CS437 - IoT
Lab 1.2
Name: Lucas Damler
NetID: ldamler2
Code Walkthrough and Demo

Issues and Thoughts

I had decided to work on my own, not because I dislike working with others, but because I wanted success, failure, and time management to solely rest on me. My schedule isn't stable with work right now and I didn't want to hinder anyone or them hinder me. I wanted all learning to be my own responsibility. That being said, I was very wrong. The first part of the lab was a breeze and I thought it would be easy going on the second part but I was terribly wrong. So here I am at 11:02 pm Sunday night with half working code pieces that don't play well together. The picarx was nothing but trouble and I spent days trying to resolve things that I don;t think would have been an issue had I been able to get the 4wd car.

For starters the loose front wheels cause the car to skew left or right even when you zero drive servo. This would throw off all calculations for picar wrapper that used dead reckoning for location updates. My original scan data worked fine but once I reimplemented it with grid to world coordinates it was all thrown off, but I had spent so many days trying to get the wrapper to work like I wanted I didn't have time to go back and fix it. The fixed ultrasonic sensor wasn't much of an issue because I was able to swap the head out as detailed in the last report. The battery was small and I had planned to use a larger one but given the size constraints there was no way to mount it or use the 2 pin connector since the car requires a 3 so as I was recording the movement and maneuvering I could get done it died and I have to wait for it to charge and hope I make the deadline. I can forgive many of the issues I faced but the wobbly wheels ruined everything and there was no solving that even with weighing it down to try and stabilize them.

I made a mistake working alone. I had to work non-stop every night after work to get where I am and it still wasn't enough. It's unfortunate butI made my choice and will have to accept the consequences.

Object Mapping

- The sensor module I wrote mapped out a 100x100 grid using a 3 average reading for accuracy because without it the car would often get stuck or perform unnecessary movements.
- The map was output in ASCII to the terminal along with position data for debugging
- Before any mapping decisions were made a bulk padding was added to the world map grid. This was actually an issue because once I stopped clearing the map it started repadding pre-existing objects but I didn't have time to fix it.
- The movement code without A* path finding was accurate and never collided with objects.

Vision Mapping

- Object detection was stored in the vision system module along with ultrasonic sensor mapping.
- There are 3 specific objects the code handled:
    1. A person in which it would stop and wait for them to move
    2. A cat because my cats like playing with it but I found the sensor for some reason didn't always detect they were there, probably because of the fur.
    3. A stop sign in which the car would stop for 3 seconds before continuing moving
- The vision module ran at 30fps and displayed to the screen at 1. Increasing framerate didn't help me enough to justify the battery cost and processing loss

Would hardware acceleration help in image processing? Have the packages mentioned above leveraged it? If not, how could you properly leverage hardware Acceleration?

- It would have and I did try to implement it but found the LiteRT code and edge coral model were not compatible with python 3.11 yet so I was unable to use it without downgrading

Would multithreading help increase (or hurt) the performance of your program?

- Multi threading would have helped up to a point and in my original I had it running with 4 threads. I ran into some trouble with the library and going back to my mistake of working alone I had to make the decision to not debug the libraries.

How would you choose the trade-off between frame rate and detection accuracy?

- The benefits of increasing framerate weren't high enough to justify the battery and process time costs. My vision mapping didn't many decisions for the objects it had so the 30 frame rate worked well enough to stop before hitting my cats

Navigation

- Random routing worked fine and it had no issues mapping out objects and picking new directions
- Navigating to an x,y point was tedious given the issues i had with the movement tracking on the picar wrapper but I eventually got it to work without obstacle detection and only if it was moving forward. Points behind caused the point navigation to go haywire
- The A* algorithm never worked to its potential. It could route a path and recalculate but I was unable to get it to work in conjunction with the sensor interrupt code. It would either calculate and execute movement before mapping so new obstacles didn't cause a reroute, or when it did stop to reroute my world grid code was off and it would hit boundaries that didn't exist. This is on top of the skewed navigation so it would move more to the x or y without adding that to the internal tracker causing all navigation to be off. I had to add an entire 20cm grid buffer to the success clause to get it to work within reason.