Dear Keijo and Joseph,

Thank you for all the hard work (time, soul...) that you put into the course.

I have joined Metropolia to learn programming and Smart Devices was for sure a brilliant lesson.

Attached codes:

LINE – Patricie – used in competition

SUMO – Patricie – used in competition

MAZE – Feihua – used in competition

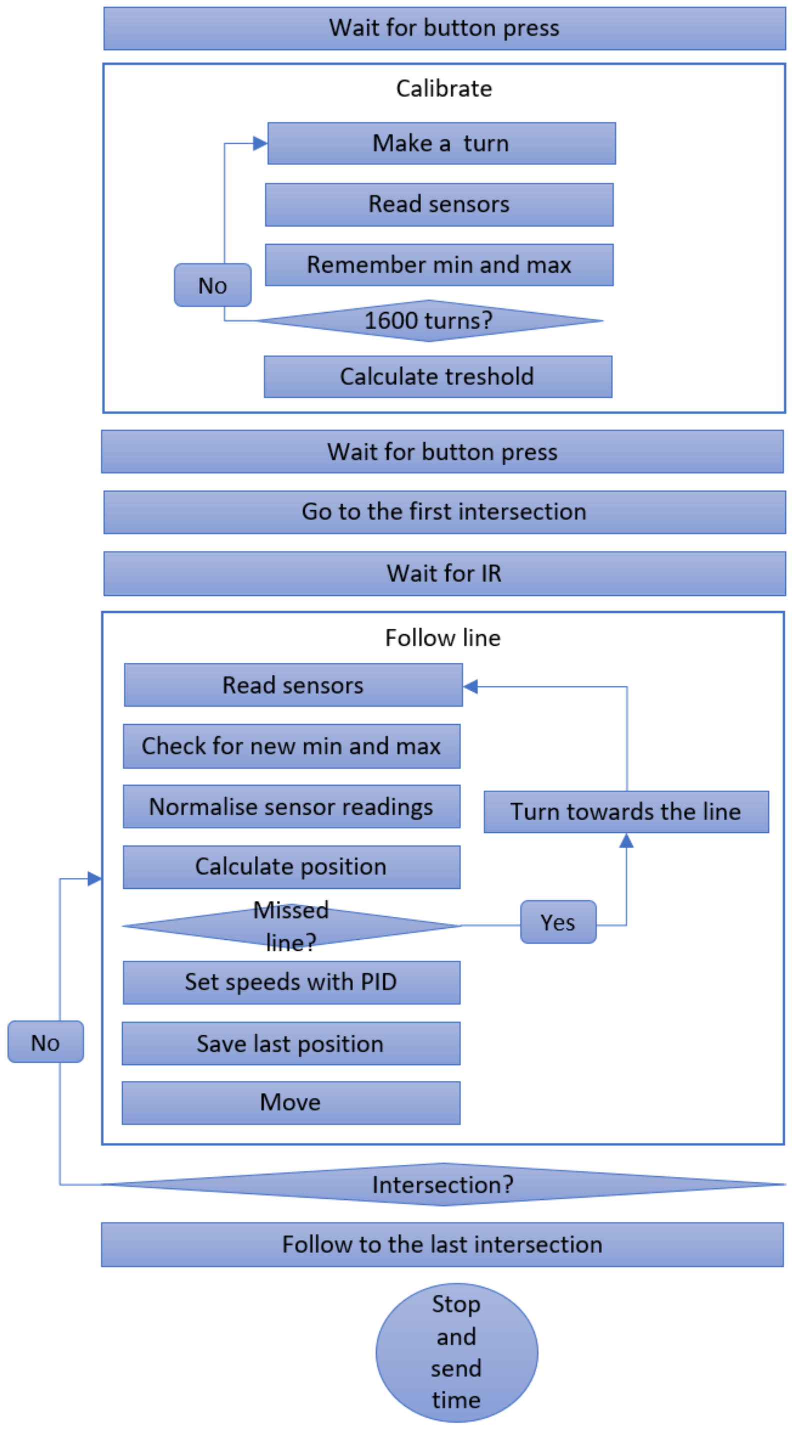
MAZE_X – Patricie – abandoned project

MAZE_R – Patricie – code from ronimk

My goal during the course was to learn as much as I can. Here is how it went:

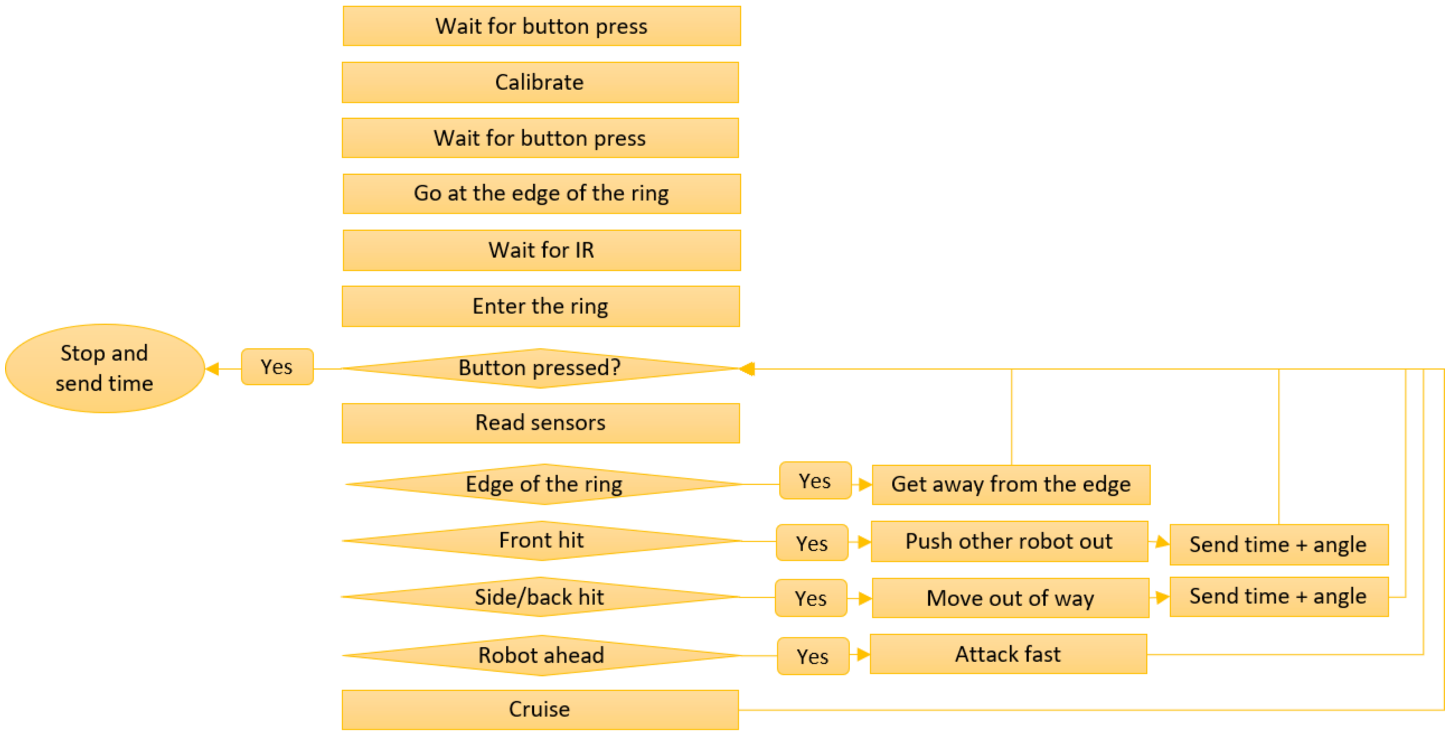
LINE

I created several codes, using 1, 2 or 6 sensors, analog and digital, PID and using conditions or weights. Together about 15 different codes and many variations. The purpose was to see how they behave and what are their strong and weak sides. My favorite code was 1 sensor PID which could do at 255 when the conditions were right, but not reliably. Highest reliable speed was 150. I added a rule there for side switching which decides which side of the line to follow. Unfortunately, I did not have enough time to adjust the PID formula for the other side. The most reliable code I made was 6 sensors weighted PID. It is reliable at speed 255 including stopping at an intersection. That was the code I picked for the competition. It sends a message when the robot misses the line.



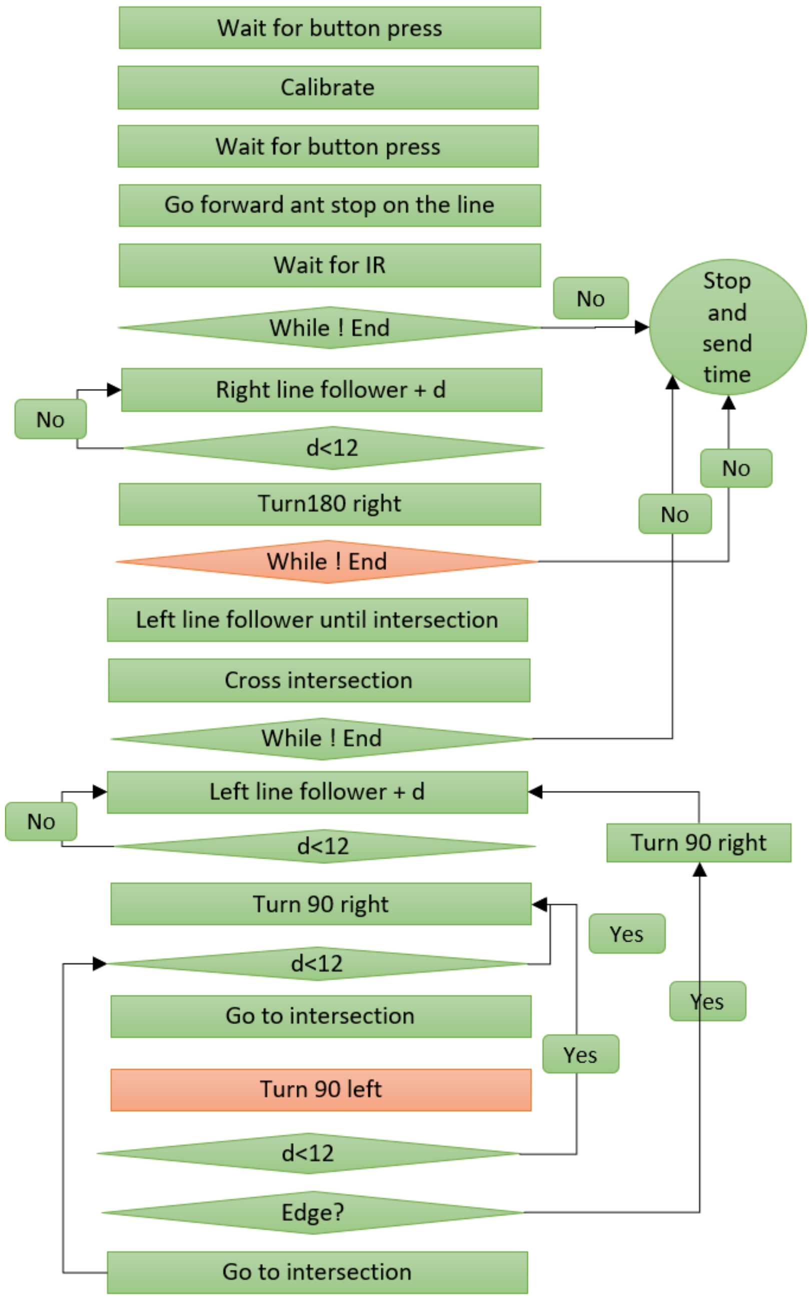
SUMO

It has also calibration for the accelerometer and sends angles after collision.

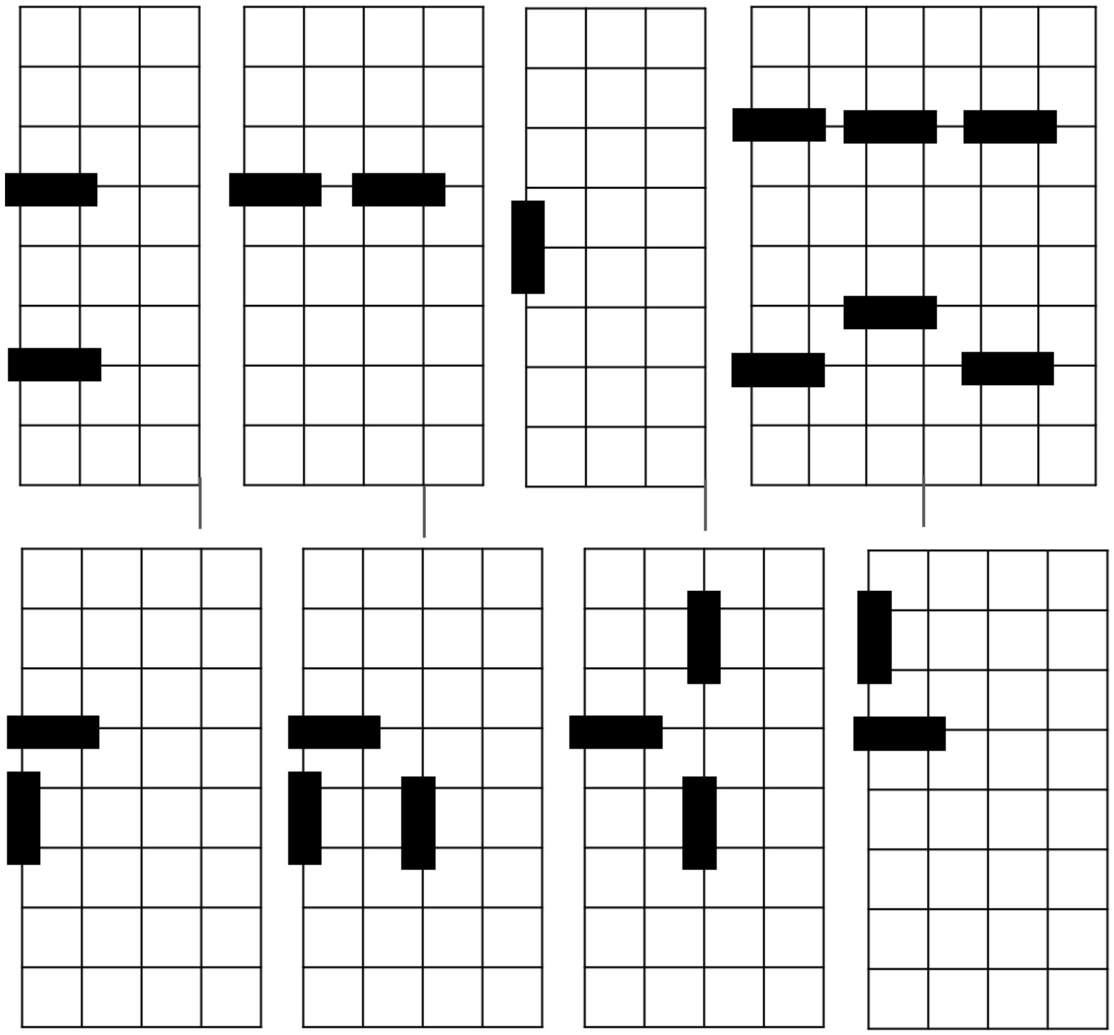


MAZE_X

My first maze attempt is attached; however, it has never been finalized. Halfway through I realized that the challenge of making the maze work without coordinates is bigger than me (and the time I had). I could use time to detect the end line or I could remember the previous reflectance values (or both). Sometimes that worked but often I had false positive readings before the end line. The concept of this maze solver was to follow the left edge of the maze and in case there were obstacles then follow the left-hand rule. But first of all the robot checked the right edge if it was free. If not, it abandoned the option. Checking the right edge first was necessary for cases when the blocks lead the robot from the left edge all to the right edge.



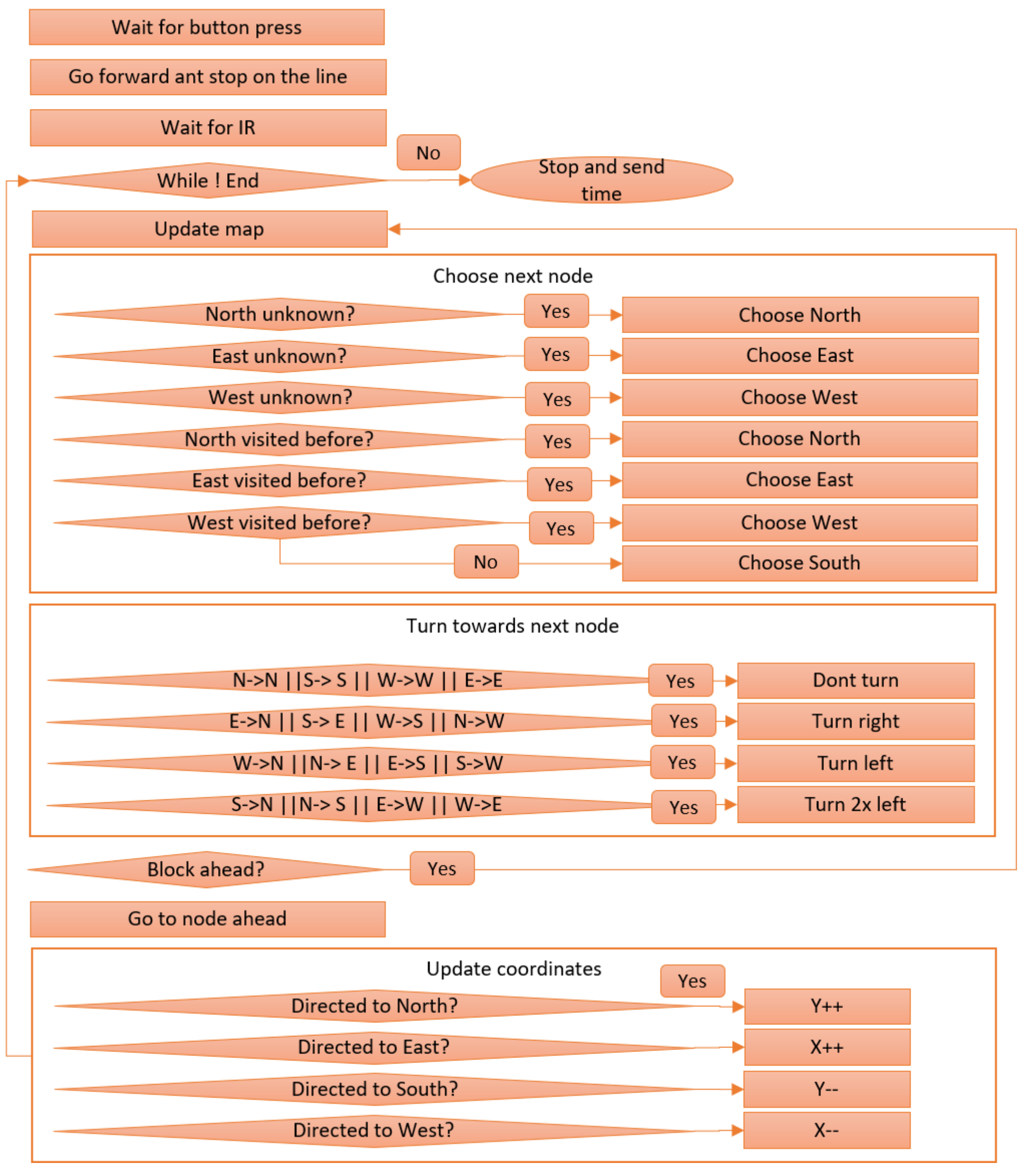
Tested scenarios



MAZE_R

With almost no time left and robot being full time used by the rest of the team I decided to take it from the other end. I went on GitHub, found codes from previous years (48 search results for “mqtt zumo maze”), tested them and picked the most reliable one. The code was at that point far from what I could write. My goal was to read it, understand it and document it. All credits for the code go to ronimk, whoever that is:
https://github.com/ronimk/Zumobot_project/blob/master/Zumo/ZumoBot.cydsn/main.c

Half of the code were in-progress attempts and many things could be simplified. I did not have enough time to pull the cleaning of the code to the end though I reduced it from 850 to 480 lines. I have learned much more from this code than if I coded the maze in my own way. It had switches, cases, custom data types and binary shifts – things I have never used before. The code was easy to understand. This is how it works:



I tried to document all my codes in such way that even I could understand them one year from now.

Functions were not used for pieces of code that are used only once. No time for that.

If you read all the way here, I would highly appreciate your feedback, especially when it comes to practices of making the code more robust and reliable across different machines.

Thank you again

Patricie