time with a nice UI. We decided to take an ESP-32 with 2 threads, one that will operate the mouse and polling the captors and the other to send the data to the external computer over Bluetooth.

4 Mouse position

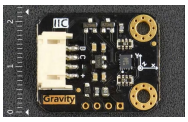
Coordinate Estimation: The mouse uses a simple coordinate system relative to its starting position in the maze.

Updating Coordinates with Forward Movement: As the mouse moves forward along a straight path, it updates its coordinates using the distance traveled calculated by the front-facing ToF sensor. Updating Coordinates at Turns: Upon detecting a turn, the mouse can update its orientation (e.g., by rotating its internal "heading" by 90°) and reset its front-facing distance. If the mouse turns left, it will update its heading to the new direction and continue moving forward, now calculating distance in this new direction.

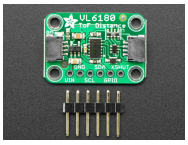
We will consider 2 position metrics, the first one, absolute position, denotes the position and the orientation of the mouse in a 16x16 cell, and the second position (relative position) that shows in which cell the mouse is.

5 Components

5.1 Sensors



(a) DFRobot Gravity I2C BMI160



(b) VL6180X short distance TOF

(c) VL53L0X long distance TOF

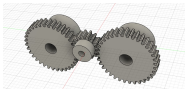
5.1.1 Explanations

1. *Gyroscope*: We didn't have a lot of choices, because we need to buy a chip mounted on a PCB. So we were left with only two models: MPU 6050 and MBI160. They have the same specifications except that BMI160 uses less energy, has integrated filters and has a higher sample rate (1600 vs 1000 Hz) making it an obvious choice.

2. *Long distance & short distance (TOF) sensors*: We have made the choice of Time of flight sensors because it minimizes interferences with the environment (lighting could be an issue) in contrario with the infrared one. The idea behind the choice of using short (5mm - 20cm) **and** long (5cm - 1.2m) distance sensors is to use the long distance one (1) for the front sensor to detect walls in front of the mouse and the four (4) short distance ones for the sides of the mouse (see 2 for more details).

5.2 Motor

(a) 16DCT Athlonix Brush 6VDC



(b) Gears (CAD)

(c) Motor controller MX1508

5.2.1 Explanations

1. *Motors*: We saw that our first choice of motors, the N20 12VDC 900RPM, was a bad decision. We believed that increasing RPMs that were already reduced was lost of energy for the motors, so we checked for another one. We got interested in motors of 10'000 rpm, because it is a common spec in a lot of successful MicroMouse projects. We then calculated the torque we would need to have a maximum speed of 4.5 m/s and an acceleration of 6 m/s^2 (see 6). We had to find a motor small enough to fit on the mouse with a voltage of 6V (not 9). It allows us to take a 7.4 V battery. The only one fitting our needs we found is the 16DCT Athlonix 10000 RPM 3.5W 6VDC (4a). It's a bit expensive, but we didn't find another one with the required specification in Switzerland. After a discussion with the TAs we ended up agreeing upon using N20 motors at first to try them out and only using the expensive Athlonix if the N20 were insufficient. This would also simplify breaking immensely and give us the option to use the positional encoders.

2. *Gears*: The choice of gears depends of the gear ratio. To compute it we needed to find an adequate number of teeth. See Figure 6 for the detailed computation.

3. *Motor controller*: We considered two possible types of H-Bridge motor controllers, either Bipolar or MOSFETs. Bipolar H-Bridges dissipate too much of the energy and need a heat sink and MOSFETs only dissipate approximatively 0.1V. So we finally chose the MOSFETs

Dual H-Bridge because we run on battery and don't want any loss. A dual H-Bridge controls 2 motors at the same time with a single chip so it suits our needs better than using 2 single H-Bridges.

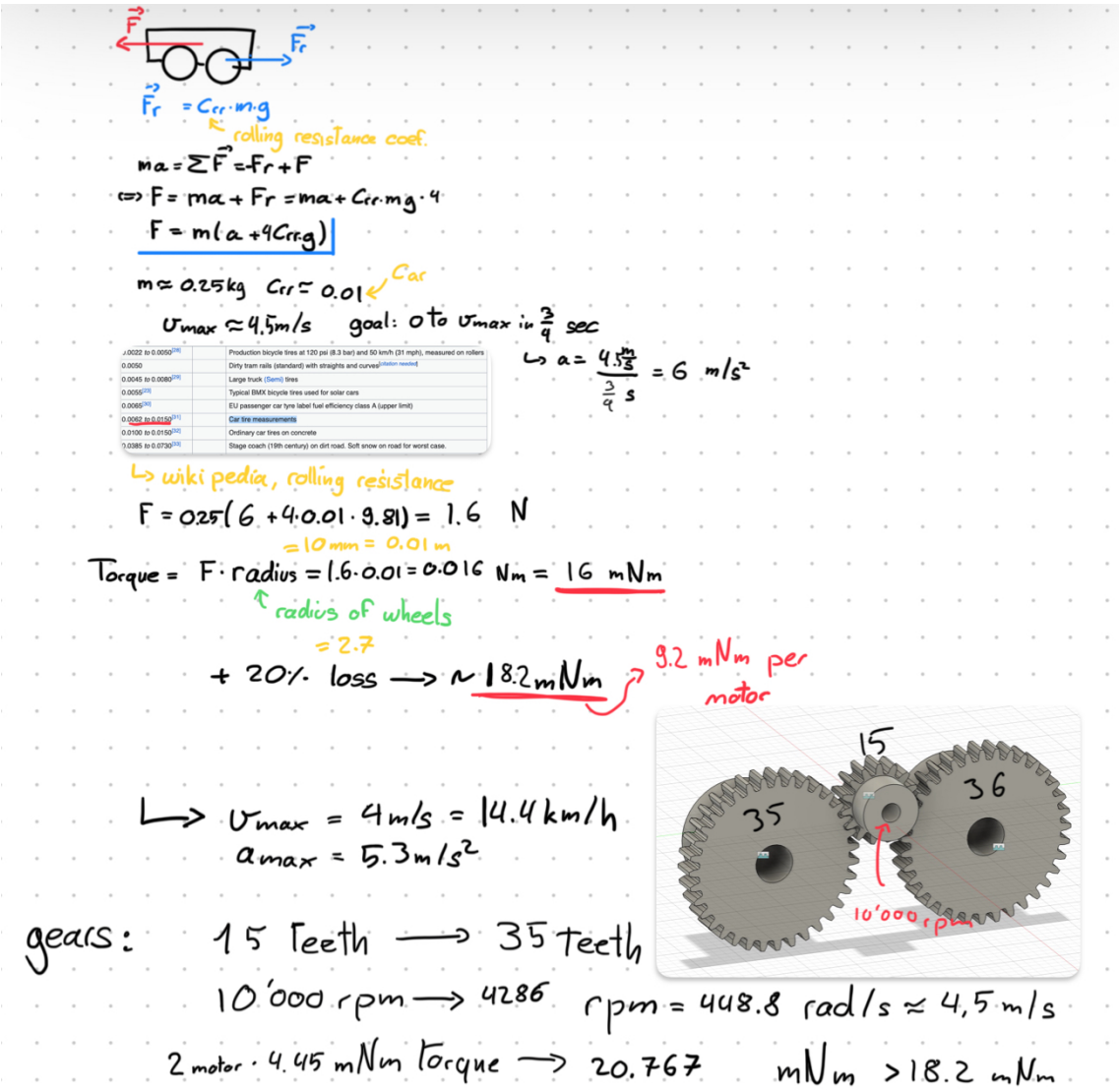


Figure 5: Calculations for Athlonix motor

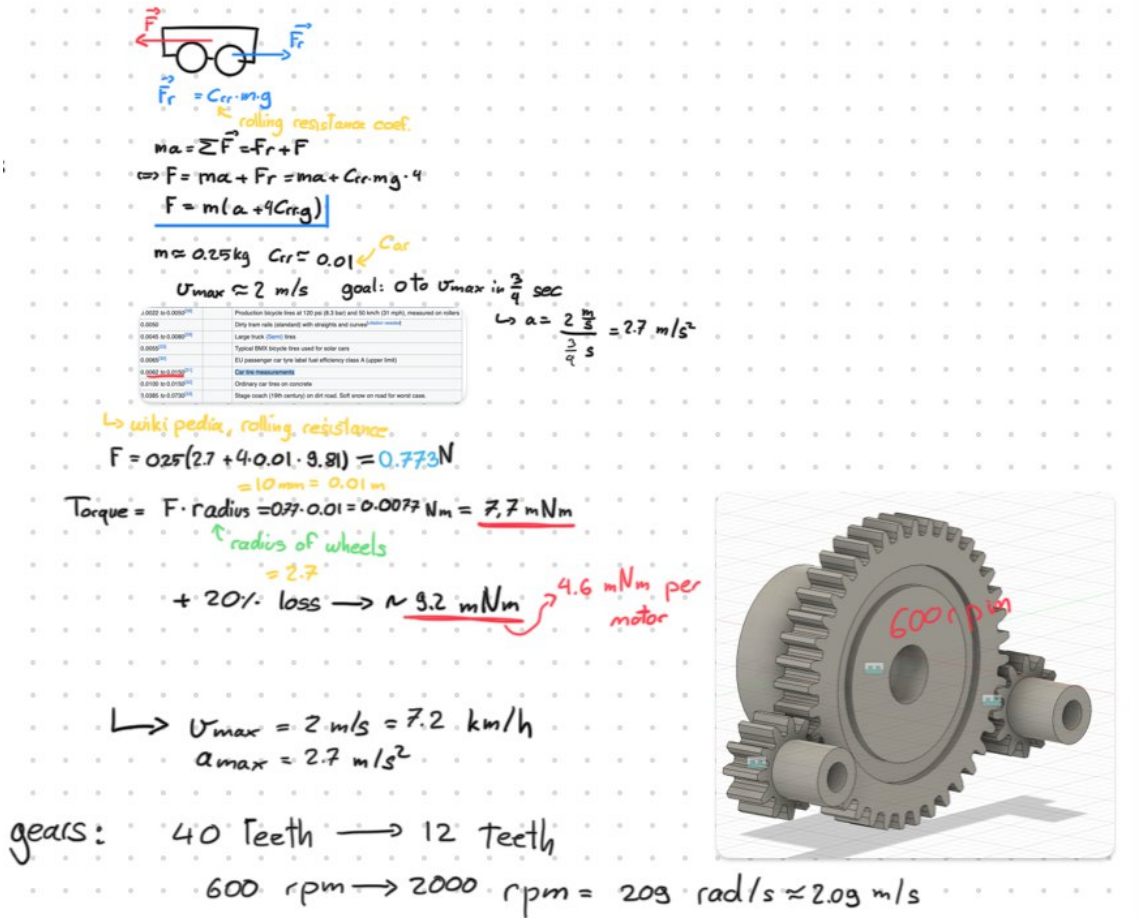


Figure 6: Calculations for N20 6V 600rpm motor

5.3 Mouse

(a) ESP32-E (N16R2)

(b) LIPO 2s 7.4V

(c) LIPO 2s 7.4V

5.3.1 Explanations

1. *Microcontroller*: We chose the ESP32-E (7a), because we need multiple threads for the debug and the algorithm, and an emitter/receiver for the Bluetooth. The microcontrollers proposed in the handbook didn't fit our requirements.

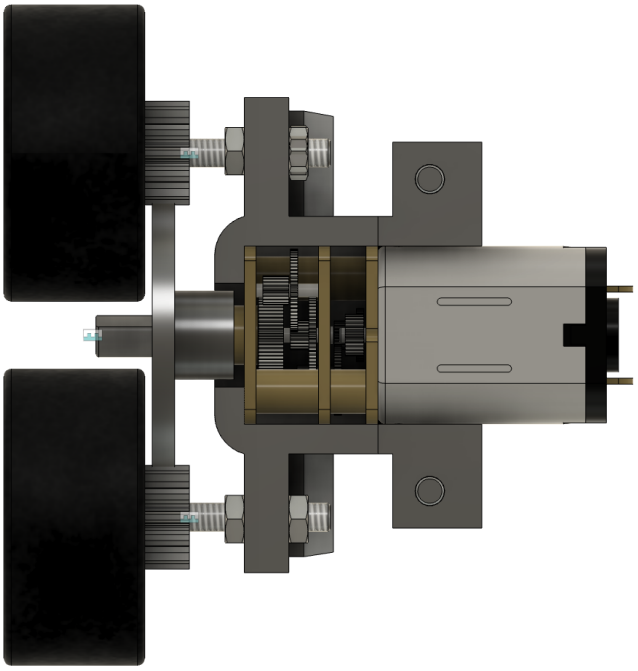
2. *Battery*: We needed a battery that could supply both motors (up to 6V). So a 2S battery with its 7.4V was sufficient. The autonomy was important too, but for the majority of the MicroMouse projects seen on Github and on the internet, the mAh was between 120 and 500 and the battery shown on 7c has 350 mAh. The number of C doesn't need to be too high, so 25C is sufficient. The last requirement is the mass of the battery, that needs to be as light as possible, this one (7c) weights 33gr. We took a second battery (see 7b) for testing, that is a little bit bigger and has 1000 mAh, but has roughly the same specifications.

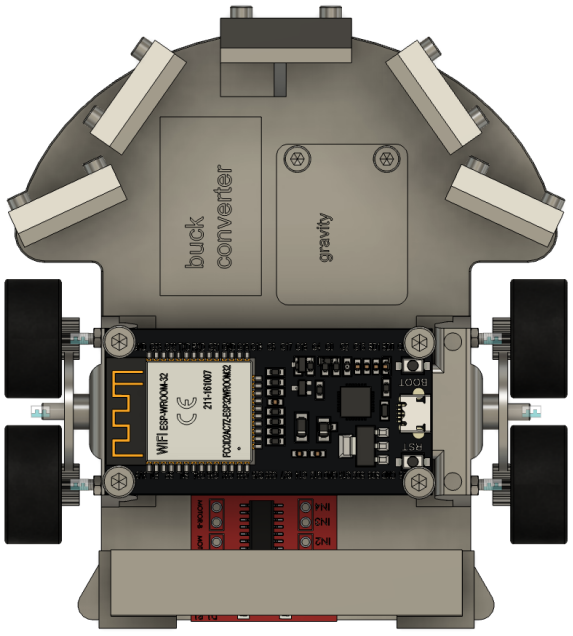
Disclaimer : This table doesn't reflect the feedback we got during the break

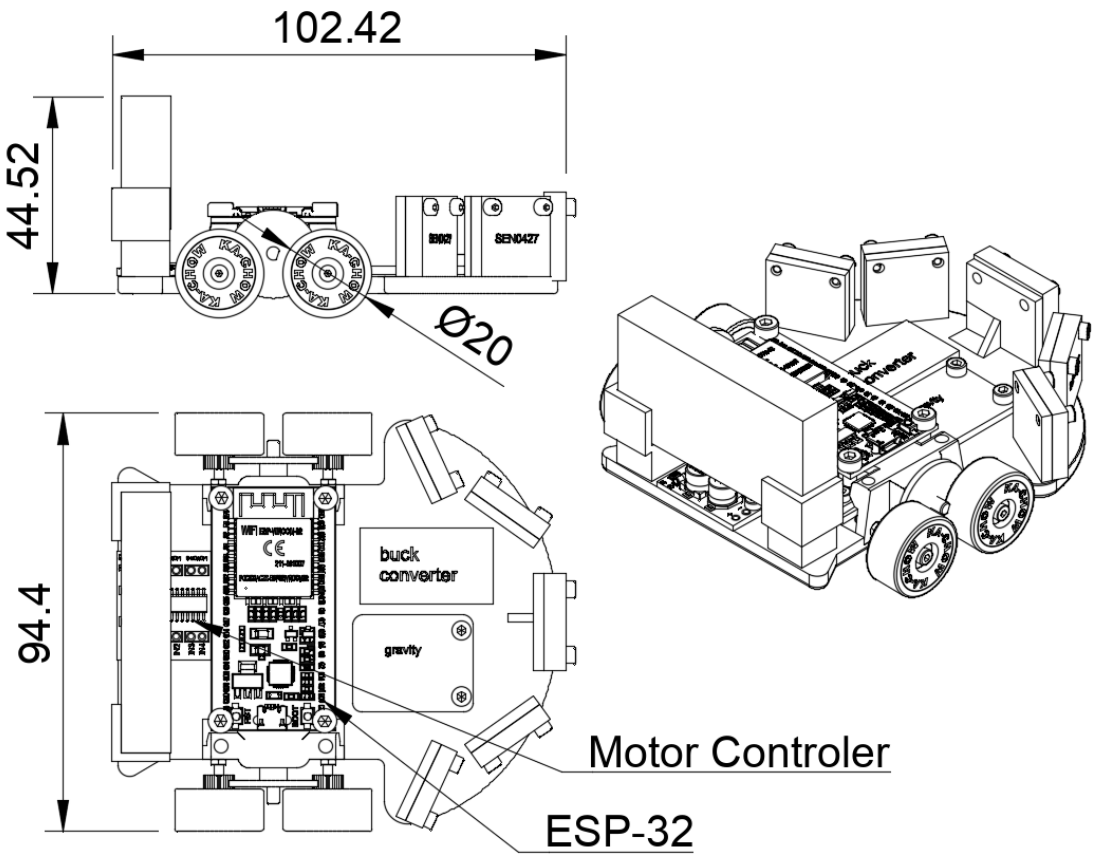
Component	Price	Quantity	Subtotal	Web Link
DFROBOT SEN0245	12.14	1	12.14	Link
DFROBOT SEN0427	6.79	4	27.16	Link
Pneus Kyosho High Grip MZW2-30	14.70	2	14.7	Link
Athlonix 16DCT 26G1 213P.1	64.30	1	128.6	Link
Firebeetle 2 esp32-e (n16r2)	11.88	1	11.88	Link
DFRobot Gravity I2C BMI160	8.99	4	8.99	Link
Engrenage plastique 35 dents	1.33	2	5.32	Link
Engrenage laiton 15 dents	7.75	1	15.5	Link
Motor controller MX1508	5	1	5	Provided in the handbook
2S 7.4V 1000 mAh LiPo	11	1	11	Provided in the handbook
7.4 V 350 mAh LiPo	6.75	1	6.75	Link
Total			**247.04**	

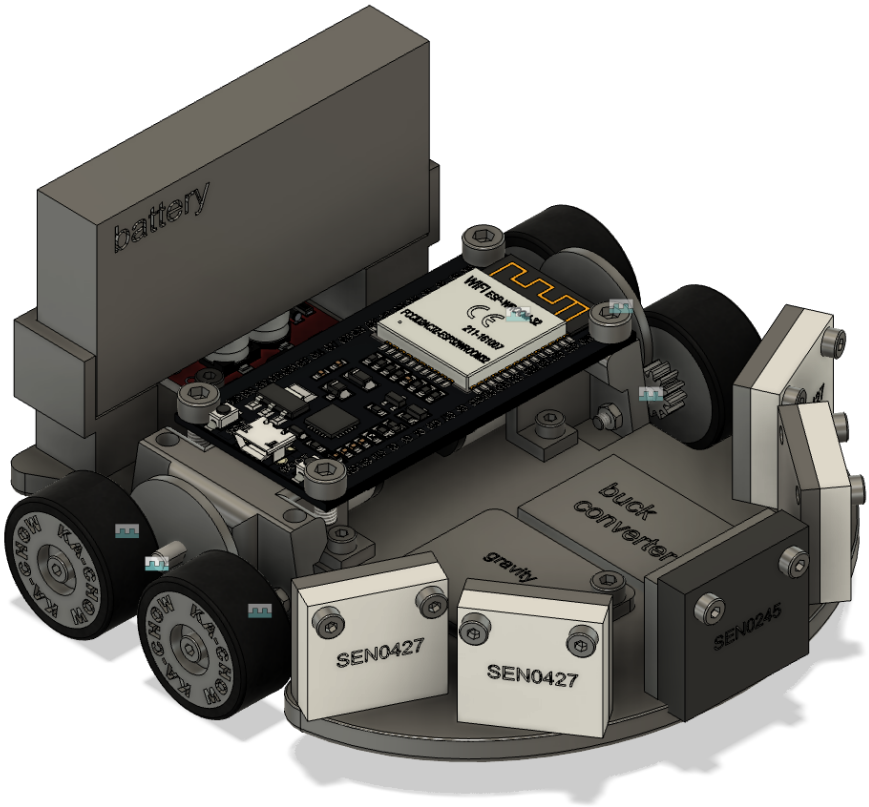
6 CAD design

See complete 3d file: git






102.42
44.52
Ø20
94.4
buck converter
gravity
Motor Controler
ESP-32


battery
buck converter
gravity
SEN0427
SEN0427
SEN0245

7 Electric diagram

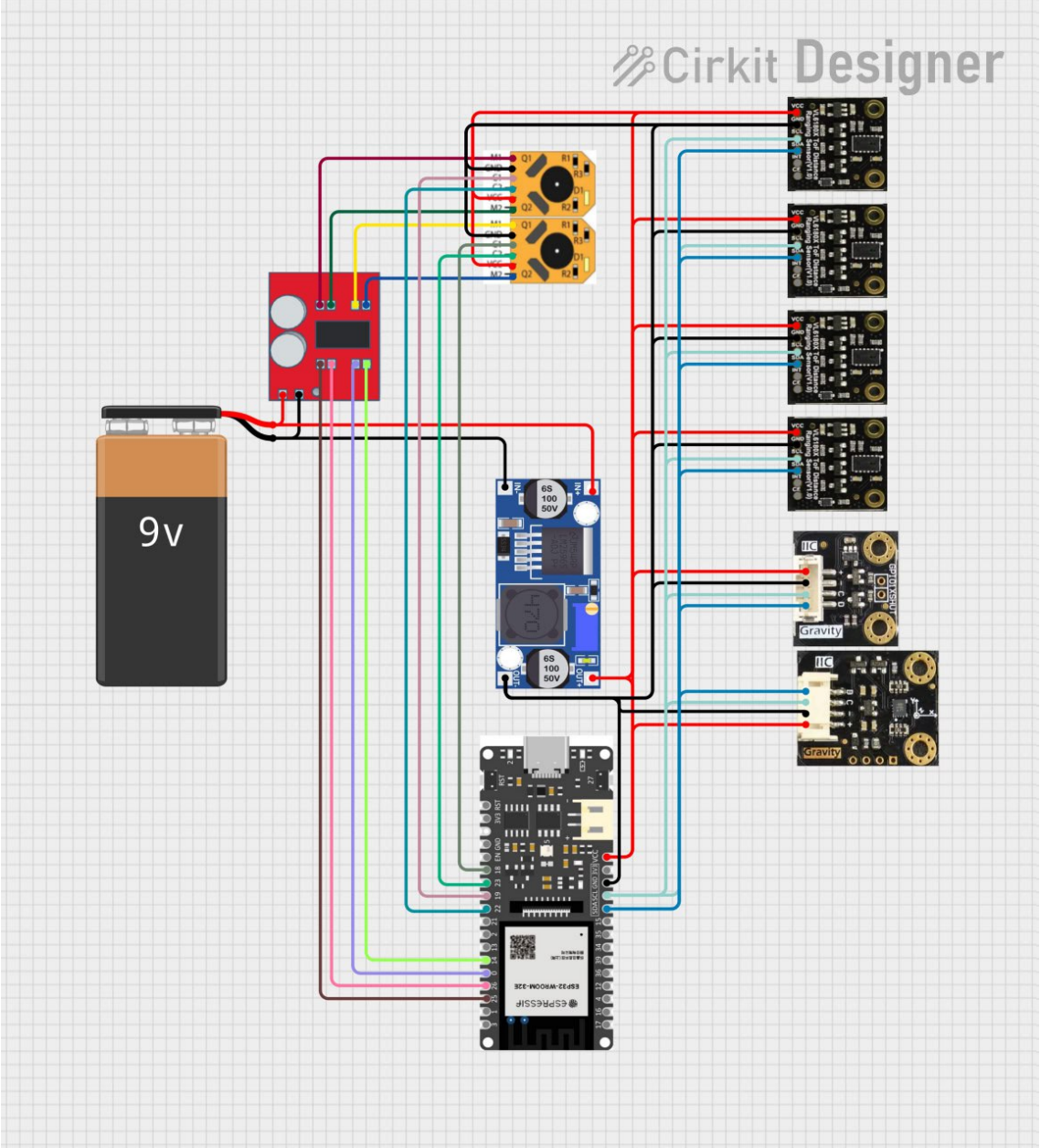


Figure 9: Mouse electronic circuit

8 Mouse overview

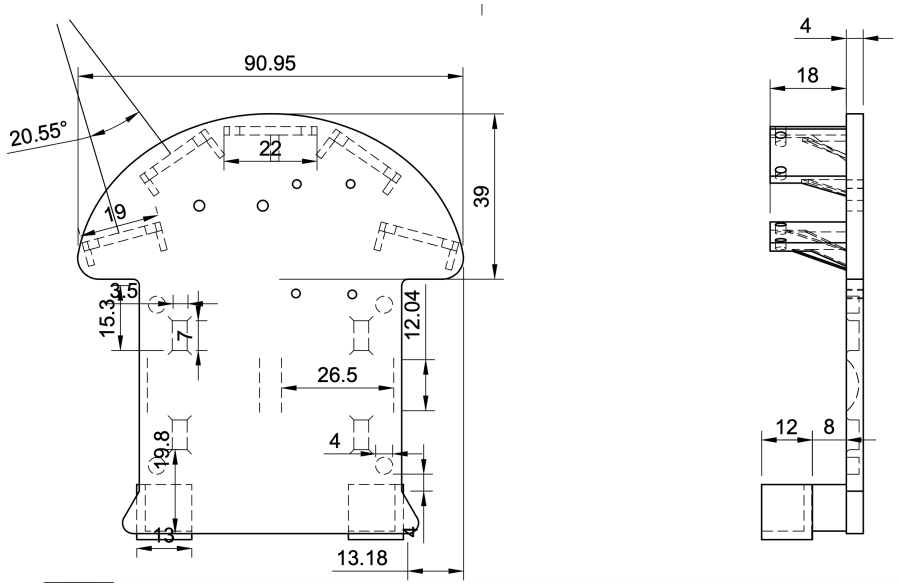


Figure 10: Mouse dimensions

The captors angle defined above are susceptible to be tuned.

9 Weight estimate

- 10 g motors → 20 g
- ≃ 8 g sensor →≃ 40 g
- 33 g battery → 30 g
- ≃ 25 g ESP → 25 g
- 2 g motor driver → 2 g
- ≃ 10 g Buck converter →≃ 10 g
- ≃ 10 g wheels →≃ 20 g
- ≃ 5 g gears →≃ 10 g
- ≃ 20 g battery protection →≃ 20 g
- ≃ 21.6 g MDF plate $\left(3.6 \times 10^{-4}\,\mathrm{m}^3 \times 600\,\mathrm{kg/m}^3\right)$ →≃ 22 g
- miscellaneous items →≃ 20 g

Total: ≃ 219 g
Maximum estimate 250 g

10 Maze building

10.1 Pillars

We chose to 3D print the pillar that will support the maze walls. We were considering using manufactured metal squares that we saw at SPOT but the dimension and the sharp edges were not ideal. We chose to elevate the joint to have more surface area in contact with the flooring (that is designed by the other team) so that if need be the pillars could link plates together.

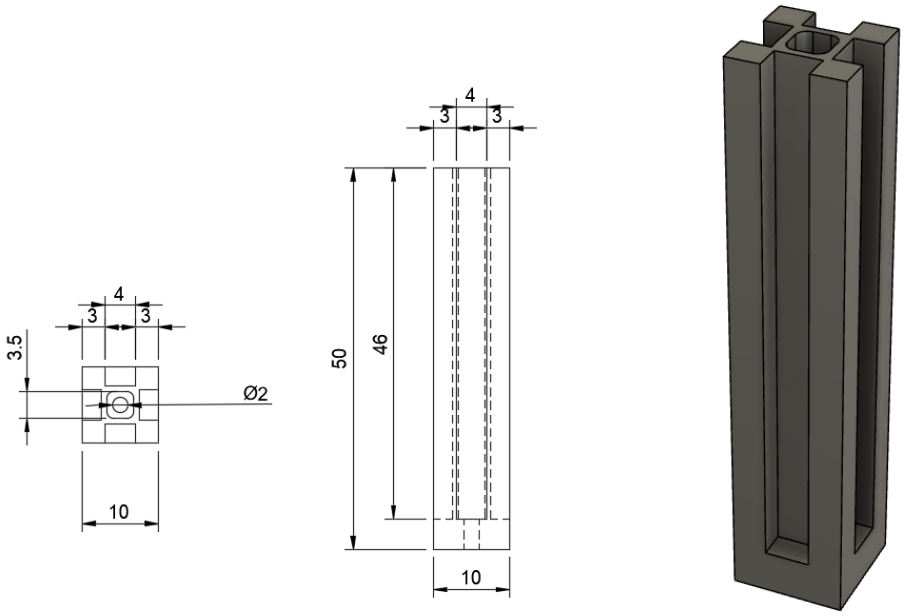


Figure 11: Pillar drawing

10.2 Walls

Official maze use 3D printed walls we decided to use layer of mdf to go faster and cheaper, we layer 3mm long plates with 2 central 4mm plate that will act as the male joint to slide in the pillars.

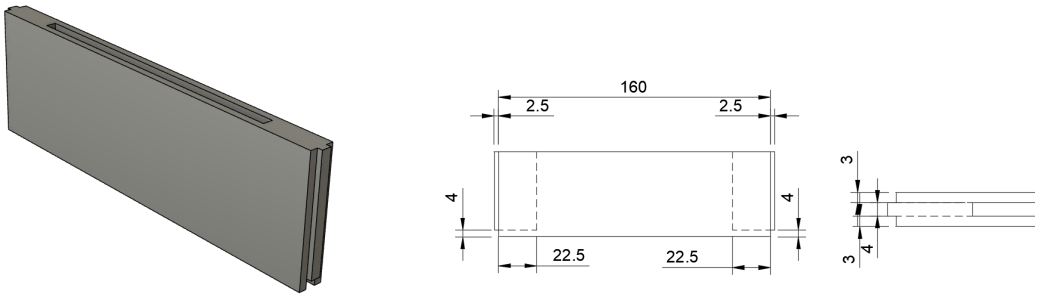
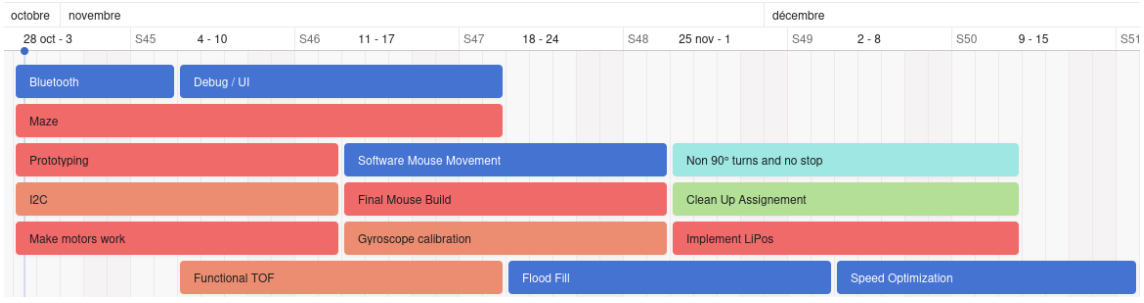


Figure 12: Wall drawing

Maze drawings link: git

11 GANTT Agenda



11.1 Maze

The cooperative process of building and finalizing the choices for the final full maze.

11.2 Bluetooth support

Choosing the Bluetooth library to use for the debugging interface, establish a way to send data over the Bluetooth to an external computer.

11.3 Prototyping

Building a rudimentary jig to test N-20 motors and breaking capacity.

11.4 I2C

Familiarize ourselves with using I2C for the sensor Bus.

11.5 Make motors work

Use the prototype to understand the different motors' properties and choose the final motor for the mouse.

11.6 Debug/UI

Software for both the mouse and the computer side of our debugging interface which will transmit information on the mouse status in real time to help us understand issues when they present themselves.

11.7 Final Mouse Build

Final mouse assembly

11.8 Software Mouse Movement

Programming turns, going forward stopping, etc.

11.9 Gyroscope Calibration

Software : calibrate and handle drift of gyroscope.

11.10 Functional Time Of Flight

Software : make sure we have a runing code to get the information from the sensors and a good way to exploit it for absolute position and relative position.

11.11 Flood Fill

Software : implement the flood fill algorithm on the mouse to give it instruction on how to resolve the maze.

11.12 Non 90 degrees turns and no stop

[optional] Build new commands for turning without stopping and make new turns that aren't 90 degrees.

11.13 Clean up assignment

Gather all issues and documentation and prepare it to be presented.

11.14 Implement Lipos

Safely implement lipo batteries if need be to be able to recharge the mouse in the future.

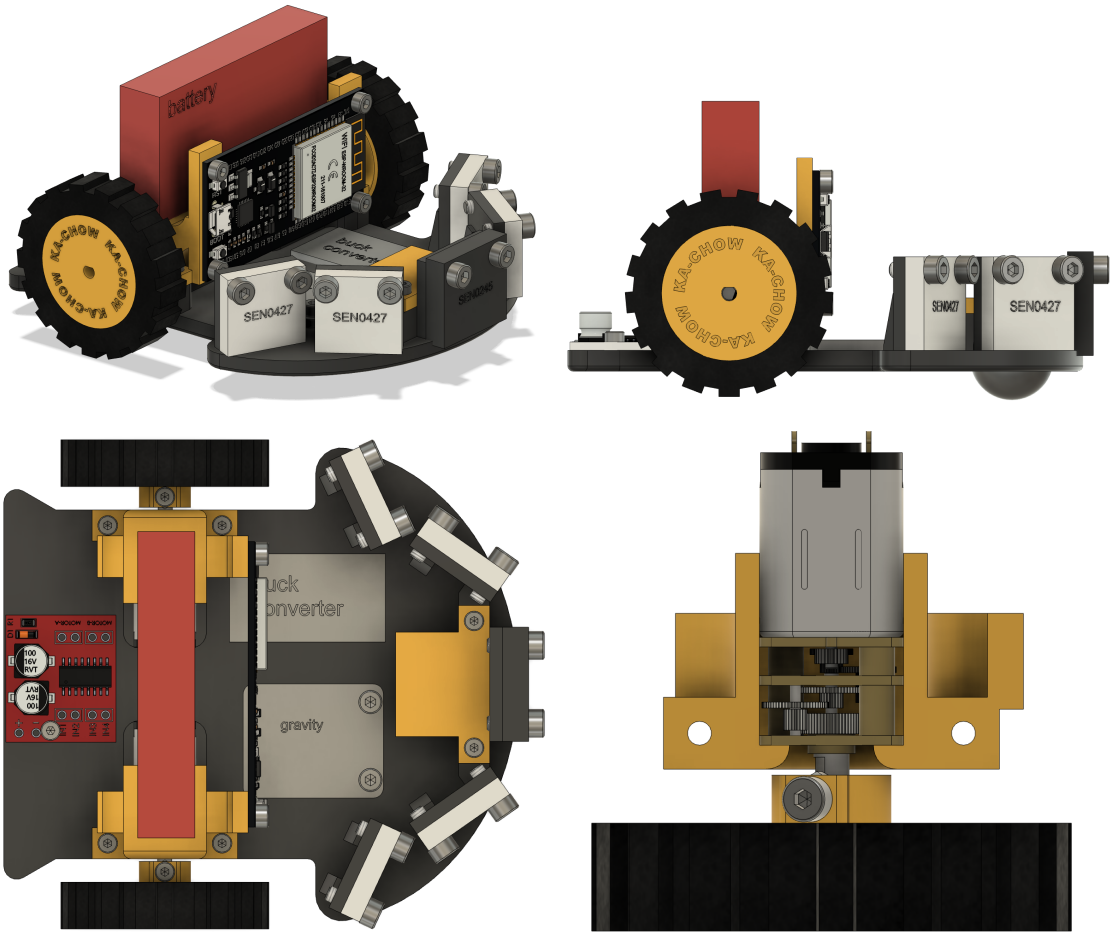
11.15 Speed optimization

Modify the floodfill algorithm to find the fastest path instead of shortest path, smooth out turns, higher max speed, etc.

12 Risk assessment

- **Mouse Size Constraint (Target: under 10cm)**
  - *Issue*: Designing a compact mouse under 10cm may restrict internal space, potentially impacting component selection and arrangement, the size restriction is especially important for diagonal movement.
  - *Consequences*: If the mouse is larger, diagonal movement may be unfeasible.
  - *Solution*: Either eliminate diagonal movement or redesign the mouse, possibly using a smaller battery or reconfiguring components to fit within the size limit.
- **Battery Energy Density and Weight**
  - *Issue*: Balancing energy density, weight, and size in lithium batteries is challenging
  - *Consequences*: Heavier batteries increase the mouse's load, requiring more motor torque for efficient movement.
  - *Solution*: Consider using lighter frame materials, such as switching from MDF to 3D-printed components, and explore using lighter, single-use batteries.
- **Sensor Reactivity and Precision in Movement**
  - *Issue*: Ensuring sensors are responsive enough for precise stopping, turning, and distance management is essential; delays could lead to inaccuracies in movement.
  - *Consequences*: Insufficient sensor reactivity could cause inaccuracies in positioning and turns, risking collisions and DNFs
  - *Solution*: Reduce speed near corners to insure our response time is sufficient, and incorporate a combination of servos and sensors to minimize errors.
- **Maze Build Quality Impact on Movement**
  - *Issue*: Variations in maze quality, such as uneven flooring, misaligned walls, or insufficient grip, could affect the mouse's stability and alignment.
  - *Consequences*: Poor maze construction, such as uneven floors or non-parallel walls, could disrupt alignment, cause slippage, and lead to potential collisions and DNFs.
  - *Solution*: To smooth transitions, place felt carpets. Wall alignment issues should be minimal due to our precise 3D-printed pillars.
- **Gyroscope Calibration for Orientation**
  - *Issue*: Achieving accurate gyroscope calibration is crucial to maintaining orientation, especially during turns
  - *Consequences*: Miscalibrated gyroscopes may complicate orientation and turn management.
  - *Solution*: Combine wheel encoders with the gyroscope for turns and rely on side sensors to assist with orientation. With four side sensors and parallel walls, calculate orientation by assessing distance variances among sensors.
- **Center of Mass Optimization for Adherence**
  - *Issue*: Proper center of mass placement is critical for balanced rotation and stability
  - *Consequences*: The centre of mass ideally aligns between the wheels along the centreline to improve rotation and stability. Misalignment could impair performance.
  - *Solution*: Since batteries contribute most to weight, position them strategically to optimize the centre of mass. Design the battery bracket to allow for easy repositioning to fine-tune the balance.

- **Structural Durability for Testing Failures**
  - *Issue*: Multiple test runs with potential crashes
  - *Consequences*: Structural failures, especially sensor damage, could make the mouse unusable.
  - *Solution*: Limit initial test speeds, gradually increasing as we get more confident in our software responsiveness and braking capacity. If diagonal paths are used, add a bumper to the base and reinforce sensor mounts as needed.
- **N20 Motors Performance**
  - *Issue*: Achieving the target speed range of 2000-3000 rpm requires accelerating with reduced N20 motors, which lowers the available torque.
  - *Consequences*: Reduced torque can lead to insufficient acceleration, possibly preventing the mouse from reaching the desired speed quickly or even failing to move altogether. This may force a reduction in maximum speed, resulting in diminished performance.
  - *Solution*: Consider upgrading to Athlonix Brush DC 6V motors and implementing a braking system. Options include:
    * Passive braking through a custom gearbox (though potentially inadequate on its own).
    * Dissipating energy as heat in a dedicated circuit, allowing the motor to act as a generator.
    * Active braking through a small servo applying friction to the wheels.
    * High-frequency reverse bursts to slow the motor gradually.
- **Making gearbox to distribute the motor energy to the two wheels**
  - *Issue*: Making a homemade gearbox could be problematic. It needs to operate with very low friction, and suitable axles for the gears aren't available. Furthermore, constructing a precise gearbox is very challenging as everything has to be mounted precisely.
  - *Consequences*: Due to friction, N20 motors may not produce enough torque, resulting in a very slow mouse, the gears could shift, come loose or break.
- *Solution*: Switch to a 2-wheel micromouse design.
  - By using larger wheels (20 mm radius), the 600 RPM (62.8 rad/s) provided by the N20 motors will be sufficient (6V or 12V). (62.8 rad/s * 0.02 m = 1.256 m/s)
  - A "front wheel" is needed. We considered using a marble for this purpose.
  - The battery should be placed closer to the center of gravity to prevent the mouse from tilting back.
  - **PROS**
    * Easier and faster to build.
    * More durable.
    * Less energy loss.
  - **CONS**
    * Slightly heavier.
    * Less stable.
    * High acceleration may cause the front to lift.


battery
SEN0427
SEN0427
SEN0427
SEN0427
gravity