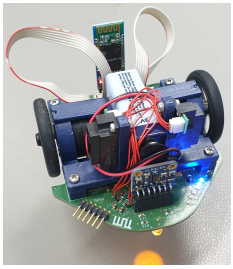
TECHNICAL UNIVERSITY OF MUNICH

Micromouse: Designing an Educational Racing-Robot from Scratch



Team Ratatouille

Authors Theo Goldfuss, Ran Gu, Niels Schlegel, Martin Schonger

Supervisor Dr. Ph.D. Alexander Lenz
Informatics 6 - Chair of Robotics, Artificial Intelligence and Real-time Systems

Oct 2020

Abstract

Team Ratatouille takes on the Micromouse challenge with the aim to design Hardware and Software of a mobile robot to navigate autonomously through an unknown maze, locate the center and race the shortest path towards it.

Its hardware is in charge of perceiving the surrounding environment and moving itself around the maze. Our micromouse is composed of one microcontroller, three sensors (one Time of Flight (Front), two Infrared (Right and Left)), three voltage circuits (9 V, 5 V and 3.3 V), one Bluetooth module, two DC motors with integrated encoders, five LEDs (two of them for balancing the mouse) and one programmer interface. The overall mouse is of size approximately 9 cm x 9 cm. For the physical design, a minimum number of 3D printed mounts for the sensors, motors and battery are created. The self-designed PCB ensures low electromagnetic interference between sensor lines as well as good heat dissipation to the ground pour by means of vias and smartly positioned traces and ground planes.

Its software is responsible of interpreting the environment by the signals from hardware and navigate the micromouse around the maze by sending the control signals back to its hardware components. The MCU dsPIC33F features the peripherals UART for PC-communication via Bluetooth, PWM for voltage control of the motors, QEI for velocity estimation, I2C and ADC in combination with DMA for sensor readings, and a Timer for calling functions at a 100 Hz frequency. A velocity PI-controller is tuned in simulation and on hardware in order to get a fast and zero steady-state error response of the motors without overshooting. For finer positioning, control to a reference pose with an intermediate direction is used. As for the maze strategy, a depth-first search-based exploration phase to build a graph model of the initially unknown maze needs to precede the race. The shortest path to the goal area is computed by a flood-fill approach.

To ensure a seamless first start-up, all board connections are tested with an oscilloscope and a voltmeter. In a second step all bought parts, such as sensors, motors and communication modules, are tested in order to get their characteristic curves for feedforward control.

The main problems were due to unforeseen PIN configurations in software, a too high initial instruction frequency of 40 MIPS, and the out-of-sync communication between the BT module and MATLAB.

The final goal of mastering the micromouse challenge will be tackled further after the report is handed in and can be tracked at our GitHub repository [28].

Contents

Abstract **1**

1 Introduction, Aims and Objectives **4**
1.1 Micromouse Challenge . . . 4
1.2 Team Goals . . . 4

2 Conceptual Design **5**
2.1 System Requirements . . . 5
2.2 Conceptual Component Overview . . . 7
2.3 Sensor System . . . 8
2.4 Wheel Placement and PCB Outline . . . 8

3 Hardware Design **9**
3.1 Components . . . 9
3.1.1 Microcontroller . . . 9
3.1.2 Sensors . . . 10
3.1.3 Voltage Regulators . . . 14
3.1.4 H-Bridge and Motor Driver . . . 17
3.1.5 Structural Elements Design and 3D Printing . . . 19
3.2 Schematic and Printed Circuit Board . . . 23
3.2.1 Electromagnetic Compatibility (EMC) . . . 26
3.2.2 Board Zoning . . . 28
3.2.3 Grounding . . . 28
3.2.4 Power Distribution . . . 29
3.2.5 Bypassing Capacitors . . . 29
3.2.6 Traces . . . 30
3.2.7 Oscillator Circuit . . . 31
3.2.8 Thermal Considerations . . . 32
3.2.9 LEDs . . . 32
3.2.10 Pull-up Resistors . . . 33
3.2.11 Buttons and Programmer . . . 35
3.2.12 Mechanical Considerations . . . 36
3.2.13 Final Steps . . . 36
3.3 Assembly . . . 36

4 Software Design **37**
4.1 Peripherals . . . 38
4.1.1 Timer Interrupt . . . 38
4.1.2 Analog-to-Digital Converter (ADC) . . . 38
4.1.3 Direct Memory Access (DMA) . . . 40
4.1.4 Inter-Integrated Circuit (I2C) . . . 40
4.1.5 Universal Asynchronous Receiver Transmitter (UART) . . . 41
4.1.6 Pulse Width Modulation (PWM) . . . 42
4.1.7 Quadrature Encoder Interface (QEI) . . . 43
4.2 Module Overview . . . 45

4.3 Controller Design and Motor Modelling . . . 45
4.3.1 PID Controller Choice . . . 46
4.3.2 Control Loop Design and MCU Implementation . . . 46
4.3.3 Manual PID Tuning Process . . . 49
4.3.4 Motor Model and Automatic Tuning Process . . . 50
4.4 Notation . . . 53
4.4.1 Coordinate Systems and Dimensions . . . 53
4.4.2 Robot State . . . 54
4.4.3 Maze Representation . . . 54
4.4.4 Global States . . . 56
4.5 Kinematic Model of a Differential-Drive Robot . . . 57
4.6 State Estimation . . . 57
4.6.1 Accuracy of Quadrature Encoders . . . 58
4.6.2 Alternative Localization Approach . . . 59
4.7 Pose Control . . . 59
4.7.1 Control to Reference Pose . . . 60
4.7.2 Trajectory Tracking Control . . . 63
4.8 Maze Strategy . . . 63
4.8.1 Naive Approach . . . 64
4.8.2 Maze Exploration . . . 64
4.8.3 Shortest Path Calculation . . . 66
4.8.4 Trajectory Calculation . . . 68

5 Testing **69**
5.1 Sensor Tests . . . 70
5.2 Communication Tests . . . 72
5.3 Motor Tests and PID Tuning . . . 73
5.4 Navigation Tests . . . 74

6 Problems Encountered and Possible Improvements **75**

7 Conclusions **77**

Bibliography **78**

Appendix **84**

A Requirement Diagram **84**

B Sensor Tests **84**

C Bill of Materials **85**

D Printed Circuit Board **87**

E Soldering **90**

F Alternative Localization Approach **92**

1 Introduction, Aims and Objectives

1.1 Micromouse Challenge

The 'Micromouse competition' started in 1977, when the IEEE spectrum magazine challenged its readers to design a small mobile robot which navigates autonomously in a random 10 x 10 (today 16 x 16) cells labyrinth [51].

Evolving into a more formalized and worldwide competition, every year more than 100 competitions in the USA, the United Kingdom, Japan and other countries take place nowadays [8].
The rules contain the following requirements [49, 54, 66]:

- The mouse must be completely self-contained and must not receive outside assistance.
- The maze consists of a quadratic 3 m x 3 m area with 16 x 16 square cells with side length 18 cm ($\sim$ 16 cm without the 1.2 cm thick walls) and 5 cm high walls.
- It is up to the designers how the robot detects the walls.
- A 2 x 2 cell block in the middle of the maze represents the goal area.
- The start square is always in one of the four corners and bounded by three walls. It always faces north (i.e. when the mouse moves forward the left-hand side is an outer wall).
- Every cell corner is adjacent to one or more walls.
- The robot has to start in one of the four corners, (1) explore the maze and find the goal ('run time'), (2) return to start, and (3) race the shortest/fastest path to the goal area.

1.2 Team Goals

The team's goals were to

- design a functioning mouse which by the end of October should be able to solve the challenge (without specifying how fast).
- design an aesthetic mouse (possibly small, symmetric) with maximal practical capabilities (maneuverability, debugging, fast prototyping and learning).
- enable every team member to have maximum learning effects in all demanded disciplines along the development process (see Fig. 1) (remote lab sessions or video recordings for members away from Munich, thorough explanations of each member's tasks and solutions, thus everybody is able to implement it).

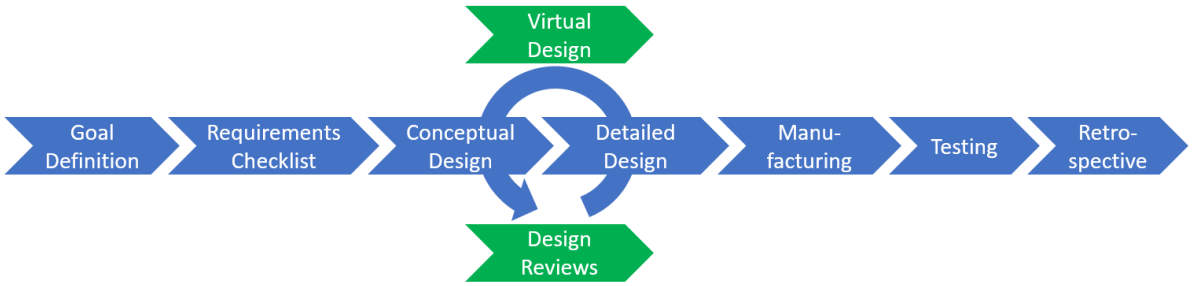


Figure 1: Development process of the micromouse

The following project related restrictions had to be considered:

- Preselected parts:
  - Microcontroller: dsPIC33FJ64MC804 from Microchip
  - Motors: Faulhaber brushed DC motor 006 SR IE2-16 with integrated encoder and gearbox with 33:1 reduction ratio
  - Wheels (2 choices): Pololu-1420 (60 mm diameter) or -1452 (40 mm diameter)
  - 16 MHz crystal in case an external oscillator is used
- Budget: consider parts less than ∼10 €, except for the motors, and coordinate with the other two teams to reduce the number of different parts
- Time: simple maze strategy by the end of September; develop more advanced strategy in parallel but aim for later completion date
- Design: create own designs in the fields of hardware selection (do not select plug and play H-bridge), PCB design, CAD design and software design (inspirations were given by our supervisor)

2 Conceptual Design

2.1 System Requirements

A requirement analysis gives all characteristics a system has and specifies what this system should do. These requirements should not focus on technical details but rather provide an overview of the system [13].

In our case the focus is on functional requirements. All requirements are structured in a requirement diagram, which can be found in Appendix A as a whole. This diagram helps to keep track of all needed functions regarding software and PCB design. To ensure readability we only display extracts. In Fig. 2 the robot is split up into the three sections: 'Sense', 'Move' and 'Communicate'.

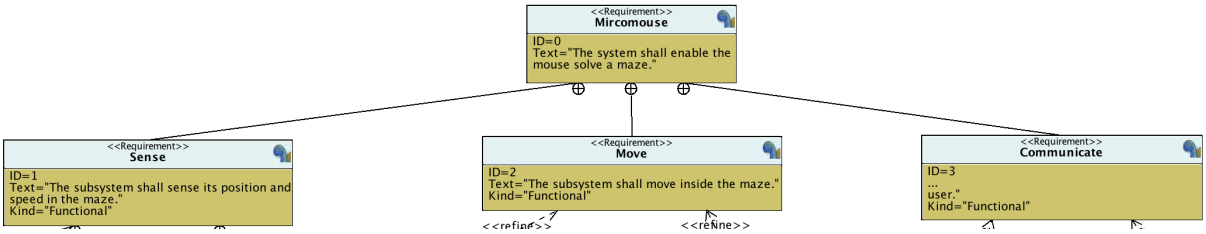


Figure 2: Requirement diagram detail with the split into Sense, Move and Communicate

The first branch is shown in Fig. 3. Sensing specifies for what the information is used and which sensor requirements are needed to gain enough information from the robot's environment.

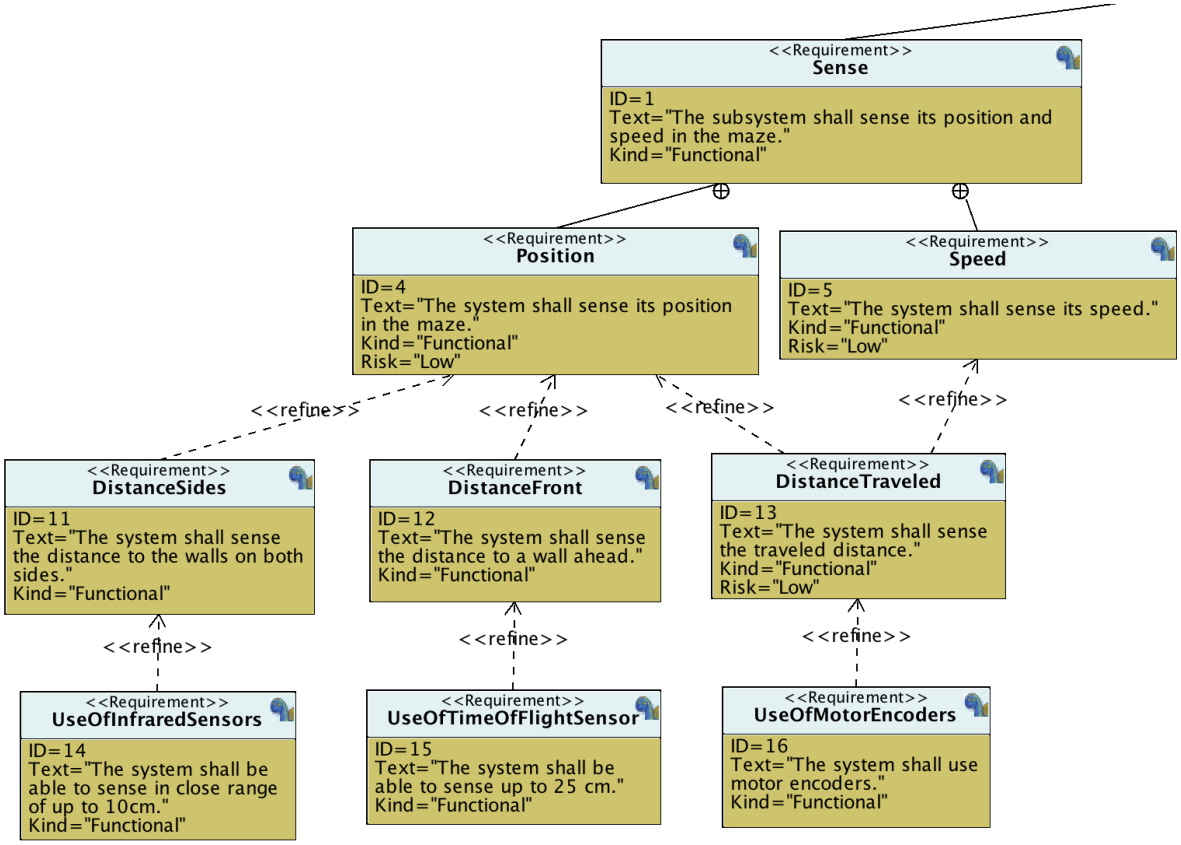


Figure 3: Requirement diagram detail of 'Sense'

Figure 4 contains movement and communication. Movement is divided by the two degrees of freedom, and communication is split into the two elements Bluetooth and LEDs.

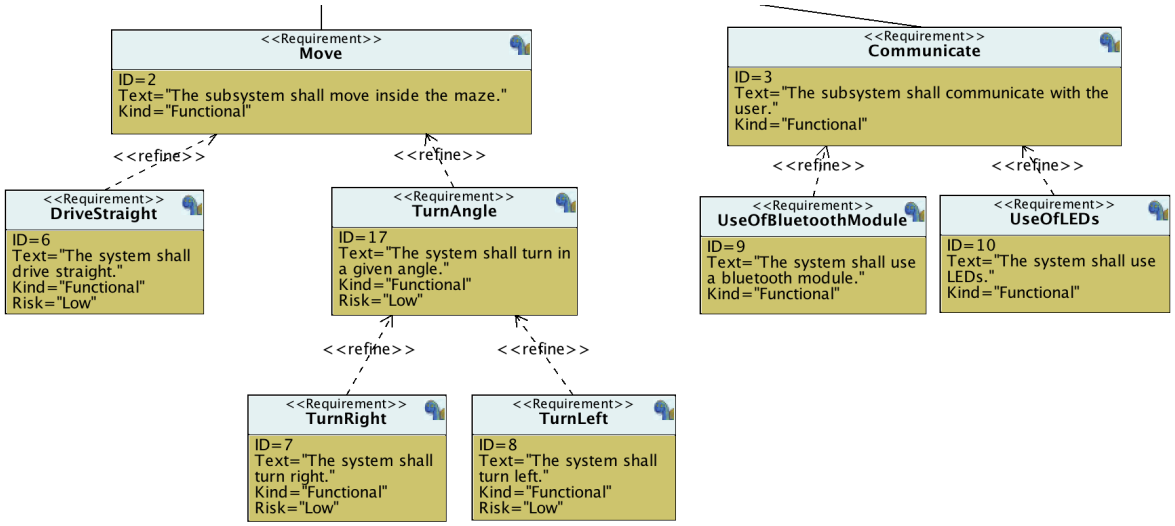


Figure 4: Requirement diagram detail of 'Move' and 'Communicate'

2.2 Conceptual Component Overview

The component overview in Fig. 5 provides a quick reference of the information flow between all involved external components, such as motors, sensors and BT module, as well as their relation to the MCU-internal peripherals. The call hierarchy of these peripherals is indicated. Further, the different voltage circuits including the voltage regulators are shown.

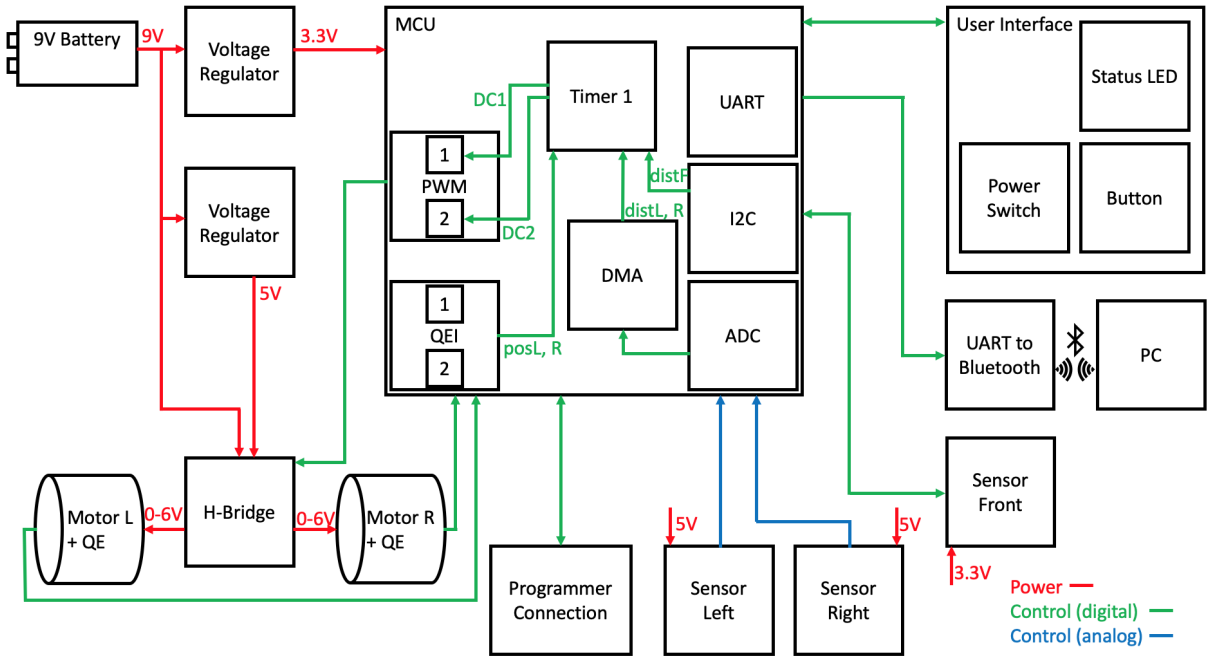


Figure 5: Conceptual component overview including information flow and voltage circuits

2.3 Sensor System

Since the floor of the maze is flat and the setup of walls has limited combinations (see Fig. 6), there exist finitely many different wall scenarios. Therefore, only a limited number of sensors is needed for detecting the surrounding walls.

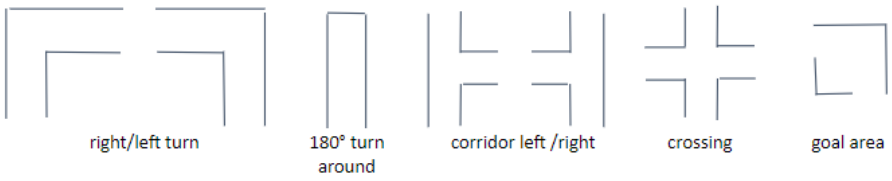


Figure 6: Maze situations to cope with

Also, only horizontally facing sensors are needed. In order to make a decision about the numbers of sensors, the design possibilities in Fig. 7 were considered. We opted for two side-facing and one front-facing sensor.

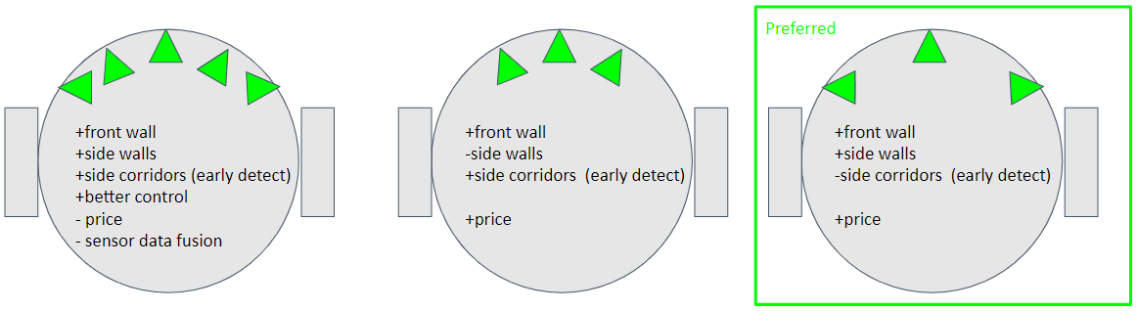


Figure 7: Alternatives of sensor configurations

To tackle all situations mentioned in Fig. 6, an evaluation on using a compass, which can measure the orientation of the robot, was carried out. A GY-271 module and an Arduino were used to perform a test on precision. The Arduino was connected to the GY-271 on a breadboard and turned with 90° steps, drawing a line at each step. The results are shown in Fig. 61 in the appendix. The third measurement with 180° is off by more than 45°.

This unconvincing performance plus the fact that the encoders in the motors can be used for the orientation measurement (see Section 4.6.1) lead to the decision of not using a compass module for our micromouse.

2.4 Wheel Placement and PCB Outline

The position of the wheels has an influence on the maneuverability of the mouse. The minimum space is 168 mm in one of the 16 x 16 squares [54]. Centered wheels lead to the ability to turn on the spot without needing more space than the maximum diameter of the mouse (see Fig. 8). Therefore, we intended to place them as centered as possible to ensure maximum maneuverability. Also, the wheels should not reach beyond the PCB outline in order to have a bumper function when hitting or sliding a wall.

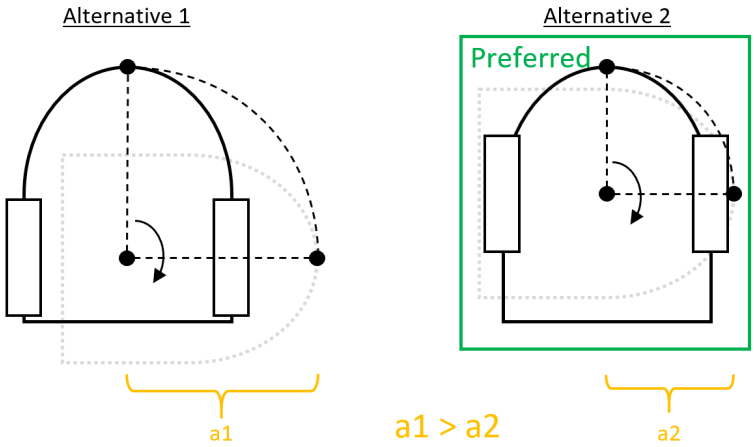


Figure 8: Turning circles of two different wheel positions

To stabilize the mouse platform, a third or fourth item touching the floor is needed. A third wheel without motor drive and spinning freely, i.e. a castor wheel, leads to unpredictable positions and driving behaviors of this wheel when turning around. Instead, two simple through-hole LEDs are placed below the PCB in the front and the rear part. Thus, no whipping effect takes place in accelerating or breaking situations. Standard 5 mm LEDs have an epoxy lens which is sufficiently durable for this prototyping purpose [76]. The wheels are placed a little more to the back in order to balance out the weight of the motors and the weight of battery and sensors in the front.

The corners shall be rounded and flattened in favor of keeping the turning radius as small as possible (max. 10 cm x 10 cm) and be able to cut corners. A compromise between keeping the PCB as small as possible and creating a sufficiently big ground plane for heat transfer is necessary. The center of gravity is kept low by placing the motors not on top of the PCB but instead create a cut out so that it lowers the motors down into the board. The SMD LEDs and Buttons are placed where they can be seen and used easily. While keeping these requirements in mind, the mouse layout shall be as symmetrical as possible.

3 Hardware Design

3.1 Components

3.1.1 Microcontroller

The chosen DSPIC33FJ64MC804-PT [22], depicted in Fig. 9, is a 16-bit digital signal controller that provides sufficient features when designing our micromouse. As shown in Fig. 5, the MCU receives data through its implemented communication interfaces: I^2C, ADC and DMA that integrate the sensor readings into code; the PC communicates with the MCU through UART over Bluetooth. With a fixed timer function, its motor control PWMs are in charge of controlling our motors, and QEI provides the actual motor velocity to the MCU. Using these features, we are able to execute the desired maze algorithm on the MCU and control our micromouse.

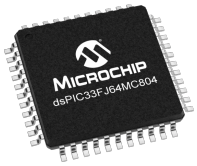


Figure 9: Microcontroller dsPIC33FJ64MC804 [47]

3.1.2 Sensors

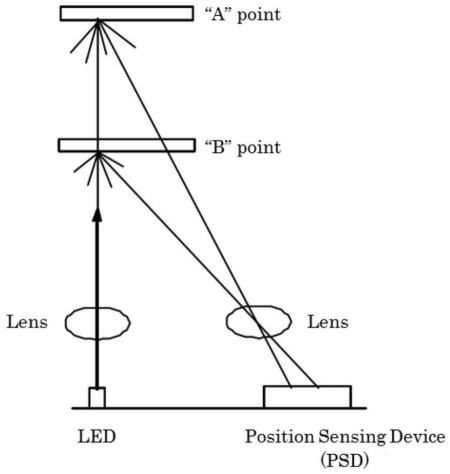
Sensors are the eyes of the robot, they provide the robot with information about its surroundings. Based on data obtained from the sensors, the robot can then make control decisions. For our project, we place multiple sensors on the micromouse to detect the presence or absence of walls and to verify its position in the maze so that we can keep our micromouse aligned at the center of the maze path and further find possible routes for solving the maze.

We evaluated several possible sensors that might fit our requirements on measuring distance.

Sensor Categories and Principles

- **IR Sensor:** An infrared sensor (IR) is a combination of an infrared transmitter that sends out invisible infrared light into the environment and an infrared receiver that detects the radiation transferred back from the transmitter [39]. The distance is calculated using triangulation of the beam of light, as shown in Fig. 10. When the position of an object varies, the angle of the reflected beam of the transmitter light changes, along with the change on the position sensing device. The distance between sensor and object is then output as an analog signal output.

(a) sharp IR GP2Y0A51S [38]

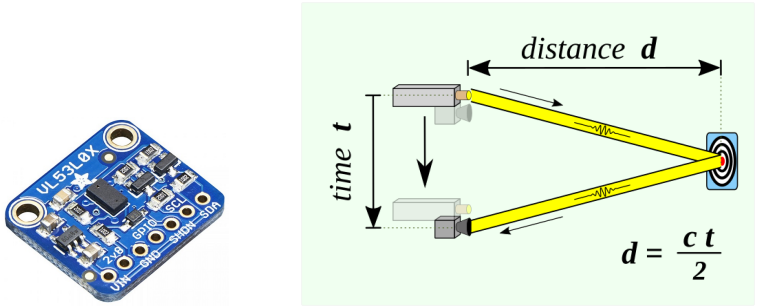


(b) simplified structure of IR sensor [6]

Figure 10: IR sensor model and its principle

- **Time of Flight(ToF) Sensors:** As indicated by its name, the sensor resolves distance between the sensor and the object by measuring the round trip time of

an emitted light signal by a laser. Figure 11 depicts this principle, along with the specific considered model.

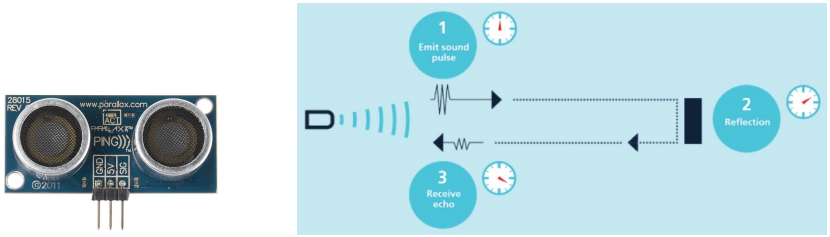


(a) ToF VL53L0X Sensor [29]

(b) ToF principle [74]

Figure 11: ToF Sensor model and its principle

- **Ultrasonic Sensors:** This type of sensor uses sound as medium instead of light. Similar to IR sensors, the emitter produces high-frequency sound waves that are out of human-hearing frequency range and the receiver detects the waves reflected by some object. The distance between the robot and the object is then calculated based on the total travel time of the sound waves as illustrated in Fig. 12. The specific model considered is also shown there.



(a) PING))) Ultrasonic Distance Sensor [7]

(b) Ultrasonic sensor principle [27]

Figure 12: Ultrasonic Sensor model and its principle

Sensor Selection

There are a few criteria that we took into consideration when choosing sensor types for our specific micromouse. First, the choice of the sensor should make the robot's design as simple as possible, requiring a minimum level of sophistication. Second, there should be as few sensors as possible, providing sufficient knowledge about the environment while minimizing redundant coverage.

Advantages and Disadvantages of Different Type of Sensors

- **IR Sensors**
  - + affordable sensors
  - + Produce analog and digital outputs
  - + Delivers high repeatability
  - - Sensitive to ambient light
  - - Provide good accuracy over shorter range but performance degrades with longer distances
- **ToF Sensor**
  - + Less sensitive to ambient light than IR sensors
  - - Power consumption and heat generation of components relatively large
  - - Unit price is more expensive compared to IR sensors
- **Ultrasonic Sensors**
  - + Provide accurate, numerical representation of distance
  - - Subject to electromagnetic interference
  - - Greater volume than other sensors
  - - Delivers low repeatability
  - - Sensitive to temperature changes

We decided to use existing IR sensors that are already on the market rather than to build our own IR transmitter and IR emitter so that we could put more effort into controlling the micromouse. We still decided to take the challenge of including two sensor types, partly due to differences in accuracy depending on the distance range. We intend to make our micromouse as light and compact as possible, so we discarded the idea of using ultrasonic sensors. Figure 13 contains a comparison of the specific sensor models considered.

	Preferred 	Preferred 	
	Infrared Distance Sensor	**Time of Flight Sensor**	**Ultrasonic Sensor**
	GP2Y0A51SK0F	VL53L0X	hc-sr04
Distance range	2cm-15cm	5 to 120 cm	2cm to 4m
Supply voltage	4.5 to 5.5 V	2.8 to 5 V	5V
Output	Analog output	digital I2C interface	Input TTL lever signal and the range in proportion
Package Size	27×13.2×14.2 mm		45*20*15mm
Current	12 mA		15mA
Costs	7.58 euro	14.87 euro	6 euro

Figure 13: Comparison and selection of distance sensors

Sensor Placement
Due to limitations on sensors' mapping distances, we chose the Sharp IR GP2Y0A51SK0F with detecting range of 2 to 15 cm and the Adafruit ToF VL53L0X sensor with a range of 5 to 120 cm.

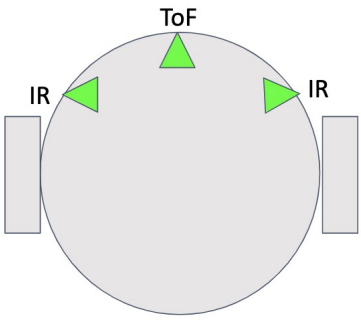


Figure 14: Sensor placement for micromouse

As shown in Fig. 14, the two infrared sensors are placed on either side of the micromouse. With their advantage of high accuracy for short distances, the detection of the existence of the side wall is guaranteed. The two IR sensors were mounted not at the edge of the robot but as close to the robot's center as possible (1.5 cm from the edge of the robot) so that they are in optimal sensing range and can provide more accurate measurements. Because of the unknown structure of the maze, larger distances may appear in the micromouse moving direction. The ToF sensor is mounted towards the front of the robot, also with an offset from the edge, in order to detect walls ahead up to 120 cm from the current location.

Electrical Specifications
The IR sensor's electrical characteristics are listed in Fig. 15. It's output voltage varies between 0.25 V and 2.5 V. On the other hand, the analog pin on the MCU requires the

input voltage to be typically max. 3.3 V (at most 3.6 V, though). Thus, the selected sensors are safe to use with our MCU.

(Ta=25°C, Vcc=5V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Measuring distance range	ΔL	(Note 1)	2	-	15	cm
Output terminal voltage	Vo	L=15cm (Note 1)	0.25	0.4	0.55	V
Output voltage difference	ΔVo	Output change at L change (15cm → 2cm) (Note 1)	1.35	1.65	1.95	V
Average supply current	Icc	L=15cm (Note 1)	-	12	22	mA

※L : Distance to reflective object

Figure 15: General Characteristics of Sharp GP2Y0A51SK0F IR Sensor [59]

The Sharp IR GP2Y0A51SK0F produces analog output that needs to be discretized and quantized by an Analog-to-Digital converter before it can be used in software (see Section 4.1.2). Furthermore, the measured values must be related to their corresponding distance (e.g. in meters). This issue is discussed in Section 5.1.

3.1.3 Voltage Regulators

To ensure proper operation of all circuitry and components, it is crucial that the supplied voltage follows load changes, i.e. remains constant for varying loads [56]. This can be achieved with the help of voltages regulators [9].

Advantages and Disadvantages of Different Types of Voltage Regulators

- **Linear Voltage Regulator:** The regulator acts as a variable resistor, automatically adjusting the resistance of the transistor to maintain a constant output voltage and continuously dissipating the difference between the input and regulated voltages as waste heat.

  + Quick response time
  + Low electromagnetic interference and noise
  + Ideal for low power applications
  - Energy inefficient if voltage drop significant
  - Often requires a heat sink

- **Switching Voltage Regulator:** The switching regulator uses controlled switches, which are set on and off at high frequency to achieve a specified average output voltage.

  + Higher power conversion efficiency (almost no loss)
  + No heat sink necessary
  - Complicated interface
  - More expensive
  - Introduces ripple noise

	Linear V Regulator LM7805(5V) / LP2950(3.3V)	Linear V Regulator LM2937ES-3.3/5.0 SMD	Switching (Buck) V Regulator LM2596S (3.3V/5V)
Temperature	0°C to +125°C / -40°C to 125°C	-40°C to +125°C	-40°C to 125°C
Current	5mA to 1A / 0.1A	500mA	3A
Input Voltage Range	7V to 20V / 2V to 30V	4.75V to 26V / 6V to 26V	4.5V to 40V
Output Voltage	4.75V to 5.25V / 3.3V,3.5 V	3.3V / 4.85V to 5.15V	3.168V to 3.432V/ 4.8V to 5.2V
Protection	Internal Thermal-Overload Protection	Reverse Battery protection, Internal short Circuit and Thermal Overload Protection	Thermal shutdown and current-limit protection

Figure 16: Comparison and selection of Voltage Regulator

We considered three different voltage regulator models, as shown in Fig 16. Since our voltage drops are not huge, from 9 V to 5 V, 9 V to 3.3 V, linear regulators are feasible. Also, because our currents and voltage drops are not very high, we can dissipate the heat energy from linear regulators. Eventually, we chose the linear voltage regulators LM2937ES-5.0NOPB and LM2937ES-3.3NOPB, because these come in a SMD package which allows heat dissipation over the ground plane.

Using a voltage regulator brings the advantage that the input voltage can vary within a certain range while the output voltage will still be within specified limits. In particular, the chosen voltage regulators will provide output voltages between 3.14 V and 3.46 V, and between 4.75 V and 5.25 V as long as the input voltage, i.e. the battery voltage, is between 6 V and 26 V [44, 45]. Therefore, we can expect a normal operation of all components that are powered via the voltage regulators even as the battery voltage degrades over time. This does, unfortunately, not apply to the motors. Moreover, the motors are always supplied with at most two thirds of the battery voltage, if it drops below 9V, the maximum velocity of the mouse decreases.

The 3.3 V and 5 V voltage regulators can be connected to the circuit in parallel as well as in series, as illustrated in Fig. 17. The total wasted energy is the same regardless of the placement of the two regulators. If we placed them in series, the first regulator would need to handle higher current. Placing them in parallel will distribute the load and only cause them both getting a little warm. Thus, placing two voltage regulators in parallel is preferred [34].

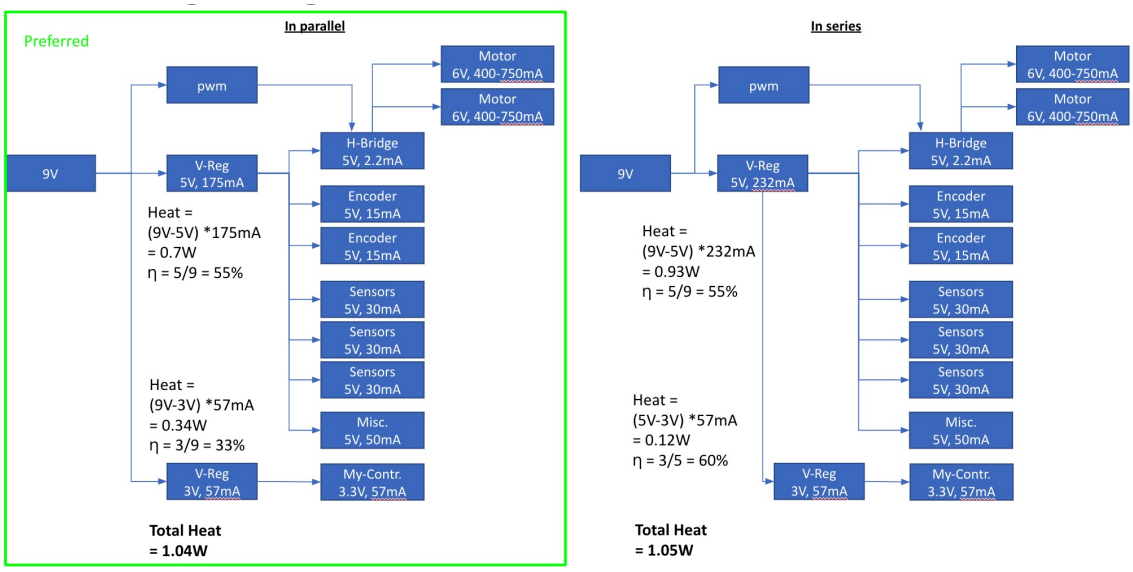


Figure 17: Different Possible Arrangements of Voltage Regulator

In order to argue for the suitability of our voltage regulators, we need to determine the maximum current that is to be expected in each voltage circuit. Further, we do a worst-case calculation for the power dissipation and compute the maximum temperature of each regulator [44, 45].

- **Components with 3.3V supply:**

dsPic	250 mA	max. current into VDD pin, [23]
Motor driver	1.8 mA	max. supply current, [63]
Encoders	2* 15 mA = 30 mA	max. current consumption, [12]
LEDs	5* 2 mA = 10 mA	operating forward current I_F, [4, 77]
TOF	40 mA	peak current, [67]
Total current	331.8 mA < 400 mA	

Under max current assumption of 350 mA, we get:
The dissipated power $P_d = (V_{in} - V_{out}) * I_{Load} + V_{in} * I_{ground} = 5.7 * 0.35 + 9 * 0.02 = 2.175W$, the max allowable temperature rise $T_{r,max} = T_{J,max} - T_{A,max} = 125 - 30 = 95°C$, and the max junction-to-ambient thermal resistance $R_{\theta JA,max} = \frac{T_{r,max}}{P_d} = 43.68$. Since $R_{\theta JA,max}$ is greater than 41.8, the regulator is safe to use at 350 mA. Further, since the value is smaller than 80, we need a heat sink, which would be the GND plane.
The max temperature it would reach is $90.92°C$.

- **Components with 5V supply:**

IR sensors	2* 22 mA = 44 mA	max. avg. supply current, [59]
Bluetooth module	30 mA	operating current, [11]
Total current	74 mA < 400 mA	

Under max current assumption of 400 mA, we get:
The dissipated power $P_d = (V_{in} - V_{out}) * I_{Load} + V_{in} * I_{ground} = 4 * 0.4 + 9 * 0.02 = 1.78W$, the max allowable temperature rise $T_{r,max} = T_{J,max} - T_{A,max} = 125 - 30 = 95°C$, the max junction-to-ambient thermal resistance $R_{\theta JA,max} = \frac{T_{r,max}}{P_d} = 53.57$. Since $R_{\theta JA,max}$ is greater than 41.8, the regulator is safe to use at 400 mA. Again, the value is smaller than 80, which is why we need a heat sink, which would be the GND plane.
The max temperature it would reach is $74.40°C$.

- **Components with 9V supply:**

Motors	2* 750 mA = 1.5 A	from motor curve in Fig. 42
3.3V regulator	331.8 mA	from above
5V regulator	74 mA	from above
Total current	1.9 A	

For the sake of completeness we include the total current in the 9 V circuits. It does not need to be considered for dimensioning the voltage regulators as it comes directly from the power supply, i.e. the battery.

3.1.4 H-Bridge and Motor Driver

The robot has two motors which have to be driven forwards and backwards and need to be controlled with different speeds. Since the microcontroller can only deliver 3.3 V and a limited current, a separated system shall deliver a voltage below 6V (max. voltage of DC motors) which can be controlled via PWM. An H-bridge, as an integrated circuit (IC-Driver), is a commonly used chip to fulfill this task. When the motor is spinning and then switched off, the inductive load of the coils can generate a high voltage or current which may destroy the transistors. From the voltage formula for an inductor

$$U_L = L \cdot \dot{I}_L$$

it can be inferred that a fast current change through the inductor results in a high voltage spike across. The time constant of the inductor

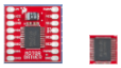
$$\tau = L/R,$$

where R is the series resistance, corresponds to the rate at which the current can change through the inductor. It takes about 5τ for the current to stop after opening the switch. The motor driver takes care that no short circuit is possible (all transistors are off for a small amount of time which results in a certain dead time). When the magnetic fields of the motor break down, protection diodes, also called flyback diodes, ensure that the transistors are not destroyed by the reverse voltage.

To make an informed decision on a specific motor driver, we compared different models in Fig. 18. Since no modules were to be used, only L293B and TB6612FNG could be considered in the end. Of the two, we chose TB6612FNG as its efficiency is above 90%,

the costs are very low and no heat sink is needed (which implies also the high efficiency of the IC-Driver). Also, it has built-in protection diodes.





Preferred

	L293B	L298N	TB6612FNG	LV8406T	Cytron MD10C
V_M supply	4.5-36V	4.5-46V	2.5-13.5V	-0.5-16V	5-30V
A supply	1 A p. ch, 2A peak	2 A p. ch 3A peak	1 A per ch; 3A peak	1.4A p.ch 2.5 peak	13A-30A
Efficiency	?	40-70%	**>90%**	?	?
Logic V	4.5-36V		2.7-5.5V	-6V	3.3-5V
V drop	? (high)	1.4V	0.05-0.13V	?	?
pwm frequ.	-	-	100 kHz	200kHz	20 kHz
Features	-	-	**standby-mode; MOSFET**	D-MOSFET	Manual Test buttons
Heatsink	heatsink needed (PCB or external mount)	internal Heatsink	**No heatsink needed**	internal Heatsink	internal Heatsink
Package	small	medium	small	big, SSOP20J	big (75x43x12)
Costs	+ (2.72€)	++ (4.58€)	+ (1.72€)	~ ++ (6€)	++ (9.76€)
Delivery T.	1 day	2-5days	1-2 weeks	not available	1-2 weeks

Figure 18: Comparison and selection of H-bridge

The selected motor driver from Toshiba has the following features:

- Short circuit and overload protection
- A small package size (SSOP24)
- Clockwise and counterclockwise/short brake/stop function modes (see AIN1 and AIN2 logic table in the datasheet [63])
- Standby (power save) mode
- Built-in thermal shutdown circuit
- Output current : 1.2A (avg) /3.2A peak (peak)
- Maximum switching frequency of 100kHz

Figure 19 shows partial circuit diagrams of the TB6612FNG output pins, including the integrated protection diodes, and how a motor is connected. The pin configurations and current flow for forward and backward mode are illustrated. The product is quite sensitive to electrostatic discharge, so the team ensured to wear an earth strap.

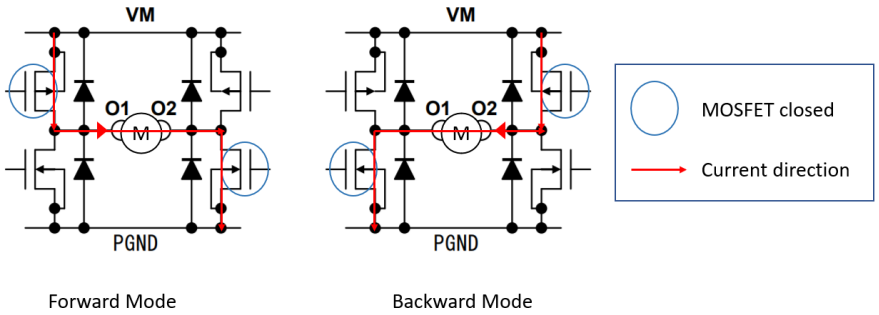


Figure 19: TB6612FNG circuit diagrams of the output pins with forward and backward configuration (adapted from [63])

3.1.5 Structural Elements Design and 3D Printing

For the design of physical elements certain guidelines shall be respected [26]:

- Minimize the part counts
- Use standard components
- Design multi-functional parts (multiple purpose)
- Determine acceptable tolerances
- Facilitate handling (assemble only from one direction)
- Design for non-error assembly (poka-yoke)

Even though the possibilities in 3D printing seem endless, it makes the production easier for FDM[1] printers (which were the only type available) when the design avoids support structures, i.e. scaffolds for spaces hanging in the air. This can often be achieved by rotating the part in the CAD/STL file so that the appropriate side of the component faces downwards.

We arrived at the final design after many iterations of refinements and repeatedly synchronizing with the PCB design back-and-forth. The end product can be seen in Fig. 20.

[1]Fused Deposition Modeling

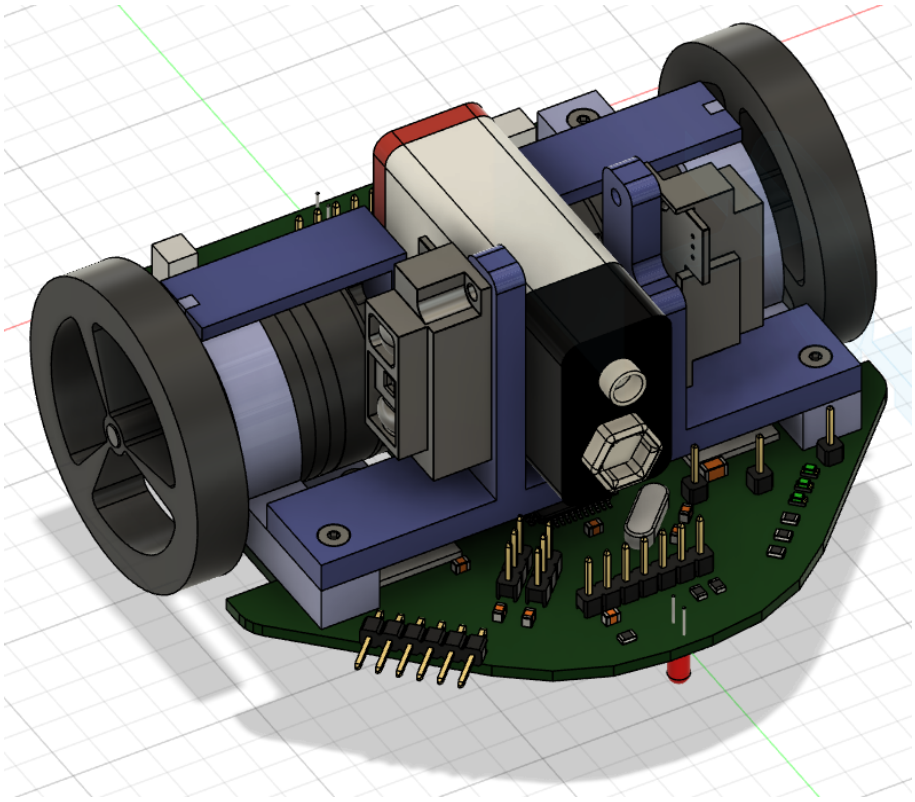


Figure 20: Computer Aided Design of micromouse

Motor Mounts Since the motors take the major part of space and weight, the motor mounts have been designed at first. After initial inspiration from a former micromouse challenge, the motor mount design has been changed in several ways to arrive at the version depicted in Fig. 21:

- Keep just enough distance to floor that the 5mm LEDs fits tightly below the board. Therefore, the motor is lowered into one of the cut outs of the board such that there is 9mm distance between floor and bottom side of PCB board (20mm (wheel radius) - (9.4mm (see drawing) + 1.6mm (PCB height)) = 9mm). Considering an average LED height of 8.6mm [77] there is 0.4mm tolerance on the front and back 5mm LED.
- Place two more holes for mounting screws below the center of gravity of the motor in order provide more stiffness.
- Implemented countersunk holes for the motor mounting and the PCB board mounts except the front wholes for mounting the sensor bridge on top. This ensures that the wheel is not interfering with the screw heads.
- Provide enough wall thickness (1.5mm) at the motor screws such that it does not break but is also thin enough that the motor shaft sticks out far enough for the wheel to be mounted and not interfere with the mountings.
- Provide enough height (6mm) such that the sensor bridge's bottom plane is high enough above the voltage regulators (5mm height) on the PCB.

- Provide a flat surface at the top of the round housing of the motor mount in order to put the battery bridge flat on top of it. To hold the bridge in place, there is a squared pin sticking out (2mm x 2mm) which fits into the cut out of the battery bridge.
- Provide possibility to put a mouse-like mask cover on top (advantages would be improved aesthetics as well as dust protection) by placing the screws not from above (as of now) but from the bottom through the PCB board to connect it with the mask cover.

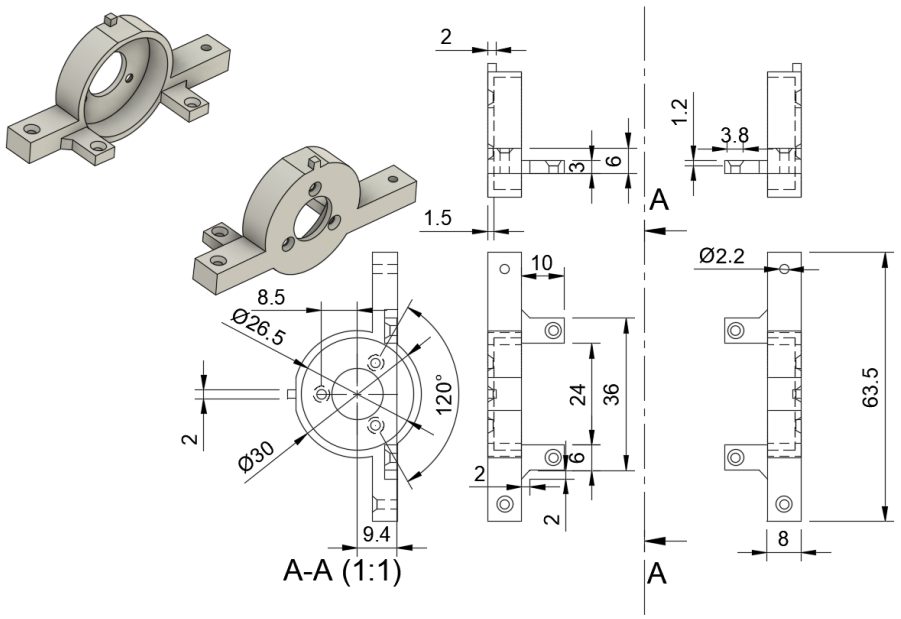


Figure 21: Motor mount left and right (line symmetric)

Sensor Bridge In order to mount the IR sensors, after thinking about two separate mounts, we eventually designed one single part which holds both sensors. As the IR sensors have a worse resolution at short distances we needed to place the sensors as centered as possible to get a higher distance to the outer borderline of the mouse. Considering the wall heights the sensors are placed as low as possible (trade-off with air space above voltage regulators). By mounting it on top of the motor mount and designing it as a bridge it comes with the advantage to save valuable space on the PCB board. Sufficient distance to the motor mounts is still kept so that they don't interfere with the sensors. Cut-outs in the sensor 'tower' mounts for the cables of the IR sensors have been included. The exact mounting dimensions for the sensors were given in their datasheet. In alignment with the battery bridge the space between the two sensor towers is 18mm wide which is wide enough for a standard 9V battery (see Fig. 22).

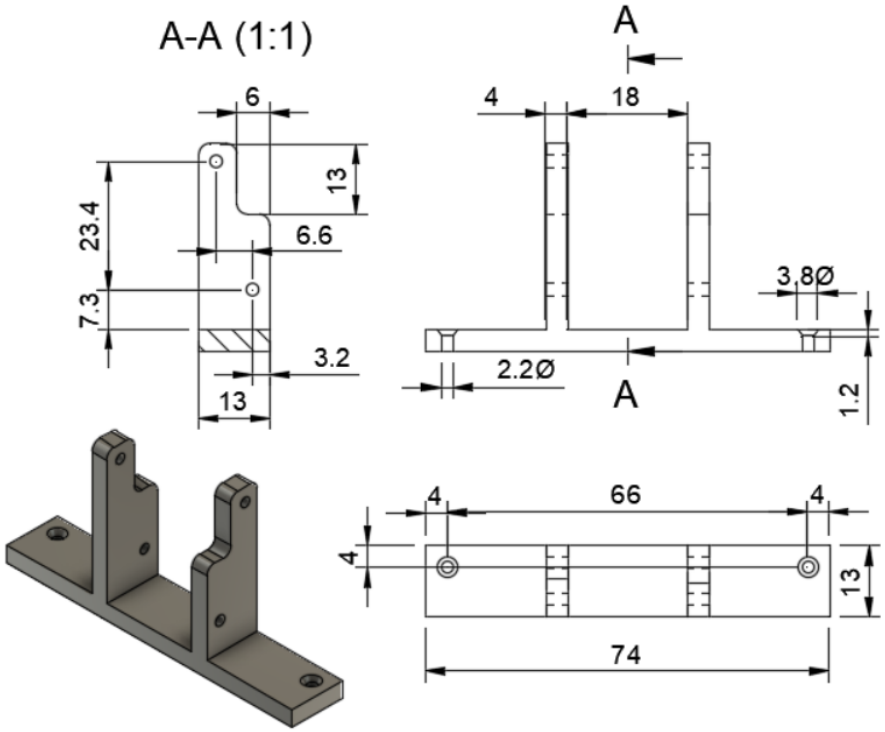


Figure 22: Sensor bridge

Battery Bridge In order to have a fixed battery placement above the PCB board the battery bridge is designed with a 18mm wide gap. With a cutout of 2.2mm x 2mm a form-locking connection to the motor mount is created (see Fig. 23).

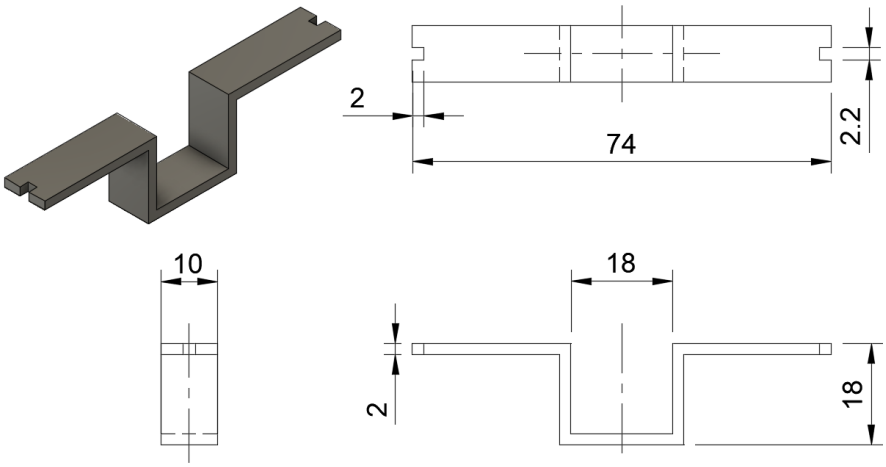


Figure 23: Battery bridge

3D Printing The parts are printed with PLA[2], 30% infill, and 3 top layers. By placing them on the right side in the 3D printer (see Fig. 24, almost no support structures are

[2]Polylactic acid

needed (only in the screw wholes). The colors symbolize the speed (red = slow for visible edges; green = fast for hidden infill regions). As maximum velocity 50mm/s are chosen.

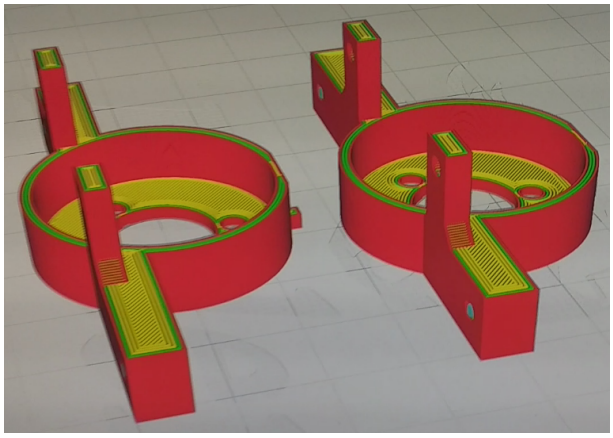


Figure 24: Placement and print settings of motor mounts in 3D printer

Painting We used two different 3D printers which had differently colored printing material available. To meet our aesthetics requirement, all 3D printed parts needed to be painted. First, all components were sanded down and cleaned. Then, a water-based blue spray paint was used to coat each part three times. The result can be seen in Fig. 30.

3.2 Schematic and Printed Circuit Board

Our goal was a pre-production prototype and therefore we designed a custom printed circuit board (PCB). There are numerous things to consider in this process which are discussed in detail in this section. The board evolved from the initial proposal in Fig. 64 into the final design depicted in Figs. 26 and 27. Figure 25 shows the corresponding schematic. In the second version, we improved on various aspects, spaced out the overall layout in order to simplify assembly, and more strictly applied electromagnetic considerations. We are proud to report that not a single issue with the PCB arose during assembly and testing.

For the design, we took some inspiration from the available development board from Microchip [1]. The board was manufactured by Beta LAYOUT[3]. The copper thickness is 35 µm and the overall thickness amounts to 1.6 mm.

Note that this section is not only intended to report on our specific board, but also to act as a reference for future projects. Also, this section is heavily based on [65], especially the parts on EMC and how to design for reduced EMI.

[3] `https://de.beta-layout.com/`

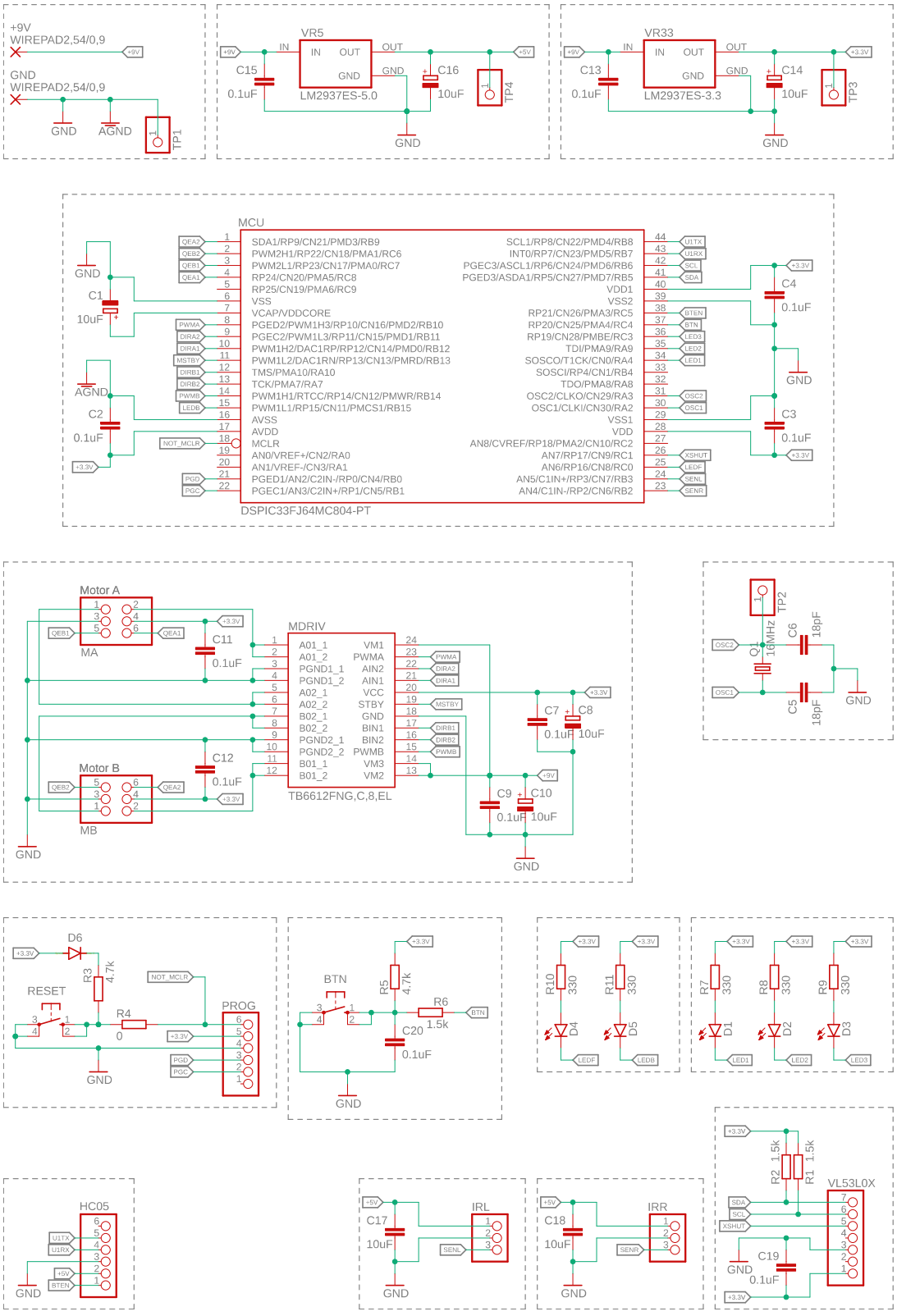


Figure 25: Schematic with components grouped by functionality

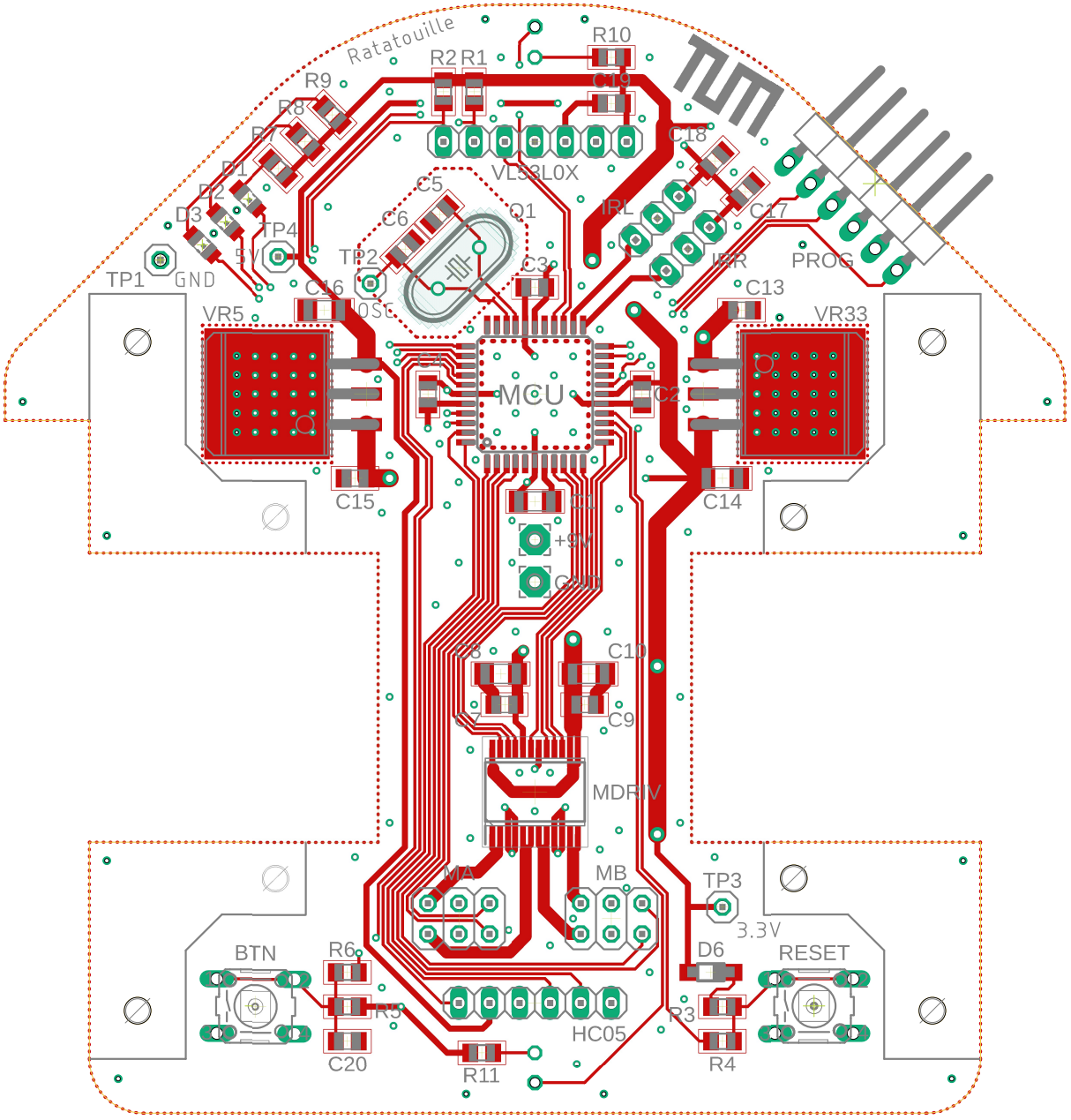


Figure 26: PCB top side without ground pour

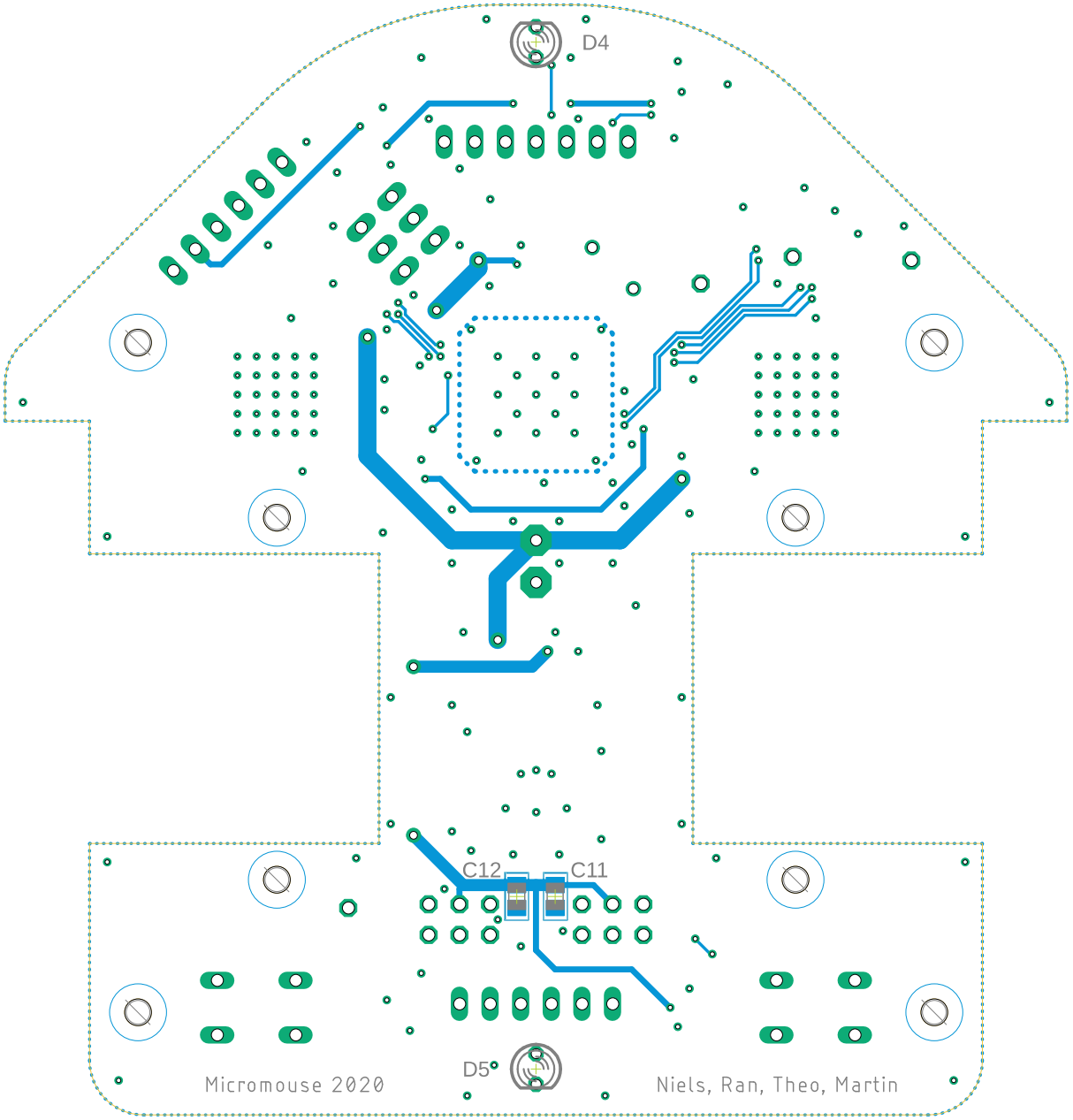


Figure 27: PCB bottom side without ground pour

3.2.1 Electromagnetic Compatibility (EMC)

Electromagnetic compatibility (EMC) regards the functionality of systems in their electromagnetic environment. Specifically, the 'unintentional generation, propagation and reception of electromagnetic energy' resulting in unwanted effects are regulated [14, 69]. One such effect is electromagnetic interference (EMI), by which externally caused disturbances of a circuit in the form of 'electromagnetic induction, electrostatic coupling, or conduction' are understood [71]. We follow the design recommendations from [65] in order to reduce EMI-related problems of our prototype.

Radio-frequency Sources
Microcontrollers generate radio-frequency (RF) noise, which is coupled to their surroundings via its pins. Particularly I/O pins are of concern here, since the corresponding traces form antennas. Furthermore, in a synchronized setting all current-switching events occur at the same time leading to large noise spikes. As the source of all RF energy the voltage regulators play an important part in the noise consideration. Therefore, we deployed carefully dimensioned linear regulators. The oscillator circuit constitutes another major noise source. Properly separating it from other components and circuits is essential.

Component Packages
Surface-mount technology (SMT) packages are generally preferred over through-hole (TH) due to their reduced inductance. Also, they often have a higher self-resonant frequency and allow for closer component placement which maximizes the effectiveness of noise-control components. We opted for 0805 and Size A (3216) packages for a good trade-off between space requirements and solderability. For the same reason the only diode was chosen as standard SOD-23 (small outline transistor) package and the TQFP (thin quad flat pack) version of the microcontroller was selected.

Where Does Noise on I/O Pins Come From?
Output pins pick up noise from the MCU power rails as well as noise capacitively coupled from adjacent pins. For input pins, the capacitance of the unused output transistor transfers noise from the power rails to the pin. The amount of noise is based on the impedance of what is connected to the pin, with higher impedance leading to more noise. It is generally recommended to configure unused I/O pins as inputs and connect them to ground to reduce capacitive coupling. To maintain flexibility we refrain from doing that. The authors of [65] state that we only have to worry about output switching frequencies of 50 kHz and above. Thus, our PWM signals with a frequency of 20 kHz should not introduce problems related to EMI.

Basic Loops
A signal, i.e. current pulse, that flows to a component requires a return path, e.g. via ground, to where it originated, forming what is called a loop. Thereby, the noise voltages and the associated currents take the path(s) of lowest impedance. Essentially, loops are antennas that are stronger the larger the laid-out area of the loop is. Hence, we can mitigate noise propagation by controlling the shape and impedance of the return paths.

Noise Types
One possible classification of noise is into differential- and common-mode. Differential-mode noise regards a signal, i.e. noise voltage, that travels to the target, does its job, and then returns. Naturally, one should not pick unnecessarily high frequencies and/or currents as this would cause more noise. Common-mode noise concerns a noise voltage that travels down both signal and return path. It is caused by a voltage drop across a shared impedance with another signal source (cf. multipoint topology as described in Section 3.2.4). The latter noise type is very common.

3.2.2 Board Zoning

The first step in PCB design is to define the general location of components on the blank board. It is important to locate the MCU next to the voltage regulator, and the voltage regulator next to where the power supply (in our case the battery cables) enters the board. We placed the analog circuits as far from the motor driver and motors as possible. The programmer connector should go very close to the MCU in order to keep the corresponding traces short. The Bluetooth module and motor connectors were simply included as pin headers and positioned with their respective size in mind. Only after the rough layout has been established the traces are drawn.

3.2.3 Grounding

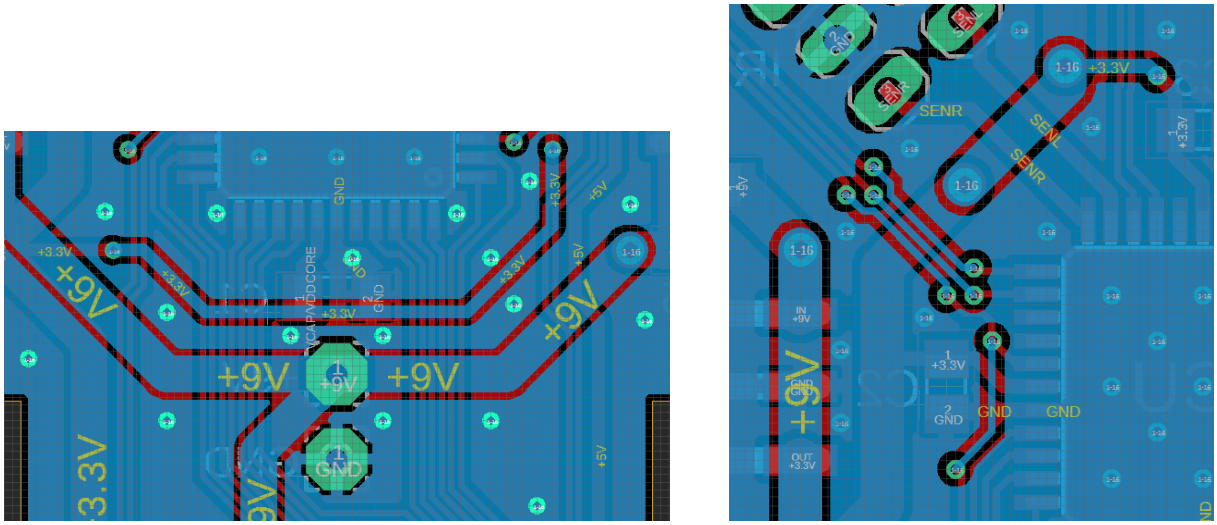
When talking about ground we can think of return paths for signals after they have done their job. Properly designing these paths is challenging, but also very important due to radiation from the signal traces. As radiation strength is proportional to the loop area, the problem can be mitigated by routing ground directly underneath the signal, i.e. providing a direct return path. Further, this lowers not only the impedance between MCU and voltage regulator, but also the impedance of signals that may become victims of noise, thus reducing the noise voltage/their susceptibility. A space-effective way of doing this is creating a gridded ground plane. The noise reduction is comparable to that of the solid middle layer ground of four-layer PCBs.

We accomplished that by keeping the bottom ground plane as uninterrupted as possible, avoiding islands and reducing the number and length of bottom traces. By jointly optimizing top and bottom layout we mitigated all dead ends and top-plane islands. The reason is that ground-fill patterns more effectively contribute to the grid when they are tied to ground at both ends. Effective tools for this are minor but selective trace re-routings and carefully placed vias to the opposite ground fill to form bridges over interrupting traces. An example is shown in Fig. 28a and a good illustration of the effects of this effort is given in Fig. 8 in [65]. Further, trace crossings were designed in a way that ground connections are opened up at the crossing, thereby prioritizing the bottom-side ground (see Fig. 28b). One could say that we tried to emulate, by careful design, many properties that make four-layer boards better.

Figures 62 and 63 depict the resulting ground pour on top and bottom side, respectively. An isolate distance of 0.3 mm was configured.

Microcontroller
Under the microcomputer, we created a solid ground pour that extends such that bypassing components and the oscillator circuit are included, thereby reducing the loop area. The ground pins of the MCU are connected to the ground pour right beneath it, which is in turn connected to the bottom side via vias. This is called a microcomputer ground. There is a reason for ICs, including our microcontroller, having multiple ground pins: it prevents voltage drops across the chip and tends to reduce losses and temperature.

Analog Circuits
In the presence of very sensitive analog circuitry one would create a separate analog



(a) A solid ground plane is emulated by bridging disruptive traces with vias and top-side ground patches

(b) Trace crossings are designed such that no dead ends are created in the bottom ground pour

Figure 28: Gridded ground and trace crossings

ground in addition to the digital ground to reduce the noise in these parts. Both grounds would only be connected at the zero potential reference point of the voltage supply. This is, however, not necessary in our case.

3.2.4 Power Distribution

The best topology for power traces is single point, where each component has its separate supply line originating from a single reference point. In that way there is no problem of common impedance coupling and, thus, reduced common-mode noise. We routed the big traces from the 9 V and 3.3 V reference points in a single-point fashion. Further away from the reference points we reverted to multipoint distribution due to space restrictions. Multipoint means that a single trace originates from the source and splits multiple times along the way to reach each target component. By carefully adjusting the trace widths we also reduce common impedance in this case (see Section 3.2.6).

3.2.5 Bypassing Capacitors

The purpose of including bypassing or decoupling capacitors between V_{DD} and ground near ICs is that any current at or above switching frequency is supplied not by the power supply but by these capacitors. If there is too much inductance in the bypassing loop and the current can not be provided, it is sourced from the supply, thereby introducing RF noise. Thus, the power traces are only for replenishing the charge in the decoupling capacitors. Measures for ensuring that are: (1) adhering to the 3:1 length-to-width rule for traces in the bypassing loop to reduce the impedance (see Section 3.2.6), and (2) keeping the loop area small by placing the capacitors as close as possible to the component or by extending the ground plane to include the bypassing capacitors rather than running ground traces (see Section 3.2.3).

Further, decoupling capacitors oppose any unexpected changes (drops and spikes) in the input voltage and keep it stable [55]. These capacitors also shunt noise caused by other circuit elements, reducing the effects on the IC [73].

Generally, ceramic capacitors are used due to their lower equivalent series resistance (ESR). For this type we opted for class 2 capacitors (e.g. X7R) as they are recommended for bypassing and decoupling applications due to their good trade-off of volumetric efficiency, and accuracy and stability [75]. A solid default choice, unless specified otherwise by any datasheet, is to place a 0.1 µF ceramic capacitor right next to components that are to be decoupled. In certain cases tantalum capacitors are suggested. These have a higher ESR than ceramic ones, which is sometimes needed by the decoupled IC. Note that the ESR is typically only specified for electrolytic capacitors, and only at a given frequency such as 100 kHz. In the case of high-frequency noise one could use a so-called decade pair consisting of a 0.1 µF ceramic and a 10 µF tantalum capacitor.

Next, we outline the selection and placement of the specific capacitors for our components. For the MCU every pair of power supply pins (V_{DD}/V_{SS}, AV_{DD}/AV_{SS}) requires a 0.1 µF low-ESR decoupling capacitor. It is recommended that the resonance frequency is in the range of 20 MHz or higher. While technically not a bypass capacitor, we also cover the CPU Logic Filter Capacitor V_{CAP} here. The on-chip voltage regulator of the microcontroller, responsible for producing the 2.5 V on which the core runs, needs a 10 µF capacitor to keep the output voltage stable. Using a low-ESR ($< 5\,\Omega$) tantalum type is recommended. The capacitors on the voltage regulators were selected according to Fig. 18 in [44] and placed according to Fig. 27 in [44]. Specifically, each regulator requires an output capacitor of at least 10 µF with an ESR between 0.01 Ω and 3 Ω, where larger capacitance values improve the transient response. Using solid tantalum capacitors is recommended. On the input side, each regulator needs a 0.1 µF capacitor. For the motor driver, capacitors for noise absorption should be connected as close as possible to the IC. We need two standard 0.1 µF ceramic capacitors and two 10 µF electrolytic capacitors. Further, there is a 0.1 µF capacitor on each encoder supply trace in order to dampen the current flow. The datasheet of the IR sensors recommends connecting a bypassing capacitor of 10 µF or more between V_{CC} and GND near the IC in order to stabilize the power supply line.

3.2.6 Traces

Inductance

The undesired inductance of traces increases with length and decreases, at a slower rate, with width. It causes voltage drops that radiate and propagate, which is especially problematic for RF energy-carrying traces. One rule-of-thumb is to stay within the 3:1 length-to-width ratio for power and ground traces between ICs and the voltage regulator. It also helps to route power and ground directly above one another as this reduces impedance and loop area (see Section 3.2.3) [40].

Crosstalk on Signal Traces

Traces running in parallel suffer from capacitive and inductive crosstalk or coupling between them. Capacitive coupling means that a rising edge on the source causes a rising edge on the victim and the same holds for falling edges, while in inductive coupling the

voltage change on the victim is opposite from the source. The former type of crosstalk is more common. The amount of noise coupled increases with the parallel running distance, the frequency, the magnitude of the potential change on the source, and the impedance of the victim. The further apart source and victim are, the less crosstalk.

To reduce crosstalk we can: (1) keep non-noisy traces away from regions where they could pick up noise, such as RF-noise-carrying traces connected to the MCU, connectors, and oscillator circuits, (2) route the return ground directly under signals that may become victims of noise (see Section 3.2.3), and (3) group noisy traces together and surround them by ground pour. Specifically, we positioned the low-current analog circuits as far away from the high-power motor driver and motors as possible. The sensor tracks were kept short and routed in a way that they are not parallel to power traces. Since the frequency on the analog traces will be similar, we put as much distance between the two traces as possible. If a crossing of noise-sensitive traces with other traces is inevitable, they should cross perpendicular.

Trace Width

Using a freely available PCB Trace Width Calculator[4] we estimated the required trace widths according to IPC-2221. For the worst case calculation of the trace from the 9 V pad to the motor driver IC, the following assumptions are made: a current of 3 A, which corresponds to twice the maximum value for both motors together (see Fig. 42), a copper thickness of 35 µm, a very safe temperature rise of 10 °C, and a trace length of 20 mm. We then obtain a required trace width of 1.37 mm (for external layers). Hence, a width of 1.5 mm for this and all other significant power traces is chosen.

Further, assuming a worst case current of 400 mA over a distance of 50 mm we calculated a minimum required trace width of 0.221 mm for any other power supply trace. By making every supply trace leading to more than one component 0.5 mm wide we stay safely above the minimum value. Any last-mile supply traces to single components are 0.25 mm wide.

3.2.7 Oscillator Circuit

The oscillator circuit constitutes a vital part of the board as its correct operation is a prerequisite for the operation of the microcontroller. It consists of a crystal and two load capacitors which resonate with the former and cause it to oscillate on its fundamental frequency. An external oscillator is used due to its higher frequency and accuracy.

To ensure proper functionality the circuit is implemented on the same side of the board as the microcontroller, thereby staying as close as possible to the respective OSC pins of the latter. The two accompanying load capacitors are placed right next to the crystal, and to shield from outside influence (e.g. from surrounding circuits) the trio is surrounded by an uninterrupted ground pour. Further, there are no traces and components right below the oscillator circuit on the other side of the board but also a solid ground pour [21, 23].

On the other hand, the oscillator circuit is a source of radio-frequency (RF) noise, primarily due to the oscillator swinging with its fundamental frequency. If crystal and capacitors are, however, properly separated from other components and traces, as well as

[4] `https://www.4pcb.com/trace-width-calculator.html`

small loop areas are ensured, then it will generally not result in problems [65]. See also Section 3.2.3.

The crystal was provided as is, i.e. without specification of the manufacturer's part number. Therefore, the unknown load capacitance C_L was assumed as 18 pF - the most frequent value on Mouser for 16 MHz HC-49 crystals. Assuming further a rather conservative stray capacitance of $C_S = C_L/2 = 9\,\text{pF}$ and using Equation 39-2 from [21] one calculates

$$C_1 = C_2 = 2 \cdot (C_L - C_S) = 18\,\text{pF}$$

for the two capacitors.

3.2.8 Thermal Considerations

During continuous operation certain components of our robot will experience a temperature increase, which needs to be accounted for. Most prominently, the calculations in Section 3.1.3 imply that it is necessary to heat sink our voltage regulators. To this end, the authors of [3] give mounting instructions, layout information and thermal advice for the TO-263 package. They suggest to place at total of 25 thermal vias underneath the chip to create a thermal connection to the bottom ground plane. It is important that enough solder is applied, possibly using a heat gun, to ensure proper heat conduction.

We keep traces that might carry high currents as short as possible. Also, the design includes space above all components to allow heat dissipation through air.

3.2.9 LEDs

For unidirectional communication with the user three surface-mount LEDs were included in the design, along with two through-hole LEDs for mechanical support. There are two basic options for realizing these diodes. Either a regular LED is driven indirectly via a transistor (bipolar junction transistor (BJT) or field-effect transistor (FET)), or a low-current LED is supplied directly from the microcontroller pin. While the former variant is certainly less restrictive in LED selection and the way to go if the current through the microcontroller is to be kept as low as possible, it is also more complex to design and implement. To keep the LED circuits simple, the latter option is chosen. Following the development board [2], the diodes will be driven negatively, i.e. light will be emitted if the output for the respective microcontroller pin is set to 0.

Any two microcontroller I/O pins are able to sink 8 mA, any four I/O pins are able to sink 15 mA, and any eight I/O pins are able to sink 25 mA [23]. Therefore, it is easily possible to directly drive five of the selected low-current LEDs at a forward current of 2 mA each.

By adding 330 Ω resistors in series with each LED, a forward voltage of 2.65 V on the diodes is achieved, leading to the specified forward current of 2 mA [4, 77]. These resistor values are obtained as $R = U/I = (3.3\,\text{V} - 2.65\,\text{V})/2\,\text{mA} = 325\,\Omega$. Regarding the required wattage rating we compute $P = U \cdot I = 3.3\,\text{V} \cdot 2\,\text{mA} = 0.066\,W$, which confirms that the chosen 0.125 W resistors are suitable.

3.2.10 Pull-up Resistors

To ensure a known signal state at all times, as well as adjust timing characteristics, pull-up resistors are deployed in some circuits [70]. Specifically, we need pull-ups for the SDA and SCL signals, and for both button circuits. This section is based on [5, 31] and all stated equations are from therein, possibly adapted.

I2C

In any digital signal information is encoded as potential changes, called low-to-high or high-to-low transitions. This also applies to the I2C clock line SCL and data bus SDA. While high-to-low transitions are rapid as they are the result of current being discharged through the low-impedance channel of a transistor, low-to-high transitions are slower because they involve current flowing through the pull-up resistor. Therefore, we only consider the signal rise time when calculating the feasible value range of this pull-up. While a lower pull-up resistance leads to faster rise times it also increases current consumption at a low signal. Higher pull-ups mean slower transitions but less current flow.

Another factor influencing the rise time is the total capacitance in the circuit. The biggest source of capacitance on a signal trace are I/O pins of integrated circuit components. Less significant sources of capacitance are nearby signal traces and parasitic package capacitances. These are in the very low pF range and will be considered by adding a margin to the calculated significant total capacitance.

When the product of pull-up resistance and capacitance is too large, the logic-high threshold will not be reached, or the signal will not remain high long enough.

In our I2C circuit there are two chips: the microcontroller and the ToF module. The calculations are done using 'worst' combination of values, i.e. the highest logic-high threshold voltage, the lowest logic-low threshold voltage, the shortest maximal rise time, and the maximum drain current. Table 1 lists all relevant specifications for calculating the pull-up value.

Sym.	Parameter		MCU	ToF	Unit	MCU datasheet reference	ToF datasheet reference
$t_{R:SCL}$	SCL/SDA rise time	max	1000	120	ns	IM21	t_R
C_P	Pin capacitance	max	10	10	pF	p.396	Ci/o, Cin
C_L	Load capacitance	max	400	400	pF	IM50, DO58	C_L
V_{IL}	Input low voltage on SCL/SDA	max	$0.3 \cdot V_{DD} = 0.3 \cdot 3.3V = 1$	0.6	V	DI18	p.24
V_{IH}	Input high voltage on SCL/SDA	min	$0.7 \cdot V_{DD} = 0.7 \cdot 3.3V = 2.3$	1.12	V	DI28	p.24
I_{max}	Current sourced by SDA/SCL	max	12	40	mA	p.357	p.23

Table 1: Specifications required for calculation of I2C pull-up resistor values. The listed parameters are taken from [23, 67].

The total bus capacitance is calculated as the sum of all involved maximum pin capacitances per signal, i.e. two times 10 pF each means 20 pF per signal. To account for all other factors we double that and conservatively assume a total capacitance of $C_{total} = 40\,\text{pF}$ per signal.

To obtain the minimum value of the pull-up resistors, we need to take into account the maximum drain current of 12 mA of the output transistors in each circuit. The second factor here is power consumption when SCL or SDA are logic-low. We choose to limit the current to 3 mA which gives

$$R_{min} = \frac{V_{DD}}{I_{max}} = \frac{3.3\,\text{V}}{3\,\text{mA}} = 1.1\,\text{k}\Omega$$

Calculation of the maximum value for pull-ups amounts to fitting an exponential capacitor charging curve to meet specified rise time requirements. The charging characteristic of an RC circuit is given in Eq. 1. The product $R \cdot C$ is called the time constant τ, which is the time in seconds required to charge the capacitor from 0 to 63.2% of the applied DC voltage. Figure 29 shows the exponential growth and illustrates that a capacitor is approximately fully charged after 5τ.

$$V_C = V_{DD}\left(1 - e^{-t/RC}\right) \tag{1}$$

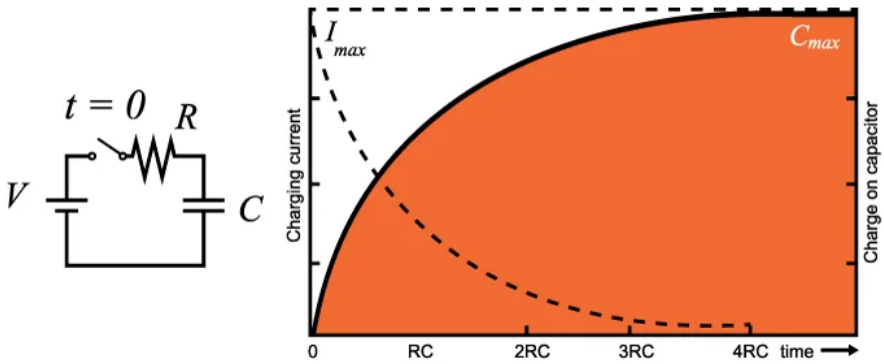


Figure 29: Capacitor charging circuit and characteristics [24])

We are interested in computing R_{PullUp} to obtain a specific rise time, thus we evaluate Eq. 1 for both the time required to reach logic-high as well as for the time required to reach logic-low, rearrange the resulting formulae, and then subtract $t_{LogicLow}$ from $t_{LogicHigh}$. The result can again be rearranged to arrive at Eq. 2.

$$R_{PullUpMax} = \frac{t_{RiseTime}}{Ln\left(\frac{V_{DD}-V_{LogicLow}}{V_{DD}-V_{LogicHigh}}\right) \cdot C_{total}} \tag{2}$$

Inserting the respective values we get

$$R_{PullUpMax} = \frac{120\,\text{ns}}{Ln\left(\frac{3.3\,\text{V}-0.6\,\text{V}}{3.3\,\text{V}-2.3\,\text{V}}\right) \cdot 40\,\text{pF}} \approx 3.02\,\text{k}\Omega$$

To summarize, we need to select pull-up resistors for SDA and SCL that are in the range from 1.1 kΩ to 3.02 kΩ. In favor of faster rise times we selected a final value of 1.5 kΩ, which is closer to the lower bound.

Button Circuits

The pull-up resistors for the button signals were chosen as 4.7 kΩ. Thus, the maximum current flow to ground when pressing BTN or RESET is 3.3 V/4.7 kΩ = 0.7 mA. Additionally, a 1.5 kΩ resistor was placed between the MCU input pin and button BTN to ensure that there is essentially no current flow into that pin. For the required wattage rating we compute $P = U \cdot I = 3.3\,\text{V} \cdot 0.7\,\text{mA} = 0.002\,31\,\text{W}$. This confirms that the selected 0.125 W resistors are more than appropriate.

3.2.11 Buttons and Programmer

The design includes two buttons for user interaction, one reset button which pulls $\overline{\text{MCLR}}$ to ground, and one general purpose button which pulls BTN to ground. Both buttons require pull-up resistors that are covered in Section 3.2.10. The buttons can be used for debugging and to change between the different modes in the final robot. For the general purpose button a capacitor was included for hardware debouncing. After every reset button press the robot is in IDLE mode.

A device reset is performed by pulling $\overline{\text{MCLR}}$ to ground. Then the program starts again from address 0. The reset circuit further encompasses the programmer connector.

The signals to the programmer, $\overline{\text{MCLR}}$, PGD and PGC, were to be kept as short as possible [23]. A forward-biased Schottky diode was put on the supply trace in order to protect the voltage regulator from high reverse voltages caused by the programmer. We use the equation $V - V_{fw} = I \cdot R$ to compute the maximum current through the diode[5]:

$$I = (V - V_{fw})R = 3.3\,\text{V} - 0.32\,\text{V}4.7\,\text{k}\Omega = 0.63\,\text{mA}$$

This current is well below the rated continuous forward current of 200 mA of the chosen diode.

3.2.12 Mechanical Considerations

For mounting the structural elements discussed in Section 3.1.5 we use M2 screws. We selected the tight norm for the through-holes which specifies a hole diameter of 2.2 mm. Apart from keeping bottom signal traces as far away as necessary to prevent any contact with nuts or washers, there are two design options for the surrounding ground plane: (1) simply let the nuts/washers touch the ground plane, or (2) include a copper-free keep-out zone such that the nuts/washers do not touch the surrounding ground plane. For the latter one needs to take into account the involved tolerances and add some extra space. In order to prevent unwanted effects we chose option two.

Initially, we intended for the M2 nuts to sit directly on the PCB and therefore included a 5.3 mm keep-out diameter in the ground plane according to the norm[6]. However, since we ended up using flat washers between nuts and board the keep-out zone should be of 5.8 mm diameter [30].

3.2.13 Final Steps

Before the board was sent off to the manufacturer, Beta LAYOUT, several steps were taken to increase the likelihood of error-free behavior. First, the Electrical Rule Check (ERC) was used to find common errors in the schematic. It detects whether all nets are properly connected and labeled, any I/O conflicts are present, and if there exist overlapping pins and ports. The consistency of schematic and board is checked as well [57]. Second, a Design Rule Check (DRC) was carried out using the design rule file provided by Beta LAYOUT.[7] The DRC checks that any manufacturer-specific constraints on parameters such as clearances, trace widths, component spacing, and hole diameters are met [58]. Lastly, manual reviews of schematic and board were carried out both by all team members individually as well as in a group session.

3.3 Assembly

Once the board design was finished and all electrical components were appropriately selected and dimensioned, the PCB was manufactured. The parts were then carefully

[5] taken from `https://electronics.stackexchange.com/a/422498`

[6] This value was obtained from `https://blogs.mentor.com/tom-hausherr/blog/tag/pcb-mounting-holes/`. While the live version of this website is no longer available, a cached version can be accessed at `http://web.archive.org/web/20150918042430/http://blogs.mentor.com/tom-hausherr/blog/tag/pcb-mounting-holes/`.

[7] `https://de.beta-layout.com/download/spezifikation/deu_Eagle_Dru_Datei_0319.zip`

hand-soldered onto it. Thereby, the two voltage regulators required the additional use of a heat-gun to ensure proper solder distribution under these components. Figures 65 and 66 show the finished board with all components soldered on, top and bottom side, respectively.

Finally, the structural elements could be screwed on along with the motors and IR sensors. Then, the wheels were attached and all three sensors as well as the Bluetooth module were connected. Inserting and connecting the battery followed by some cable management completes the mouse. The fully assembled state is depicted in Fig. 30.

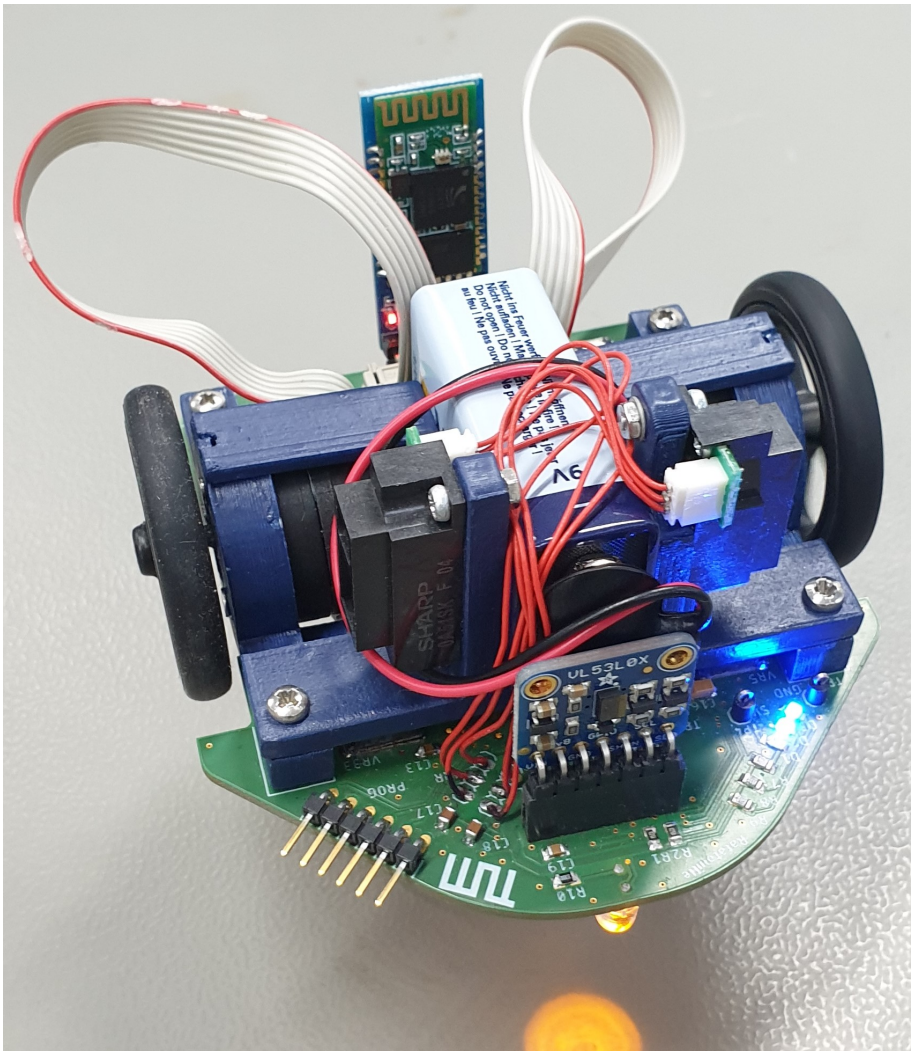


Figure 30: Fully assembled micromouse

4 Software Design

The program on the microcontroller is the brain of our robot. We opted for a modular software architecture in order to facilitate collaboration and to allow for easy exchange of components. Section 4.2 gives an overview.

Configuration-wise, we used PLL to scale the 16 MHz frequency of the external oscillator to 26.6666 MIPS. By setting the Internal External Start-up Option Configuration bit

(IESO), automatic clock switching to the external oscillator after start-up was enabled.

We opted to follow the Common C Interface (CCI) syntax in order to enhance the portability of our code [50]. For example, this simplifies the declaration of the Timer1 interrupt function to `void __interrupt(auto_psv) _T1Interrupt(void)`.

4.1 Peripherals

The microcontroller's peripherals form the foundation of all basic and advanced functionality of our software. They are realized as distinct hardware components within the chip to which certain tasks can be outsourced.

4.1.1 Timer Interrupt

In order to simplify the design of the following embedded control system, the code is designed to run at a fixed cycle time. This would also allow the application of digital filter algorithms for robot localization.

Therefore, we use Timer1 to generate accurate, cyclic events with minimal overhead.[8] Using [16], we set up the timer interrupt. For convenience, we created a function which takes the desired timer period in nanoseconds as argument and, if possible, sets the prescaler (TCKPS) as well as the period register (PR1) for Timer1 accordingly. The timer period in milliseconds can be obtained as

$$T = \frac{T_{CY}}{1e6} \cdot TCKPS \cdot (PR1 + 1) \tag{3}$$

where T_{CY} denotes the instruction cycle period in nanoseconds.

4.1.2 Analog-to-Digital Converter (ADC)

As our selected IR sensors provide an analog voltage output, an Analog-to-Digital Converter (ADC) is needed to transform the analog output into a digital signal, which can then be converted into a distance value in meters. Our microcontroller has an integrated multi-channel 10 bit ADC [23]. The change in voltage required to alter the ADC's output is called quantization error Q. It can be calculated as [18]:

$$Q = \frac{V_{ref+} - V_{ref-}}{2^k} = \frac{5v}{2^{10}} = 4.88\,\text{mV}, \tag{4}$$

where k is the resolution in bits, V_{ref-} refers to GND, and V_{ref+} is the supply voltage of 5V. It means that a voltage range of 4.88 mV is associated with each binary code. Since we have 10 bit resolution, Q is relatively small. Further, the quantization error can influences the quantization noise. For our 10-bit system, the Signal-to-Noise (SNR) ratio in decibel is calculated as [18]:

$$SNR_{DB} = 20 \cdot log(\frac{2^k}{2^0}) = k \cdot 20 \cdot log(2) = k \cdot 6.02\,\text{dB} = 60.2\,\text{dB} \tag{5}$$

[8] It takes four instruction cycles from the time of the interrupt to the execution of the first instruction of the interrupt service routine (ISR), and another three instruction cycles to return from the interrupt.

To make sure we have automatic transfer data from ADC to memory, we set ADDMABM to 1 so that DMA buffers are written in the order of conversion. We use simultaneous sampling by setting SIMSAM to 1, thereby we need to use the 4-channel mode due to $AN4$ and $AN5$ only being available for simultaneous sampling on channels 2 and 3, respectively.

The analogue to digital conversion consists of two stages: sampling and conversion. We set ASAM to 0 since sampling starts not immediately after the last conversion. The sampling time can then be calculated as [18]:

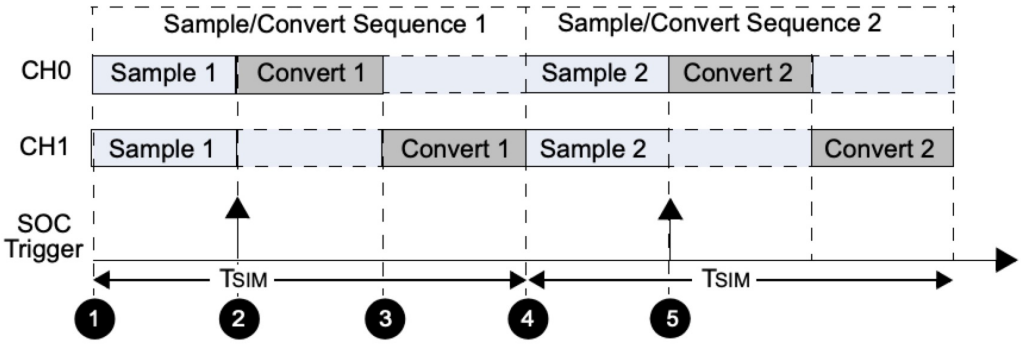
$$T_{SMP} = SAMC < 4:0 > \cdot T_{AD} \tag{6}$$

The Start of Conversion trigger ends the sampling phase and begins the analog-to-digital conversion. We set SSRC to 111 so that an internal counter enables this trigger automatically. Now we have to calculate the conversion time T_{CONV}. With 10 bit operation, it can be obtained as [18]:

$$T_{CONV} = 12 \cdot T_{AD} \tag{7}$$

where T_{AD} is the ADC clock period.

For our micromouse, simultaneously sampling multiple signals ensures that the snapshot of the analog inputs from left and right sensors occurs at precisely the same time for all inputs. Figure 31 illustrates this for the case of 2 channels.



Note 1: CH0-CH1 Input multiplexer selects analog input for sampling. The selected analog input is connected to the sample capacitor.

2: On SOC Trigger, CH0-CH1 sample capacitor is disconnected from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0 is converted to equivalent digital bits.

3: The analog voltage captured in CH1 is converted to equivalent digital bits.

4: CH0-CH1 Input multiplexer selects next analog input for sampling. The selected analog input is connected to the sample capacitor.

5: On SOC Trigger, CH0-CH1 sample capacitor is disconnected from the multiplexer to simultaneously sample the analog inputs. The analog value captured in CH0 is converted to equivalent digital bits.

Figure 31: 2-Channel Simultaneous Sampling (ASAM = 1) [18]

The total time taken to sample and convert multiple channels with simultaneous sampling can be calculated as [18]:

$$T_{SIM} = T_{SMP} + (M \cdot T_{CONV}), \tag{8}$$

where T_{SMP} is the sampling time, T_{CONV} is the conversion time, and M is the number of channels selected by the CHPS$< 1:0 >$ bits.

Currently, the timing settings of the ADC module are conservatively set to high values. Depending on the output resistance of the sensors, these timing characteristics can be optimized in the future in order to achieve the highest possible conversion rate. An important fact here is that a smaller sensor output resistance leads to a shorter charging time for the capacitors of the ADC's S/H units.

After conversion, the results need to be read from the ADC buffer, either by the CPU or by the DMA peripheral (see next section).

In practice, the sensor measurements suffer from a certain amount of noise. As a remedy, a two-stage filtering procedure can be applied. First, the last 2-3 bits of the ADC output are discarded. Figure 54 confirms that this is a reasonable choice. Second, we don't take the most recent cropped measurement directly, but always work with the median value of the last couple of measurements. This effectively removes occasional error spikes.

4.1.3 Direct Memory Access (DMA)

The Direct Memory Access (DMA) peripheral transfers data from data source locations to data destination locations without intervention of the CPU. After the analog-to-digital conversion, we use DMA to transfer multiple ADC readings from the peripheral buffer to the data RAM, which is accessible by the CPU. The DMA peripheral is set to continuous Peripheral Indirect Addressing mode with ping-pong disabled. We configure the peripheral such that the data transfer is automatically initiated by a DMA request. IRQSEL (DMA Peripheral IRQ number) is set to 13 which corresponds to ADC1. Furthermore, we set the start address (DMA4STA) to point to adcData array of integers. Since we use 4 channels in the ADC, DMA4CNT is set to 3.

4.1.4 Inter-Integrated Circuit (I2C)

The ToF sensor on the front of the robot uses Inter-Integrated Circuit (I2C) to communicate with the microcontroller. I2C is a bidirectional, two-line communication protocol often used for intra-board communication. There is a data line (SDA) and a clock line (SCL) on the board. I2C has several modes operating at different speeds (Standard-mode, Fast-mode, etc.) requiring different setup. A lot of different parts can be connected on an I2C bus. Every part is software addressable. There are two roles in I2C: master and slave. Both can transmit or receive data. One can use a single or multiple masters [32].

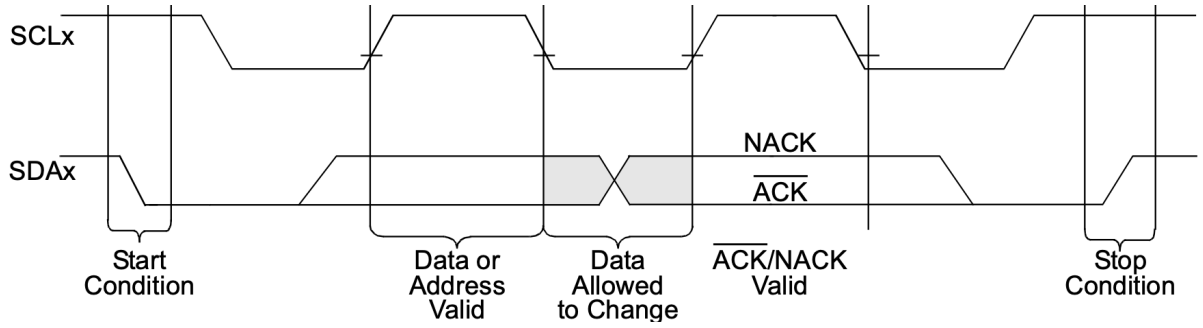


Figure 32: I2C bus data transfer, taken from [15]

For a single master configuration important protocol features are:

- Start condition: SCL is high, SDA high to low transition
- Stop condition: SCL is high, SDA low to high transition
- ACK (Acknowledge): Transmitter releases SDA, receiver pulls SDA low
- 7-bit slave address: Seven bits of data followed by a data direciton bit

The start condition, stop condition and acknowledge are illustrated in Fig. 32.

The dsPIC33FJ64MC804 we are using has an I2C module using two pins as interfaces. These two pins are used as the clock and data line as explained before. The I2C master with 7-bit addressing is supported [23].

The code implemented to enable I2C communication is split up into several functions to ensure maximum flexibility. The function `initI2C` sets all registers to reach the settings we need. With `i2cSTART`, `i2cSTOP`, `i2cRESTART`, `i2cACK` and `i2cNACK` all parts of the I2C protocol are covered. In `getDistanceFront` these protocol step functions are executed in the correct order to get the distance measured by the ToF in the front.

On both the SDA and SCL line we used a pull-up resistor (see Section 3.2.10). In our design we are in single master mode with only one slave which makes the XSHUT line irrelevant.

4.1.5 Universal Asynchronous Receiver Transmitter (UART)

The integrated UART module of the dsPIC33FJ64MC804 microcontroller enables the reception and transmission of data. This communication is full-duplex and asynchronous. By default, a message consists of a start bit, eight data bits and a stop bit. The two lines transmit (U1TX) and receive (U1RX) are needed [19].

We use the UART module to send and receive data over the HC-05 Bluetooth module. This setup serves the communication with a computer to debug and develop our robot. Our simple custom protocol consits of: one byte specifying the datatype, one byte giving the length of the following payload, and the data payload. The baud rate can be calculated with Eq. 3-1 in [19]:

$$BaudRate = \frac{F_{CY}}{16 \cdot (U1BRG + 1)}. \tag{9}$$

We initialize the UART module by setting all registers accordingly in an `initUART1()` function. A ring buffer has been implemented to transmit data asynchronously in the background. With the `txEnqueue()` function we can push data into the ring buffer which gets sent with First In, First Out logic (FIFO). Internally, this function makes use of transmit interrupts to keep the transmit register U1TXREG (and the underlying buffer) filled. This ensures a smooth UART and finally Bluetooth communication with the computer.

In the end we plan to stream the current location with grid cell, x, y and ϕ as well as velocity with the tangential and angular part to the computer for visualization.

During development, the UART connection has also been used for sending commands to the robot. Thereby, it is important to always read the received data, even if unused, to avoid a receive buffer overflow (OERR). The effective receive buffer size is five bytes.

4.1.6 Pulse Width Modulation (PWM)

In order to set different motor velocities and therefore supply the motors with varying power, a pulse width modulation (PWM) can be applied. Instead of an analogue voltage regulation via physical voltage regulators, the power can be a controlled digitally by putting the standard voltage supply of 9 V ON and OFF for a definable time within a fixed period (see Fig. 33).

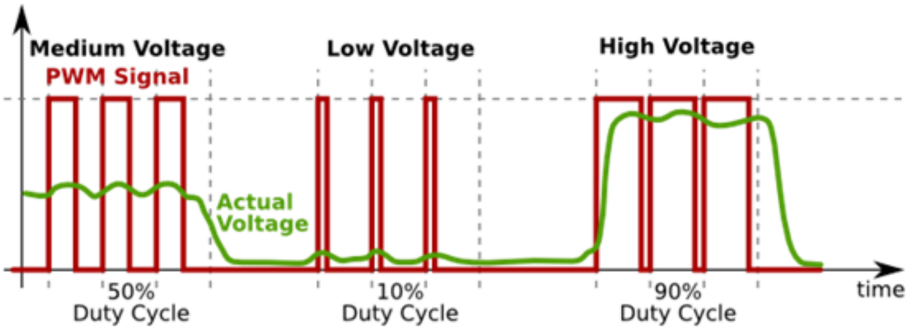


Figure 33: Different duty cycles and resulting voltages with pulse width modulation (PWM) [68]

The frequency of the PWM should not be low because then no smooth average voltage level would be achieved but instead a high fluctuation would occur ('ripple effect') and the load would follow the pulse. Also, below 20 kHz the human ear would notice disturbing noise. A too high PWM frequency would be limited due to low-pass properties of the load and the possible switching frequency of the transistors in H-bridge and MCU which would limit the current flow. Therefore 20 kHz are chosen as a good compromise between frequency and resolution.

Now the period value for the MCU can be calculated by Eq. 10 with the MCU frequency F_{CY} and the desired PWM frequency F_{PWM}. The Prescaler is by default chosen as 1:1.

$$PTPER = \frac{F_{CY}}{F_{PWM} \times PxTMRPrescaler} - 1 = \frac{26.666\,\text{MHz}}{20\,\text{kHz} \times 1} - 1 = 1332 \tag{10}$$

As you can see in Fig. 34, the edge-aligned mode is chosen which creates a step-wise increase of the PWM Timer until the Duty Cycle Register value (P1DCx) is reached. During this time the PWM module outputs a high signal (if it is a high (H) Pin PWMxH). Then it returns to and remains at zero until the end of the period is reached [46].

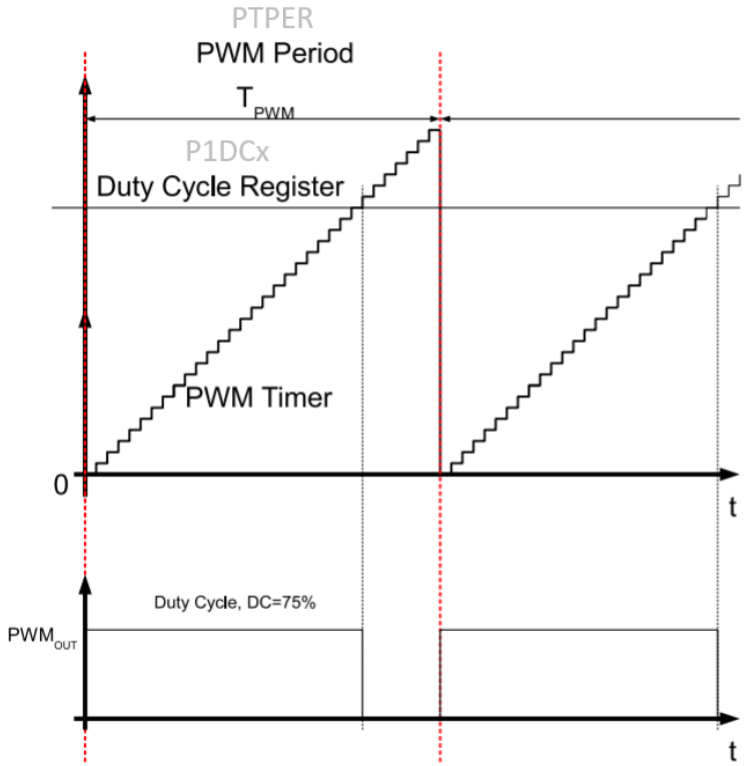


Figure 34: Parameters to set DC and period time for PWM module [42]

In order to configure 100% Duty Cycle the register P1DCx has to be set to $2 \times (PTPER + 1)$ and scaled down accordingly (DC 100% $\Leftrightarrow$ P1DCx = 2666 (9 V); DC = 66% $\Leftrightarrow$ 1777 (6 V)) [46].

Two channels of the same PWM module (PWM1) are used which results in synchronized PWM signals for both PWM pins (by just setting P1TPER once). But there are different duty cycles possible (P1DC1, P1DC2), thus separately controllable velocities for left and right motor are manageable. The PORT and TRIS registers controlling the pin are disabled when a pin is enabled for the PWM output.

4.1.7 Quadrature Encoder Interface (QEI)

The motors have integrated encoders. These encoders generate 16 pulses per revolution. Instead of counting periods, the rising and falling edges can be counted to create a higher sub-resolution by setting the QEI peripheral to 4x mode. With a gearbox of 33:1 reduction ratio, 2112 impulses are sent to the MCU in one revolution. In order to detect the direction of the rotation, two 90° shifted phases (A and B) are provided (see Fig. 35). Thereby, only certain logic combinations are possible when turning forward and backward (e.g. if A is HIGH and B turns from LOW to HIGH, the motor is turning forward). A third signal,

the index slot, gets send only at a certain position of the motor shaft. This feature is not needed in our application.

The POSCNT variable counts the number of impulses from the encoder. A `long` variable is introduced to store the most significant bits in case of an overflow of POSCNT, which occurs after 65536/2112 = 31.03 revolutions which corresponds to approx. 4m of travel. If POSCNT reaches the value held by MAXCNT, a defined QEI-interrupt occurs and the POSCNT value is reset to zero (see Fig. 36) and is added to the long variable. The long variable represents the bits 16 to 47 and the POSCNT register represents bits 0 to 15. Thereby a higher distance can be saved and kept track of (2^{32}/2112 revolutions). For respecting forward and backward rotations it is checked in the interrupt if the POS2CNT value is higher or lower then 65535 in order to detect an under- or overflow (in an overflow the POSCNT starts off again at 0 which is lower than 65535) [17].

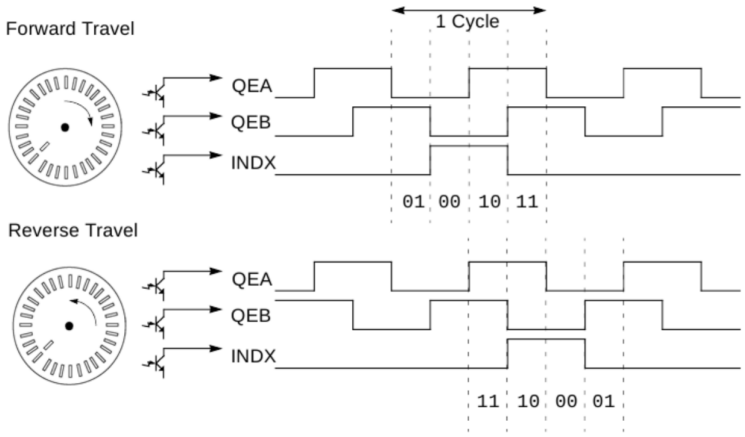


Figure 35: Motor encoder with over/underflow interrupts to detect direction of motor [17]

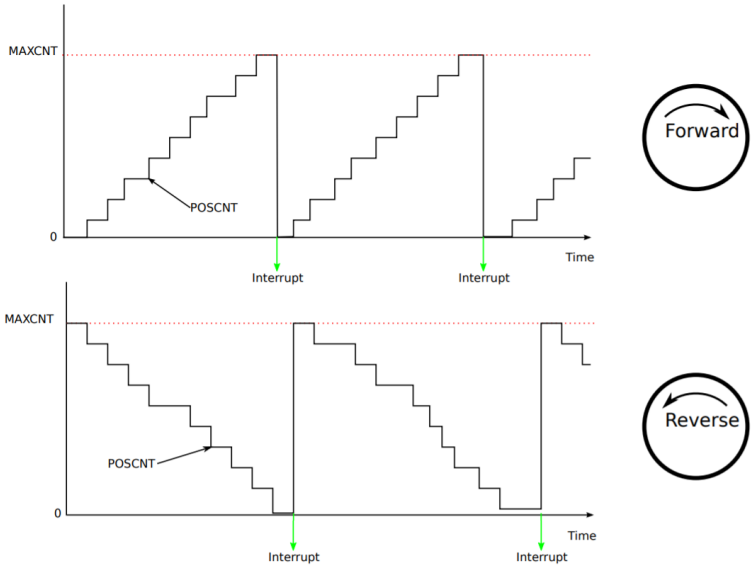


Figure 36: Motor encoder with over/underflow interrupts to detect direction of motor [41]

Since the motors are mounted in reverse position the left motor the QEI output has to be negated for using the same PID logic as for right motor.

4.2 Module Overview

Fig. 37 gives an overview of the key components of the software system. Similar functionality is grouped into modules with separate .h and .c files. It can be seen that the timer interrupt acts as a periodic dispatcher for most processes. The state module keeps track of the robot's state (see Section 4.6). High-level operations such as path and trajectory planning are coordinated by the logic module (see Section 4.8). Lastly, the control module is responsible for reaching the immediate target state or staying on a given trajectory (see Sections 4.3 and 4.7).

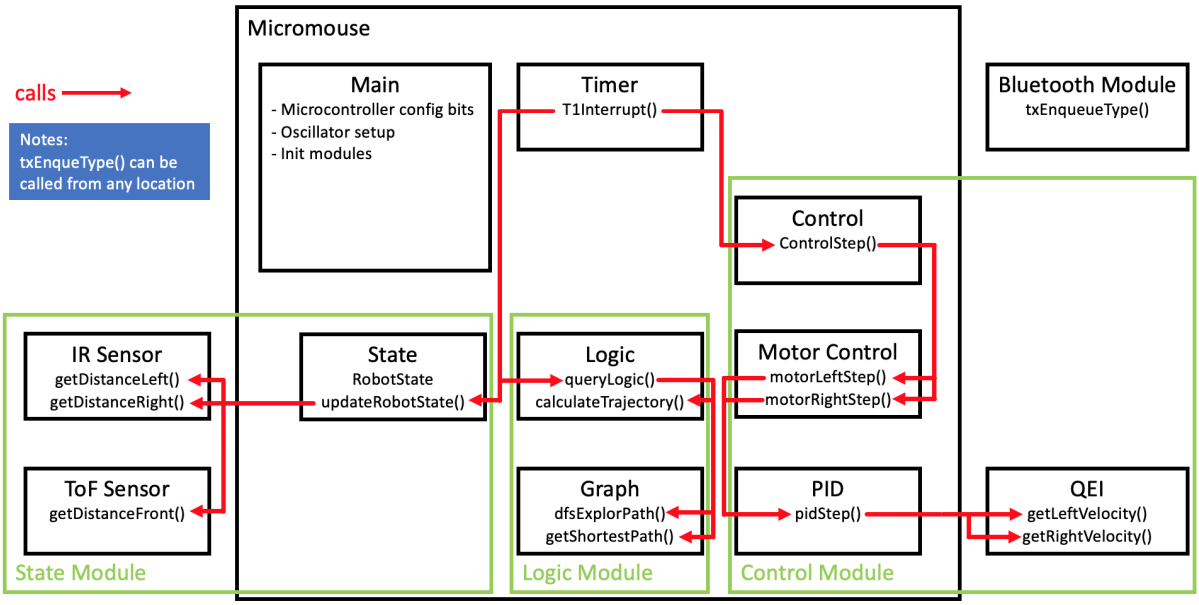


Figure 37: Overview of the major software modules and their structural and functional dependencies. Note that the Logic module interacts with the Control module by updating the global target state and, depending on the pose control strategy, specifying whether the current target state is intermediate or final.

4.3 Controller Design and Motor Modelling

If the mouse model and the environment model were accurately known and there were no disturbances from the environment, an open-loop control without feedback loop would be sufficient.

Since there are several error sources (limited resolution during integration, unequal wheel diameter, variation in the contact between wheel and floor due to variable friction), the mouse needs feedback from the sensors. For the purpose of knowing how fast and how far the mouse has travelled a closed-loop PID control can be implemented.

4.3.1 PID Controller Choice

The goal of a good control is to achieve low (or no) steady-state error, a fast response, and minimal overshoot. These goals cannot be approached independently, instead careful tuning of the control parameters can provide the best performance [10].

The simplest controller is the Proportional-Controller which amplifies the feedback error by a proportional gain value. The advantage is a simple controller setup with reasonably good dynamic behavior [78] as it responds quickly to current errors. A possible disadvantage is the lack of stationary relative accuracy, i.e. the system's output value reaching the reference value for the time t approaching infinity, without an unacceptably large gain.

With the Integral-part zero steady-state error can be achieved as the error of the feedback loop is integrated up and the I-part increases until the error reaches zero. Pure integral control also enables the fading over of high-frequency noise. It is often analogized by 'looking into the past' in order to get a better response. The disadvantage is that it is slow in responding to higher error jumps.

The Derivative-part tries to predict the future and can therefore quickly react to high jumps in the feedback error which his especially suitable for deadband in the system's dynamics. It is commonly used to reduce overshoot in the step response. As a downside the derivative term amplifies high-frequency sensor noise. This is often tackled by a derivative filter [37].

For choosing the right controller two approaches can be used. The first assumes that we have no model of the system's dynamics and we can tune the controller directly on hardware. In our case this is possible since we cannot destroy the system by testing different Gain-values as long as the maximum voltage is limited. The second one depends on simulating a model, tuning the controller in simulation and then applying it on the hardware. Both approaches are shown in the following chapters.

Before actual testing on the mouse, a PI-Controller is assumed to be sufficient for the motors as the motor in combination with the gearbox friction has a dampening effect. This would reduce overshooting and would make a D-controller avoidable which has the benefit of preventing any noise amplification. This assumption shall be tested on the mouse later on.

4.3.2 Control Loop Design and MCU Implementation

Depending on the sensor data received, we can define different states to which the mouse reacts with different actions (state condition = TRUE $\rightarrow$ Action x). For example, the simplest action is to drive straight (= action) with 2 walls on each side (= condition) with a constant speed. Therefore, a velocity control on both motors has to be integrated.

Breaking the control into smaller components, the first approach is to implement a motor control for one motor as depicted in Fig. 38.

As seen in Algorithm 1 the feedback loop takes the current velocity of the motor and subtracts it from the desired velocity (in the unit of encoder counts per 10ms) to calculate the error. This error value is then multiplied with the three PID parts. The proportional part is simply calculated by multiplying the error term by the Kp-gain.

The integral part is calculated by accumulating the error in every interrupt ($error_{aggregated} = error_{aggregated} + error_{previous}$) and multiplying it with the Ki-gain.

Therefore, we save the error in the current interrupt in a variable for use in the next interrupt. There is a limiter module (limiter1) following the I-part in order to limit this value because it can get very large. Furthermore an anti-windup is integrated by clamping (shutting off the further rising of the aggregated error in case of actuator saturation at min and max limit).

The derivative part (has to be tested if necessary) can be calculated by taking the difference between the current error and the previous error which gives a valid approximation of the real derivative since all time steps are the same ($error_{derivative} = error_{current} - error_{previous}$). This also eliminates infinity reaching problems of real derivatives of step responses.

Besides the PID controller (the numbers of for the Kp, Ki and Kd values are arbitrary) there is also a feedforward term which directly, without consideration of the current error, supplies a certain gain for the PWM module for the motor. This can be done if the motor's characteristic curve of PWM & resulting velocity is known. By setting different PWM duty cycles (determined by the P1DCx variable) in a test setup, the resulting velocity of the motor can be determined by counting the encoder impulses per interrupt period.

This resulting curve can then be inverted in order to get the needed relationship of 'desired velocity' as input and 'PWM duty cycle' (P1DCx) as output (see Fig. 58). In case the error turns negative (desired velocity is lower then actual velocity) the motor direction has to be changed (it is actually not changing the direction but the magnetic field is reversed causing to slow down the motor). This is implemented by a simple if-condition. Based on the sign of the PID output we set the DIRA1/DIRA2 and DIRB1/DIRB2 outputs.

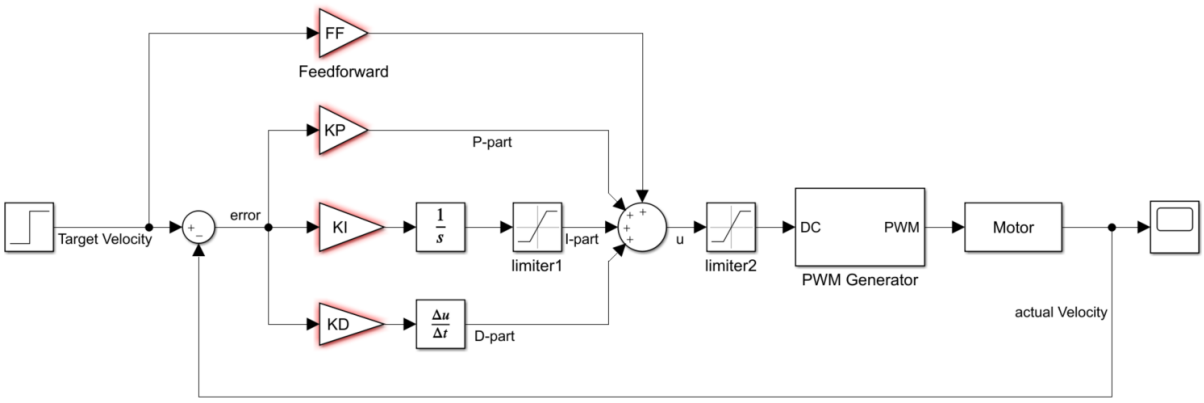


Figure 38: Motor velocity control

As we need for every motor a PID controller we created a struct containing all parameters required for a single PID controller, and then simply use two instances of this struct.

For the update interval, i.e. the frequency at which the encoder sensor is read, commonly used frequencies are $10 - 100$ Hz ($100 - 10$ ms period). We choose 10 ms as it allows a minimum velocity detection of 6 mm/s for 1 impulse in 10 ms as you can see in Fig. 39. This can be inferred from the conversion of impulses to travelled distance. A higher update interval (e.g. 50 ms) can detect lower velocities but does not provide sufficiently fast updates of changes.

Algorithm 1: PID control implementation

input: Actual velocity, desired velocity, PID parameters
output: Total Gain for PWM peripheral (motor)

```
Initialize PID parameters;
Set PID parameters;
while True do
    every timer interrupt (10ms):
    current error = desired velocity - actual velocity;
    Proportional term:
    Pout = current error * Kp;
    Integral term:
    aggregated error += current error;
    Iout = aggregated error * Ki;
    Iout = limiter1 (Iout);
    Differential term:
    Dout = (current error - previous error) * Kd;
    TotalOut = Pout + Iout + Dout + feedforward;
    TotalOut = limiter2 (TotalOut);
    previous error = current error;
    return TotalOut (+ forward, -backward);
end
```

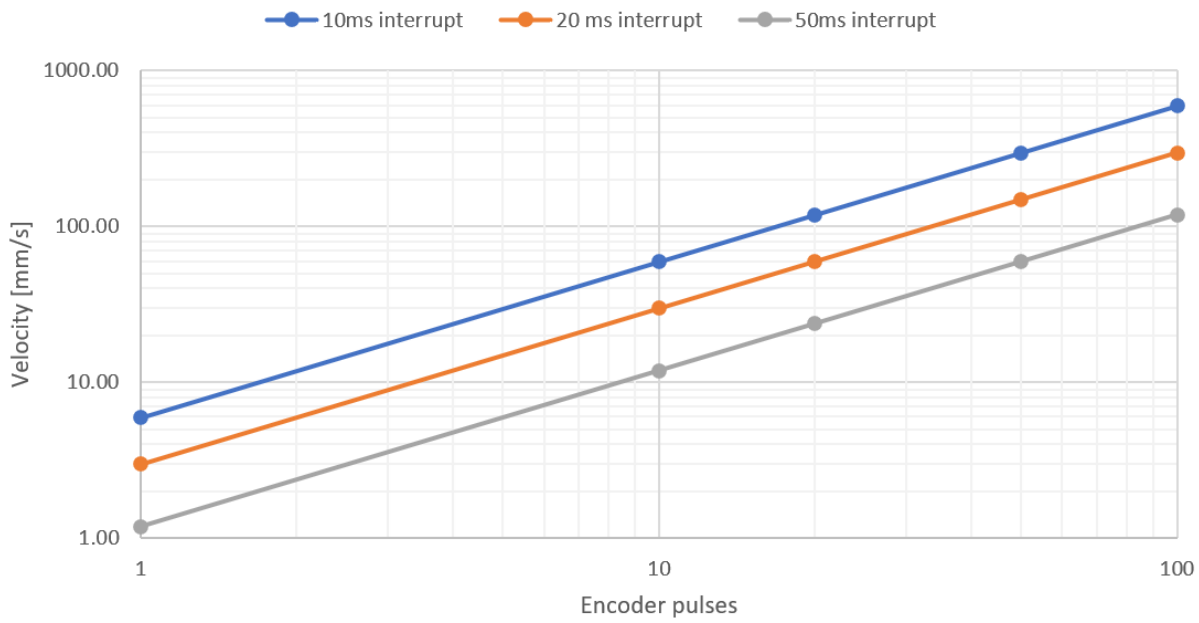


Figure 39: Relationship between update interval for closed-loop control and velocity

After both motor's PID parameters are tuned well (see Section 4.3.3), a driving straight action can be implemented (see Fig. 40). Therefore, the difference between the left and right sensor's reported distance is calculated, forwarded into another PID controller (maybe proportional part is sufficient), and added to the error1 (+) and subtracted from the error2 (-).

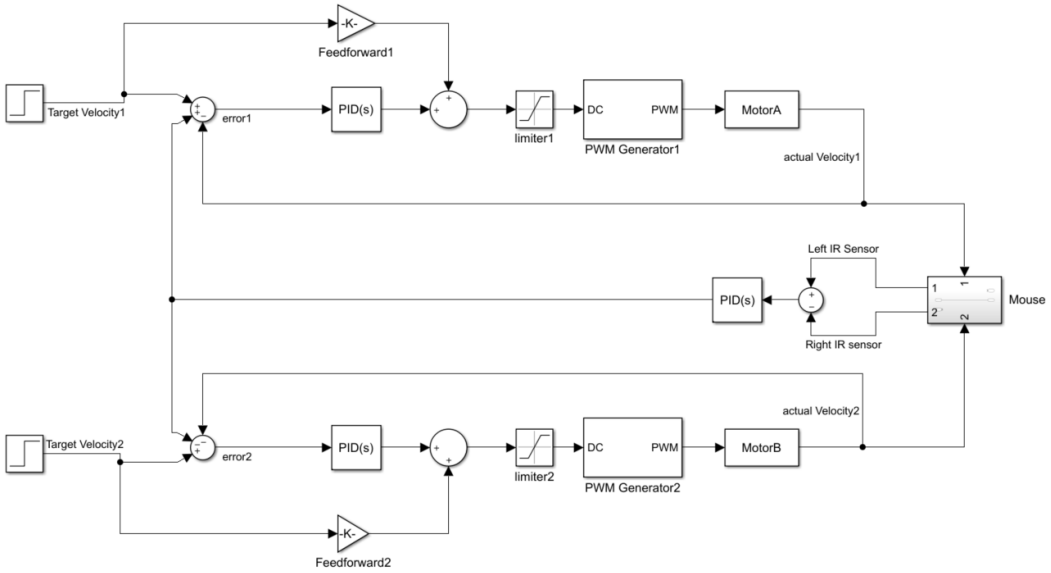


Figure 40: Velocity and driving straight control

Please note that this control loop design does not necessarily match the later explained maze strategy algorithms but shows the iterative testing approach.

4.3.3 Manual PID Tuning Process

For manual PID tuning the Ziegler-Nichols tuning rules are a often used methodology (besides Cohen-Coon). There are disadvantages when applying it on fast responding or second-order systems like overshooting which can be tackled by alternative tuning rules. In the end some fine manual adjustments by instinct might still be needed [48].

To tune the PID gains a step response is applied on the system by setting the desired velocity to a certain value below the maximum value.

Starting with a P-controller the proportional gain is increased until the system's response (actual velocity) is oscillating with a stable (= constant) amplitude and in a regular (= periodical) way. The proportional gain is recorded and further indicated as Ku, the corresponding period duration as Tu. In Fig. 41 the values Ti and Td can then be derived. The term T is the interval in which the PID cycle is called (here 10 ms). The terms Ki and Kd can be calculated according to the following formulae [35]:

$$Ki = Kp * \frac{T}{Ti} \quad \text{and} \quad Kd = Kp * \frac{Td}{T}$$

Rule Name	Tuning Parameters
Classic Ziegler-Nichols	Kp = 0.6 Ku Ti = 0.5 Tu Td = 0.125 Tu
Pessen Integral Rule	Kp = 0.7 Ku Ti = 0.4 Tu Td = 0.15 Tu
Some Overshoot	Kp = 0.33 Ku Ti = 0.5 Tu Td = 0.33 Tu
No Overshoot	Kp = 0.2 Ku Ti = 0.5 Tu Td = 0.33 Tu

Figure 41: PID-tuning rules [48]

4.3.4 Motor Model and Automatic Tuning Process

There are different possibilities to model the motor and gearbox system, either a state-space model can be derived from physical formulae or prebuilt modules can be used, for instance in MATLAB/Simscape.

For those modules the characteristics of the motor have to be derived (see Fig. 42), which can be found in or calculated from the motor datasheet .

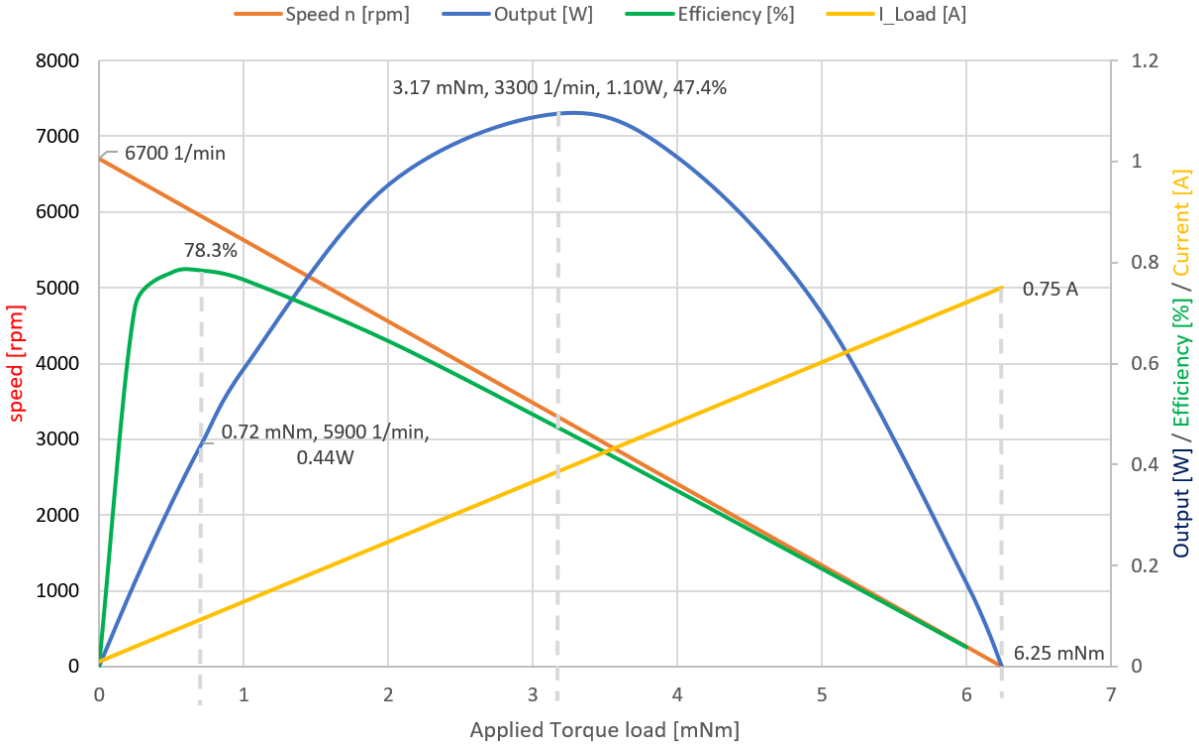


Figure 42: Faulhaber brushed DC motor 006 SR IE2-16 characteristic curve (calculated from datasheet [12, 64])

For the DC motor module, the settings are permanent magnet, model parameterization by stall torque (6.247 mN m) and no-load speed (6700 rpm) (as these values can be calculated from the datasheet [12], see also Fig. 42), 6 V supply voltage, rotor damping parameterization by no-load-current (9.78 mA) and 0.68 g cm^2 rotor inertia. The following simple gear is set to conversion ratio 33 and efficiency 0.6 (as indicated in the datasheet [12]). A Torque source simulates for instance the roll resistance.

The PWM module is set to frequency = 20 kHz, 0% DC = 0 and 100% DC = 1777 which corresponds to an output of 6V. The output is averaged in order to save simulation

effort. For the H-bridge a resistance of 0.5 Ω was considered (see datasheet [63]) and for the free-wheeling diode 0.1 Ω (inferred from similar diodes).

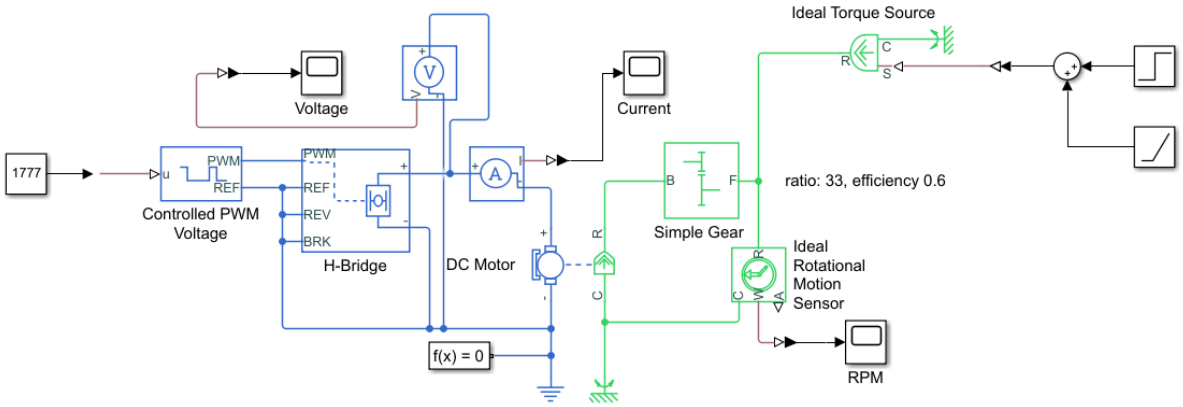


Figure 43: Motor plus gearbox model in Simscape (MATLAB))

To validate the model it is applied with 100% Dutycycle (1777) and an ideal torque on the gearbox shaft in varying strengths (see Fig. 44). As we cannot prove the gearbox behavior (except by hardware identification tests) the gearbox is taken out of the simulation and calculation except for the velocity determination (rpm). You can see that the curves are similar to each other. The slightly different behavior of the simulation can be explained by the PWM and H-bridge module which lower the voltage supply because of voltage drops due to resistances whereas the calculations are considering a nominal voltage of 6 V.

Simulink offers an automatic PID-Tuning ('PID-Tuner'). Therefore a discrete PID Controller has been placed before the system plant (see Fig. 45 bottom right). For the automatic tuning process the system plant has to be linearized at a steady state time which can be selected after simulation (see Fig. 45 left). Then the response time can be set (see Fig. 45 top) to either fast (short duration until the target term is reached but overshoot behavior) or slow (no overshoot but long duration until steady state is reached). Also the transient behavior can be set to either aggressive or robust. At a medium response time and robust transient behavior setting (P-gain = 3.153, I-gain = 163.1) the step response reaches the desired value after 0.3 seconds (zero steady state error) without overshoot (see Fig. 45 center top; the controller effort can be seen below). Note that the conversion is different from the implementation on the MCU as the target and actual value is in rounds per minute and not feedbacked as encoder pulses. In order to be applied to the mouse either the model has to be changed or the mouse has to convert the QEI-values into rounds per minute.

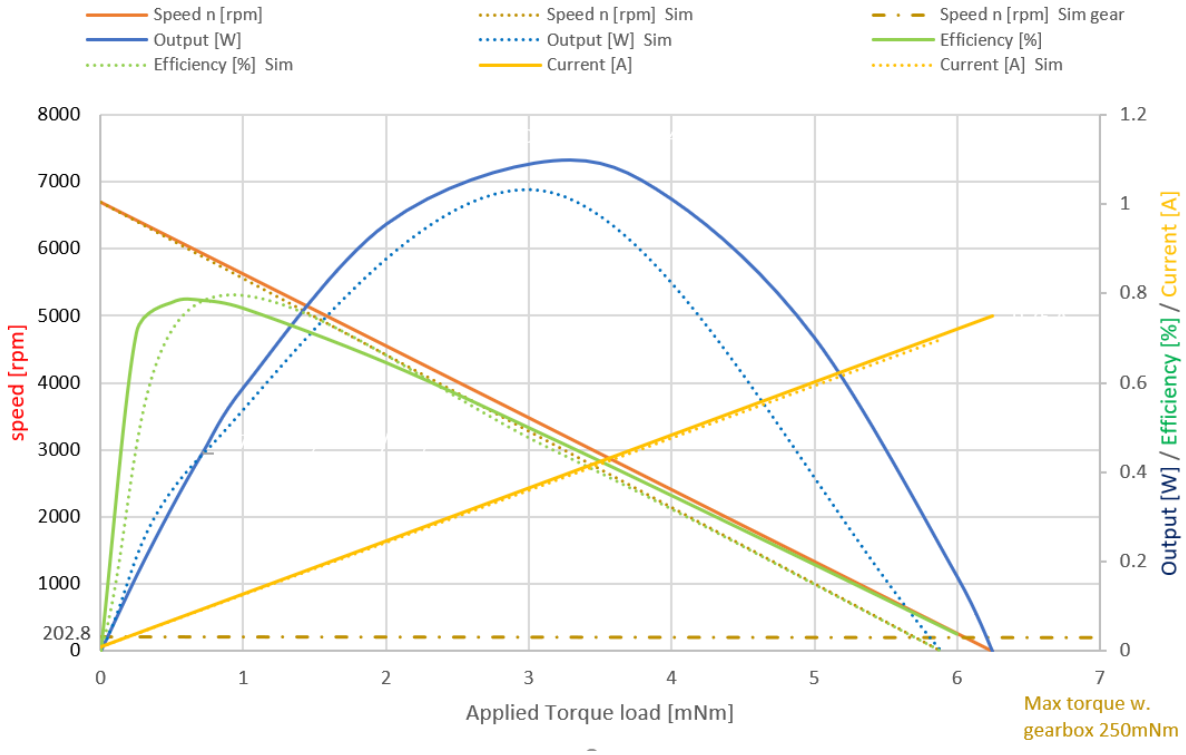


Figure 44: Comparison between motor model and simulation

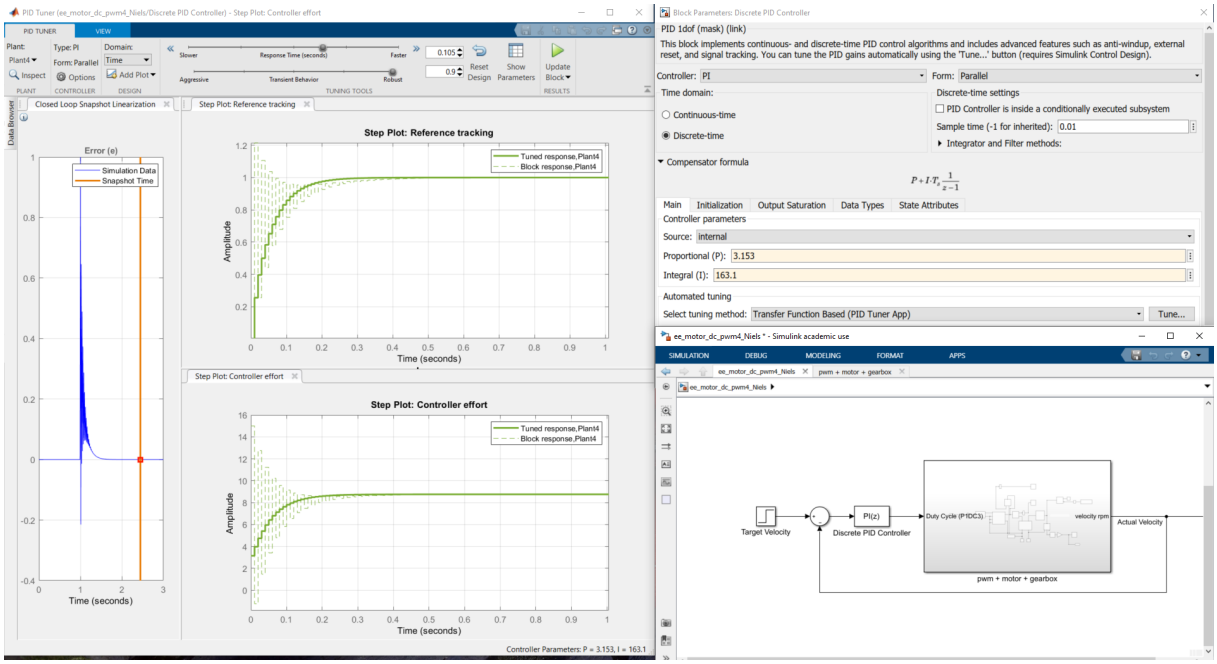


Figure 45: Automatic PID-Tuning-Process in MATLAB Simulink

4.4 Notation

Here, we introduce the necessary notation and underlying assumptions for the rest of the report.

4.4.1 Coordinate Systems and Dimensions

The global two-dimensional coordinate system is defined to have its origin $\{0\}$ at the center of cell 0. The X_0 and Y_0 axis are oriented such that positive Y_0 points toward the exit direction of cell 0, i.e. towards cell 1. Furthermore, we define a robot coordinate system that has its origin $\{R\}$ fixed to the center of the robot, thus it moves with the robot. Its axes are denoted X_R and Y_R, where X_R points in the direction of forward motion. This is illustrated in Fig. 46.

Additionally, various dimensions that are used in the software are defined. These are also listed in Table 2.

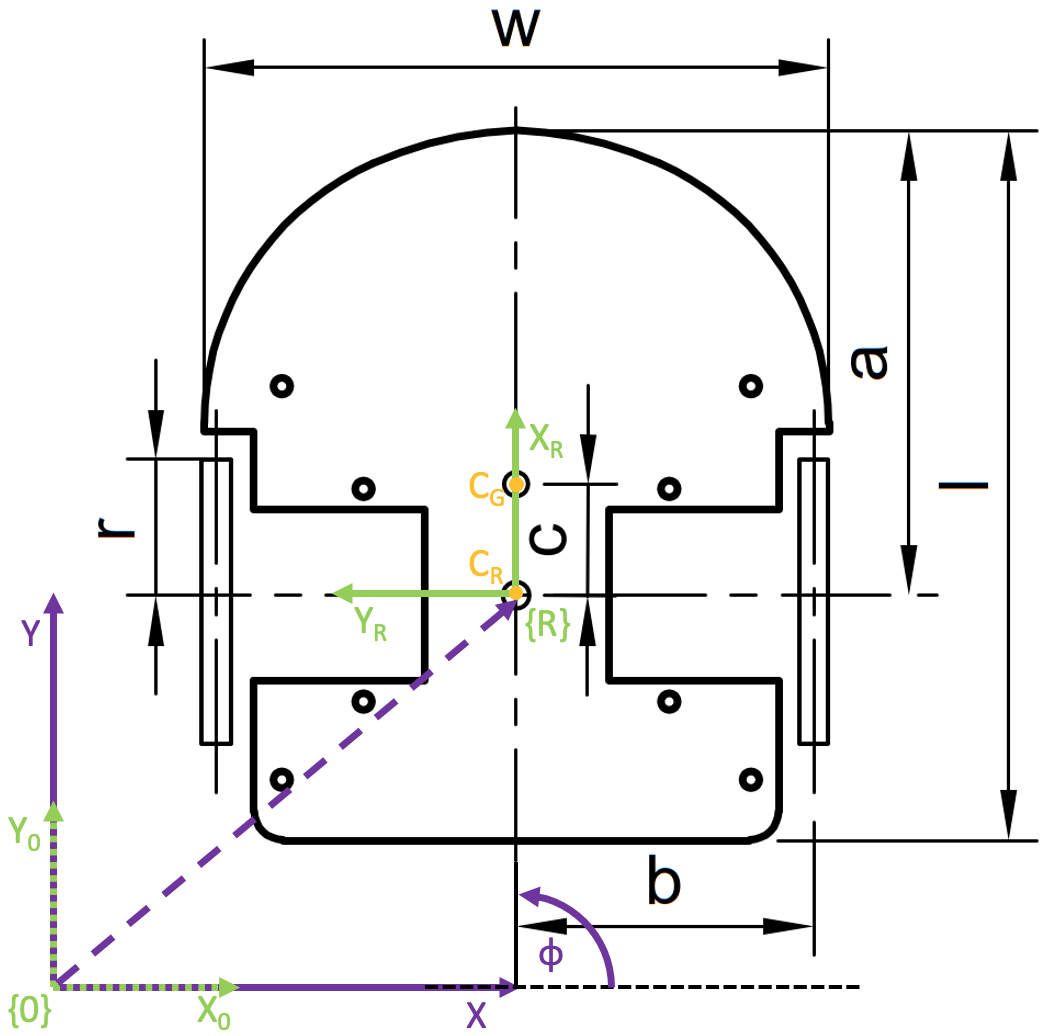


Figure 46: Dimensions (black), coordinate systems (green), and robot state variables (violet)

Description	Symbol	Value	Unit
total robot length	l	92.00	mm
total robot width	w	88.00	mm
wheel radius	r	20.00	mm
distance from CR to front	a	56.00	mm
distance from CR to CG	c	15.00	mm
distance from CR to wheel center	b	41.00	mm

Table 2: Definition of the distances in Fig. 46 with their corresponding values

4.4.2 Robot State

Internally, we represent the robot state $q(t)$ at any given time as the triple of x position, y position, and orientation angle φ,

$$q(t) = \begin{bmatrix} x \\ y \\ \varphi \end{bmatrix}$$

These are defined as follows: x is the distance from $\{0\}$ to $\{R\}$ along X_0, y is the distance from $\{0\}$ to $\{R\}$ along Y_0, φ is the angle in degrees between the x axes measured in counterclockwise direction from X_0 to X_R. Their definitions are indicated in Fig. 46. More formally, φ can also be expressed as [61]:

$$\varphi \coloneqq \text{atan2}(Y_0 \text{ component of } X_R, X_0 \text{ component of } X_R).$$

4.4.3 Maze Representation

Internally, the maze cells are uniquely identified with the numbers 0 through 255. The start area is always defined as 0, the adjacent cell which is not separated from cell 0 by a wall gets number 1, and so on, until 15. Then the sequence continues with 16 for the cell next to cell 0, and the pattern repeats. An example is shown in Fig. 47. Hence, the positive x direction corresponds to going from cell 0 to 16 and the positive y direction corresponds to going from cell 0 to 1. When looking at the maze from above the positive x direction is called 'right' and the positive y direction is called 'up'.

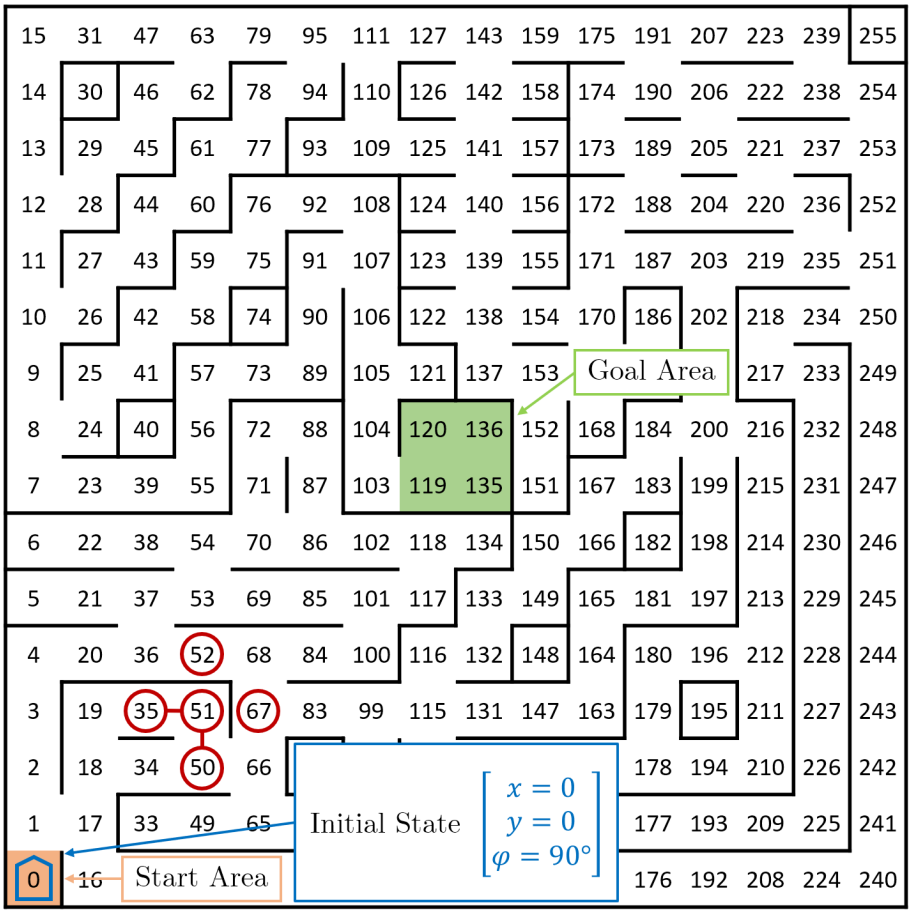


Figure 47: Identification of maze cells, illustration of start and goal area, and indication how the graph elements (nodes, edges) align with the maze

Given a specific cell number the neighboring cells can be determined by simple arithmetic, i.e. adding or subtracting 1 or 16. For example, if we are at cell 51, facing upwards, and we need to compute the cell to the right, then this would amount to $right = 51 + 16 = 67$.

Implementation-wise, the maze is treated as an undirected, unweighted graph. The nodes are implicitly defined as the numbers 0 through 255, corresponding to the cells in the maze. For fast neighbor access we store an adjacency list for each node. For example, the neighbors of node 51 can be found by going through the array `adj[51]`. Refer to Fig. 48 for an illustration. To allow for efficient execution of Algorithms 4 and 2 the graph structure includes two arrays, each storing a value per node. If the memory footprint was an issue, these arrays could be fused. However, it is cleaner to keep them separated.

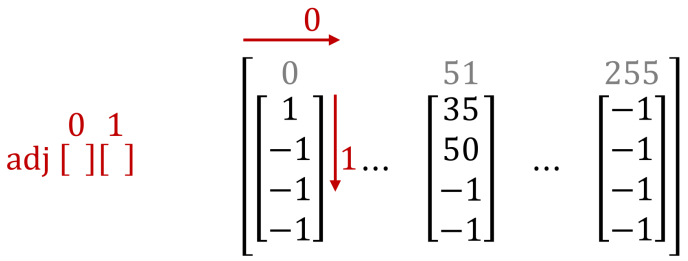


Figure 48: Implementation of the maze topology as adjacency lists

4.4.4 Global States

At the highest level, we distinguish 5 (6 when counting POWEROFF) modes the robot can be in. Fig. 49 gives an overview and specifies the transitions between modes. These can take the form of a button press or fulfillment of a guard condition. For better readability, the basic transitions into POWEROFF and IDLE are only mentioned in the accompanying note.

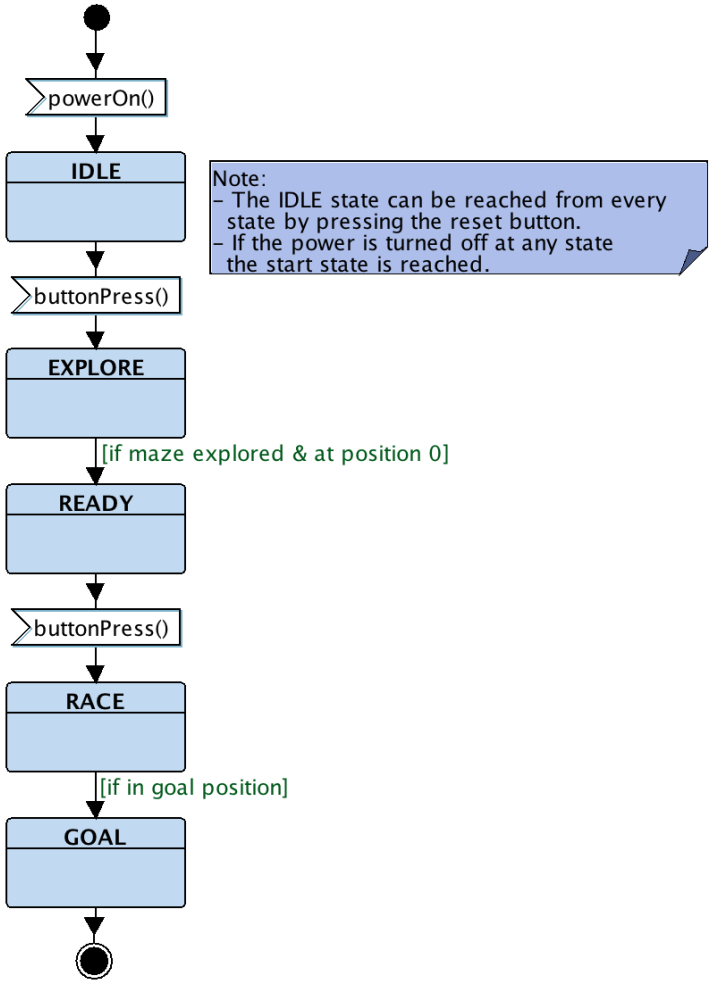


Figure 49: Global modes together with transitions and guard conditions

4.5 Kinematic Model of a Differential-Drive Robot

Kinematics concerns the geometric relationships in the system regarding velocities, without considering the causes of motion. Therefore, first-order differential equations are used. We assume no slip in driving and lateral direction. This section is based on Klančar et al. [36] and all stated equations are from therein.

We adopt the following notation: the distance between both wheels is L $(= 2b)$, the velocities of the wheel centers are v_L and v_R, the angular wheel velocities are ω_L and ω_R, the tangential velocity of the robot is v and its angular velocity is ω. Further, let the instantaneous center of rotation (ICR) be the point around which both wheels have the same angular velocity.

In local (robot) coordinates, the internal kinematics are given by

$$\begin{bmatrix} \dot{x}_R(t) \\ \dot{y}_R(t) \\ \dot{\varphi}(t) \end{bmatrix} = \begin{bmatrix} v_{X_R}(t) \\ v_{Y_R}(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ -\frac{r}{L} & \frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_L(t) \\ \omega_R(t) \end{bmatrix} \tag{11}$$

In global coordinates, the external kinematics can be expressed as

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\varphi}(t) \end{bmatrix} = \begin{bmatrix} cos(\varphi(t)) & 0 \\ sin(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{12}$$

Complementary to the kinematic model, the dynamic model considers the physical causes of motion, i.e. forces, energies, mass, and inertia. Therefore, second-order differential equations are needed. See [36] Eqs. 2.68 and 2.69 for a state-space representation of the combined kinematic and dynamic model. However, for our simple robot we don't take dynamics into account.

4.6 State Estimation

At each timer interrupt the internal state representation needs to be updated to reflect the physical movement of the robot. There are several data sources that may be utilized for this process. Let the current time be t. We first consider only the encoder readings. This section is based on Klančar et al. [36] and all stated equations are from therein, possibly adapted.

To perform odometry, i.e. compute the robot pose from measured tangential and angular velocity, the kinematic model is integrated. This is called forward kinematics. In a discretized setting a numerical integration scheme can be applied when constant v and ω are assumed during the sampling time T_s. Specifically, we use trapezoidal integration:

$$\begin{aligned} x(k+1) &= x(k) + v(k) \cdot T_s \cdot \cos\left(\varphi(k) + \frac{\omega(k) \cdot T_s}{2}\right) \\ y(k+1) &= y(k) + v(k) \cdot T_s \cdot \sin\left(\varphi(k) + \frac{\omega(k) \cdot T_s}{2}\right) \\ \varphi(k+1) &= \varphi(k) + \omega(k) \cdot T_s \end{aligned} \tag{13}$$

Note that v and ω are not measured directly but rather computed from the angular wheel velocities ω_L and ω_R, which are obtained from the encoder measurements. First, we get the wheel center velocities as

$$\begin{aligned} v_L(t) &= r \cdot \omega_L(t) \\ v_R(t) &= r \cdot \omega_R(t) \end{aligned} \tag{14}$$

The angular velocity of the robot can be calculated as

$$\omega(t) = \frac{1}{L} \cdot (v_R(t) - v_L(t))$$

We obtain the instantaneous radius of rotation, i.e. the distance between the robot center and the ICR, as

$$R(t) = \frac{L}{2} \cdot \frac{v_R(t) + v_L(t)}{v_R(t) - v_L(t)}$$

Finally, the tangential velocity is computed:

$$v(t) = \omega(t) \cdot R(t)$$

Additionally, the three distance sensor readings can be taken into account when estimating the new state, in particular the position. The infrared sensors can be used in two ways. First, during state updates we check whether the current orientation angle φ modulo 90° is approximately equal to 0. If additionally the side-facing sensors report readings that suggest the presence of a wall on one or both sides, this distance information is included in the calculation of the next state at $t + 1$. Second, when driving straight, the measurements of the side-facing sensors can be monitored in order to detect the presence of corners on either side. This information can then also be used to correct the internal state representation at $t + 1$. When driving straight, the time-of-flight sensor measurements may be included in the position update as long as there is a wall in front of the robot within sensing range.

In the source code, state estimation is implemented in the function `updateState()`.

4.6.1 Accuracy of Quadrature Encoders

Initially, we navigate based on the encoder readings only. Therefore, it is important to know how accurately, in theory, this can be done. For this purpose, we distinguish between translation and rotation of the robot.

Translation

The counts per wheel rotation are

$$cpw = 16 \cdot 4 \cdot 33 = 2112$$

And we get the distance travelled per count as

$$\frac{d\pi}{cpw} = 0.0595\,mm$$

Rotation
The distance travelled by each wheel when the robot rotates on the spot by 360° is

$$b \cdot \pi = 257.6106\,mm$$

From this we can calculate the counts per full robot rotation as

$$\frac{257.6106\,mm}{0.0595\,mm} = 4329.5899$$

Finally, the resolution in degree per count is obtained

$$\frac{2\pi}{4329.5899} = 0.0015\,rad = 0.0831°$$

We see that the theoretical accuracy of encoder-based navigation is rather high. However, as small errors due to timing delays and physical inaccuracies will inevitably occur and may even accumulate over time, using sensor measurements in addition to encoder readings is indispensable for correct robot navigation.

4.6.2 Alternative Localization Approach

Our design goal is to be able to get a relatively accurate position and velocity signals when controlling the motion of micromouse. However, the velocity estimation that is based on the differentiation of position measurements tends to have a greater error for educational mobile robots. A Kalman filter can be used to improve the velocity estimation with a multi-sensor fusion model and a robot motion model. Since our model is nonlinear, we cannot use a normal Kalman filter (KF), but need to implement the extended Kalman filter (EKF).

After some research, our team decided not to include the Kalman filter approach, since implementing a KF, in comparison with our existing standard approach, will increase the design control complexity but not incrementally improve our localization accuracy [43, 62]. Refer to Appendix F for the necessary mathematics of a KF.

4.7 Pose Control

There are two options for controlling the pose of our robot: control to a reference pose without specifying the path between current and target pose, and trajectory tracking where a target trajectory is computed and the robot is controlled to follow/approach it. Although the former approach is generally more difficult for our case of a nonholonomic robot, we focused on that first because it is easier to compute a sequence of discrete target poses rather than obtaining a twice-differentiable trajectory through all desired waypoints. Nonetheless, considerations were also made regarding the second option. This section is based on Klančar et al. [36] and all stated equations are from therein, possibly adapted.

As mentioned in Section 4.3.2 the control can be decomposed into a feedforward (calculated from the reference pose/trajectory under the assumption that there is no disturbance) and a feedback part (to combat inaccuracies and disturbances). This is called

two-degree-of-freedom control and especially important for nonholonomic systems. Feedforward control is mainly applicable to trajectory tracking and one advantage of it.

Another possibility is to deploy a cascaded control scheme as a way of separating the motor dynamics from the position control. There, a basic position controller that takes the target position per wheel is discussed. In contrast, here we deal with the problem of actually determining these target positions for both wheels when a desired robot pose is given, and further finding suitable target angular wheel velocities for reaching them. This is done implicitly. The position controller computes the desired tangential and angular velocity of the robot, converts them into appropriate angular wheel velocities and passes them to the previously discussed velocity PID controllers.

4.7.1 Control to Reference Pose

In this scenario, the reference pose q_{ref} is defined by a position (x_{ref}, y_{ref}) and an orientation φ_{ref}. The path from the start/current pose $q(t)$ to q_{ref} can be defined explicitly and adapted during the motion, or implicitly. In principle, any feasible path can be chosen, but some are preferable over others in terms of travel time, distance, accelerations, etc. Control to a reference pose can be split into two separate tasks: orientation control and forward-motion control. These cannot be implemented independently.

Orientation Control
The orientation error is computed as

$$e_\varphi(t) = \varphi_{ref}(t) - \varphi(t) \tag{15}$$

From the orientation part of Eq. 11 it follows that our system is of integral nature. Therefore, the control error can be driven to zero by a proportional controller as $\omega(t) = K_\omega \cdot e_\varphi(t)$. If more flexibility in terms of transients shape is desired, we refer the reader to page 65 in [36] for a second-order transfer function.

Forward Motion Control
An important difference between reference pose control and trajectory tracking is that in the former setting the velocity should decrease when approaching the goal, while in the latter the velocity at all times is specified by the trajectory. Thus, a reasonable choice here is a proportional controller that takes the distance from the target location as input. With the position error defined as

$$D(t) = \sqrt{(x_{ref}(t) - x(t))^2 + (y_{ref}(t) - y(t))^2}, \tag{16}$$

this amounts to $v(t) = K_v \cdot D$. Care needs to be taken for very small errors regarding overshooting, as well as very large errors regarding physical limitations such as maximum velocity and acceleration of the robot. One possibility for the latter case is to low-pass filter the controller output.

Simple Feedforward Control
Using inverse kinematics we can compute the feedforward control variables required to

reach a specified pose. While this is generally difficult, a simple solution is to consider translation and rotation separately. Thus, to reach a desired pose, assuming the direct path is unobstructed, the robot would first perform a rotation, then drive straight, and do another rotation. The required control variables $v_L(t)$ and $v_R(t)$ can be calculated by rearranging Eqs. 2.8 and 2.9 in [36] and using the fact that $v_R = v_L$, $\omega = 0$, $v = v_R$ for straight motion and $v_R = -v_L$, $\omega = \frac{2v_R}{L}$, $v = 0$ for rotation only. The actual control variables can then be obtained from Eq. 20 and combined with the respective feedback signal from above.

Combined Orientation and Forward Motion Control
We opted for a control strategy which does not control the robot directly to the reference pose, but to an intermediate direction during the approach. This ensures that we arrive at the target position with the specified orientation φ_{ref}, which is important within the maze so we are correctly oriented for the next trajectory segment.

First, we define the direction from the robot center to the reference point as

$$\varphi_r = \text{atan2}(y_{ref} - y(t), x_{ref} - x(t))$$

Figure 50 illustrates this and all other parameters of this strategy. Hereby, $r > 0$ is a design parameter which influences the curve radius or rather the bulginess of the robot path proportionally. The two depicted angles α and β are of central importance. They are formally defined in Eq. 3.14 in [36]. These angles are always of the same sign and point in the direction of approach.

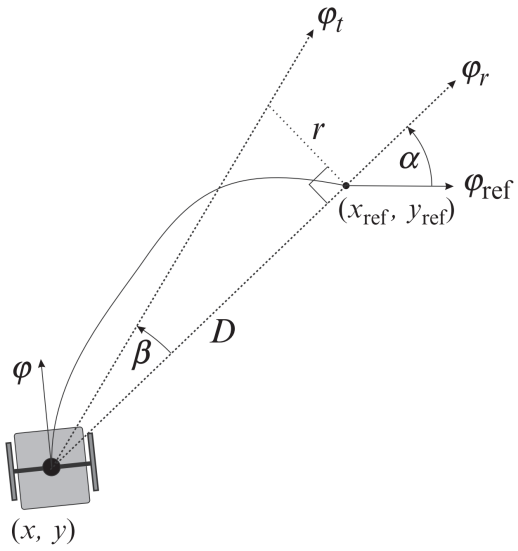


Figure 50: Control to reference pose using an intermediate direction (reproduced from Fig. 3.7 in [36])

In the first phase, large absolute values of alpha indicate that driving straight to the reference point is suboptimal since the orientation error will be significant when reaching it. In this case, we should rather control the robot to an intermediate direction that is shifted away from the reference orientation (towards the current orientation $\varphi(t)$).

This intermediate direction is defined as $\varphi_t(t) = \varphi_r(t) + \beta(t)$. Upon getting closer to the reference point, α decreases and β increases. Once they become equal, we enter the second phase of the control strategy. The target orientation is switched to $\varphi_t(t) = \varphi_r(t) + \alpha(t)$. Note that there is no bump in the trajectory since the two angles are equal at the transition.

The computation of the orientation error is adapted from Eq. 15 as

$$e_\varphi(t) = \varphi_r(t) - \varphi(t) + \begin{cases} \alpha(t) & |\alpha(t)| < |\beta(t)| \\ \beta(t) & else \end{cases} \tag{17}$$

and the position error $D(t)$ is calculated as in Eq. 16. The angular velocity control $\omega(t)$ and the translational velocity control $v(t)$ are commanded via the upgraded proportional relations in Eq. 18. Especially in the start phase when the first control gains are sent and the robot is at a standstill, the velocity and acceleration should be limited. We also need to pay attention when robot drives over the reference point. This is achieved by checking whether the absolute orientation error exceeds 90° and possibly correcting the orientation error by 180° before it enters the controller.

$$\begin{aligned} \omega(t) &= K_\omega \cdot \arctan(\tan(e_\varphi(t))) \\ v(t) &= K_v \cdot D(t) \cdot \text{sign}(\cos(e_\varphi(t))) \end{aligned} \tag{18}$$

These are then combined to compute the wheel center velocities as

$$\begin{aligned} v_L &= v - \frac{L}{2} \cdot \omega \\ v_R &= v + \frac{L}{2} \cdot \omega \end{aligned} \tag{19}$$

and the angular wheel velocities can be obtained as

$$\begin{aligned} \omega_L(t) &= \frac{1}{r} \cdot v_L(t) \\ \omega_R(t) &= \frac{1}{r} \cdot v_R(t) \end{aligned} \tag{20}$$

Keep in mind that while the robot is always controlled to an orientation slightly shifted from the final target orientation φ_{ref} by β or α, this shift is driven towards zero and the robot orientation is forced towards φ_{ref}. When the robot is very close to the target pose zero velocity commands are sent.

This pose control strategy is implemented as `controlStep()` in the source code and is currently set as default.

From Reference Pose to Reference Path Control

Important to note is that the above strategy will bring the robot to a halt at the target state. While this is perfectly fine in the exploration phase (see Section 4.8.2), it is undesired behavior during the racing phase. There, we would like the robot to pass intermediate points on its trajectory with a certain velocity instead of stopping at each point.

One solution to this issue is to lower-bound the target velocity computed by the controller depending on whether there is another target state planned after the current target state. It is, however, not straightforward to integrate the lower bound with tangential and angular velocity calculations. This should to be done in a way that the robot does not get off course. One would likely need a distinction based on the current segment type (straight, left curve, or right curve) as well as on the next segment type. Then the velocity commands can be modified accordingly.

Alternatively, we could check if we are in close proximity[9] to the immediate target state. Then, the gains from the immediate target to next target are calculated and 'blended' with the current gains.

4.7.2 Trajectory Tracking Control

When the desired trajectory is known as a twice-differentiable function of time, trajectory tracking can be implemented to ensure that the robot follows it as close as possible, even in the presence of inaccuracies and disturbances.

In the feedback part there are different possibilities of defining the error, such as expressing it in the global coordinate system or in the local robot frame. In either case, it is common to linearize the error model around some equilibrium point in order to use any linear controller. This process is described in Section 3.3.3 of [36].

For trajectory tracking often a two-degree-of-freedom control is used. Then, in contrast to the reference pose control strategy with an intermediate direction as discussed in Section 4.7.1, the feedback signal does not have to do all the work and smaller control gains are sufficient for reducing the control errors. This is a big advantage of trajectory tracking and makes it more robust to interference.

The feedforward part, using inverse kinematics, calculates the required control gains for the robot to follow the trajectory in a disturbance-free world. This is possible because, under this assumption, the tangential velocity of our differential-drive robot will always be tangent to the trajectory. Refer to [36] Eqs. 3.16 and 3.18 to obtain $v(t)$ and $\omega(t)$. The actual control variables can then be obtained from Eqs. 19 and 20.

We refer the reader to Section 3.3 in Klančar et al. [36] for a comprehensive explanation of trajectory control.

While trajectory tracking has not been implemented yet, we have already devised a strategy for generating the trajectory as discussed in Section 4.8.4.

4.8 Maze Strategy

At the highest level, we implemented the global state machine from Fig. 49, whereby the states therein are referred to as modes. Depending on the current mode, it coordinates all actions of the robot, such as path planning (with graph nodes) and trajectory generation (with robot states). The target state for the position controller is also updated here. Therefore, a custom ring buffer for planned target states is implemented. The logic module further utilizes the abstract and discretized graph representation of the maze (see Section 4.4.3), as well as the the internal robot state representation (see Section 4.4.2).

[9] What it means to be near a state can be calculated based on the control gains, where larger gains correspond to a greater area.

It is queried at every timer interrupt, right after the state estimation and just before a pose control step is performed.

In IDLE mode, the software waits for the user to press the BTN button in order to start the exploration phase. Thus, it is assumed that the robot is located at the start position.

When the robot is in EXPLORE mode, the logic checks if the current state approximately coincides with the target state. If this is not the case, nothing needs to be done. Otherwise, the next element in the buffer is set as target state. If the buffer is empty, the exploration logic is queried for a new node sequence to follow. This sequence is stored in the buffer and the first element set as target state. When the exploration logic signals that there is nothing left to explore, the global state machine transitions to READY.

At this point, the robot is back in cell zero and oriented towards cell one. Then, the shortest path calculation is performed (see Section 4.8.3) and, in turn, the obtained node sequence is passed to the trajectory generator, which computes a sequence of target states. Once the BTN button is pressed, the RACE mode is enabled, the pose sequence is pushed into the empty buffer, and the first element set as target state.

In RACE mode, the states in the buffer are set as target states one by one. Here, additional logic needs to be implemented to achieve a more or less smooth velocity profile. Refer to the end of Section 4.7.1 for possible approaches. Once the final target state has been reached, the robot is located in one of the four goal cells and the state machine transitions to GOAL.

4.8.1 Naive Approach

A simple strategy for driving in the maze is to use predefined motion primitives such as drive straight and turn on spot. Therefore, the basic pose control discussed at the beginning of Section 4.7 can be utilized. Once in the maze, the mouse would select one of the motion primitives depending on the current wall situation.

While this naive approach is probably unsuitable for racing the maze, it could prove as a valuable tool for debugging.

4.8.2 Maze Exploration

Given that the robot will be placed into an unknown maze, i.e. it has no prior knowledge of the specific labyrinth topology, an exploration phase needs to precede the racing phase. The robot has to build up an internal representation of the reachable part of the maze, which can subsequently be used for path calculations and trajectory planning.

To this end, a decision needs to be made at every intersection in the maze. The alternatives are to go left, straight, right, or back. There are a variety of approaches to this task, which can be grouped into local and global strategies.

A most naive example for the former category would be to simply choose randomly at intersections. While the probability for the robot to reach the goal area and/or visit the entire reachable part of the labyrinth would certainly be greater than zero and increase the longer the exploration phase is permitted to run, it is definitely not the most efficient option. Another example of a local strategy are deterministic rules which are applied at each intersection. One of such rule could be to always follow the right wall. If the maze

structure has certain properties, like the goal area not being part of an island, then this method is a feasible and deterministic option for reaching the goal area. However, the micromouse rules don't prohibit the goal area from being an island. A common problem to all local strategies is that they are unaware of already visited cells and therefore bound to be inefficient.

We opted for a global strategy that is conceptually similar to a depth-first search. The robot maintains an internal representation of the maze in the form of an undirected, unweighted graph. The entire logic part of the software works on this discretized structure. This graph is initialized as an empty graph without edges. Each time the absence of a wall is detected the corresponding edge is added to the graph. When a new node is explored, the node from which it has first been reached is set as its parent.

At each node the algorithm checks if any neighbor has not yet been explored, i.e. its parent has not yet been set. If any such node exists, it is set as new target state. Otherwise, the algorithm backtracks along the chain of parent nodes until a node with at least one unexplored neighbor is reached, thereby remembering the intermediate nodes to reach this neighbor. The resulting sequence of nodes is then passed on to trajectory planning and the exploration logic is consulted again only once the target has been reached. This procedure is depicted in Algorithms 2 and 3.

Currently, no optimizations are implemented. A straightforward way of saving time in the exploration phase is to check for any cell c, before exploring it, whether its direct neighbors have all been explored. If this is the case, then cell c does not need to be explored and the robot can move on. Another possible optimization is to compute from any about-to-be-explored cell c the reachable set of nodes $\mathcal{B}$ in the direction(s) of unknown territory and check if that set contains any goal node. If this is not the case and there does not exist a shorter path from cell 0 to cell c within $\mathcal{B}$ than the shortest path from cell 0 to c in the already explored region, then the entire node set $\mathcal{B}$ can be marked as explored because no node in $\mathcal{B}$ will ever be part of any shortest path from cell 0 to the goal area.

Algorithm 2: Explore(s) calculates the path from s to the next unexplored node

input: start node s
output: path p

```
1  Initialize p ← ∅.
2  Let next ← DFS(s).
3  while next has not been found as direct neighbor of s and set of unexplored
     nodes in reachable graph ≠ ∅ do
4  |  Let s ← parent(s).
5  |  Add s to p.
6  |  Let next ← DFS(s).
7  end
8  if next represents the next node to explore then
9  |  Add next to p.
10 end
```

Algorithm 3: DFS(s) performs one step of depth-first search

input: start node s

```
1 for each neighbor nb of s do
2 |   if parent(nb) has not been set then
3 |   |   Set parent(nb) ← s.
4 |   |   Return nb as next unexplored target node.
5 |   end
6 end
```

Algorithm 3 has a runtime complexity in $\mathcal{O}(n)$, where n is the number of nodes. Therefore, Algorithm 2 has a runtime complexity in $\mathcal{O}(n \cdot n)$. The space requirements of the former is constant, while the latter needs $\mathcal{O}(n)$ space. Note that the runtime classification could be done more precisely considering that all graphs we are dealing with contain only vertices of a maximum degree of 4.

The exploration algorithm is implemented as `dfsExplorePath()` in the source code.

4.8.3 Shortest Path Calculation

Assuming that the maze has been explored and a corresponding unweighted graph has been constructed, the shortest path calculation can be done on the abstract graph structure. To this end, we decided for a simple but effective algorithm based on the breadth-first search. Algorithms 4 and 5 depict the detailed procedure. It can be illustratively described as filling the maze with water starting at cell zero, whereby each cell remembers by which cell it has been filled. Once the goal area has been filled the shortest path is determined by tracing backwards via the stored 'parent' cells until the start cell is reached. This technique is called flood-filling [53]. Figure 51a shows part of the shortest path for the maze in Fig. 47.

As stated in [25], Algorithm 4 is an appropriate and effective approach for the use case of a small micromouse robot in a maze. An advantage of the algorithm is that it does not require the maze to have single cell wide passages, thus working well for any given input.

Algorithm 5 has a runtime complexity in $\mathcal{O}(n + m)$, where n is the number of nodes and m is the number of edges. Consequently, Algorithm 4 also has a runtime complexity in $\mathcal{O}(n + m)$. The space requirements of both algorithms are proportional to the size of the graph.

If desired, the algorithm can easily be extended to calculate all shortest paths. Among those the fastest one can be chosen based on criteria such as different styles of turns or diagonal running [52].

The shortest path calculation outlined above applies to the setting of one start node and one target node. As the goal area of the maze consists of four cells, Algorithm 4 is simply executed four times, once with each goal cell as start node. In the end, one shortest of the four calculated paths is selected and passed on to the trajectory planning stage.

The shortest path calculation is implemented as `getShortestPath()` in the source code.

Algorithm 4: ShortestPath(s,e) calculates the shortest path from s to e

input: start node s, end node e
output: path p

Set $depth(s) \leftarrow 0$.
Call BFS(s,d).
Initialize $p \leftarrow \emptyset$.
Let $cur \leftarrow e$.
Add cur to p.
while $cur \neq s$ **do**
    Let $cur \leftarrow$ any neighbor of cur having $depth$ of exactly one less than the $depth$ of cur.
    Add cur to p.
end

Algorithm 5: BFS(s,d) performs recursive breadth-first search

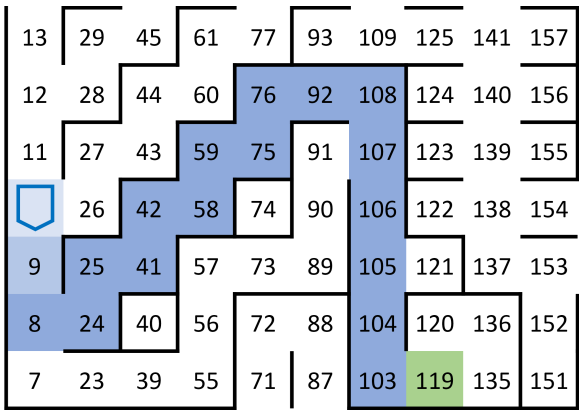
input: start node s, current depth d

for *each neighbor nb of s* **do**
    if $depth(nb)$ *has not been set* **then**
        Set $depth(nb) \leftarrow d + 1$.
    end
end
for *each node nb having* $depth(nb) = d + 1$ **do**
    Call BFS(nb, $d + 1$).
end

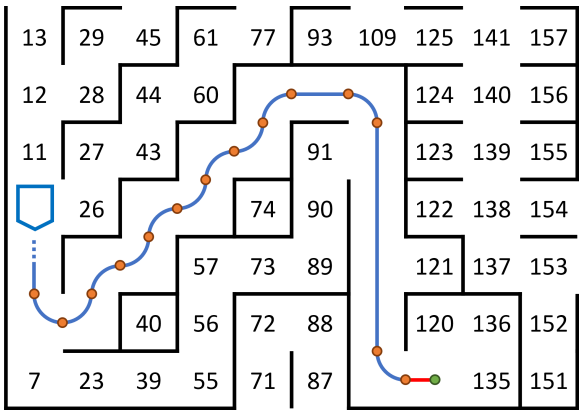
4.8.4 Trajectory Calculation

Given a path as sequence of maze cells, or graph nodes, it is necessary to transform it into a sequence of target poses. The behavior of the robot at any point in the pose sequence is determined by the pose control strategy as described in Section 4.7.

We devised a simple but effective strategy for setting the intermediate states. Except for the first and the last point, all waypoints are located centrally between two adjacent cells. For straight segments (consisting of at least three cells), there are only points at both ends between the first and second cell of the segment, and between the penultimate and the last cell of the segment. For curves (consisting of exactly 3 cells arranged over corner), there are points between both pairs of cells. There are no duplicate points. The first point of the sequence is located in the center of the first cell, and the last point is correspondingly located at the last cell's center. As an example for this path conversion, Fig. 51b shows the resulting sequence of waypoints when taking the node sequence from Fig. 51a as input. Further, the blue path approximately resembles the actual path driven with the pose control strategy outlined in Section 4.7.1.



(a) Path as sequence of graph nodes. The blue nodes mark intermediate path nodes and the green node marks the final node on the depicted path.



(b) Trajectory as sequence of target states. The orange states designate intermediate states and the green state represents to the end state. All target orientations are tangential to the path. Correspondingly, the blue lines show trajectory segments where the velocity is not to be reduced when reaching the end and the red line marks a segment where the robot should stop at the end.

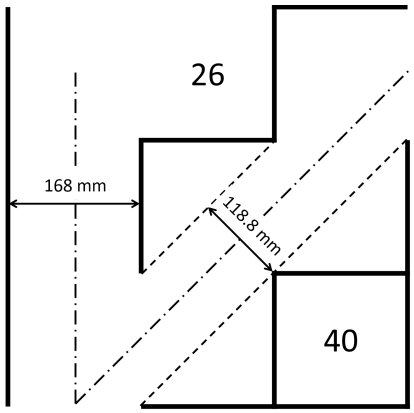
Figure 51: From path to trajectory

Smooth Trajectory via Interpolation

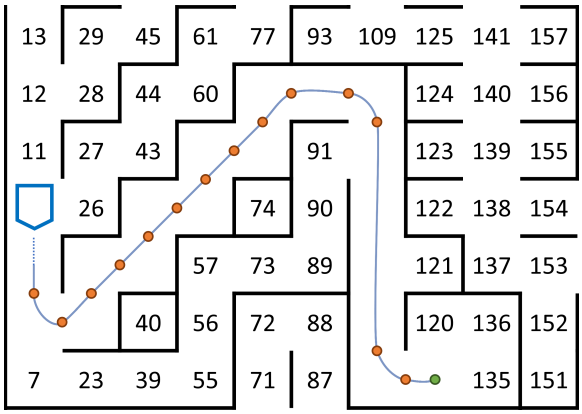
For the future use of trajectory tracking, as introduced in Section 4.7.2, we experimented with different combinations of waypoint sets and interpolation techniques. As it turns out, the sequence of target states computed above works exceptionally well together with MATLAB's `makima` function. Compared with cubic spline interpolation the path is less bulgy in the vicinity of curves and generally remains close to the cell centers on straight passages. More formally, the makima algorithm 'avoids excessive local undulations' [33].

Once the control and odometry modules have achieved a sufficient level of accuracy, we can attempt diagonal driving. As illustrated in Fig. 52a the narrowest diagonal passage is 118.8 mm wide. Hence, our mouse of width 88 mm fits just fine with more than 10 mm on either side when centered.

Figure 52b shows the makima-interpolated trajectory for the example scenario with diagonal driving enabled. It seems to be near-optimal.



(a) The small outer dimensions of our robot, specifically the width of 88 mm allow it to drive diagonally through the maze, as the smallest opening is 118.8 mm wide.



(b) Trajectory computed with MATLAB's makima function, which performs Modified Akima piecewise cubic Hermite interpolation through the orange waypoints. The resulting path is feasible. Compared to a cubic spline interpolation, the trajectory is less 'curvy'.

Figure 52: Diagonal driving and trajectory interpolation

A possible improvement would be to make use of the entire width of the passage before curves. The curve would be entered at the outside edge, nearly touch the inner corner, and exit towards the other outside edge. Thereby, the turn radius would get bigger and the turn could be taken at a higher velocity [72].

Apart from computing the trajectory in terms of target poses, the velocity profile between start and end needs to be determined, i.e. how fast the robot should drive on each section of the trajectory. Important factors to consider here are the maximum acceleration and deceleration of the system. For example, the velocity could be high in straight passages, low for turns and medium during diagonal driving.

5 Testing

To be able to detect faults and to guarantee full functionality of the micromouse robot several tests need to be performed. A first visual inspection by eye and with a microscope did not show any anomalies. Subsequently, we performed a verification of all connections by using a voltmeter on one of the test points (GND, 3.3V and 5V) and on all planned electrical consumers on the board. Everything was connected as expected. To familiarize ourselves with the oscilloscope we built a LC circuit, charged the capacitor and connected

the probes as shown in Fig. 53. Afterwards, we were able to verify the first settings of our microcontroller by testing the oscillator with the oscilloscope.

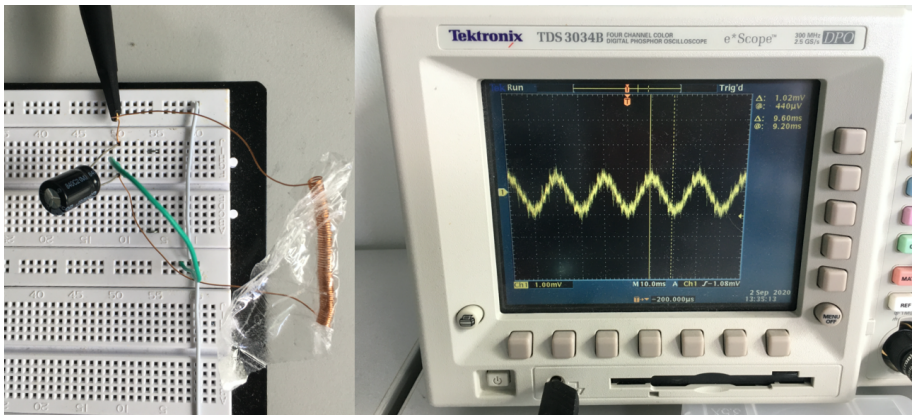


Figure 53: LC circuit with a self made quick fix coil and the oscilloscope measurement

For further software testing the board connected with the PICKIT4 and the MPLAB debugger as well as lines in code to light up a LED were very helpful to find flaws.

5.1 Sensor Tests

Testing the infrared sensors was performed by recording the IR sensor values. The board is supplied via the power supply unit in the lab. The sensor on the mouse, a ruler and the reflective white wall are aligned. The sensor values are streamed through the UART and Bluetooth to MATLAB and noted down along with the distance between sensor and reflective surface as seen in Fig. 54.

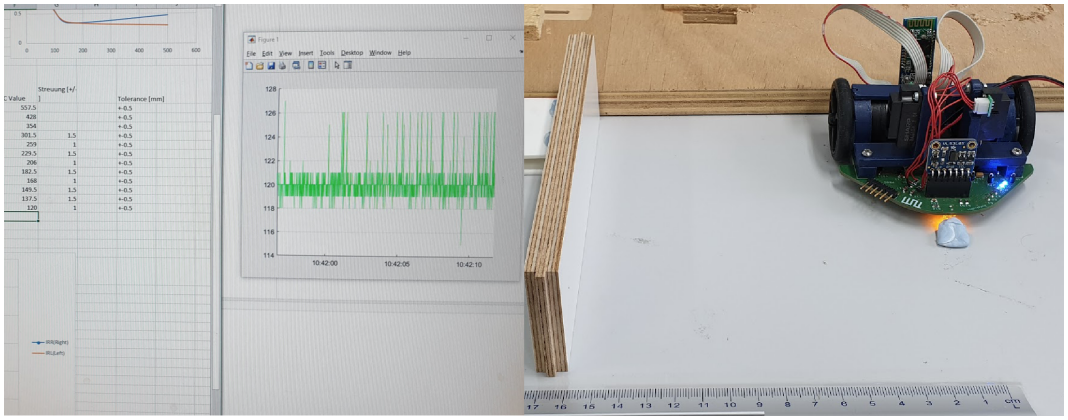


Figure 54: Infrared sensor test setup

In Figure 55 the results are visualized and compared to the expected behavior from [60].

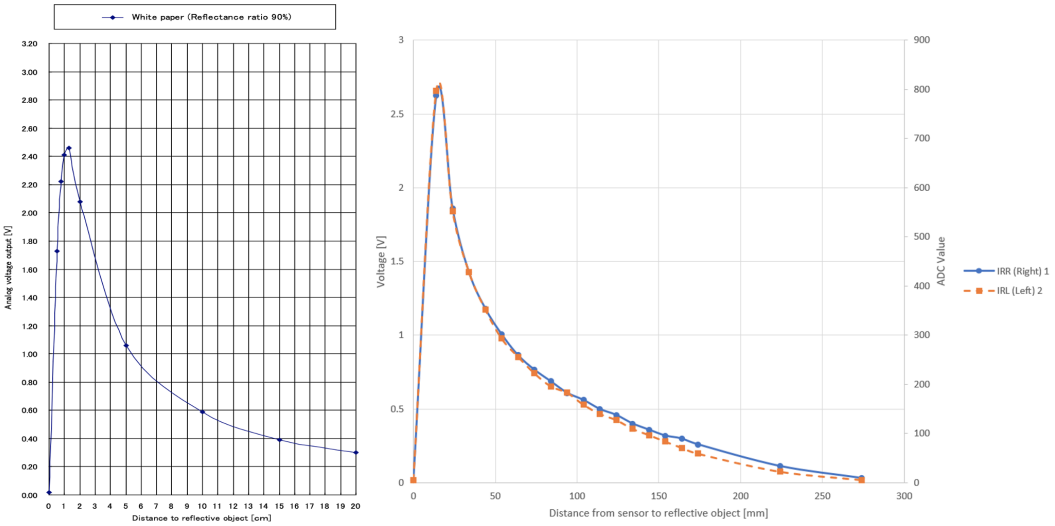


Figure 55: Infrared sensor test results comparison with [60]

To integrate the IR sensors into the code, we created our own conversion function by sensor testing. Thus, the approximate distance (in mm) corresponding to the ADC value can be derived through power regression. The formulae are derived from Fig. 56. Note that only those values are taken into account that are important for the maze case (range from 40mm to 150mm measured from mouse midpoint).

- IR left:

$$distance = 6815.5 \cdot x^{0.776} \qquad R^2 = 0.9981 \tag{21}$$

- IR right:

$$distance = 5580.8 \cdot x^{0.744} \qquad R^2 = 0.9979 \tag{22}$$

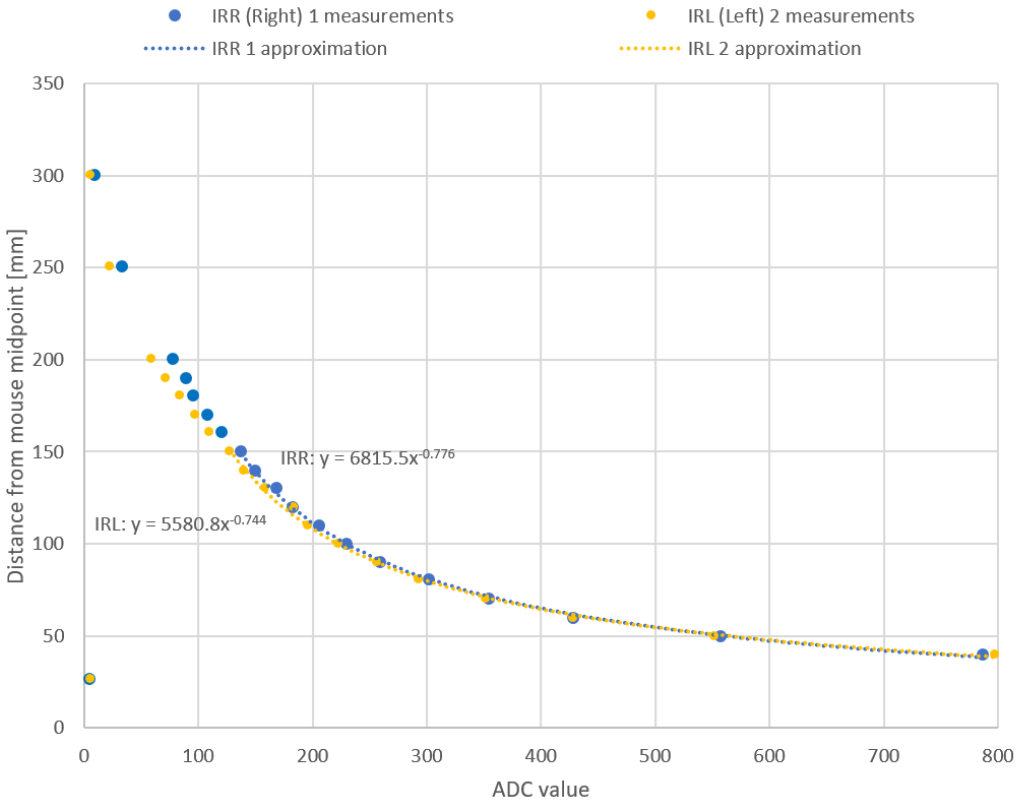


Figure 56: ADC to distance conversion by regression function

To conclude, the infrared sensors work as expected. A measurement with these sensors over 20 cm is not recommended due to the small change in voltage as seen in Fig. 55.

5.2 Communication Tests

One channel of communication from our microcontroller to the user on a computer is enabled by Bluetooth. The string "Micromouse Ratatouille" is enqueued into the ring buffer of the UART serial communication. This is transmitted to the Bluetooth module where a logic analyzer is connected to all pins to check proper function of the ring buffer and the UART peripheral. The channel can be analyzed with Saleae Logic software and as seen in Fig. 57 is working properly. The second step is to plug in the module and connect the computer via Bluetooth. On the computer all Bluetooth traffic can be visualized using e.g. the comman line tool cat or MATLAB. All data is received, our communication works well and can now be used to debug and understand our robot.

Figure 57: UART message received over logic analyzer and visualized with Saleae logic program

5.3 Motor Tests and PID Tuning

For determining the feedforward control, for each Duty Cycle value the pulses per 10 ms are counted and this pair of values is plotted in Fig. 58. The parameters for the feed forward control are calculated from these results as inverse of the proportionality constant. The line through our measured values follows: $DC = 25.356 \cdot Encoder\ pulses/10ms + 16.642$ for Motor A and $DC = 24.258 \cdot Encoder\ pulses/10ms + 14.447$ for Motor B. This is now used to reach the target velocity directly, thus the PI-controller has less 'workload' and can concentrate on counteracting disturbances.

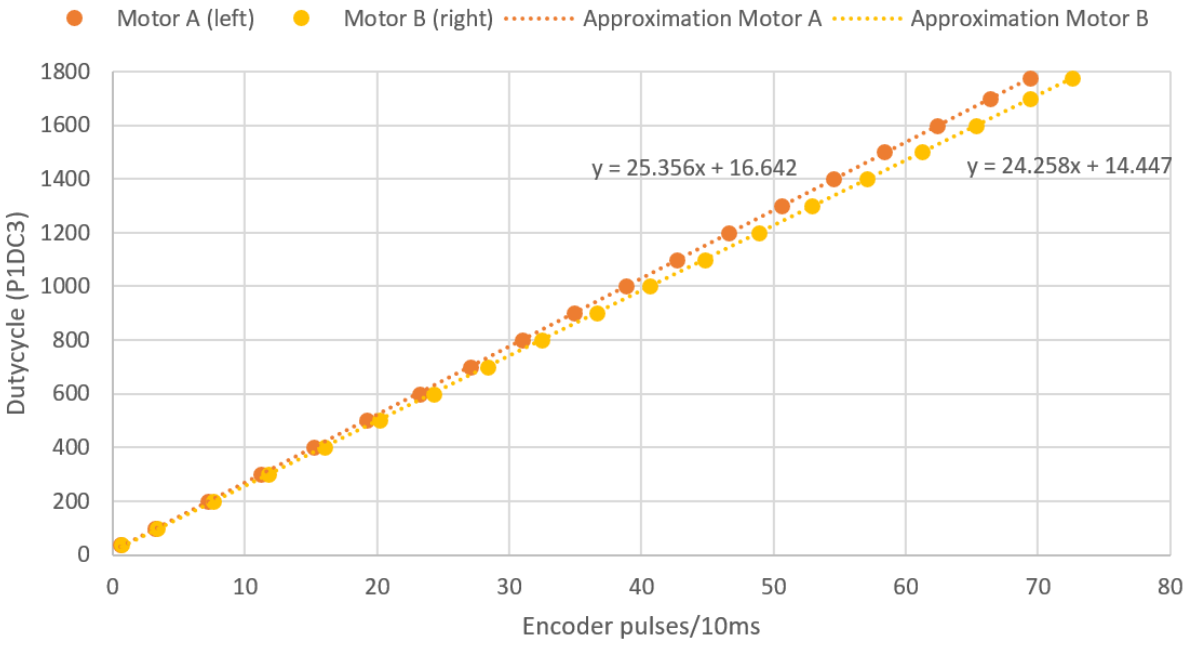


Figure 58: Characteristic curve of the motors from velocity to Duty Cycle

First attempts on PI tuning showed that a P-gain of 5, I-gain of 0.1, limiter1 (-270,270), limiter2 (-1777, 1777) and feedforward (m = 4.88, b = 3.71) for a desired velocity of 110 impulses per 50ms provide a fast response, no overshoot and zero steady state error (note: the 50ms interrupt was set for testing purposes which also needed different feedforward values. In later tests 10ms interrupts are considered). After the motor is stopped for a certain amount of time, the aggregated I-part is high and decreases very slowly, such that

the motor stays above the desired velocity of 110 impulses per 50 ms (see Fig. 59). In the next tests this can be handled by a decay function for the I-part.

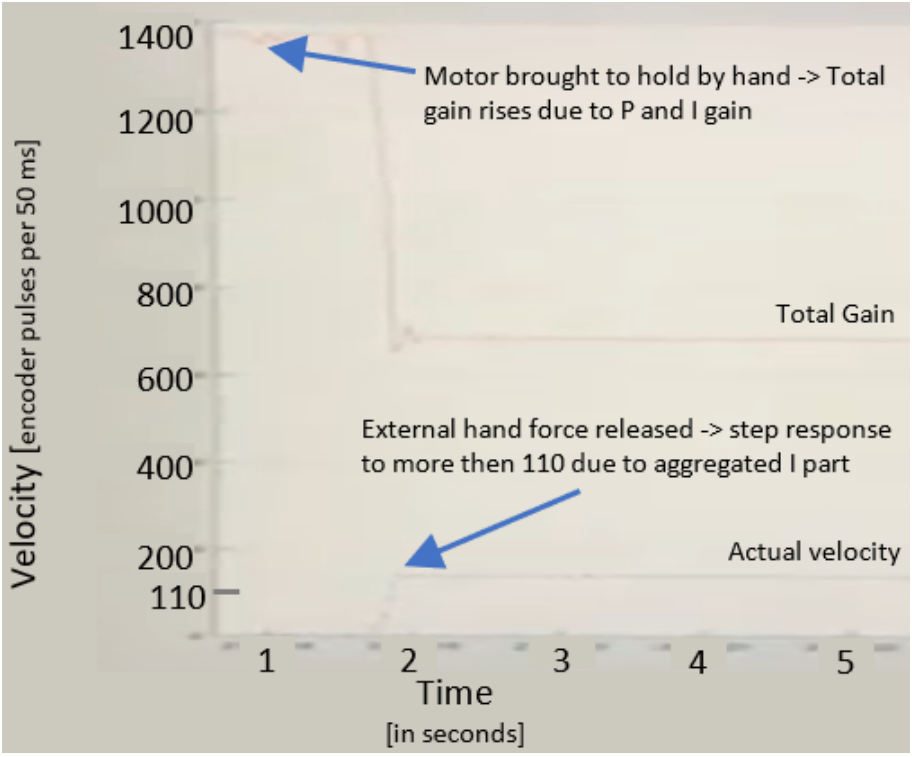


Figure 59: PI Controlling after forced hold of motor

The Tuning procedure of Ziegler-Nichols is planned to be executed to improve the results. When only a P-Controller was applied, a P-gain of 11 lead to a major oscillation

5.4 Navigation Tests

Several qualitative tests were performed to see how our control and navigation approaches are working. More tests are planned to reach our final goal of finishing the maze.

The first verification is a straight line control test with different settings. Driving straight over a specific distance in meters and stopping worked with two walls on the sides, the IR sensor readings and no QEI or PID involved. The test worked fine as well for the case of no walls with QEI and PID to calculate the velocity.

As a second verification turning on spot with a predefined angle in radian with our complete control was performed. This was not possible due to bugs in the code on our microcontroller.

The third verification was designed to proof if the robot could turn after a straight motion. Therefore the target position and orientation is specified. The full available control to reference pose strategy was used. It works with limitations, the final position was reached but the path was rather curvy.

It is intended to clarify several scenarios in the upcoming weeks:

- Turn on spot with a specified angle in radian and simple QEI/PID control

- Curve with specified radius (tangential and rotational motion combined)
- Verifying straight, turn on spot, curve, curve after straight in different maze scenarios: walls present/absent on different sides
- Place mouse in maze and see how different topology is handled: decisions at different types of intersections (2-way,3-way,4-way), behavior in situations such as short/long straight passages, curve after straight passage, diagonal driving
- Does the exploration phase work entirely for different mazes?
- Does the path planning work for different mazes?
- Does the racing phase work as expected with higher velocity?

6 Problems Encountered and Possible Improvements

During the development we encountered some minor problems and discovered some designs which can be improved for the next prototype:

Software

- Output-Pin configuration does not work:

  - Possible reason 1: by enabling the PWM module (`P1TCONbits.PTEN = 1`) the module takes control of the associated pin output drivers, i.e. disables the output driver, and thereby messes up the corresponding TRIS register.

  - Possible reason 2: changing the output mode for pin pairs (e.g. `PWM1CON1bits.PMOD3 = 1`) followed by disabling PWM for a pin (e.g. `PWM1CON1bits.PEN3L = 0`) resets the corresponding TRIS register such that the pin is configured as input [20, 46]; It could be resolved by having a second output pin configuration in the main file.

- Sometimes the robot would not run after programming/reset. Initially, we thought it was due to 40MIPS. When we switched to 26.6MIPS, it seemed indeed fixed but after some time the problem arose again. We found out via debugging through `main()` that it consistently works when a delay between the clock switch (from internal to external) and the next setup code (`initIO()`) is inserted. Therefore, we just put a for loop there and the problem is solved.

- We intentionally left some analog pins free because we anticipated I2C to be difficult and weren't sure we would be able to get it working. If this turns out to be true, then we can simply solder a wire from a third IR sensor to an unused analog pin and get the distance reading without much additional effort. However, we should have been more careful at choosing the analog pins for the sensors because with AN4 and AN5 we now have to use 4 channels of which we ignore 2. For the potentially third IR sensor, we would then use AN0, which is currently converted as one of the 2 discarded channels.

- Out-of-Sync communication with Bluetooth in MATLAB problems were due to the fact that we implemented our own protocol for the UART communication. It is essential for any receiver that they start receiving somewhere between two protocol packets, each consisting of multiple UART blocks. Without any measures for synchronization this is not always fulfilled. Against expectations we discovered that the out-of-sync problem occurs rather frequently. Therefore, we implemented that the robot waits with transmission until it receives a 'start packet' from the communication partner, e.g. MATLAB. Then the other end is guaranteed to receive as first UART block the beginning of a protocol packet. Later on, we could utilize the built-in break sequence (UTXBRK) for a seemless periodic synchronization.
- Time delays in live-MATLAB plot occur probably because the memory gets overloaded after a certain time which is related to the number of signals transferred. If we restart the MATLAB code it works again. Another approach is rather than plot the data, we write it into a file and plot it afterwards.

Hardware

- Battery and IR sensor cables get ripped off from the PCB board. A potential solution is to design a strain relief which is easily implemented by an extra hole where the cables are led through and fixed by cable ties. Furthermore, cable channels or guiding structures would make handling easier.
- The tolerance of the used 3D printer could have been better considered to make sure the M2-screws are fitting through the wholes. This was not a major issue as the screws could be forced in. The form-fit-Pin of the battery bridge had to be rasped a little bit to fit.

Collaboration

- The student version of Fusion 360 did not allow to smoothly work together as the files had to be exported and uploaded to GitHub manually. The integration function between eagle and fusion could also been explored better for a more efficient development process.

Improvement Possibilities

- Swap QEI channels for the left motor such that counting upwards corresponds to forward motion of the mouse.
- Swap motor supply tracks of the left motor such that both motors have the same direction setting for forward and backward motion.
- Include power on/off switch directly on PCB.
- Include a voltage divider on the battery voltage to measure the actual supply voltage via ADC and DMA $\rightarrow$ adjust the PWM cap (which is 2/3 for $9\,\mathrm{V} \rightarrow 6\,\mathrm{V}$) to ensure constant max velocity of the robot

- Create round or a 'triangle connection' instead of the sharp 90° angled edges for the Battery bridge as the parts will last longer.
- Put a mouse-faced cover on top of the robot (advantages would be improved aesthetics as well as keep out dust).

7 Conclusions

The Ratatouille mouse design has delivered all intended functionality so far, such that the maze challenge could soon be tackled with more time in the lab. The remote collaboration with all team members (due to Corona) has not disturbed the quality of the project. The different disciplines united in the team (Informatics, CAD, 3D Printing, Physics) and characters (one for perfectionism, one for pushing the timeline, one for try out without much planning, one for research the state of the art) allowed for a well-distributed workload. One of the success factors was a fast exchange possibility within the online version control platform GitHub [28]. For a better debugging process a change and outcome list to know what leads to certain behaviour would have helped.

It was important to celebrate small milestones (LED blinks for the first time, Motor is moving, Bluetooth Communication shows graph plot, etc). because the next issue will show up and requires energy and perseverance.

In order to immediately start the maze race after soldering, the software should have been implemented earlier instead of starting it once the robot was ready to be used. For instance, the communication between the PC and the Bluetooth module would have helped immediately for debugging.

For the testing of the micromouse it was crucial to divide the test steps into smaller separated steps to capture what is working and what not.

All in all, the project opened up a lot of 'black boxes' (e.g. PCB design) and the team could apply a lot of content from lectures in real life.

Bibliography

[1] *16-Bit 28-Pin Starter Development Board User's Guide*, DS51656B, Rev. B, Microchip Technology Inc., Dec. 2008. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/51656B.pdf.

[2] *16-Bit 28-Pin Starter Development Board User's Guide*, DS51656B, Microchip Technology Inc., 2008. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70000195g.pdf.

[3] "AN-1797 TO-263 THIN Package," *Texas Instruments, Application Report SNVA328A*, p. 4, 2013. [Online]. Available: https://www.ti.com/lit/an/snva328a/snva328a.pdf.

[4] *APTD2012LVBC/D: 2.0 x 1.25 mm SMD Chip LED Lamp*, DSAO7912 / 1203015436, Rev. 4B, Kingbrigh, Aug. 2019. [Online]. Available: https://www.mouser.de/datasheet/2/216/APTD2012LVBC-D-1101043.pdf.

[5] R. Arora, "I2C Bus Pullup Resistor Calculation," *Texas Instruments, Application Report SLVA689*, p. 5, 2015. [Online]. Available: https://www.ti.com/lit/an/slva689/slva689.pdf.

[6] B. de Bakker. "IR Sensor Principle." (n.d.), [Online]. Available: https://www.makerguides.com/sharp-gp2y0a710k0f-ir-distance-sensor-arduino-tutorial/.

[7] Bladuino. "How to Connect HC-SR04 Ultrasonic to Bladuino Pro Mini." (2014), [Online]. Available: http://bladuino.blogspot.com/2014/08/how-to-connect-hc-sr04-ultrasonic-to.html.

[8] D. Christiansen. "The Amazing MicroMouse Contest." (2014), [Online]. Available: https://spectrum.ieee.org/consumer-electronics/gadgets/the-amazing-micromouse-contest. Accessed: September 2020.

[9] E. Coates. "Power Supply Basics." (2020), [Online]. Available: https://learnabout-electronics.org/PSU/psu10.php. (Accessed: September 2020).

[10] D. Collins. "Motion Control Tips." (2017), [Online]. Available: https://www.motioncontroltips.com/faq-how-are-pid-parameters-configured-for-variable-frequency-drives/. Accessed: September 2020.

[11] Components101. "HC-05 - Bluetooth Module." (2018), [Online]. Available: https://components101.com/wireless/hc-05-bluetooth-module. Accessed: September 2020.

[12] *DC-Gearmotors: Series 2619 SR IE2-16*, Dr. Fritz Faulhaber GmbH & Co. KG, Feb. 2020. [Online]. Available: https://www.faulhaber.com/fileadmin/Import/Media/EN_2619_SR_IE2-16_DFF.pdf.

[13] A. Dennis, B. H. Wixom, and D. Tegarden, *Systems Analysis and Design with UML*, 4th. Wiley Publishing, 2012, ISBN: 1118037421.

[14] "Electromagnetic Compatibility (EMC), DIN EN 61000-2-2 VDE 0839-2-2:2020-05, German version," VDE VERLAG, Berlin, DE, Standard, May 2020.

[15] *dsPIC33F/PIC24H Family Reference Manual Inter-Integrated Circuit (I2C)*, DS70000195G, Microchip Technology Inc., 2015. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70000195g.pdf.

[16] *dsPIC33F/PIC24H Family Reference Manual Section 11. Timers*, DS70205D, Rev. D, Microchip Technology Inc., Sep. 2011. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/70205D.pdf.

[17] *dsPIC33F/PIC24H Family Reference Manual Section 15. Quadrature Encoder Interface (QEI)*, Microchip Technology Inc., 2010. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70208B.pdf.

[18] *dsPIC33F/PIC24H Family Reference Manual Section 16. Analog-to-Digital Converter (ADC)*, DS70183D, Rev. D, Microchip Technology Inc., Apr. 2012. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70183D.pdf.

[19] *dsPIC33F/PIC24H Family Reference Manual Section 17. UART*, DS70188E, Microchip Technology Inc., 2012. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70188E.pdf.

[20] *dsPIC33F/PIC24H Family Reference Manual Section 30. I/O Ports with Peripheral Pin Select (PPS)*, DS70190E, Rev. E, Microchip Technology Inc., Apr. 2012. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/70190e.pdf.

[21] *dsPIC33F/PIC24H Family Reference Manual Section 39. Oscillator (Part III)*, DS70216D, Rev. D, Microchip Technology Inc., Apr. 2012. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70216D.pdf.

[22] *dsPIC33FJ64MC804*, Microchip Technology Inc., n.d. [Online]. Available: https://www.microchip.com/wwwproducts/en/dsPIC33FJ64MC804.

[23] *dsPIC33FJ64MCX02/X04: 16-bit Digital Signal Controllers (up to 128 KB Flash and 16K SRAM) with Motor Control PWM and Advanced Analog*, DS70291G, Microchip Technology Inc., 2012. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/70291g.pdf.

[24] L. EETech Media. "Capacitor Charge and Time Constant Calculator." (n.d.), [Online]. Available: https://www.allaboutcircuits.com/tools/capacitor-Charge-and-time-constant-calculator/. Accessed: October 2020.

[25] I. Elshamarka and A. B. S. Saman, "Design and Implementation of a Robot for Maze-solving Using Flood-fill Algorithm," *International Journal of Computer Applications*, vol. 56, no. 5, 2012.

[26] Formlabs. "Design for Manufacturing With 3D Printing." (n.d.), [Online]. Available: https://formlabs.com/blog/design-for-manufacturing-with-3d-printing/. Accessed: September 2020.

[27] M. GmbH. "Ultrasonic Technology: Ultrasonic Principle." (n.d.), [Online]. Available: https://www.microsonic.de/en/support/ultrasonic-technology/principle.htm.

[28] T. Goldfuss, R. Gu, N. Schlegel, and M. Schonger. "Ratatouille GitHub Repository." Private due to sensitive information. (2020), [Online]. Available: https://github.com/mimouse/main. (Accessed: October 2020).

[29] GRobotronics. "Adafruit VL53L0X Time of Flight Distance Sensor - 30 to 1000mm." (n.d.), [Online]. Available: https://grobotronics.com/adafruit-vl53l0x-time-of-flight-distance-sensor-30-to-1000mm.html?sl=en.

[30] T. Hausherr, "PCB Design Optimization Starts in the CAD Library," IPC APEX EXPO 2012, 2012. [Online]. Available: https://www.mikrocontroller.net/attachment/364589/PCB_Design_Optimization_Starts_in_the_CAD_Library.pdf.

[31] M. Hughes. "I2C Design Mathematics: Capacitance and Resistance." (n.d.), [Online]. Available: https://www.allaboutcircuits.com/technical-articles/i2c-design-mathematics-capacitance-and-resistance/. Accessed: October 2020.

[32] *I2C-bus specification and user manual*, UM10204, NXP Semiconductors, Apr. 2014. [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[33] T. M. Inc. "Modified Akima piecewise cubic Hermite interpolation - MATLAB makima." (2020), [Online]. Available: https://www.mathworks.com/help/matlab/ref/makima.html. (Accessed: October 2020).

[34] Jippie. "What is a Voltage Regulator." (2015), [Online]. Available: https://electronics.stackexchange.com/a/186091.

[35] R. Keim. "Embedded PID Temperature Control: Ziegler–Nichols Tuning." (2016), [Online]. Available: https://www.allaboutcircuits.com/projects/embedded-pid-temperature-control-part-6-zieglernichols-tuning/. (Accessed: September 2020).

[36] G. Klancar, A. Zdesar, S. Blazic, and I. Skrjanc, *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*. Butterworth-Heinemann, 2017.

[37] C. Knospe, "PID control," *Control Systems Magazine, IEEE*, vol. 26, pp. 30–31, Mar. 2006. DOI: 10.1109/MCS.2006.1580151.

[38] E. Komponente, *Cloudflare*, n.d. [Online]. Available: https://eckstein-shop.de/Sharp-GP2Y0A41SK0F-Analog-Distance-Sensor-4-30cm.

[39] B. Ladewig, M. McReynolds, B. Sustr, and K. Williams. "Final Report For The Micromouse." (Apr. 2003), [Online]. Available: https://github.com/kathleenwest/MicroMouse/blob/master/Final%20Report.pdf.

[40] A. Lenz, "Lecture 7: PCB Prototyping and some Electronics," TUM Practical Lab Course: Micromouse - Building an Educational Racing Robot From Scratch, 2020. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/2321352/mod_resource/content/1/Lecture7PCBsAndPrototypingElectroncis.pdf.

[41] A. Lenz, "Lecture 5: Review and Encoders," TUM Practical Lab Course: Micromouse - Building an Educational Racing Robot From Scratch, 2020. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/2301189/mod_resource/content/1/Lecture5ReviewEncElectronics.pdf.

[42] A. Lenz, "Lecture 4: Pulse Width Modulation (PWM) and Motor Control," TUM Practical Lab Course: Micromouse - Building an Educational Racing Robot From Scratch, 2020. [Online]. Available: https://www.moodle.tum.de/pluginfile.php/2274434/mod_resource/content/1/Lecture4PWM.pdf.

[43] Li, Guangxu, Qin, Dongxing, and Ju, Hui, "Localization of Wheeled Mobile Robot Based on Extended Kalman Filtering," *MATEC Web of Conferences*, vol. 22, p. 01 061, 2015. DOI: 10.1051/matecconf/20152201061. [Online]. Available: https://doi.org/10.1051/matecconf/20152201061.

[44] *LM2937 500-mA Low Dropout Regulator*, SNVS100F, Rev. F, Texas Instruments Incorporated, Jul. 2014. [Online]. Available: https://www.mouser.com/datasheet/2/405/lm2937-405746.pdf.

[45] *LM2937-2.5, LM2937-3.3 400mA and 500mA Voltage Regulators*, DS100113, Texas Instruments Incorporated, Aug. 2005. [Online]. Available: http://users.ece.utexas.edu/~valvano/Datasheets/LM2937-3.3.pdf.

[46] Microchip. "dsPIC33F/PIC24H Family Reference Manual Section 14. Motor Control PWM." (n.d.), [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/70187e.pdf. Accessed: September 2020.

[47] Microchip Technology Inc, *Microcontroller dsPIC33FJ64MC804*, n.d. [Online]. Available: https://www.microchip.com/_images/ics/medium-dsPIC33FJ64MC804-TQFP-44.png.

[48] I. Microstar Laboratories. "Ziegler-Nichols Tuning Rules for PID." (n.d.), [Online]. Available: http://www.mstarlabs.com/control/znrule.html#Ref4. (Accessed: September 2020).

[49] R. Misra and R. Adler. "Micromouse Competition Rules." (2018), [Online]. Available: https://ewh.ieee.org/reg/2/sac-18/MicromouseRules.pdf. Accessed: September 2020.

[50] *MPLAB XC16 C Compiler User's Guide*, DS50002071K, Rev. K, Microchip Technology Inc., Jul. 2020. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/50002071K.pdf.

[51] N. A. Polytechnic. "History of Micromouse Challenge." (2017), [Online]. Available: http://www.micromouseonline.com/micromouse-book/history/. (Accessed: September 2020).

[52] W. D. Pullen. "Think Labyrinth: Maze Algorithms." (2017), [Online]. Available: http://www.micromouseonline.com/micromouse-book/mazes-and-maze-solving/examples/. Accessed: October 2020.

[53] W. D. Pullen. "Think Labyrinth: Maze Algorithms." (2015), [Online]. Available: http://www.astrolog.org/labyrnth/algrithm.htm. Accessed: August 2020.

[54] I. (Robotics Society of America. "Maze Solving / Micromouse Rules." (2011), [Online]. Available: http://robogames.net/rules/maze.php. Accessed: September 2020.

[55] S. Sattel. "What Are Decoupling Capacitors?" (n.d.), [Online]. Available: https://

www.autodesk.com/products/eagle/blog/what-are-decoupling-capacitors/. Accessed: August 2020.

[56] S. Sattel. "What is a Voltage Regulator?" (n.d.), [Online]. Available: https://www.autodesk.com/products/eagle/blog/what-is-a-voltage-regulator/. (Accessed: September 2020).

[57] S. Sattel. "Schematic Basics Part 3: ERC." (n.d.), [Online]. Available: https://www.autodesk.com/products/eagle/blog/schematic-basics-part-3-erc/. (Accessed: September 2020).

[58] S. Sattel. "Design Rule Check: PCB Layout Basics 3." (n.d.), [Online]. Available: https://www.autodesk.com/products/eagle/blog/design-rule-check-pcb-layout-basics-3/. (Accessed: September 2020).

[59] Sharp, *GP2Y0A51SK0F: Distance Measuring Sensor Unit, 2 to 15 cm*, n.d. [Online]. Available: https://www.mouser.de/datasheet/2/365/gp2y0a51sk_e-1360360.pdf.

[60] *Sharp GP2Y0A51SK0F*, OP13007EN, SHARP, n.d. [Online]. Available: https://www.mouser.de/datasheet/2/365/gp2y0a51sk_e-1360360.pdf.

[61] R. Stafford. "How can I find the angle between two vectors, including directional information? - MATLAB Answers." (2015), [Online]. Available: https://www.mathworks.com/matlabcentral/answers/180131-how-can-i-find-the-angle-between-two-vectors-including-directional-information/#answer_169149. (Accessed: October 2020).

[62] J. Su, C. Lee, and C. Chen, "Sensor fusion algorithms for encoder resolution enhancement in educational mobile robots," in *2016 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*, Aug. 2016, pp. 1–5. DOI: 10.1109/ARIS.2016.7886612.

[63] *TB6612FNG: Driver IC for Dual DC Motor*, TOSHIBA Corporation, Oct. 2014. [Online]. Available: https://www.mouser.de/datasheet/2/408/TB6612FNG_datasheet_en_20141001-708260.pdf.

[64] *Technical Information Faulhaber Motors*, Dr. Fritz Faulhaber GmbH & Co. KG, Jun. 2020. [Online]. Available: https://www.faulhaber.com/fileadmin/Import/Media/EN_TECHNICAL_INFORMATION.pdf.

[65] Texas Instruments, "PCB Design Guidelines for Reduced EMI," *SZZA009 November*, 1999. [Online]. Available: https://www.ti.com/lit/an/szza009/szza009.pdf.

[66] Y. University. "UK Micromouse Maze Solver Rules." (n.d.), [Online]. Available: https://www.cs.york.ac.uk/micromouse/Rules/Maze_Solver_Rules.pdf. Accessed: September 2020.

[67] *VL53L0X: World's Smallest Time-of-Flight Ranging and Gesture Detection Sensor*, DocID029104, Rev. 2, STMicroelectronics, Apr. 2018. [Online]. Available: https://www.st.com/resource/en/datasheet/vl53l0x.pdf.

[68] Wiki.nus. "Analog Output." (2020), [Online]. Available: https://wiki.nus.edu.sg/display/Arduino/Analog+Output. (Accessed: October 2020).

[69] Wikipedia contributors, *Electromagnetic compatibility*, [Online; accessed 8-October-2020], 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Electromagnetic_compatibility&oldid=966509408.

[70] Wikipedia contributors, *Pull-up resistor, The Free Encyclopedia*, [Online; accessed 7-October-2020], 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Pull-up_resistor&oldid=958261724.

[71] Wikipedia contributors, *Electromagnetic interference*, [Online; accessed 8-October-2020], 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Electromagnetic_interference&oldid=976042558.

[72] Wikipedia contributors, *Racing line — Wikipedia, the free encyclopedia*, [Online; accessed 15-October-2020], 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Racing_line&oldid=959804758.

[73] Wikipedia contributors, *Decoupling capacitor*, [Online; accessed 10-October-2020], 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Decoupling_capacitor&oldid=933630042.

[74] Wikipedia contributors, *Time of flight — Wikipedia, the free encyclopedia*, [Online; accessed 15-October-2020], 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Time_of_flight&oldid=977748557.

[75] Wikipedia contributors, *Ceramic capacitor*, [Online; accessed 9-October-2020], 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Ceramic_capacitor&oldid=979732106.

[76] I. Wikipedia Foundation. "Light-Emitting Diode." (2020), [Online]. Available: https://en.wikipedia.org/wiki/Light-emitting_diode. Accessed: September 2020.

[77] *WP7113LSYCK/J3: T-1 3/4 (5mm) Solid State Lamp*, DSAN8288 / 1101033257, Rev. 4B, Kingbrigh, Mar. 2019. [Online]. Available: https://www.mouser.de/datasheet/2/216/WP7113LSYCK-J3-535774.pdf.

[78] S. Zacher and M. Reuter, *Regelungstechnik fur Ingenieure: Analyse, Simulation und Entwurf von Regelkreisen*. Springer Fachmedien Wiesbaden, 2014, ISBN: 9783834822161. [Online]. Available: https://books.google.fr/books?id=VLnEBAAAQBAJ.

Appendix

A Requirement Diagram

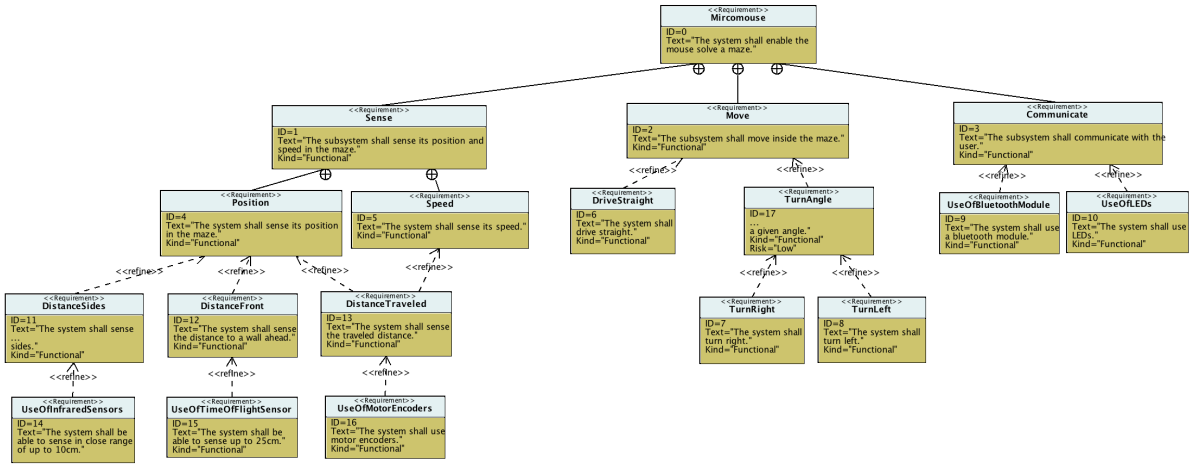


Figure 60: Full requirement diagram

B Sensor Tests

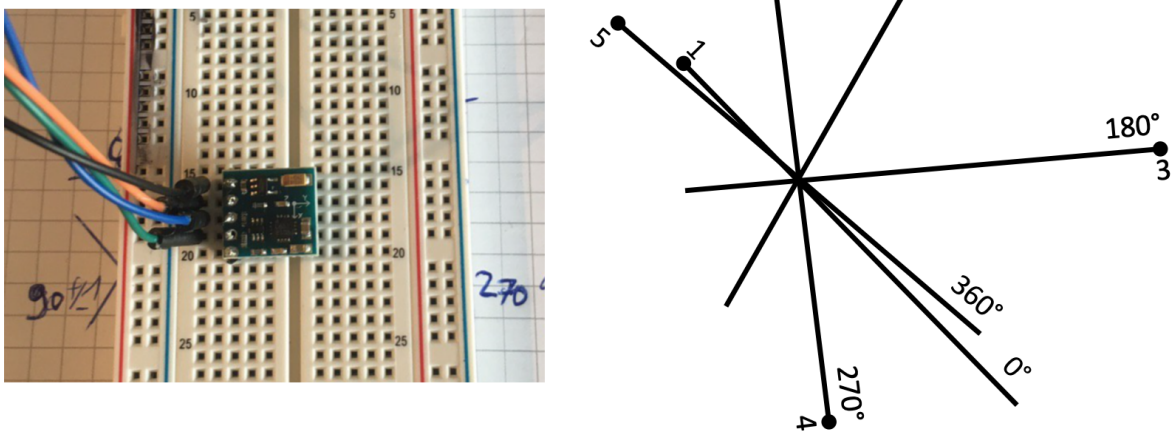


Figure 61: Compass test with Arduino

C Bill of Materials

Qty.	Designator	Description	Mfr.	Mfr. Part Number
1	MDRIV	Brushed DC Motor Driver, 2 Channel, 1 A, 13.5 V, 100 kHz, SMD, SSOP-24	Toshiba	TB6612FNG,C,8,EL
11	C2, C3, C4, C7, C9, C11, C12, C13, C15, C19, C20	Capacitor, MLCC, 0.1 uF, 50 V, 5%, X7R, SMD, 0805	KEMET	C0805C104J5RACAUTO
2	C17, C18	Capacitor, MLCC, 10 uF, 10 V, 5%, X7R, SMD, 0805	KEMET	C0805X106J8RACAUTO
2	C5, C6	Capacitor, MLCC, 18 pF, 50 V, 1%, high SRF, C0G, SMD, 0805	KEMET	CBR08C180F5GAC
5	C1, C8, C10, C14, C16	Capacitor, Tantalum, 10 uF, 10 V, 10%, 2 Ohm, SMD, A	KEMET	T495A106K010ATA2K0
4	TP1, TP2, TP3, TP4	Connector, TP, Loop, 1.02 mm	Keystone Electronics	5001
1	Q1	Crystal, 16 MHz, 10 PPM, 18 pF, Radial, HC-49/S	TXC Corporation	9B-16.000MEEJ-B
1	MCU	Digital Signal Controller, 40 MHz, 16 bit, 64 kB, +125 °C, SMD, TQFP-44	Microchip	dsPIC33FJ64MC804-E/PT
3	LED1, LED2, LED3	Diode, LED, Blue, 1.85 V, 2 mA, 800 mcd, SMD, 0805	Kingbright	APTD2012LVBC/D
2	LED4, LED5	Diode, LED, Yellow, 1.85 V, 2 mA, 800 mcd, TH, 5mm	Kingbright	WP7113LSYCK/J3
1	D1	Diode, Schottky, 240mV, 200mA, 30 V, SMD, SOD-123	Vishay	BAT54W-HG3-08

Qty.	Designator	Description	Mfr.	Mfr. Part Number
1	VL53L0X	Distance Sensor Module, ToF, VL53L0X Sensor, 5-200 cm, 3.3 V, 30 ms	Adafruit	3317
2	IRL, IRR	Distance Sensor, IR, analog, 2-15 cm, 5 V, 16.5 ms	Sharp Microelectronics	GP2Y0A51SK0F
1	R4	Resistor, MF, 0 R, 0%, 1/8 W, SMD, 0805	Vishay	WSL080500000ZEA9
3	R1, R2, R6	Resistor, TKF, 1.5k, 1%, 1/8 W, SMD, 0805	Vishay	CRCW08051K50FKEA
5	R7, R8, R9, R10, R11	Resistor, TKF, 330 R, 1%, 1/2 W, SMD, 0805	Vishay	CRCW0805330RFKEAHP
2	R3, R5	Resistor, TKF, 4.7k, 1%, 1/8 W, SMD, 0805	Vishay	CRCW08054K70FKEA
2	BTN, RESET	Switch, Tactile, SPST, 1 N, TH, 6x6 mm	Omron	B3F-1000
1	VR33	Voltage Regulator, LDO, 3.3 V, 5%, 500 mA, SMD, TO-263-3	Texas Instruments	LM2937ES-3.3/NOPB
1	VR5	Voltage Regulator, LDO, 5 V, 5%, 500 mA, SMD, TO-263-3	Texas Instruments	LM2937ES-5.0/NOPB

D Printed Circuit Board

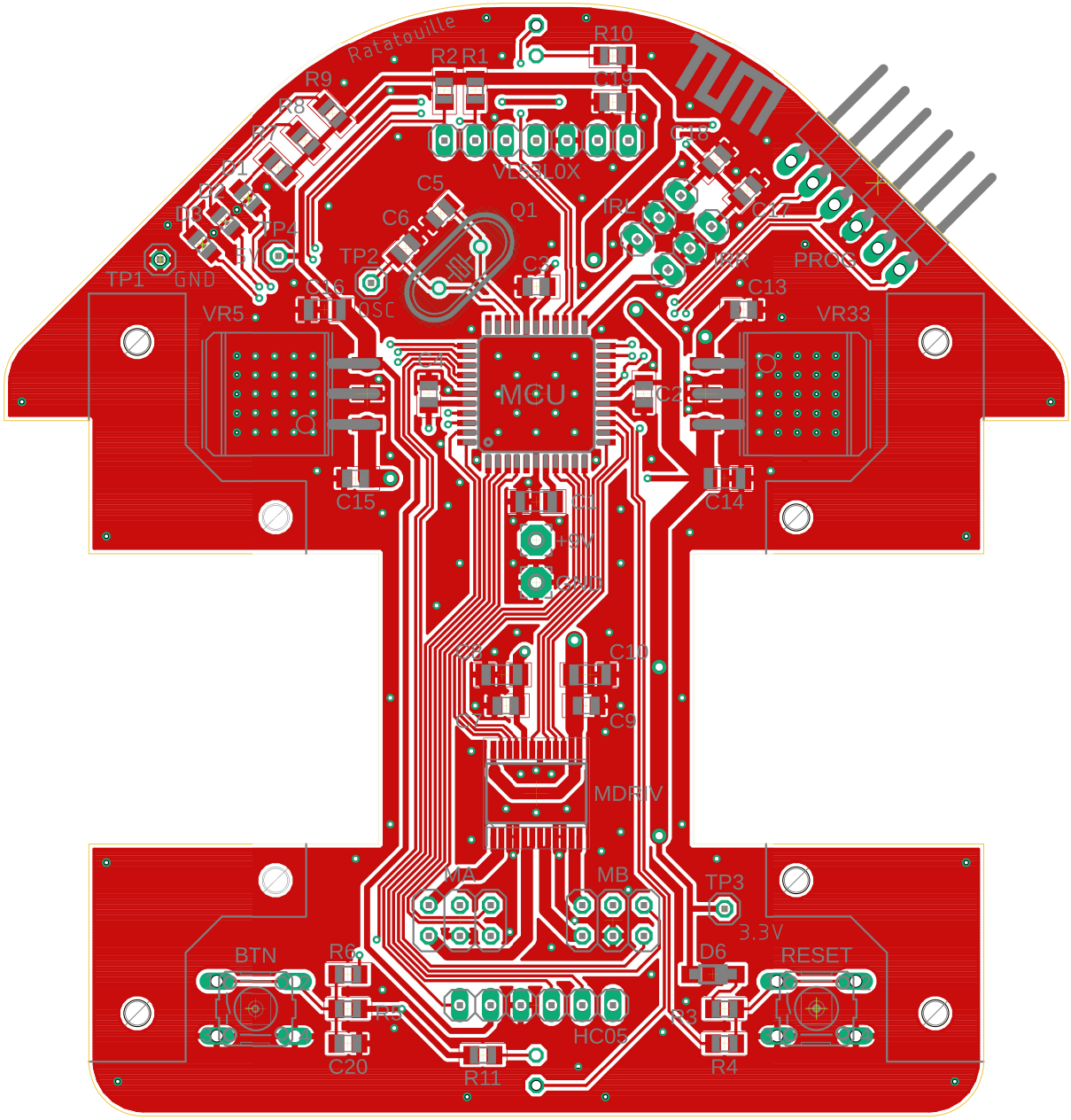


Figure 62: PCB top side with ground pour enabled

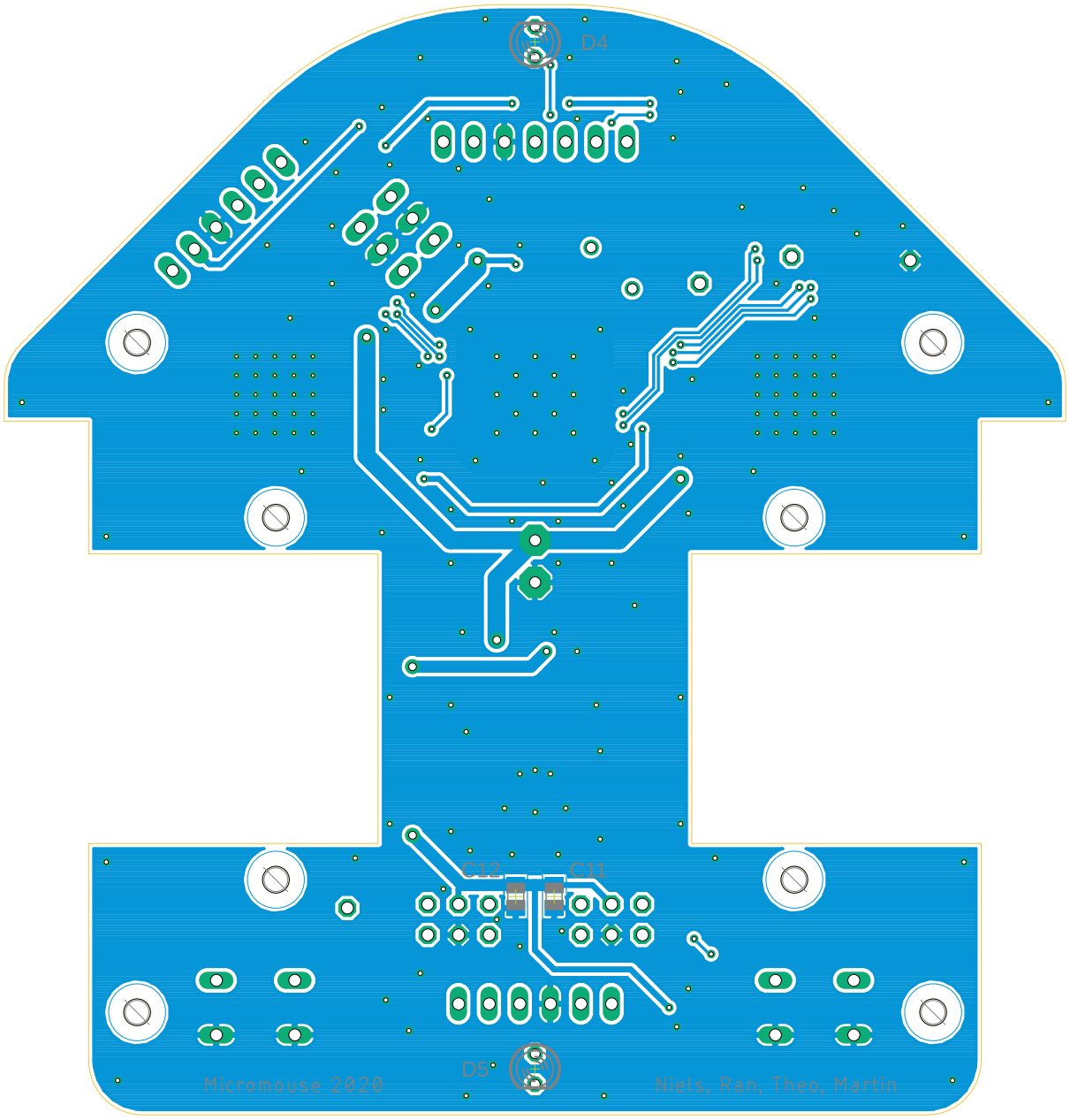


Figure 63: PCB bottom side with ground pour enabled

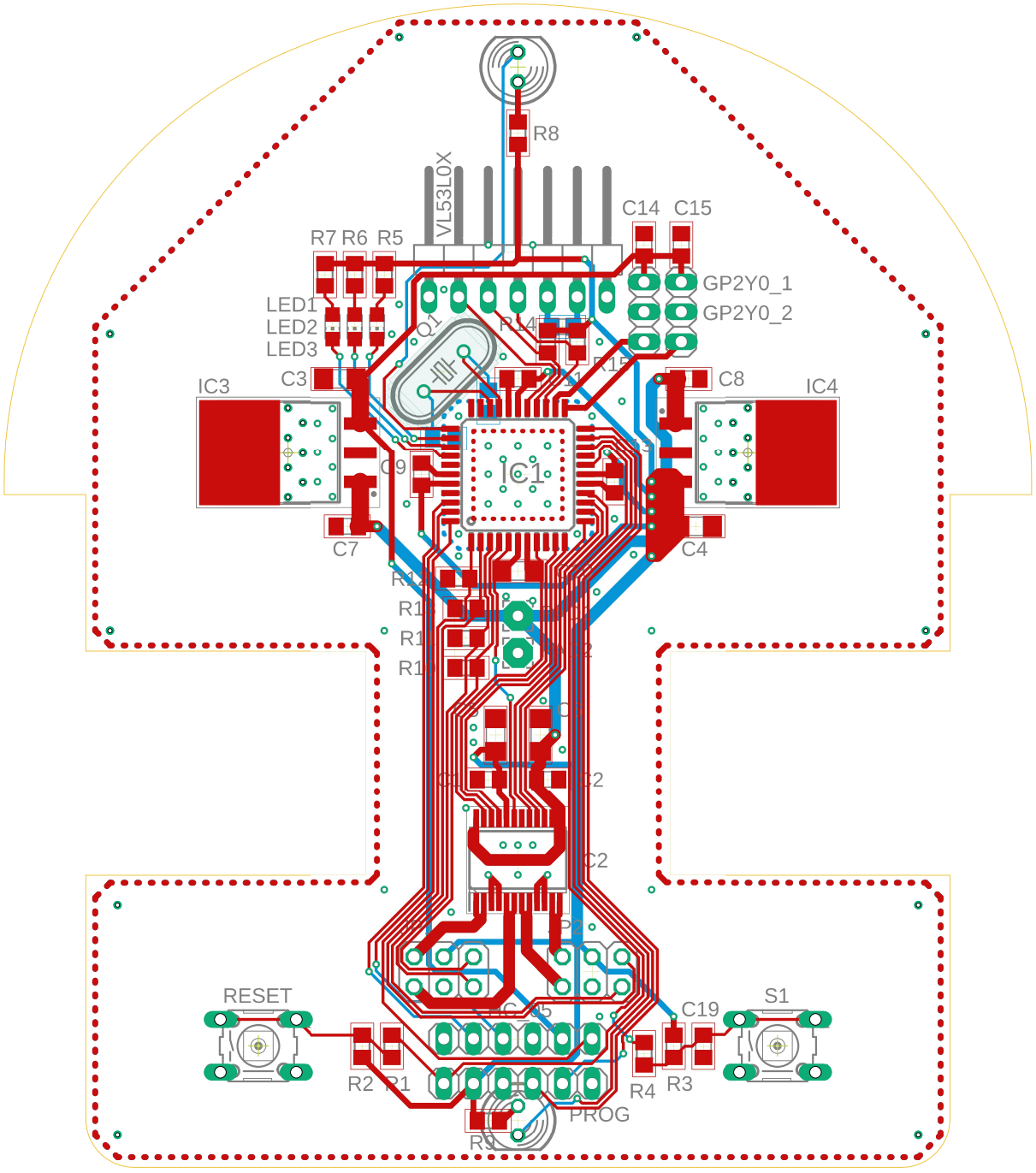


Figure 64: First iteration of the PCB

E Soldering

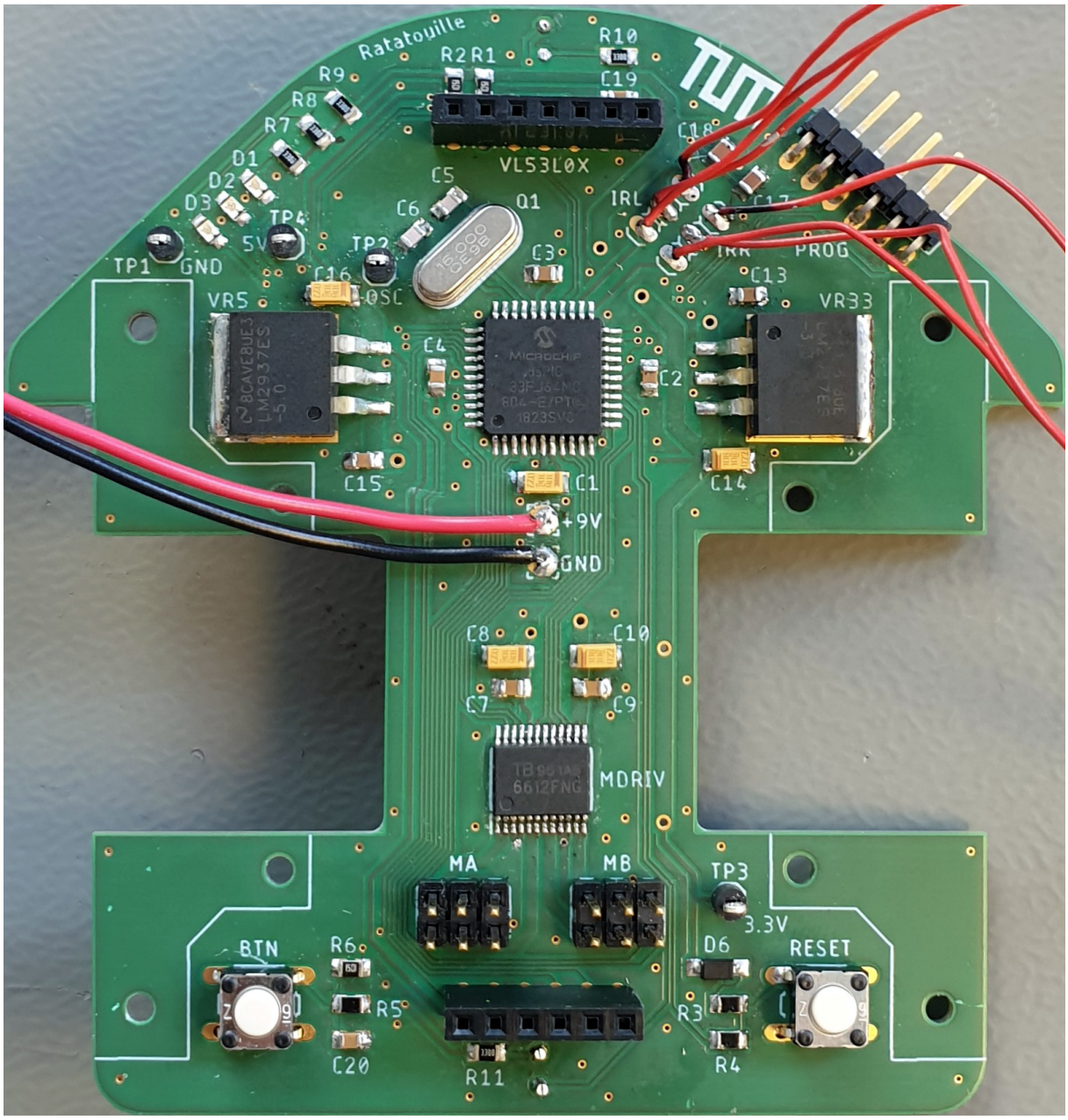


Figure 65: PCB with all components soldered on, top side

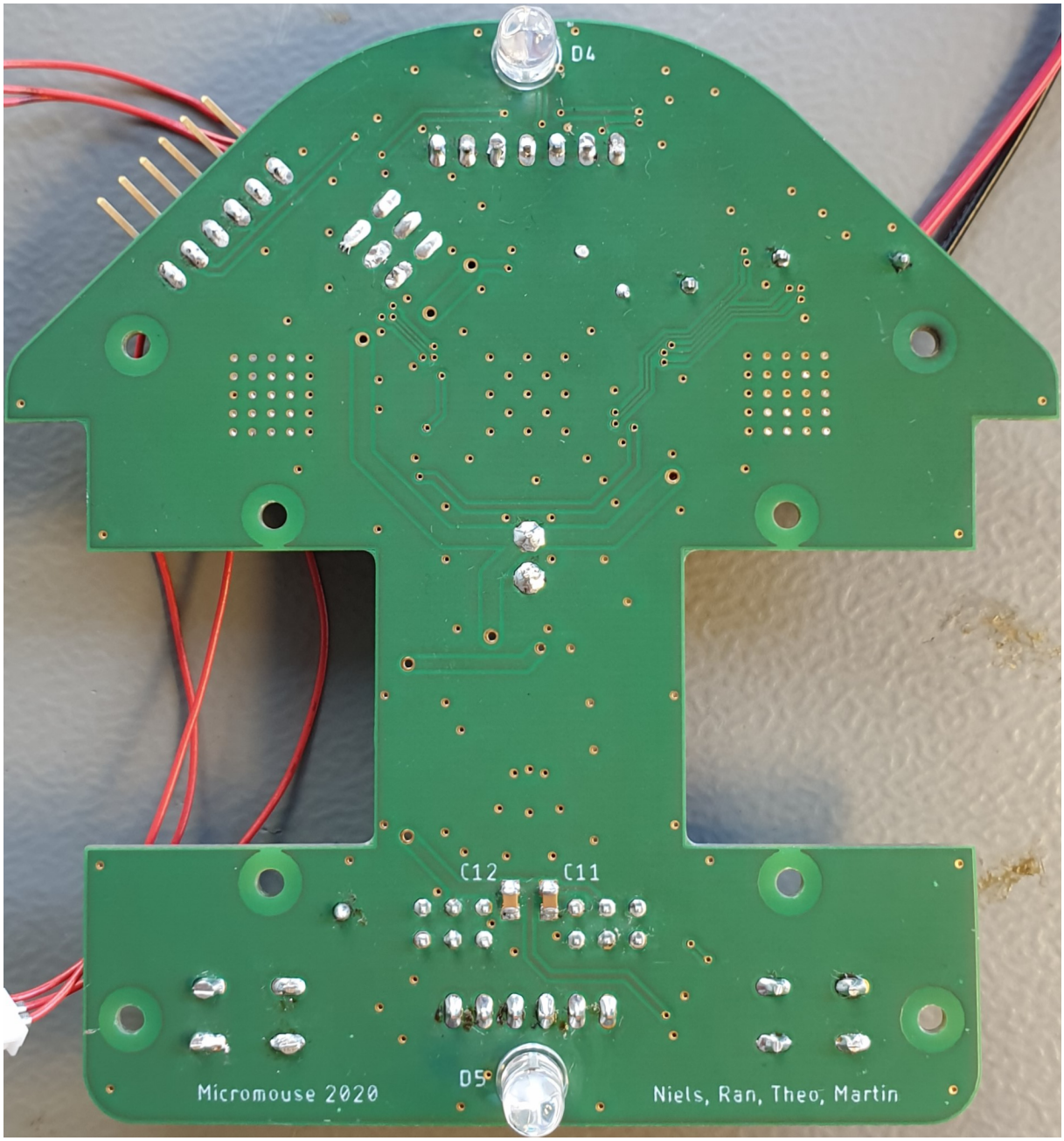


Figure 66: PCB with all components soldered on, bottom side

F Alternative Localization Approach

Brief calculation of Kalman fiter is explained below.

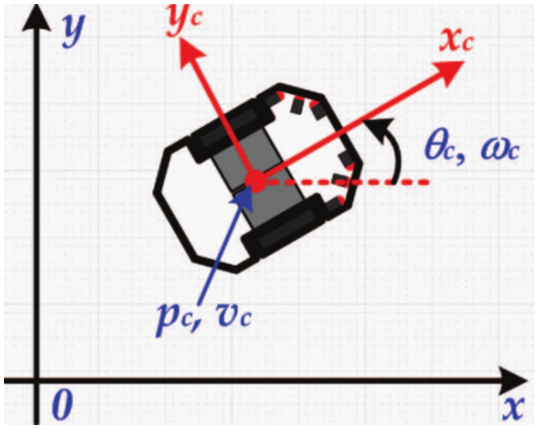


Figure 67: Differential driven micromouse [62]

P_C,θ_c,v_c,$w_c$represent its center position, orientation angle, center velocity, and angular velocity. Suppose sampling time interval is h, the corresponding discrete-time model of the 67 is described as follows [62]:

$$\begin{bmatrix} p_C(k+1) \\ v_C(k+1) \end{bmatrix} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_C(k) \\ v_C(k) \end{bmatrix} + \begin{bmatrix} \frac{h^2}{2} \\ h \end{bmatrix} a_c(k) + \begin{bmatrix} w_1(k) \\ w_2(k) \end{bmatrix},$$
$$= Fx(k) + Ga_c(k) + W(k), \tag{7a}$$
$$z(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_C(k) \\ v_C(k) \end{bmatrix} + v(k) = Hx(k) + v(k), \tag{7b}$$

where $W(k)$ and $v(k)$ is the input disturbance and measurement noise with Gaussian distribution at time k. $z(k)$ is the measurement output variable. The dynamics for a Kalman filter of the micromouse can be derived from followings [62]:
Prediction:

$$\hat{x}_k(k+1) = F\hat{x}_k(k) + Ga(k), \tag{8a}$$
$$\hat{z}(k+1) = H\hat{x}_k(k), \tag{8b}$$
$$P_k(k+1) = FP_k(k)F^T + Q_k, \tag{8c}$$

Update:

$$K_{k+1} = P_k(k+1)H^T(HP_k(k+1)H^T + R_{k+1})^{-1}, \tag{9a}$$
$$\hat{x}_{k+1}(k+1) = \hat{x}_k(k+1) + K_{k+1}e_z(k+1), \tag{9b}$$
$$e_z(k+1) = z(k+1) - \hat{z}(k+1), \tag{9c}$$
$$P_{k+1}(k+1) = U_{k+1}P_k(k+1)U_{K+1}^T + K_{k+1}R_{k+1}K_{k+1}^T, \tag{9d}$$
$$U_{k+1} = I - K_{k+1}H, \tag{9e}$$