# Micromouse: Designing an Educational Racing-Robot from Scratch Report

Natalia Poliakova,[1] Autor 2,[2], Autor 3[3], Autor 4, [4] Autor 5, [5] Autor 6 [6]

[1]Department of Informatics, Technical University of Munich
An Unknown Address, Wherever, ST 00000, USA
[2]Another Unknown Address, Palookaville, ST 99999, USA

**Our abstract - short overview of the whole report contents? Probably should be left untouched till we finish with the main report body.**

# Contents

# 1 Introduction

## 1.1 Micromouse competition

According to the general description of the micromouse competition, ”in this contest the contestant, or team of contestants, must design and build an autonomous robotic mouse capable of traversing a maze of standard dimensions from a specified corner to its center in the shortest time.” (*1*).

General example of a competition setup can be seen on the Fig. 1

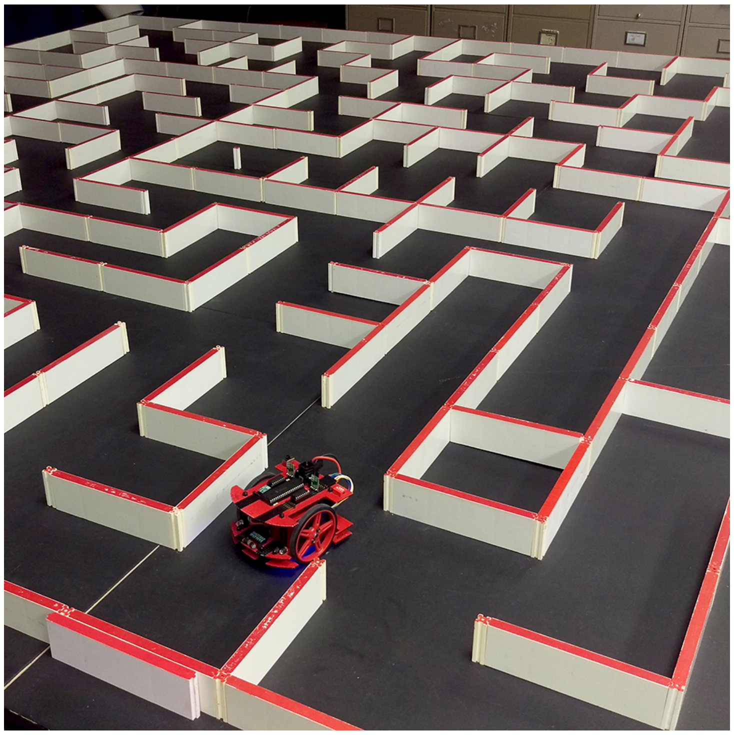


Figure 1: Micromouse competition photo: labyrinth and robot example (*2*)

## 1.2 Aims and Objectives

Short summary of the general competition rules is as following (*1*):

- Self-Containment: A Micromouse shall be self-contained (no remote controls).

- Method of Movement: A Micromouse shall not jump over, fly over, climb, scratch,cut, burn, mark, damage, or destroy the walls of the maze.
- Maze Dimensions: The maze is composed of 18cm x 18cm unit squares arranged as 16 x 16 units. The walls of the units of the maze are 5 cm high and 1.2 cm thick
- Multiple Paths: Multiple paths to the destination square are allowed and are to be expected. The destination square will be positioned so that a wall-hugging mouse will NOT be able to find it.

In the case of our project, the aforementioned rules were used in a slightly modified way. The labyrinth is reduced in size to 8x8 units in order to fit better to the project conditions. The micromouse competition consists of 2 runs: in the first run, the robot is going through the maze and, according to arbitrary chosen algorithm, constructs the maze map; in the second run the mouse should reach the center of the maze (found in the first run) as quickly as possible, according to the composed map of the labyrinth.
The main goal of the project was set to: ”get as close to the realization of the procedure described above as possible”. The ways and approaches that were chosen to reach this goal are named and described properly in the next section.

## 1.3 Tools

The list of tools used throughout the whole length of the project:

- Microchip MPLABX - an IDE, used to set-up, configure and program a microcontroller. Used together with XC16 compiler from Microchip.
- MathWorks MATLAB (*did any of us except for Alex actually use it?*)

- Autodesk Eagle - PCB design software, used to create schematic diagrams, organize the component placement and route the PCB.
- Autodesk Fusion 360 - CAD/CAM design software, used to build in 3D all parts of the casing for the robot.

## 2 Conceptual design and justification of the design

In order to set the right design goals and milestones during the semester, adequately estimate the workload and organize the working process in the most effective way, we needed to: analyze the initial design conditions and restrictions, roughly formulate the implementation blocks along with respective skill-sets of all project members and - after that - write down the most suitable working program. Both steps are described in detail below.

### 2.1 Initial design conditions

The main question to answer was to grasp and implement using the knowledge acquired the logic behind the ”how do we build a robot that should drive in the labyrinth and be able to make intelligent decisions (turns) based on the observations (made by sensors)?”

There are some predefined conditions and tools that we used as a starting point in answering this question:

- Geometric constraints

  As was mentioned in the previous section, the geometric parameters of the maze in our case were as following: 8x8 identical squares with a side of 16cm. Therefore, our ”mouse” was logically required to be smaller than 16cm in width and length, ideally the size that would allow it to move freely while performing any type of movement within the maze (therefore leaving at least 2-3 centimeters of distance to the surrounding walls)

- Power supply

  According to the unified formal requirements of the Micromouse competition, the robot should be autonomous, which infers carrying it's own power supply in form of a standard 9V battery.

- Sensing the environment

  The initial condition in regards to sensing simply states that the robot should be able to perceive the environment and make informed decisions about the next movement (moving straight or turning in one of the directions - the turn angle was set to be fixed at 90 degrees in order to simplify the design) based on the analysis of the information received from the sensors. The sensor type that is the most efficient for the cause - simple infrared proximity sensor. We had an option to either order the desired amount of sensors or to build them ourselves.

- Motors

  In order for the "mouse" to move, we were provided with predefined 2619-SR-IE2-16 motors and sets of wheels of different diameter.

- Microcontroller

  Initially, for the educational purposes and in order to gain some knowledge in configuring and setting up the microcontroller to control the needed peripherals, a pair of dsPIC30F4011 microcontrollers was provided. Appropriate for the goal task microcontroller was decided to be chosen later. Overall, it had to have all required pins and interfaces, such as:

  - PWM outputs for the motor control

  - analog inputs to receive data from sensors

  - enough extra pins to connect peripherals (such as LEDs)

- flexible and convenient pin mapping in order to satisfy the geometric constraints

- PCB

  The printed circuit board for the robot was to be designed independently, according to all the aforementioned conditions.

- Casing

  In order for all of the components (such as the board, the motors and the battery) to hold together in one single ”mouse” unit, it was set that a custom casing must be designed and printed.

Based on those conditions and also tools and devices provided, we came up with the implementation plan, which can be seen in detail in the next part.

## 2.2 Work program and Gantt chart

I would suggest to fit here the part with ”design decisions based on the aforementioned conditions (justification)” *This is where I’m a bit lost. We could include some ”ideal” version of this chart here, but where exactly should we describe all the changes in planning and organization and why they had to be done at each stage of building the prototype and the final version of the mouse? Should it be described here? Or later in the ”problems and challenges”? Also I think maybe here we should talk about all the initial learning stage we had to go through in the first half of the praktikum (maybe devote a subsection for this here or later in the report)*

# 3 Acquired Knowledge

Some introductory info.

### 3.1 Studying dsPICs

Some theoretical knowledge. Something like listing a short summary of everything we've studied.

### 3.2 Fusion

We can mention a workshop here

### 3.3 Eagle

Not sure if we should mention it here, but why not.

**your suggestions for other subsections are highly appreciated**

## 4 Hardware design

In this section, the hardware design and implementation of the micromouse robot are described. First, the corresponding electrical circuit, known as scematic, is broken down to its components and thoroughly presented. In the next subsection, the actual components chosen for the hardware implementation are listed. The model for the final Printed Circuit Board (PCB) follows right after, where the chosen electrical components are placed in the computer model. Last, but not least, the protective casing for the micromouse is described.

### 4.1 Schematic and Components Description

The first step in the development of the hardware is the drawing of the electrical circuit. The realization of a moving micromouse demands for the combination of a variety of components connected to each other. Exactly this is described in the so called schematic of the circuit, which resembles a drawing on paper. The schematic of the micromouse can be seen in Fig. 2.

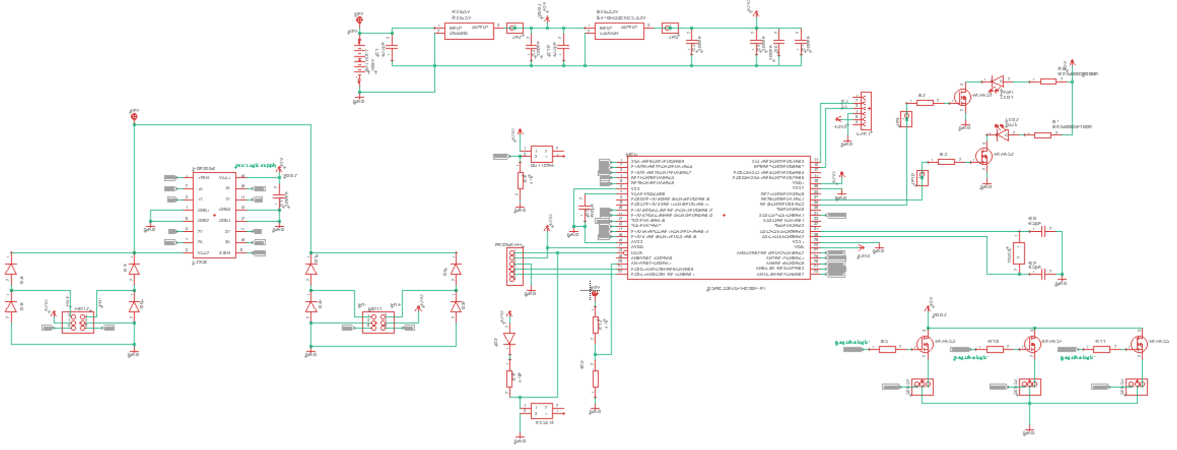


Figure 2: Schematic overview of the micromouse

This can look intimidating at first and definitely not readable in such a miniature picture. So, let's break it down to its components and describe its sub-circuits.

With the exception of the first subsection, the components will be presented from left to right and from top to bottom, so that the reader may always refer to the overview schematic to get a clear picture of a component's relation to the whole.

### 4.1.1 Microcontroller (MCU)

The microcontroller (mcu) can be seen in Fig. 3. Notice that the symbolic representation of the mcu, specifically when it comes to the location of its pins, doesn't correspond to the real model. For the purpose of the schematic, we don't really care about the real location of the pins. As will be seen later, this becomes of importance when placing the components on the board to be printed.

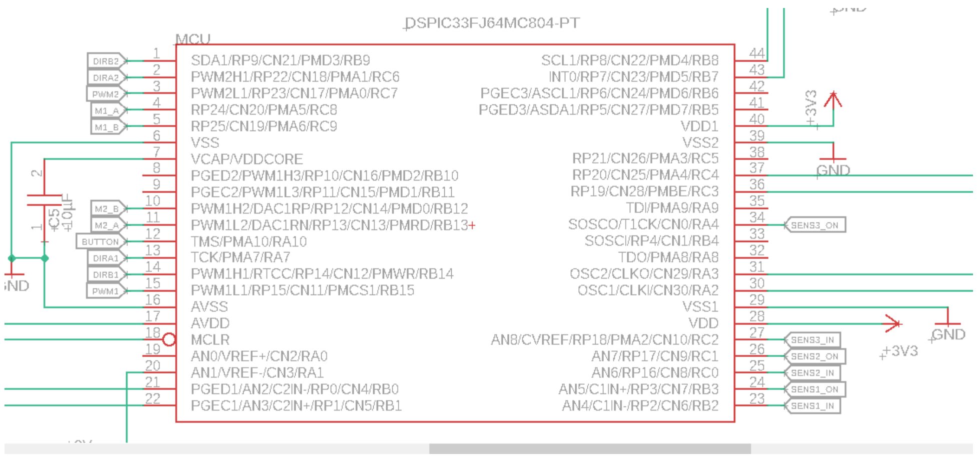
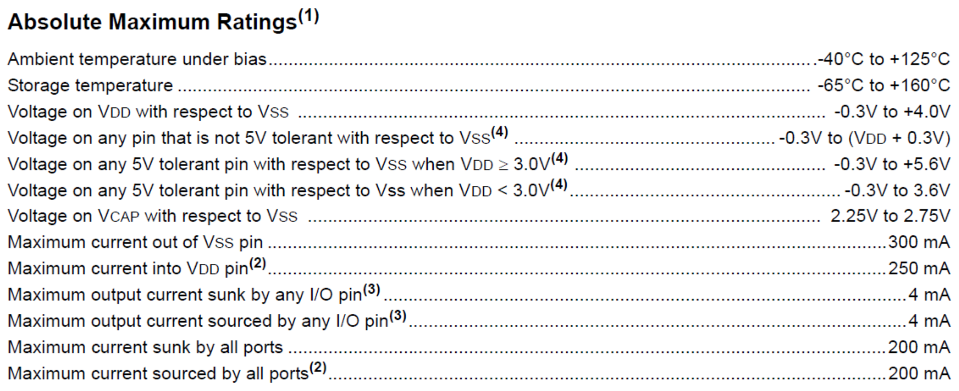


Figure 3: The dsPIC33FJ64MC804 microcontroller

The assignment of mcu pins to components is presented in the Software Section and will not be repeated here. Please refer to that Section for understanding the functionality of its pin in our circuit.

Another point to notice about the pin connections is the convenient label feature that Eagle offers. Notice how some of the components are wired to the pins of the mcu, while other connections are represented as labels. Labels in the circuit are the equivalent of wires and greatly contribute to the readability and modularity of the schematic. As long as two wires anywhere in the circuit share the same label, they are connected.

A last comment has to be made about the pins of the mcu: The pins, as everything in our circuit, are very real components and therefore are subject to electrical limitations. This is a very important point to keep in mind, when driving any load, such as LEDs or a motor, but also when connecting input signals, such as sensors. Specifically, in the section ”Electrical Characteristics” in (*3*), we find maximum values for individual pins and for all utilized pins combined. These can be seen in Fig. 4. These characteristics lead to some important design decisions, specifically named in later subsections.



**Absolute Maximum Ratings**(1)

| | |
|---|---|
| Ambient temperature under bias | -40°C to +125°C |
| Storage temperature | -65°C to +160°C |
| Voltage on VDD with respect to VSS | -0.3V to +4.0V |
| Voltage on any pin that is not 5V tolerant with respect to VSS(4) | -0.3V to (VDD + 0.3V) |
| Voltage on any 5V tolerant pin with respect to VSS when VDD ≥ 3.0V(4) | -0.3V to +5.6V |
| Voltage on any 5V tolerant pin with respect to Vss when VDD < 3.0V(4) | -0.3V to 3.6V |
| Voltage on VCAP with respect to VSS | 2.25V to 2.75V |
| Maximum current out of VSS pin | 300 mA |
| Maximum current into VDD pin(2) | 250 mA |
| Maximum output current sunk by any I/O pin(3) | 4 mA |
| Maximum output current sourced by any I/O pin(3) | 4 mA |
| Maximum current sunk by all ports | 200 mA |
| Maximum current sourced by all ports(2) | 200 mA |

Figure 4: Electrical limitations for the pins of the mcu

### 4.1.2 Votage Regulators

The necessity for voltage regulators, which can be seen in Fig. 5, arises from the need for different voltages present in our system. Specifically, as found in (*3*), the mcu considers 3.3V as logical one and it is this voltage we need to provide to it (see VDD pins). The distance sensors used (taken from a previous micromouse model) need 5V. And yet the motors need 6V, as can be seen in (*4*). Notice that we are using a battery of 9V. Therefore, we need a way to generate the required 3 different voltages.

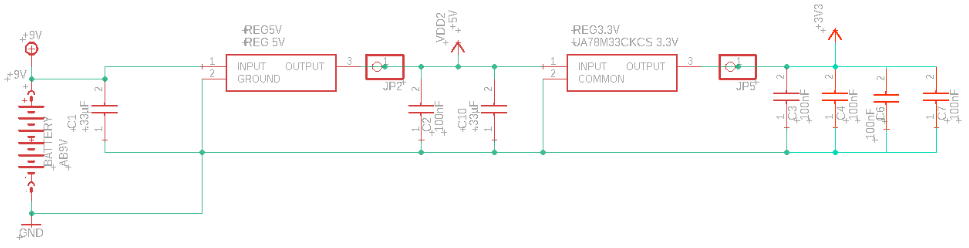


Figure 5: The battery and voltage regulators

When it comes to the motor, we can regulate the voltage we feed it with, by regulating the maximum duty cycle of the PWM (explained in another Section). So, we provide it with a fraction of the available 9V. For the distance sensors and the mcu, 2 voltage regulators are used in succession. They provide our components with steady voltage, necessary for functioning properly.

Now, notice the capacitors used in Fig. 5. Capacitors are used in the input and output of the voltage regulators. Additionally, a bunch of capacitors can be observed at the far right part of the sub-circuit. All of these capacitors have both a special name and function. They are called decoupling capacitors and their role is to help in providing steady voltage. Specifically, they are always physically connected close to their corresponding component and should a temporary lack of electrons happen, they provide from their surplus, keeping the voltage steady. The capacitors at the far right are only symbolically placed there in the circuit. In reality, they are connected between the pins of the mcu that connect to VDD (3.3V) and ground (GND).

### 4.1.3 H-bridge and Motors

Next, the motors and H-bridge sub-circuit will be presented. First, let us consider the motors. As can be seen in Fig. 6, the motors are symmetrically connected to the H-bridge and to the mcu. So, let us concentrate on one of them, the left. The same explanation applies for the other one.

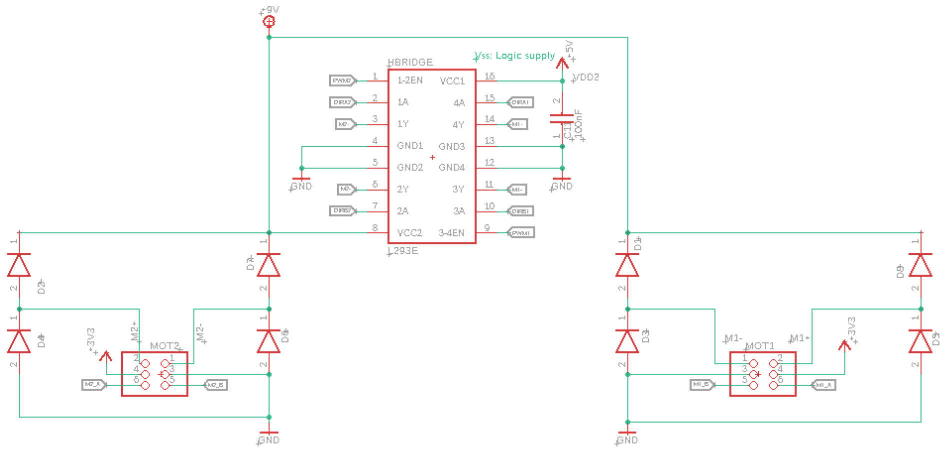


Figure 6: The motors and the H-bridge

In Fig. 7 taken from the motor datasheet, the motor pins can be noticed. One pair of pins (1, 2) refers to the voltage we feed to the motor. Remember that we are using PWM (max 6V) to control how fast the motor turns and notice that if the voltage is reversed, the rotation of the motor is also reversed. As we will shortly see, for this purpose, the H-bridge is needed. The function of reversing the motor's rotation is hihgly desirable, since we want our 2-wheels micromouse to be able to move backwards and to turn on spot (imagine one wheel moving to one direction and the other to the opposite direction with the same velocity).

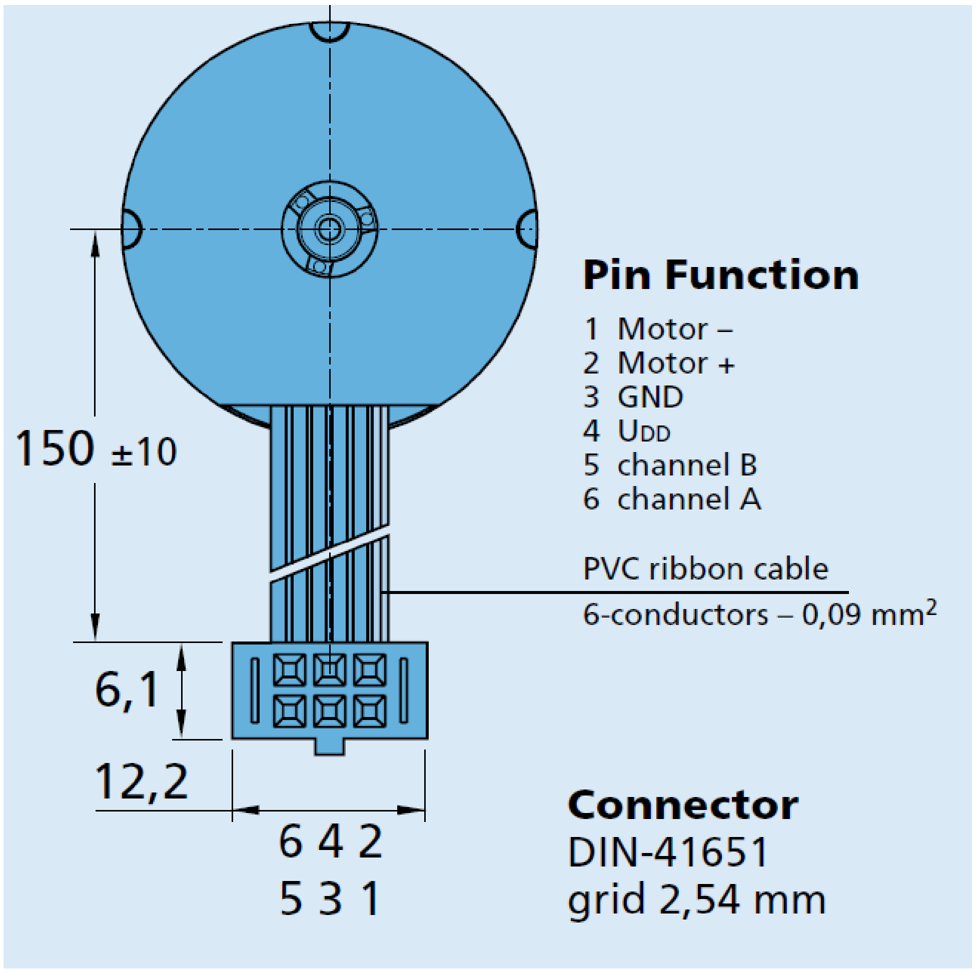


Figure 7: The motor pins as presented in its datasheet

Another pair of pins (3, 4) refers to the voltage feed of the motor encoder. And the last pair of pins (5, 6) are used for the output of the motor encoder signals. These signals encode the motor's rotational position every moment and are fed to the mcu. In the mcu, with the help of the quadrature encoder (QE) module, they can be translated to real position in space.

Finally, let us consider the H-bridge. As mentioned, the need for the H-bridge arises from the function of reversing the motor polarity and thus its rotation. Without the use of an H-bridge, one would have to physically reverse the wires for the motor feed. If it is impractical with the use of wires, it is impossible with the tracks on a PCB. Instead, the H-bridge can conveniently reverse the voltage. First, have a look at the exemplary circuit in Fig. 8, take from the datasheet of the H-bridge (*5*).

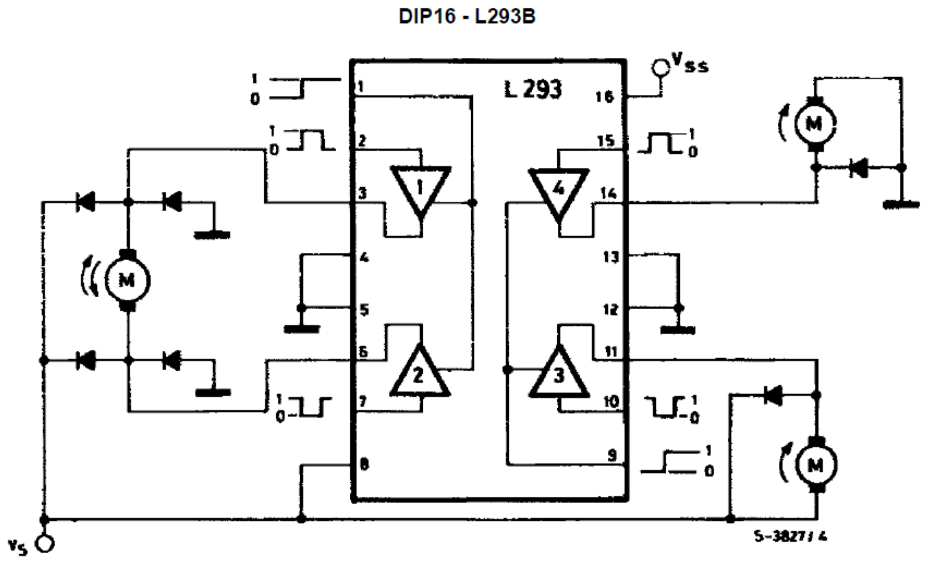


Figure 8: Exemplary motor connection from the H-bridge datasheet

We feed our PWM in pin 1. Pins 2 and 7 are used for controlling the direction of rotation of the motor. The truth table is simple: If one direction pin is 1 and the other 0, then the logical 1 defines the direction. If both of the pins are set to logical 0 or 1, the motor breaks. Notice how the pins 3 and 6 feed the voltage to the motor and are coupled with the direction pins.

As a last comment, in both Fig. 8 and Fig. 6 notice the existence of diodes. Again, these diodes have both a special name and a special function. They are called catch diodes and their use is nicely described in (*6*). In the same source, an extensive description of the functionality of an H-bridge can be found, in a beginner friendly way.

For our purposes, it suffices to say that the presence of catch diodes is necessary for the smooth functioning of our brushed motor. Consider the switching of direction phases or simply the breaking of the motor. The remaining current that was supplied to the motor has to go ("escape") somewhere. It is not a good idea to dissipate the current to our pins, without an alternative. The catch diodes provide this alternative.

A commonly used type of diode for this purpose is the Schottky diode (also the case in our circuit), which is characterized by a fast recovery time. Generally, when a diode switches from forward conducting to reverse current blocking mode, for a short period of time (recovery time) reverse current flows through the diode. Exactly this phenomenon is utilized in the catch diode case.

#### 4.1.4 Extra Button

Naturally, the more input options we have for our micromouse robot, the more flexible the commands we can issue and the more complicated the software can be. Therefore, we added a button (switch), which can be seen in Fig. 9.

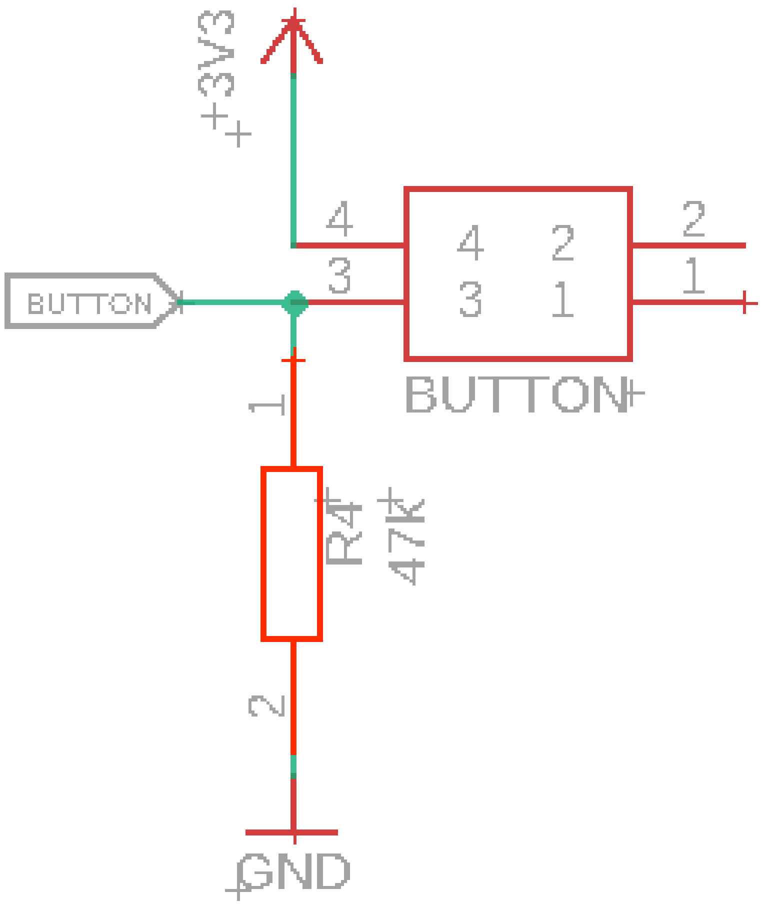


Figure 9: An extra button for input

A typical pull-down resistor of 47k is used, to make sure the mcu pin never has an undefined logical value (”floating”). When it comes to the resistance value, it is somewhat arbitrary in that it does its job. A different value could have been selected. For the current that flows through the resistor, when the button is pressed, we have:

$$V = I * R => 3.3 = I * 47 => I = 0.07mA$$

If a much smaller value for the resistance was used, then the current would be much larger, leading to unnecessary power dissipation on the resistance and heating. Also, in this case, there is the possibility for the input to the mcu pin to be stuck in a low, logical 0 level, no matter if the button is pressed or not.

### 4.1.5 Programmer

Describe the usage of the programmer.

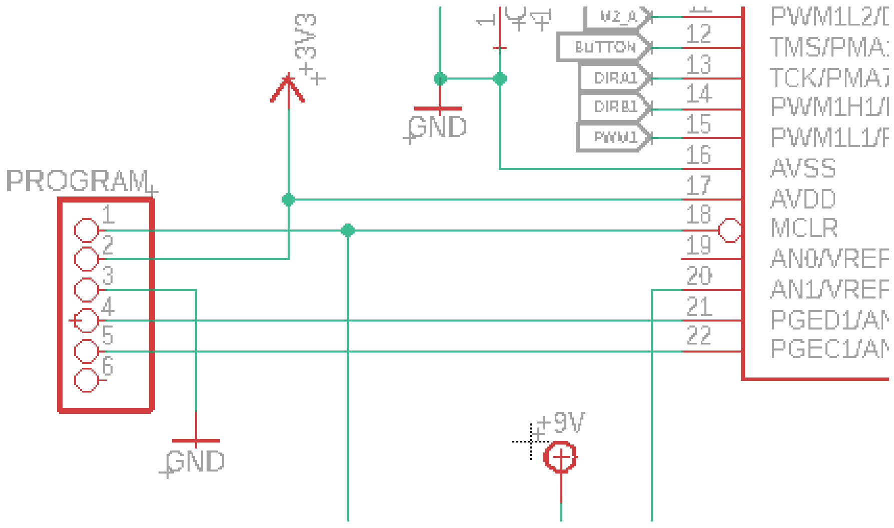


Figure 10: The programmer

### 4.1.6 Reset Button

Describe the need for a reset button. Describe the negative logic and pull-up resistor.

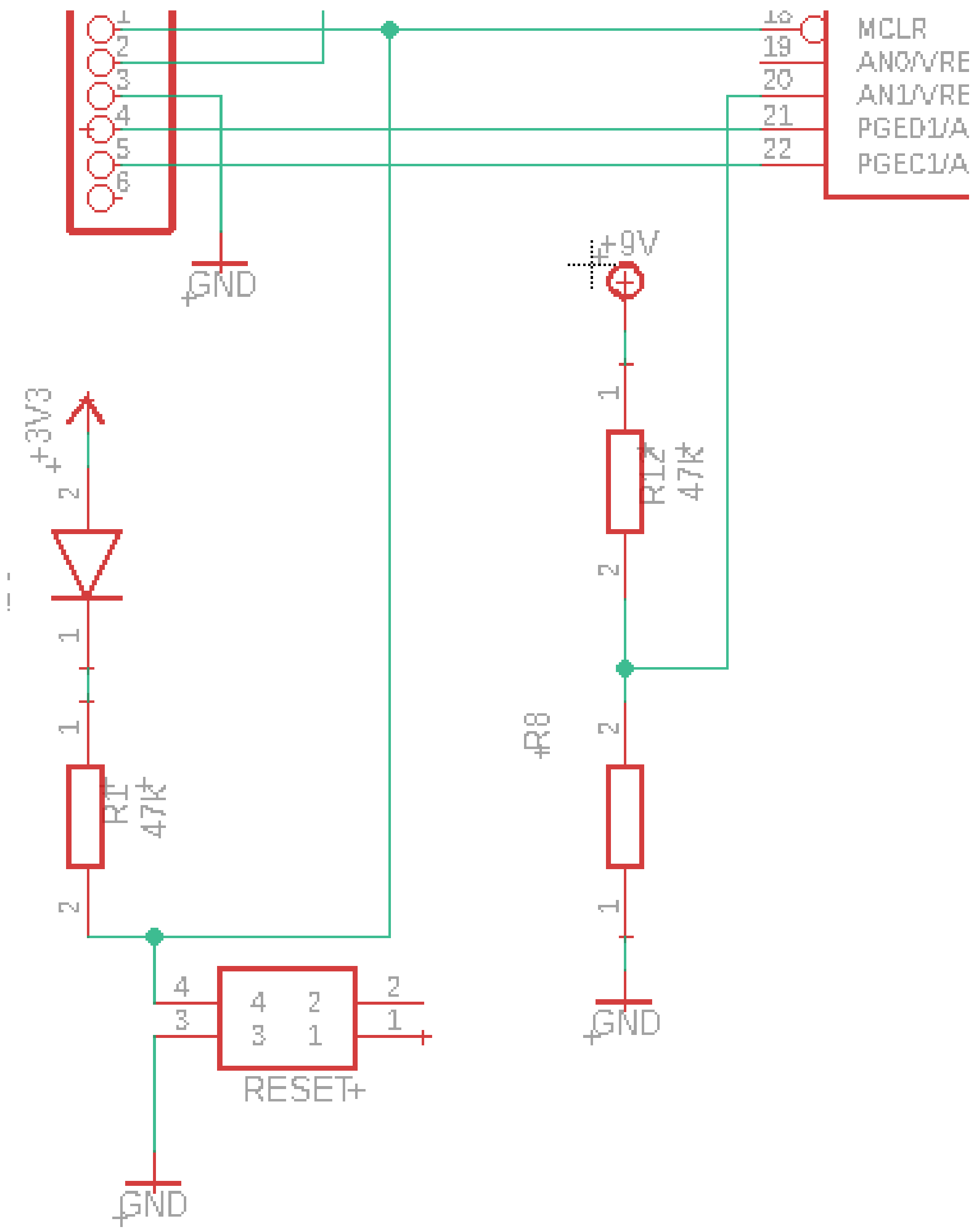


Figure 11: The reset button and the battery voltage measurement circuit

### 4.1.7 Battery Measurement

Describe the need for measuring the battery voltage level and adjusting the PWM accordingly. Describe the concept of voltage divider and the values chosen.

### 4.1.8 Universal Asynchronous Communication (UART)

Describe the need for UART, to issue commands at will.

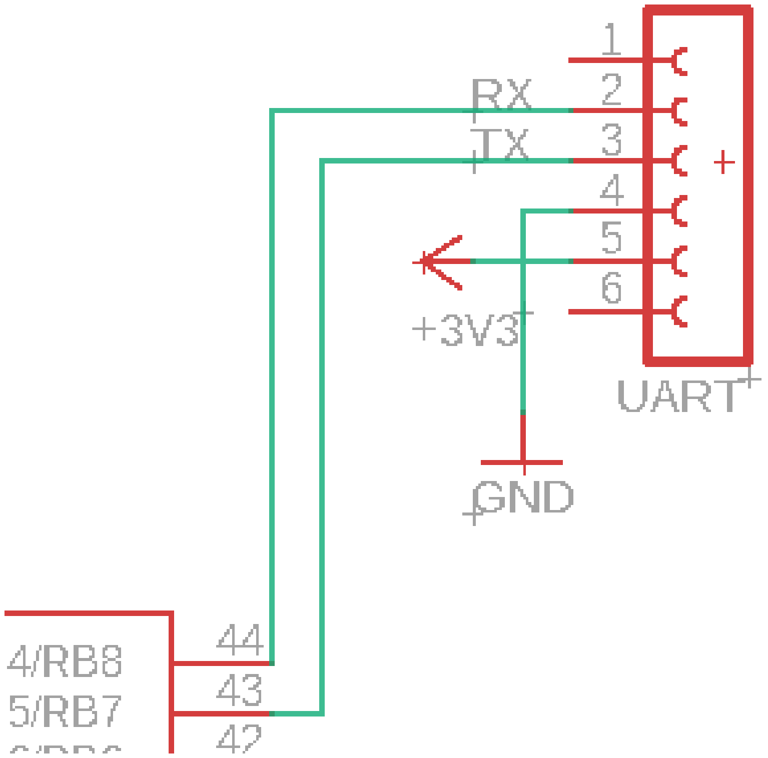


Figure 12: The UART module

### 4.1.9 LEDs

Describe the need for mechanical stability and the usage of LEDs. Say why we want to blink them. Describe why transistors are needed and briefly the way they are connected.

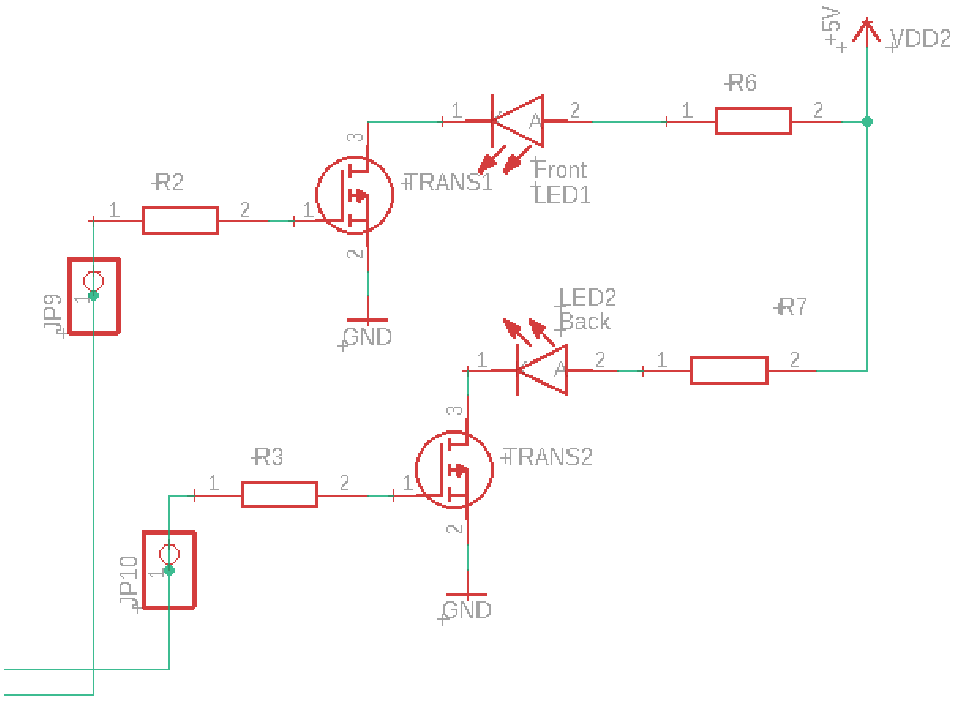


Figure 13: The LEDs used for mechanical stability

### 4.1.10 Oscillator

Describe the choice for an external oscillator and the need for capacitors (mcu datasheet).

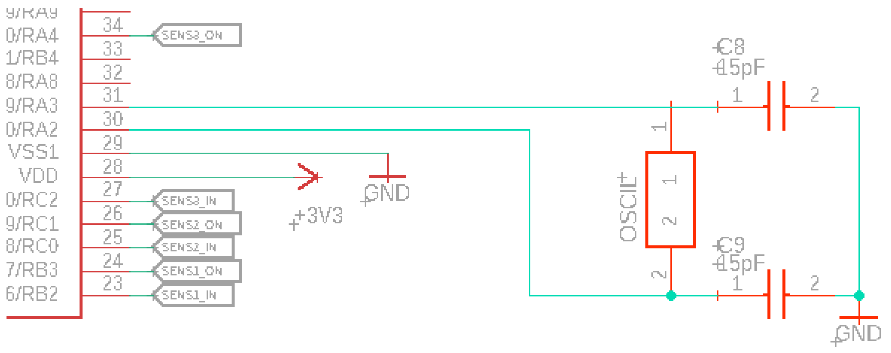


Figure 14: The external oscillator

#### 4.1.11 Distance Sensors

Describe the need for sensors and the way they are connected. Explain why we chose to have transistors there.

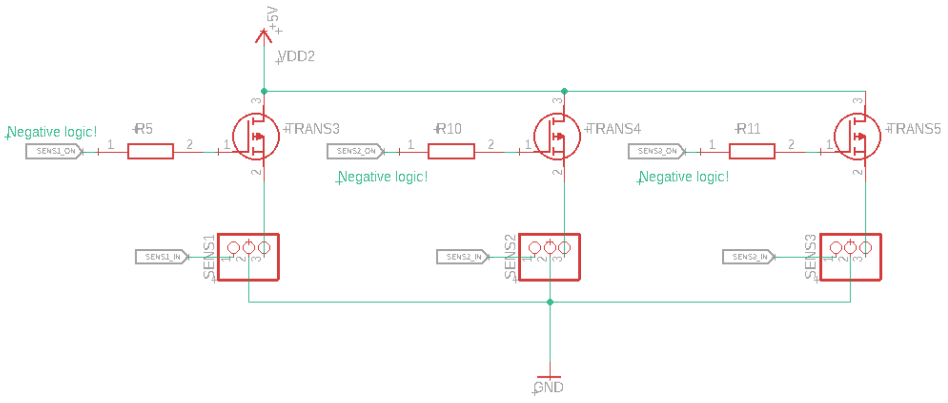


Figure 15: The distance sensors

## 4.2 Components

List the actual components we ordered. No need to explain the values for resistors and capacitors. Explain specific choices if needed.

## 4.3 Printed Circuit Board (PCB)

Comment on the restrictions of the PCB, dimensions and shape. Explain the components' placement choice. Describe the concept of 2 layers and via points. Describe the concept of ground plane. Comment on electrical noise, its sources and why it doesn't really matter in our case.

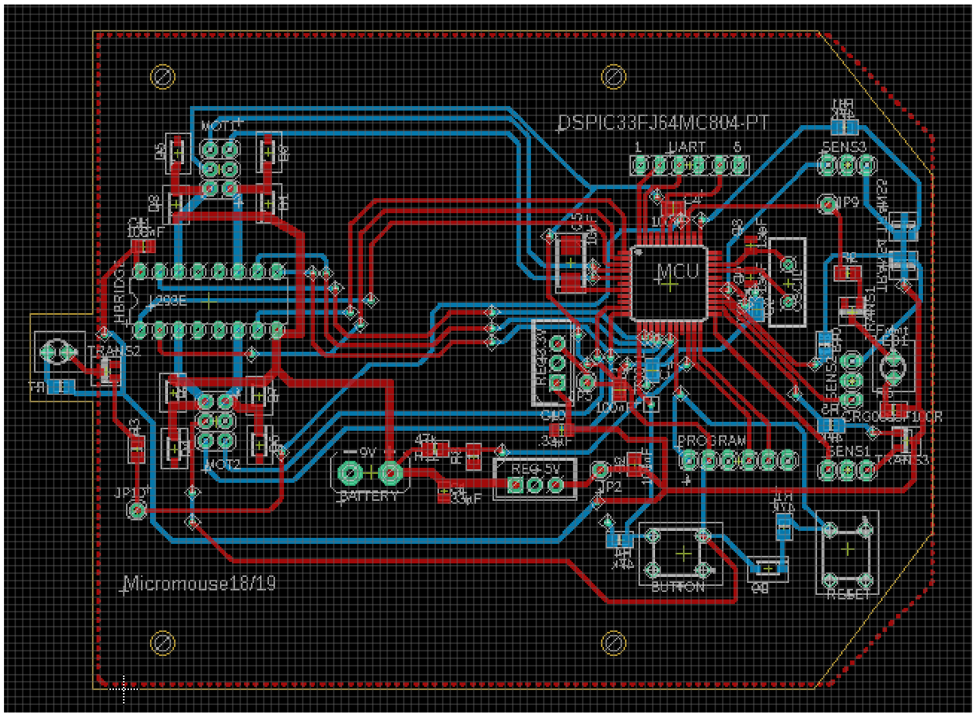


Figure 16: The PCB with both top and bottom tracks visible

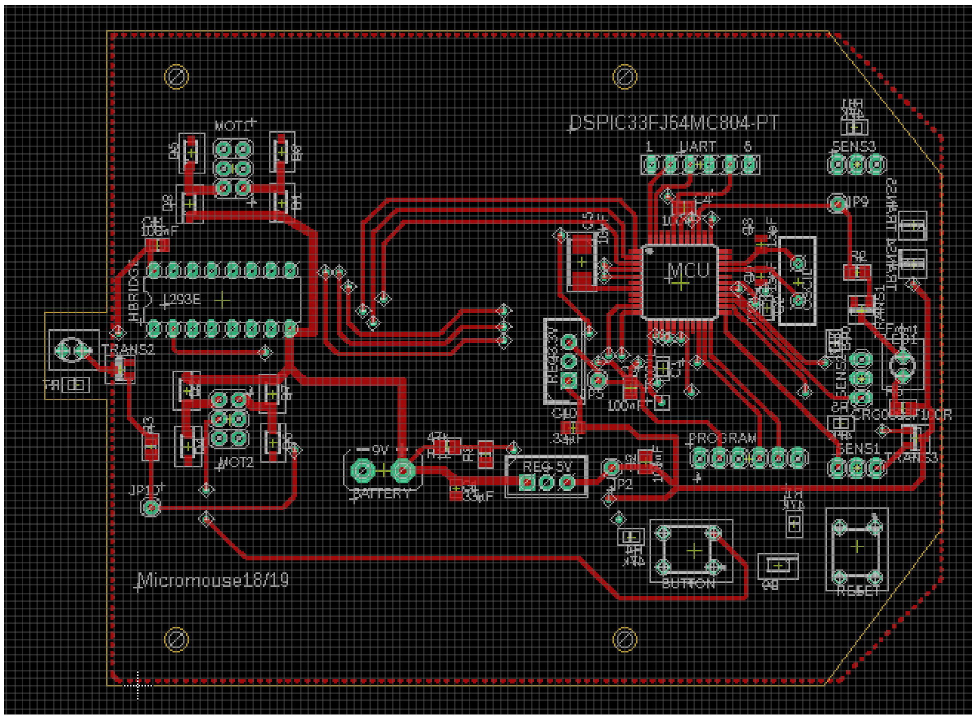


Figure 17: The PCB with only top layer tracks visible

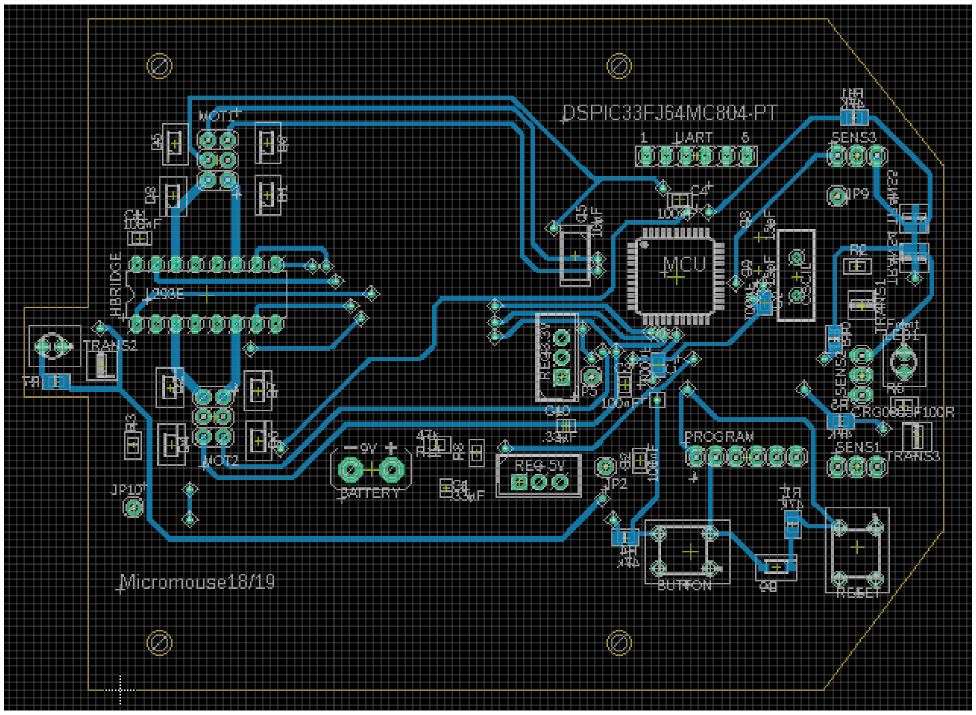


Figure 18: The PCB with only bottom layer tracks visible

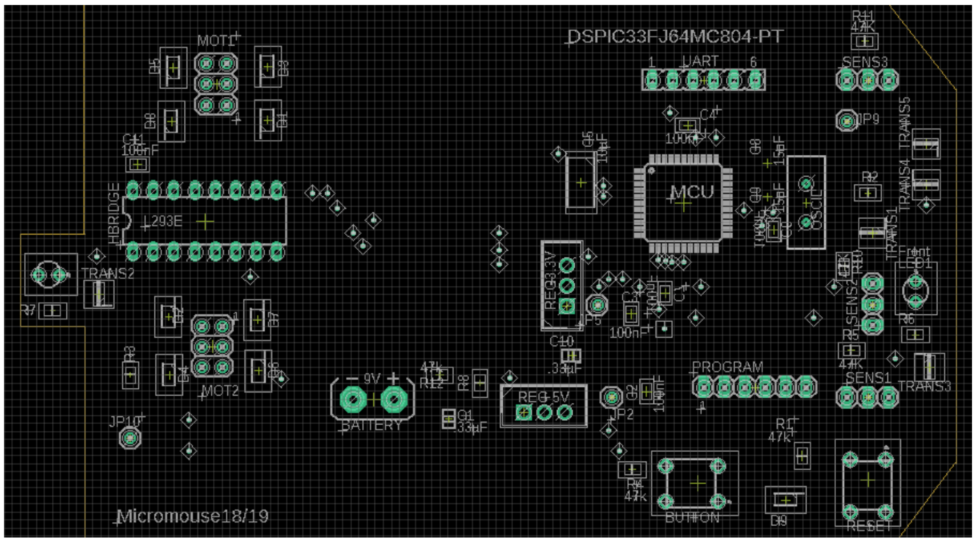


Figure 19: The PCB without tracks, only components visible

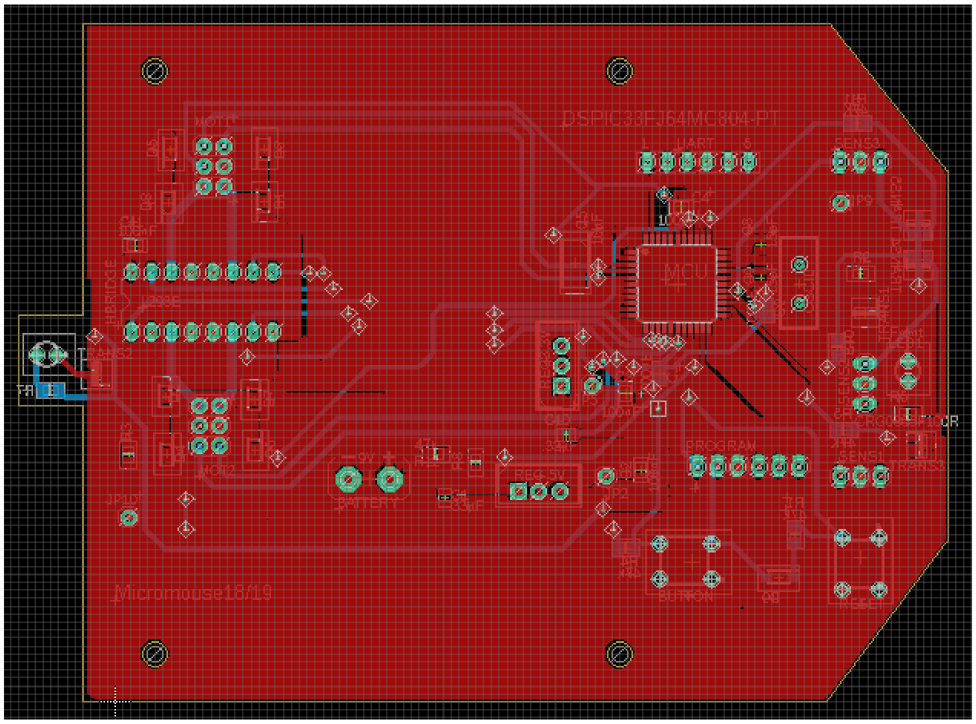


Figure 20: The PCB with the ground plane visible

## 4.4 Casing

Present the need for casing. Describe it. Present necessary pictures. Describe important calculations.

# 5 Software design

some general info

## 5.1 Peripherals

The description of our modular architecture, the work principles of the separate modules and basically ”how we control the peripherals” such as motors, leds, timers, etc. Logic comes a bit later in here. Also all info on DMA and pin remapping and our pretty mapping table are also welcomed here I think

**Timer 1**

Short description on how we worked out the time

**Pulse Width Modulation (PWM)**

how we chose the parameters

**Quadrature Encoder Inferface(QEI)**

how does qei work?

**Analogue-to-Digital Conversion(ADC)**

**Direct Memory Access (DMA) - does this belong here?**

**Universal Asynchronous Receiver Transmitter (UART)**

## 5.2 Pin Mapping

maybe a diagram of which peripheral maps to which pin

## 5.3 Controller Design and Approach

Here goes the whole logic of the mouse movement control (how should it behave when is faced with the wall or on contrary - with the gap in the labyrinth). PID controller design and the logic behind it.

### 5.3.1 Proportional-Integral-Derivative Controller (PID)

### 5.3.2 State Machine

# 6 Summary of tests

Here goes everything practical we could possibly test - for example the speeding curve of the motor, the temperature conditions of the board (when we'll solder it - whether it is able to work without overheating and such) Of course, the PID working values (such as the error convergence rate) Possible subsections:

- casing tests

- schematic tests

- software logic tests

# 7 Implementation challenges

During the whole implementation process we've faced a significant amount of challenges of a diverse nature. For a major part of them we were able to find a feasible solution that was implemented or at least managed to come up with some solution propositions. All the challenges can be split up in the application categories, which a listed below.

## 7.1 Organizational challenges

1. Organization of groups

   - Problem encountered:
     The main problem from the beginning of the practical course was to organize all the project members into equally skilled groups. The splitting principles were also not clear.

   - Solution found:
     The first solution we came up with was to split up into 2 groups of 3 people, which proved efficient enough (or rather the inconveniences didn't show up) at the initial working stages such as acquiring the initial knowledge and getting familiar with the tools. After some time it became clear, that this group separation doesn't work anymore, because the workload of the whole project stretched throughout the remaining time was too much for 3 people to carry out, so it was decided to unite the groups and continue working on the project jointly. This changing of structure mid-project also led to further complications in regards to management within a group, which will be discussed next.

2. Organization and management within a group

- Problem encountered:

  In terms of the management within the group, the major problems that were encountered are:

  - Lack of high-level working plan and working structure that everyone can stick to.

  - No clear role assignment for the project members, lack of the task management.

  - Lack of day-to-day updates and communication.

  - No unified repository for everyone to work with and further complications tied with that (follows the problem with groups separation mentioned previously)

- Solution found:

  The solution that helped to fix (to some degree) first two problems was to (following the unification of 2 groups into one single unit) get rid of the multitasking paradigm and to assign some specific task or ”area of expertise” to each member of the project. Thus, after such decentralization the need for a centralized high-level working plan was decreased, because each participant could organize the working process to fulfill own needs the best. This of course led to deepening the problem of lacking communication and the solution was not the most efficient in terms of implementation speed. The communication problem tied to the absence of a single unified repository for each project member to work with was solved in the second half of the project using GitHub.

## 7.2 Software design challenges

## 7.3 PCB design challenges

## 7.4 Casing design challenges

*Well, here go all the problems we faced coupled with our solutions for them. Potentially the longest part in the report. Can be split in parts similarly to the previous sections (talking about* *software, board and casing)*

# 8 Conclusions

Here - short summary of the achieved results, maybe some general words and praises for our final mouse version, just something positive to end the mouse story well.

## 8.1 Expectations

Slightly controversial part, we could omit it (I would like not to though) or rephrase it somehow. Basically - what did we expect from the course. Not sure if this part actually belongs here, but for now it works.

## 8.2 Propositions

Our suggestions to maybe somehow improve or better organize some parts of the praktikum or the task or whatever.

## Appendix

some extra figures

## References

1. R. Misra, R. Adler, *Region 2 IEEE SAC 2018 - University of Pittsburgh* (2018).

2. J. Wu, Micromouse project (2015). Https://jerry1100.github.io/projects/micromouse.

3. Microchip Technology Inc., *dsPIC33FJ32MC302/304, dsPIC33FJ64MCX02/X04 and dsPIC33FJ128MCX02/X04 Data Sheet, High-Performance, 16-bit Digital Signal Controllers* (2011).

4. FAULHABER, *DC-Gearmotors 100mNm*.

5. STMicroelectronics, *L293B, L293E, PUSH-PULL FOUR CHANNEL DRIVERS* (2003).

6. A. Tantos, H-bridges - the basics, `http://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics`.

Some guidelines from the template

1. Please follow the style for references outlined at our author help site and embodied in recent issues of *Science*. Each citation number should refer to a single reference; please do not concatenate several references under a single number.

2. Please cite your references and notes in text *only* using the standard LaTeX `\cite` command, not another command driven by outside macros.

3. Please separate multiple citations within a single `\cite` command using commas only; there should be *no space* between reference keynames. That is, if you are citing two papers whose bibliography keys are `keyname1` and `keyname2`, the in-text cite should read `\cite{keyname1,keyname2}`, *not* `\cite{keyname1, keyname2}`.

Failure to follow these guidelines could lead to the omission of the references in an accepted paper when the source file is translated to Word via HTML.

## Handling Math, Tables, and Figures

Following are a few things to keep in mind in coding equations, tables, and figures for submission to *Science*.

**In-line math.** The utility that we use for converting from LaTeX to HTML handles in-line math relatively well. It is best to avoid using built-up fractions in in-line equations, and going for the more boring "slash" presentation whenever possible — that is, for `$a/b$` (which comes out as $a/b$) rather than `$\frac{a}{b}$` (which compiles as $\frac{a}{b}$). Likewise, HTML isn't tooled to handle certain overaccented special characters in-line; for $\hat{\alpha}$ (coded `$\hat{\alpha}$`), for example, the HTML translation code will return [^$(\alpha)$]. Don't drive yourself crazy — but if it's possible to avoid such constructs, please do so. Please do not code arrays or matrices as in-line math; display them instead. And please keep your coding as TeX-y as possible — avoid using specialized math macro packages like `amstex.sty`.

**Displayed math.** Our HTML converter sets up TeX displayed equations using nested HTML tables. That works well for an HTML presentation, but Word chokes when it comes across a nested table in an HTML file. We surmount that problem by simply cutting the displayed equations out of the HTML before it's imported into Word, and then replacing them in the Word document using either images or equations generated by a Word equation editor. Strictly speaking, this procedure doesn't bear on how you should prepare your manuscript — although, for reasons best consigned to a note, we'd prefer that you use native TeX commands within displayed-math environments, rather than LaTeX sub-environments.

**Tables.** The HTML converter that we use seems to handle reasonably well simple tables generated using the LaTeX {`tabular`} environment. For very complicated tables, you may want to consider generating them in a word processing program and including them as a separate file.

**Figures.** Figure callouts within the text should not be in the form of LaTeX references, but should simply be typed in — that is, `(Fig. 1)` rather than `\ref{fig1}`. For the figures themselves, treatment can differ depending on whether the manuscript is an initial submission or a final revision for acceptance and publication. For an initial submission and review copy, you can use the LaTeX `{figure}` environment and the `\includegraphics` command to include your PostScript figures at the end of the compiled PostScript file. For the final revision, however, the `{figure}` environment should *not* be used; instead, the figure captions themselves should be typed in as regular text at the end of the source file (an example is included here), and the figures should be uploaded separately according to the Art Department's instructions.

## What to Send In

What you should send to *Science* will depend on the stage your manuscript is in:

- **Important:** If you're sending in the initial submission of your manuscript (that is, the copy for evaluation and peer review), please send in *only* a PostScript or PDF version of the compiled file (including figures). Please do not send in the TeX source, `.sty`, `.bbl`, or other associated files with your initial submission. (For more information, please see the instructions at our Web submission site, http://www.submit2science.org/ .)
- When the time comes for you to send in your revised final manuscript (i.e., after peer review), we require that you include all source files and generated files in your upload. Thus, if the name of your main source document is `ltxfile.tex`, you need to include:
  - `ltxfile.tex`.
  - `ltxfile.aux`, the auxilliary file generated by the compilation.
  - A PostScript file (compiled using `dvips` or some other driver) of the `.dvi` file generated from `ltxfile.tex`, or a PDF file distilled from that PostScript. You do not need to include the actual `.dvi` file in your upload.

- From BIBTEX users, your bibliography (`.bib`) file, *and* the generated file `ltxfile.bbl` created when you run BIBTEX.
- Any additional `.sty` and `.bst` files called by the source code (though, for reasons noted earlier, we *strongly* discourage the use of such files beyond those mentioned in this document).

## References

1. R. Misra, R. Adler, *Region 2 IEEE SAC 2018 - University of Pittsburgh* (2018).
2. J. Wu, Micromouse project (2015). Https://jerry1100.github.io/projects/micromouse.
3. Microchip Technology Inc., *dsPIC33FJ32MC302/304, dsPIC33FJ64MCX02/X04 and dsPIC33FJ128MCX02/X04 Data Sheet, High-Performance, 16-bit Digital Signal Controllers* (2011).
4. FAULHABER, *DC-Gearmotors 100mNm*.
5. STMicroelectronics, *L293B, L293E, PUSH-PULL FOUR CHANNEL DRIVERS* (2003).
6. A. Tantos, H-bridges - the basics, `http://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics`.