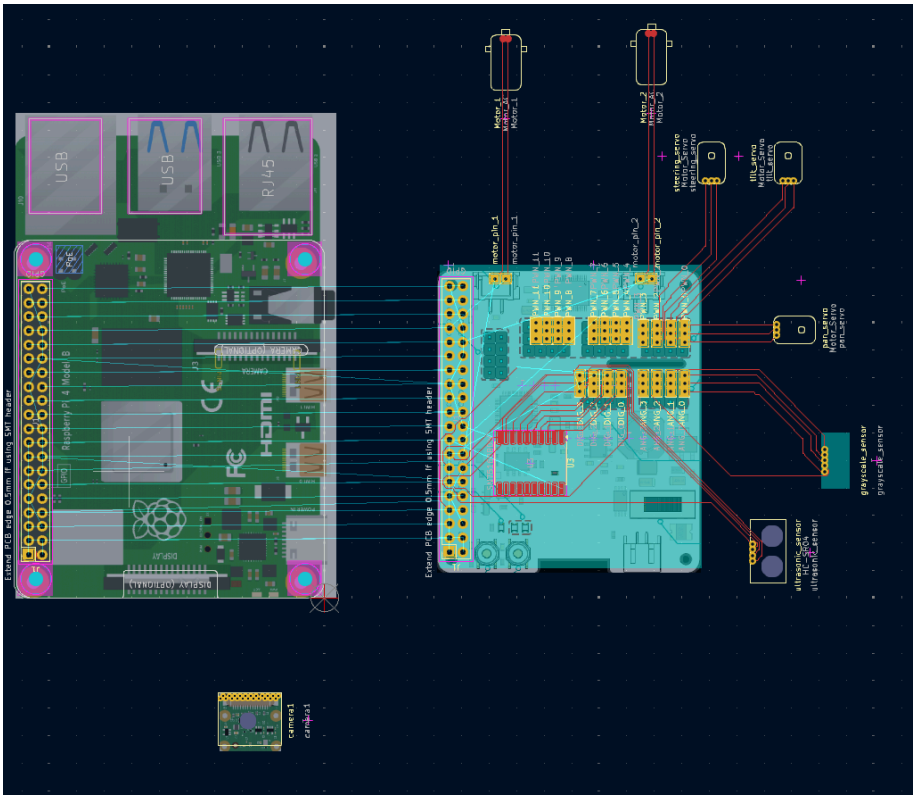
CS437 - IoT
Lab 1.1
Name: Lucas Damler
NetID: ldamler2
[Code Walkthrough Video]
[Self Driving Car Demo Video]

PCB Design:

I faced several challenges during the pcb design phase. For starters I am not a “real” engineer. One of my worst classes in undergraduate was ELE 100 in which I actually did worse in that than the x86 assembly class. I struggle to understand circuits and can say with certainty I never want to have to be in charge of physical hardware design. That being said, in the same way I would probably never want to write assembly regularly as a job I can appreciate having the ability to read through it at the very least. Especially as someone who enjoys the dangers of writing C sometimes you have to read through an assembly dumb and it’s nice to be able to understand what I’m seeing. I will probably avoid a job that requires me to do production quality PCB design at all cost but in the same vein I really appreciate being able to look at a schematic and understand the complexities of it even if I couldn’t design it well myself

The other main issue I faced was being stuck with the picar-x most of the footprints and symbols didn’t exist so I had to spend days researching and reading through spec sheets trying to figure out the different components and pieces that made the robot hat. My design ultimately isn’t perfect and probably isn’t printable but the struggle of trying to put this together really did help me get a greater understanding of hardware design choices and I know will be invaluable when I begin planning and designing my final project.

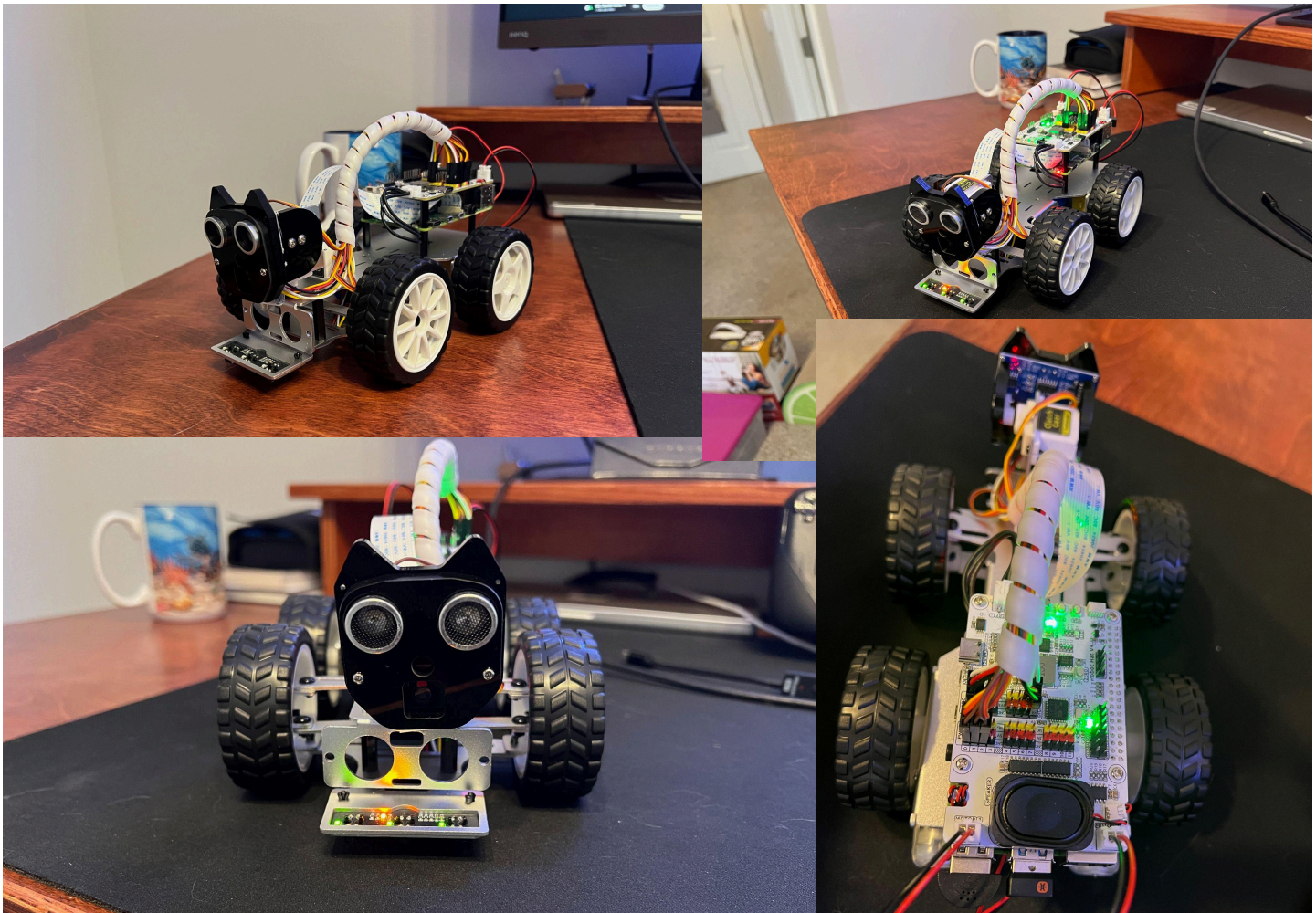


Assembly

Assembly was straightforward and simple for the picar-x. Deciphering the wire input for a couple of the modules was a little trick trying to make sure I didn't short something out. Luckily nothing went went

You may have noticed the head of the picar-x looks perhaps a bit suspiciously like a cat. As noted in my submission for lab 0 I had originally tried to start this project with the Freenove dog kit mentioned in the lab 0 instructions. During the break when I tried to get it working I realized it struggles to move on medium shaggy carpet even after zeroing it which is 90% of my apartment. Ultimately I had to give up but by that time the 4wd kit was out of stock. You may still be wondering why the cat head? When looking through the code and testing out the different modules I found an issue with the design of the picar-x. While having a lower mounted ultrasonic module is beneficial to detecting many objects it limits the ability to do an ultrasonic environment mapping since the module can't pan. Since I wasn't using the dog kit I had purchased and couldn't return it I decided to swap their heads. This way I can pan and tilt both the camera and ultrasonic module which will hopefully help with SLAM mapping in part 2. Unfortunately the head can't tilt down so I lose the ability to see lower object's but I think I gain a lot more and can compensate for that using the camera later on.

One other issue I saw with this design that I couldn't compensate for was the low grayscale sensor. Small bumps and ledges can get caught on it before the wheels have a chance to mount the terrain. Another small issue seems to be the lower quality servos. Even after zeroing several times the wobbliness of the front wheels cause the car to drift one way or the other slightly when traveling longer distances. I have not found a solution for that yet.



The Code

As of now the code is pretty simple and a little crude. I've never been a fan of python so my python design skills are a little lackluster. I do have a lot of experience in highly available distributed messaging apis and found some of that was transferable to this.

The car starts in the zeroed position and goes forward until the ultrasonic sensor detects an object within a certain distance. It then backs up, randomly chooses left or right, then moves forward until the sensor detects another object. I did run into a few issues in my initial object detection code that are described as follows:

1. The ultrasonic sensor would always detect an object when it first turns on and would cause the first action to be an "evasive maneuver" by backing up
2. During evasive maneuvers the synchronous code would back up and often turn into an object because it had to finish it's action before checking the sensor again
3. Getting stuck turning back and forth when encountering a corner like object
4. As noted above the grayscale sensor gets caught on low terrain some times
5. The higher placed cat head can miss detecting shorter objects and crash into them
6. The car occasionally gets caught on corners

Resolving issue 1 was easy enough by averaging 3 ultrasonic readings it seemed to eliminate the issue entirely.

Having a lot of experience with async code and race conditions I implemented a task event loop that allows the forward drive code to run alongside the cliff and object detection code in which forward and evasive maneuver tasks can be interrupted during movement should either sensor detect a danger. There was an issue with the initial async code where backing up would stutter and continually interrupt while the object it was trying to avoid was in the threshold. This was fixed with an extra backup state.

When choosing a random direction during obstacle avoidance I found the car could occasionally get stuck if it kept alternating between left and right when coming into contact with a corner-like position. Without working ahead on part 2 of the lab or utilizing the panning sensor I added I attempted to code some maneuver logic that would record the last n turns and see if it meets or exceeds the turn threshold within a certain time limit. If it's made too many turns in a short period of time the code will back up the car and do a really 180ish degree spin by rotating the wheels in opposite directions before proceeding. It works maybe 70% of the time.

Other than bending the grayscale sensor up I don't have a good solution for that other than keeping it on semi-level terrain.

As of now the code does not compensate for the higher ultrasonic sensor since it can't tilt down. Ideally I want to work towards using the camera with object detection to see if I can't compensate for that issue somehow. There are extra free digital pins and I have a spare ultrasonic sensor so theoretically I could add it back to the lower front and add a third sensor task. All I would need is 4 free wires since the pins from the original Freenove kit aren't compatible.

This code doesn't try to solve the corner catching issue yet. Ideally in future iterations of this code will pan and store the environment data in a map where I may be able to add a buffer for the wheel size.

Conclusion

I opted to work on this and future projects alone. I work in and lead a team of engineers professionally and have no problem working with others but in this case I want success and failure to be entirely dependent on me. I want to be forced to do things I struggle with like PCB and circuit design. Even writing python code which I really REALLY don't like. I think python has utility but is ultimately an overrated language for what a lot of people use it for especially at enterprise scales, but I digress.

Forcing myself to do things I'm uncomfortable with or don't like doing is how I became a principal engineer for my company in under 4 years. The struggle is what made me better at software development. Working on my own so I can better understand things for myself allows me to collaborate better later on with others.

I absolutely hated designing the pcb because I'm bad at it and while I'm no professional I really appreciate the experience of being forced to do something I'm not good at. I don't work with hardware a lot but I do have a keen interest in low level programming which can't be done without understanding the hardware. I'm looking forward to part 2 of this lab and more difficult challenges in embedded/device programming in the future.