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

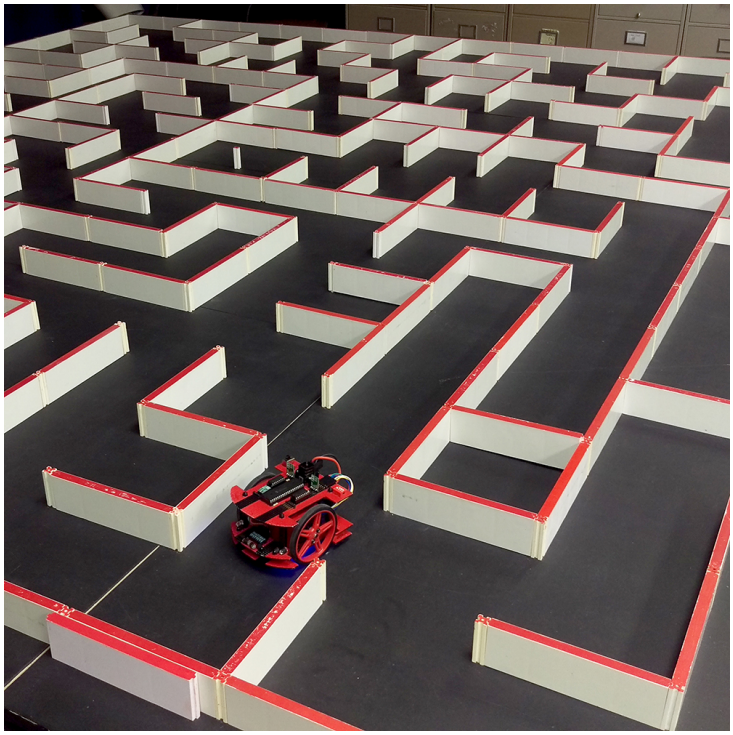

Figure 1: Micromouse competition photo: labyrinth and robot example (*2*)

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

Some introductory info

4.1 Schematics

General plan, choosing components, etc.

4.2 Components

The spreadsheet with all the data

4.3 PCB

Board design and specifications *The complete description of our schematics design and board design and what we did and why Also including all calculations we've managed to collect throughout the whole course of said design. Pretty pictures and some tables with calculated values would be amazing.*

4.4 Casing

The whole journey on the casing design. Pictures of the final 3D model and printed model of course, some calculations and data on the sizes and maybe geometrical values

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

5.2 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.

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

7.1 Organizational challenges

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.

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 [ˆ(α)]. 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.