Embedded Systems – Autonomous Car Project

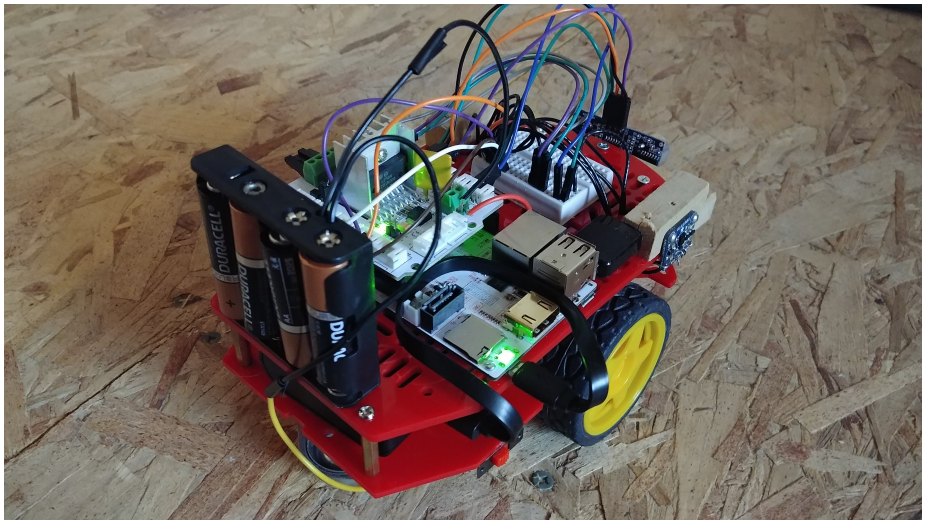
Sean Meyer, Scott Meyer, Ameen Sassi, Andrew Bentley

Introduction

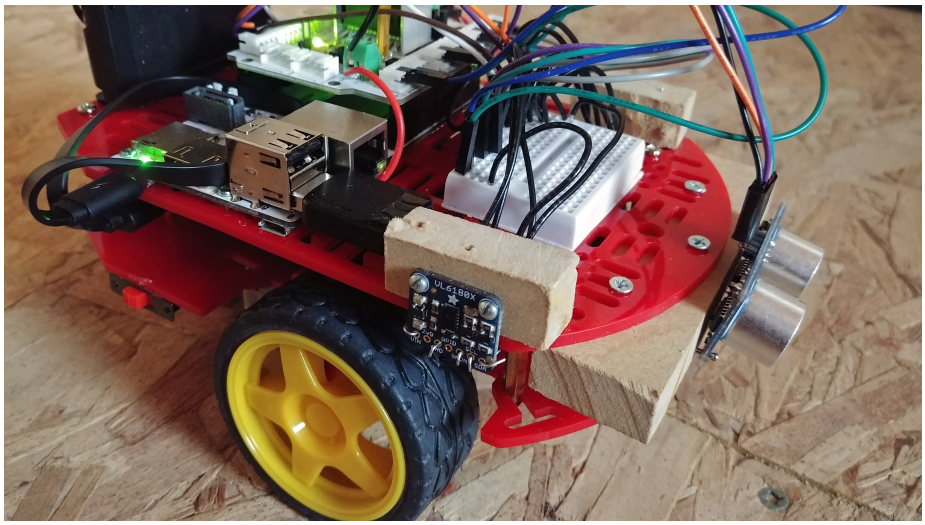
This is the Final Project Report for Sean Meyer, Scott Meyer, Ameen Sassi, and Andrew Bentley. We developed an autonomous car that is capable of following a path to reach its destination. It should be able to detect if there is an obstacle and reroute itself if necessary.

Overview of System

Our autonomous car is built using several different devices. The car itself is a simple plastic car with two wheels connected to the motors and a third stationary 'wheel'. The motors are powered by a rechargeable battery pack. The program is run using a PCDuino and Shield mounted onto the car. It is powered by a battery pack of 4 AA batteries.



Our sensory data is provided by 2 laser sensors that are mounted on the left and right side of the car and an ultrasonic sensor on the front of the car. We connected a USB WIFI router to be able to connect to the PCDuino remotely. We also used a Breadboard for connecting multiple sensors into a single slot on the PCDuino.



Challenges

Hardware

The very first challenge was getting the car and sensors set up, along with making them all work together. The cables we initially had didn't completely work, and we had to find a way to mount everything to the car so that it wouldn't fall off or move around. We ended up getting a lot of different small screws from a hardware store and cutting some small wooden blocks that we could screw onto the car, and then screwed the sensors into those blocks. Connecting the wires from those sensors to the motor shield, without them falling off, was also difficult and required a lot of trial and error.

In addition to the setup troubles, our group had issues getting multiple laser sensors working together. The issue stemmed from the fact that, by default, only one sensor can be read from over I2C at once using the Shield and PCDuino. We got around this by modifying the Adafruit_VL6180X.cpp library to take an I2C address. This allowed us to run our 2 laser sensors at once without having to shut one down to read from the other. While we eventually got it working, it was very hard to debug when the problem could be a hardware failure, a wiring issue, a coding mistake, a misunderstanding of the library, etc.

Another challenge was interfacing with the motors and sensors. We are not really sure what type of motors are on the car but whatever they are, they were somewhat difficult to work with. Step motors would have allowed us to better control the car in its movement compared to the motors that we were working with. On top of this, when we were trying to hook up the lasers and get the

sensor data from them, the libraries included with them were not designed or documented very well at all. This made the coding that retrieves the sensor data difficult to develop for. The libraries also forced us to build the code to drive the car in C as the libraries we needed were either not very good or nonexistent in Python.

An issue that especially plagued us during the testing phase was that the car was moving very inconsistently. This was due to both the sensors and the motors. While the sensors do give relatively good readings, they can be a bit jittery, jumping up and down. We had to correct for this at all times when dealing with the sensors. The ultrasonic sensor in particular seems to give measurements at large outliers sometimes, meaning that an average does not work well. Because of this, we instead used a running median. The motors were also a big issue. Our car would switch between turning too much and turning too little. This was all with the same exact numbers. This is likely to do with the amount of power that the motors were pulling from the batteries. The amount of power from the batteries was changing by enough to change how much the motors turn each wheel during each test. In addition to this, the batteries lose charge as we continue to run tests which means that the batteries are providing less power to the motors the longer we go. This causes more inconsistencies. Despite our best efforts to fix these problems with a lot of error correction, using the sensors to determine the car position and correct it, it was still a big challenge for us.

Software

We came across another challenge when we tried finding a way to represent the map. We didn't have an image of the map available when we started development and had to imagine what format the map would be given to us. At first, we decided that we were going to only use the car's sensors to navigate the map, but after struggling with the hardware, we decided to use nodes and represent the map as an adjacency list. It wasn't a very difficult challenge compared to the others, but we decided to code the map creation, breath-first search, and driver programs in Python. Some of us had more experience with Python than others so we had to take some time to learn the syntax and how to create classes, functions, and objects. However, it became easier as the project went on.

The next challenge was figuring out how to traverse the map within the program. One of the issues we never quite figured out was how the car would figure out which specific node it was currently in. We proposed the idea to add a counter for every second the car was in motion and update its current node based on that, but the motors were very inconsistent and we later decided against the idea. In the end, we decided on having the car figure out where it was based on the intersections it passed. This also proved challenging as sometimes the sensors would sense the same intersection twice and mess up its positioning.

Further difficulties arose with making our C code and Python code communicate. We developed the code that gathers the sensor data as well as the code to make the car drive in C code. The path finding algorithm that we built so that the car knows where is it and how to navigate to the end of the course was created in Python. Two different coding languages talking to each other is not impossible but it adds another level of complexity to our project. We managed to overcome this challenge by having the C code start up the Python code and having them communicate through strings. The C code provides a string to the Python which uses it to determine the next step. The Python then returns a string to the C, telling it where to go.

The last challenge that we had was one mostly of time. We were starting development on better versions of our existing functions. However, we did not have the time to test them properly in time for the official run. This required us to attempt to slap in some of the proposed new features onto the old functions so we did not risk breaking the entire system right before it was due.

Proposed Algorithms

Our system for the car was designed in two separate parts. The first is our actual code which runs the motors, gets data from the sensors, and converts the information. We broke down driving the car itself into separate pieces, and created some algorithms for each (of varying complexity). The main components were algorithms to do the following: Fix Rotation, Drive Centered, Turn Left/Right. Below is a basic description of what each part does:

- Fix Rotation - Turns the car in place (without moving forward/back) so that it is rotated parallel to the walls around it. For example, if the car is put into the track at a 90 degree angle (front and back to the walls), this algorithm attemps to turn the car so the walls are at the left and right side. This is done by keeping track of the distances of the side laser sensors and turning in place until the shrinking one stops shrinking.
- Drive Centered - Attempts to drive forward while keeping the walls equally distant from each of the side sensors. This function handles both driving straight and driving along the curved path, as well as correcting when the car moves too close to any one side. The main idea is to change the speed of either the left or right motor when changes in left/right sensor data are detected. We keep track of what the sensors used to be returning and what they are returning now, and using an average of that data try and come up with how much the motor speed needs to change to make a correction. This causes the car to self-correct and try and stay within the center of the "lane".
- Turn left/Right - This used a fixed delay and speed to turn 90 degrees, based on testing within the track. While it wasn't completely accurate,

using the fix rotation algorithm both before and after turning allowed it to work reasonably well.

The second part is our path finding algorithm that helps the car keep track of its location in the track and find the quickest path to the end.

- The Map Creation algorithm was simple. A .txt file held all the information for each node and when it is run through the program, an adjacency list is created. Each node holds the following information: id, chckpnt (a flag that is set when the node is an intersection or turn), blocking (a flag that is set when a node is blocked by an obstacle), and a list of the connecting nodes.
- The Breadth-first search was a basic breadth-first search algorithm. When the goal node was found, a list was created with a list of nodes from the goal back up to the root of the tree (which would hold the starting node.) The list would then reverse and a list from start to finish would be returned. It is built on the idea that the map can be split up into 1 foot by 1 foot sections. It uses a text file that contains an adjacency map that lists each section and what is connected to it.

Evaluation and Results

Our car didn't quite make it to the end of the track, although it consistently got close and was able to navigate through the curved area. The biggest issue was that we didn't do a good job keeping the car centered at all times, which caused the sensors to become more and more inaccurate (hitting surfaces at curved angles, getting max distance readings because of how turned the car was, etc). We added a lot of error handling and self-correction so that the car would almost never get totally stuck, and could keep driving, but it would lose track of where it was along the path due to these issues (basically it would think it had hit an intersection when it hadn't).

In terms of the map creation and path finding programs, the code ran very well. It could create an accurate representation of the map and find the fastest route to the end. It was never able to re-path in the case of an obstacle, but we're pretty sure we would be able to figure it out if given more time.

Overall, the system performed reasonably well, and our code to get unstuck and correct rotation worked great, but it definitely could have been better. In addition, the mapping code could have been improved had we had more time on the project. We could do better if we were starting again from scratch, but overall we're happy with the performance.

Lessons Learned

One lesson that we can take away from the project is that writing code in multiple languages adds a complexity to any project in that the two different languages need to be able to communicate with each other. While we were able to get around it in a fairly clean manner, it probably would have been best to simply stick with one language for the entire project for simplicity's sake.

Another big lesson learned was to focus more effort on trying to keep the car in as "good" a state as possible at all times, rather than focusing on correcting bad situations. A lot of our effort went into code that would recover from bad situations, and this code worked very well, but we've learned that it's better to try and avoid those situations in the first place as much as possible.

Something we already knew, but really got hammered home, was that working with real world machinery, like the sensors and motors that we had, there is going to be a lot of inaccuracy. While we knew that the motors and sensors would not be perfect going into the project, we didn't fully appreciate the scope of that problem and plan for solving it right away. For example, if we were to do something like this again in the future, one of the first steps would be averaging/filtering the sensor data to get the most accurate readings possible, and working forwards from there. Attempting to create our driving algorithms while still using suspect sensor readings (before we started smoothing the data) added delays and mistakes that could have been avoided.

Lastly, we learned that when working on the software we should constantly test it together with the hardware. After completing the driver program, the car ended up running into a lot of issues while trying to use the path given to it. If we constantly tested the software with the hardware, small issues and problems would be easily found and solved. In addition to this, we learned that we should have the people working on the software work more closely with the people working on the hardware. Being more acquainted with the hardware, how it works and its inaccuracies would assist greatly in designing and modifying the software to better fit the car.

Conclusion

We all learned a lot throughout the duration of the project. While our project is not perfect, we are proud of what we were able to accomplish. There are a few things that we could have done differently and several challenges that we could have either moved past with more time or avoided completely by avoiding some of our missteps. With more time and effort, we could have created a system that is truly autonomous. That being said, in our minds, this was a success because we learned a lot and got to play around with some interesting technology.

Setup of Car

The setup of the car is fairly simple. The code was loaded onto our PCDunio. From there, we simply need to compile it and run it. The car will then start moving, provided that the motors are turned on.

Video

URL of the video: https://youtu.be/utSghOVPHnk

Division of Work

- Sean: Development of the Driving algorithm, Setting up the sensors
- Scott: Development of the Driving algorithm, Setting up the sensors
- Ameen: Development of the pathfinding algorithm, Initial setup of the car
- Andrew: Development of the pathfinding algorithm, Create the report