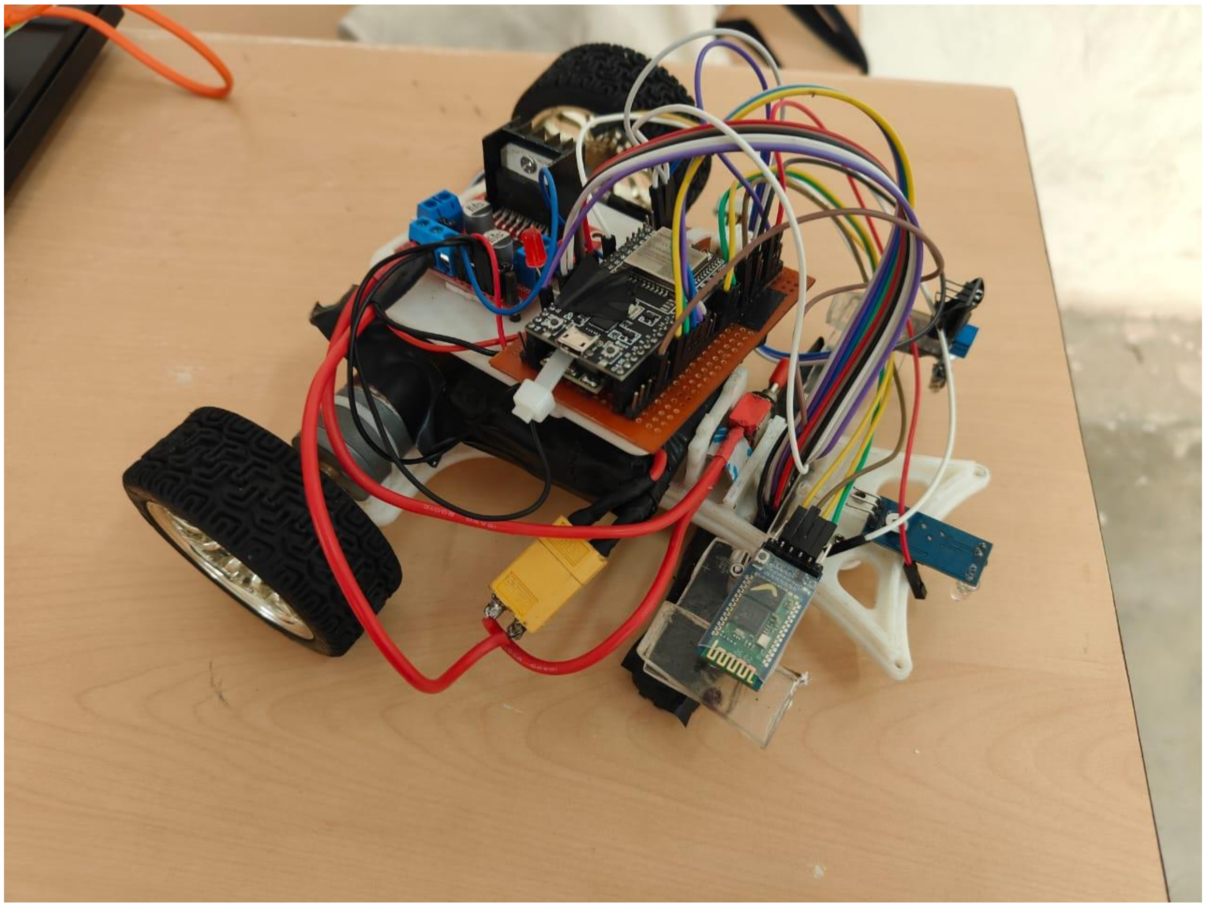


ZIPLINE

Introduction

So this october, Our special team decided to participate in the IITB Techfest Zonals as is it's tradition, we usually participate in 2 competitions, Cosmoclench & Meshmerize, with 2 teams for each competition

Cosmoclench is simple at heart, you make a RC bot with a human driver with an arm or gripper capable of picking up a specified object and traversing an obstacle course while carrying it

In Meshmerize (what I participated in), you make a line follower that can solve line mazes, this is different from making a micromouse since those solve mazes with physical walls, we are operating with inverted mazes (no walls, just lines)

Not sure how to structure this document as I am writing it in retrospect, but I think I'm just going to start talking and organize the result at the end 👍

This is mostly from my own experience and I'm sure details will be missing in places I wasn't too involved in – especially chassis and electronics and such, It is my hope that my team members will help fill in the gaps

Design

Since we were given a deadline to come up with a parts list and a basic vision for our bot after **many** discussions we decided to build a bot with the following:

- **L298 Motor Driver**: Because they are relatively easy to come by and we were already familiar with using one, not to mention we already had some of these, These properties came in handy since we burnt through 4 of em just doing construction and it didn't give us problems once we cleaned out any motor related electrical issues,

Overall I think it was a alright decision and I don't think I would swap it out unless for something a bit more robust with a better voltage regulator

- **ESP32 WROOM 38pin**: We decided on this board because it had sufficient pins to accommodate our 16RC and enough left over to connect everything else besides having built-in Bluetooth and Wi-Fi, though that didn't help much as during testing using Wi-Fi would cause it to heat up alarmingly and using BT would disable pins that we depended on so we resolved to attaching a BT module (HC-05) 🤡

I don't think this was a bad choice but knowing more about the quirks of the ESP and being **very very careful** with that 3.3v and 5v rail difference would have saved us a lot of headaches and debugging time & in all honesty considering our experience level an arduino mega would have served us better

- **Polulu 16RC QTR IR Sensor**: This was our main sensor and so nearly everything was designed *around* it, It's an expensive sensor too and not one that's easy to find so I'd say we were lucky to call dibs on it before the other team did though in the end it didn't matter

I'm just disappointed we were not able to make the full use of this sensor, since we weren't able to harness it for anything besides providing error values to our PID control system

- **3D Printed Chassis**: *<HARSH WRITE SOMETHING THERE THANK YOU>*
- **N25 Motors w/ Encoders:** We were recommended to use these motors by several people and not knowing any better decided it was as good an idea as any, and I'm not entirely sure but I think they are part of the reason that doomed us, We weren't able to source fresh motors from anywhere so after quite some time we finally got our hands on motors from a senior but after modification to change their gearboxes, we unfortunately ended up with differing gear ratios and encoders that were not corresponding to each other

The critical flaw that while we were aware of it only discovered how detrimental it would be at the end of the competition is that the motors cut out at something like 3v, since they are designed to operate at around 10-11v and due to this we were not able to get PID to make as smooth movements as we wanted.

I think that about covers all the mission critical parts and other smaller parts aren't of too much consequence and I have probably forgotten them 😅

Code Architecture

Okay so I will probably talk the most here since I think I spent time writing code for the Bot, Initially while waiting for components to arrive, I wanted to at least be able to complete an implementation of LSRB with some kind of Maze solving and then a way to find the optimal path,

To support this research, I created a very simple simulator in C with Raylib (undefinedDarkness/line-follower-sim) wherein one can easily test drive various algorithms and see how the bot would behave traversing several mazes, Eventually I hope to make it easier to use and potentially create a web based version to make it more accessible

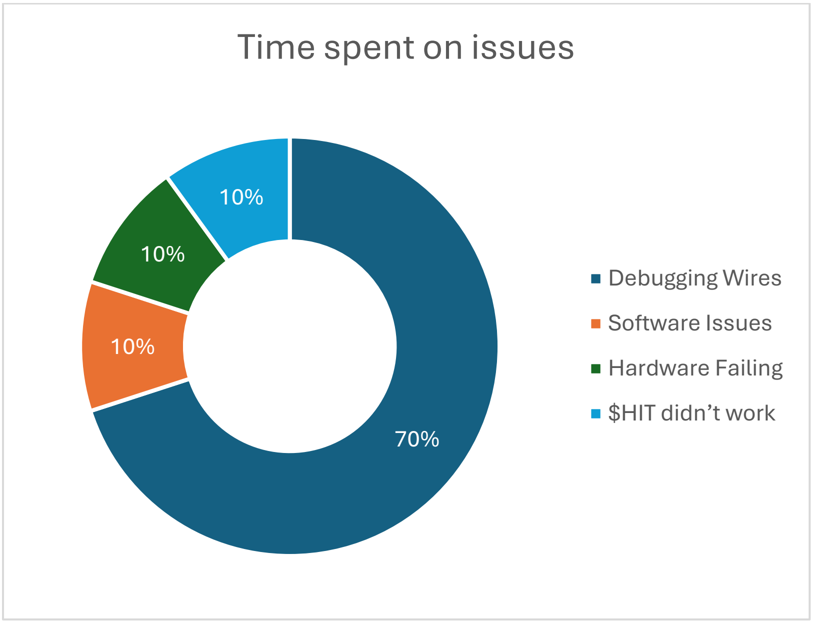
Using this simulator, we were able to successfully implement a working version of the algorithm described in 1410.4145, and after that utilizing a simple matrix mapping we were able to get to finding a "more optimal" path using A*

In the end, we only got as far as implementing LSRB on the bot and even that used to not work quite right for while the logic was sound we were simply unable to reliably detect intersections which was a problem caused by the combination of simply brittle intersection detection code and a sub optimal PID (caused by hardware issues)

This document was also helpful, Slide 1

Implementation and issues faced

This is easily going to be the longest section as compared to our planning and initial building phase, ironing out all the issues took forever and we never really finished in the end, If I had to guess our time was spent something like this



The Story

Our initial plan was something like this:

1. Gather components
2. Build a bot that can follow a line (Line Follower)
3. Figure out how to do the maze solving part

Considering we had only drafted this plan on September 20th and we had to *go for the competition* on the 8th of October, we were none too confident but we tried our best with that time limit looming over our heads but by some grace of God before we were set to go to Bangalore, The competition was postponed to the tail end of October on the 26th, Which worked out really handily for us since we didn’t have anything to show on the 8th, We had just had a couple days to work on it after spending like 2 days sourcing a N25 motor and getting it all assembled and as such had very little time to tune PID on the bot so it was barely able to follow a line.

Even getting our components was an adventure, most of the parts we had decided on we were able to get quickly but the initial set of motors we acquired, didn't work so we had to sit on our hands for about a week and a half to get a set that worked at all, and we burned quite a few of the motor drivers so we had to get and keep spares for that, The sensor we got from the team luckily since it's a hard part to find on its own

As the timeline was shifted, We all took a large breath of relief, It had given us just enough breathing room to actually have something to go to Bangalore with.

This is where we had one of our fumbles I’d say is that we weren’t able to keep our high pitched intensity from that hectic week (before the 8th) continuing on till the 26th, We slowed down by a lot and for a couple works not much work was done since we had our 2nd batch of mid sem exams, Even during the exams we worked what little we could, creating charts for testing and doing whatever small modifications we could do while still waiting for some parts.

<- *Not enough detail here, if anyone can write about what all we did here, that’d be great* ->

During this time, we spent a good week and a half debugging issues with the bot and during this time we burnt through like 5 L298N’s but by the end we had a mostly functioning bot that we could trust wouldn’t fail suddenly on us.

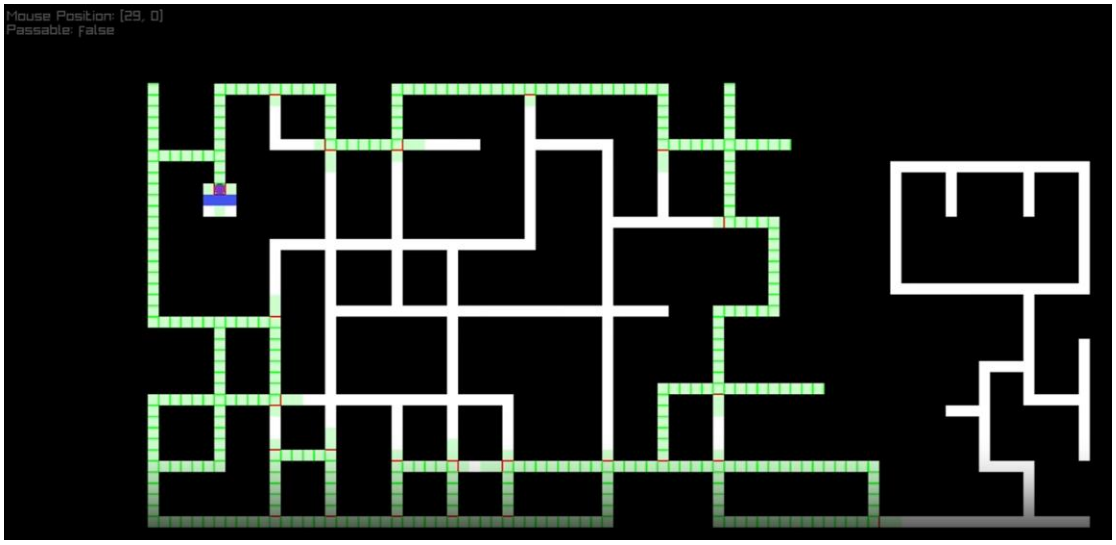
Obviously I’m glossing over heavily here as it’s been a while since then and my memory fails me.

Some of the issues we faced:

- Weak connections due to unreliable jumper cables
- Issues with power since the ESP32 MCU works on a 3.3V rail compared to 5V

After a couple days work on tuning the PID to a reasonable level we started work on the maze solving part (This is like 1 week before the competiton now)

We had already written a fully working algorithm for LSRB maze navigation and later A* for path optimization along with a specialized simulator for the same


Mouse Position: [29, 0]
Passable: false

And porting it to ESP, wasn't a big deal since we had already written it in C11, but this is where we faced one of big issues, **Intersection Detection**

For some reason, no matter what we did we cou

ldn't get the intersection detection to work smoothly, our QTR16 while fine for the PID control, simply couldn't reliably detect the intersections and in the end we even tried a janky solution with 3 array of individual IR sensors positioned far enough to not cause issues *(if only they worked),* In the end we figured out that it was also due to our N25 motors not being precise enough to do as minute movements as we required for smooth PID so our PID was more jank than we could allow (*that was the death kneel)*

It's been a while so I'm not really able to write with as much detail and knowledge like I could at the time but I will say it was a very let's say *humbling* experience and I learnt a lot internally even if I couldn't list it out even if I wanted to.

Oh yeah, the motors we got had encoders which we spent a lot of time trying to get working as it would allow us to accurately measure the distance we had travelled and use that in our control code (to stop at the end of every 30cm segment) but that didn't simply work as in they were inconsistent with themselves and eventually on a senior's recommendation we even wondered if we should try doing wheel odometry and like taking several readings of the encoders at various speeds to try and derive some kind of relation to get some distance out.