https://github.com/lovemich/AVC_Team-Epsilon

AVC FINAL REPORT

300381661

Lavanya Sajwan
Lab: Wednesday 10am-12pm

LECTURERS - Elf, Bryan Ng, Arthur Roberts

Abstract

This report consists of Team Epsilon's design, coding, testing, as well as the final run of the autonomous vehicle built. This project required co-operation and communication between members of the team. The group's final product was able to successfully complete quadrants one, two and three and it was also able to enter quadrant four.

Introduction

The task given to Team Epsilon by the clients was to navigate a maze. According to the rules of this challenge, each team was initially given a Raspberry Pi unit, a PCB (printed circuit board), a basic chassis, an H-bridge, camera, motors and wheels. These were to be the starting point of the robots and teams had to build on to it. Each team was also given an extra $100 "Arthur dollars", virtual money that teams could choose to spend on additional parts such as sensors.

Each team had restrictions such as the time restraint of six weeks – one of the weeks being a break, when the teams were able to have access to the labs, the programming language being used, the hardware provided as stated above and the total time of 15 minutes for the final run through.

During this Autonomous Vehicle Challenge (AVC), everyone was learning to build and test a functioning robot that was able to operate without human interaction, while working with new people. Skills learnt during lectures were used to get our vehicle through a maze. This project also called for the team to maintain frequent record keeping and to develop a report. Teams did this to keep the customers happy with the product and to fulfil their specific requirements. The aim was to apply what each member had learnt in lectures and during assignments to synthesise a way for the robot to complete the maze by overcoming the obstacles in each quadrant and all done within the time limit of 15 minutes.

The objectives for this were to put together a functioning robot, produce a working program, undergo rigorous testing, showcase the design during a final run, and to write a report, all while keeping the specifications in mind as well as staying within the given budget and sticking to weekly goals and tasks. Some of the key specifications chosen by the clients comprised of:

1. Teams - the teams were not chosen, and were put together based on assignment marks.
2. Aesthetic - the robot had to be aesthetically pleasing. It also had to be easy to disassemble so that it was easy to take apart in case the team made a mistake as well as for reusability reasons. Therefore too much solder was not ideal.
3. Maze - The robots had to finish the maze within the 15 minutes time frame without any interaction and were to go from start to finish.

Team members benefited during this challenge as each individual developed their coding, hardwiring, networking and teamwork skills. Overall, this challenge gave the team insight on what it is like to be in an engineering based environment.

Team Epsilon's group aim was to complete as much of the maze possible while also having a short time by going as fast as possible.

Background

The AVC, is the autonomous vehicle challenge; it called upon teams to make a vehicle that was able to operate without human interaction. This is different to automate - which relies on the basis of human control. There are five levels of automation, as stated from the National Highway Traffic Safety administration (NHTSA) of USA. At level zero, the vehicle is automated. At level one there is specific-function automation, which means that the driver controls everything else apart from some functions. At level two, the combined function automation means that there are at least two primary functions that the vehicle can control by itself and simultaneously, while the human controls the rest. At level three, limited self-driving automation occurs, which means that the car senses when the driver should take control of the car based on the safety conditions. At the last level, level four, the vehicle drives without human control, which was required for the challenge the team was undertaking. This was because the vehicle made within the team should've been able to navigate the obstacle course by itself by sensing the changes in the environment around it *(User: 122.15.201.20, last modified 15th May 2016*).

This challenge was interesting as self-automated cars are now closer to becoming a reality and so the little vehicle made within the groups, gives a small scaled view of how these cars work. Testing on scaled autonomous cars started at the very least in the 1920's picked up in the 1950's *(User: 122.15.201.20, last modified 15th May 2015)* and have continued to the present day; so much that self-driving cars currently drive on in cities *(https://www.google.com/selfdrivingcar/, ND)*. These are highly beneficial to the wider public when available, as people who are unable to have the freedom to drive by themselves due to loss of motor control and blindness will be able to "drive" to places themselves. And as an automated car fades out, and autonomous becomes the mainstream, road accidents will dramatically reduce. However, problems with these are that they are unable to account for the human mind's ability to make split decisions. It can't process whether another object on the road will suddenly swerve etc. (*Ghose, posted May 14, 2015).*

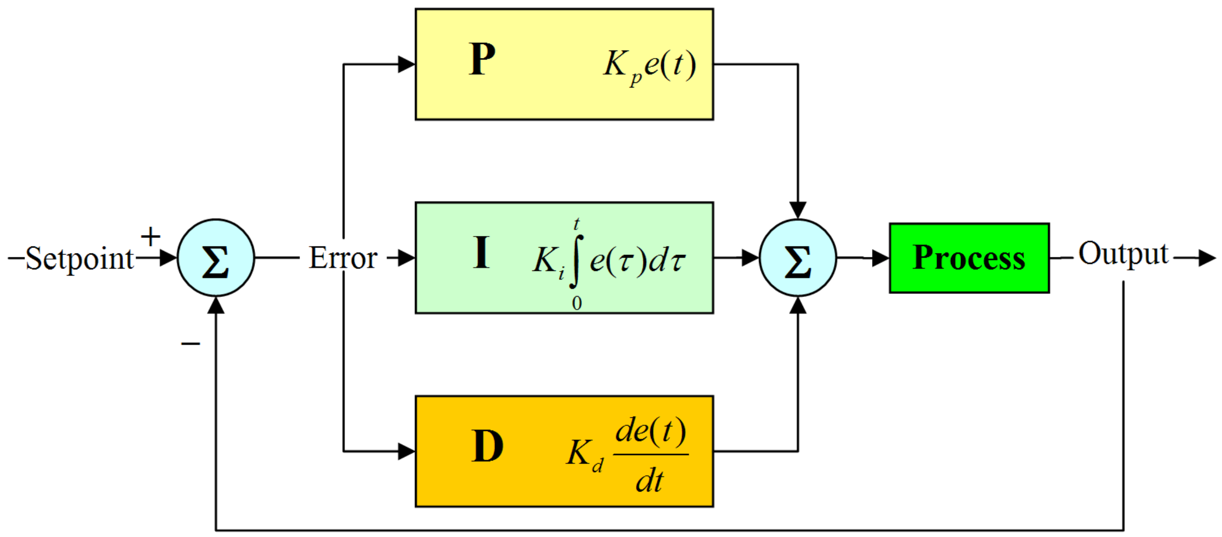
A Raspberry Pi was used to program the autonomous vehicle in C++ programming. It was used because it could directly connect to the computer monitor to program and was also directly able to interact with the components connected to it such as the motor and wheels. The motors on the vehicles were connected to the H-bridge on the RPi which let the voltage flow in both directions so the motors were able to go backwards and forwards and then let the wheels go backwards and forwards.
(*https://www.raspberrypi.org/help/what-is-a-raspberry-pi, ND/)*
(Eldrige, last modified 12 April 2016)

As prior mentioned, the programming language C++ was used. This was because teams were told to use this and it was used in the 101 course assignments. This language is able to perform low-level machine code commands, unlike Java and Python which are used in other courses being taken. The LibE101.so library was used as the code would be unable to compile without it and was the one the team was told to use. *(Eldrige, last modified 12 April 2016*)

Computer-Aided Design (CAD) was used to design the 3D printed objects used on the autonomous vehicle. CAD is used as a way of stimulating what the designs will look like before physically incorporated on an object. Team Epsilon used the FreeCAD program as it was easier for us to 3D model and manipulate rather than code for them if the team were to use OpenSCAD. This is also because Team Epsilon's robot was very minimalist in its use of 3D printed objects and they were very small, so coding for the designs would have been time consuming. Overall FreeCAD was user friendly and easier for beginners to use. *(Eldrige, last modified 12 April 2016*)

A server is a program that allows requests from other users of the same or from different computers to fulfil the service needed to be done. In the networking part the physical gate was controlled by the gate server and requests were sent to it in order to open it.
(*Rouse, last modified June 2014*)
(Eldrige, last modified 12 April 2016)

Error signals are a way for the system to understand how far away it is from where the program wants it to be. Control theory is used to teach objects complex behaviour such as how Team Epsilon "taught" the autonomous vehicle. PID control is the control system used to control the robot's movements. PID stands for proportion, integration and differentiation. The proportional gain is how much the vehicle has to steer to come closer to the desired places. At a high proportional gain, the vehicle can spin out of control, while with too low of a proportional gain, the vehicle won't be able to correct itself accurately. With only proportional control factored into the output, the vehicle will be crooked once at the centre line so the controller will overshoot the line and won't properly follow it. The integral term accounts for correction based on future errors and go back to the original performance after the error has occurred. The derivative term is how fast the vehicle moves perpendicular to the line. This derivative gain is how much the vehicle opposes the proportional gain. This helps the vehicle to stay close to the line and needs to be as close it can to the value of the proportional gain. If it is too small the robot will still oscillate from the line and if it is too high, the system will take a while to correct itself and will thus be overdamped. The combination of these values will be the output of Team Epsilon's vehicle. (*AerospaceControlsLab, published on July 21, 2015*)

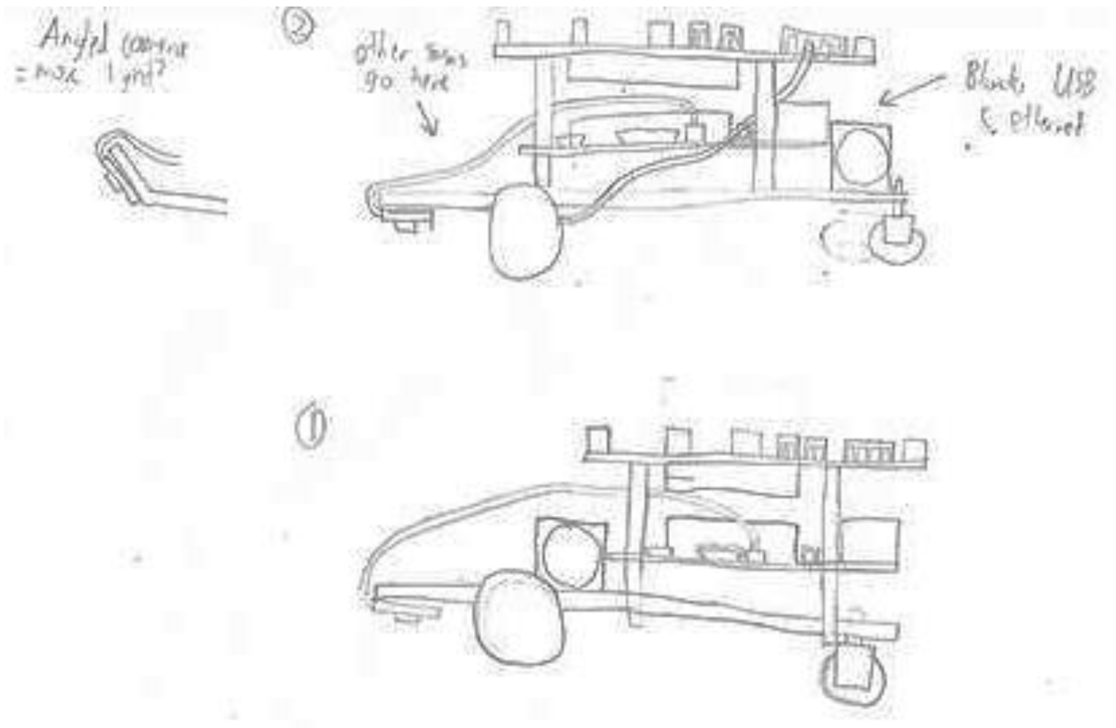


http://stm32f4-discovery.com/wp-content/uploads/pid-controller-diagram.png

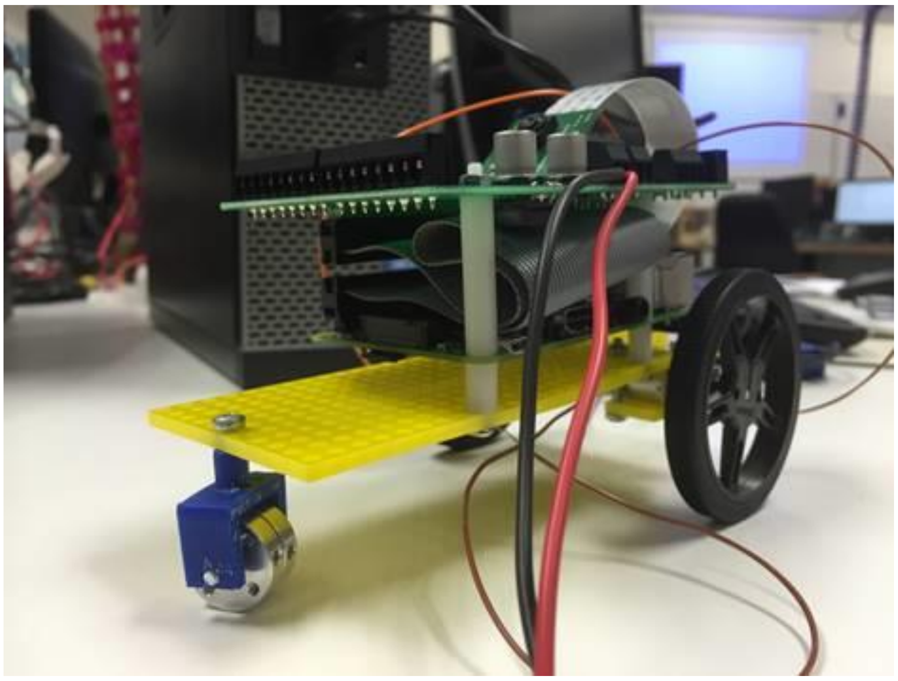
Method

Hardware

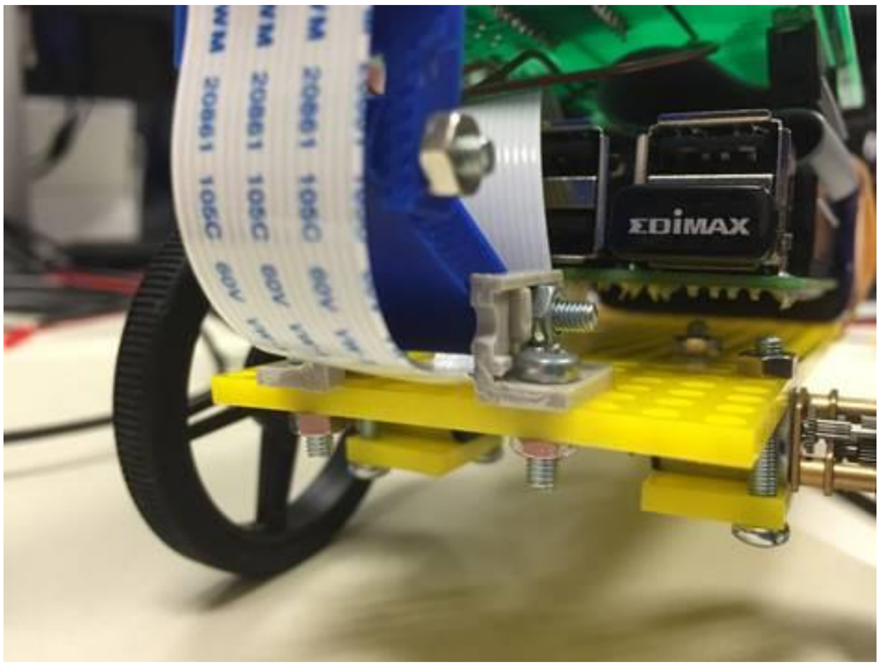
The hardwired vehicle was quite close to the initial design. The low and long design was based on racing cars to reduce the minimal air friction in the room so that the vehicle was able to achieve the quickest time that the team desired.



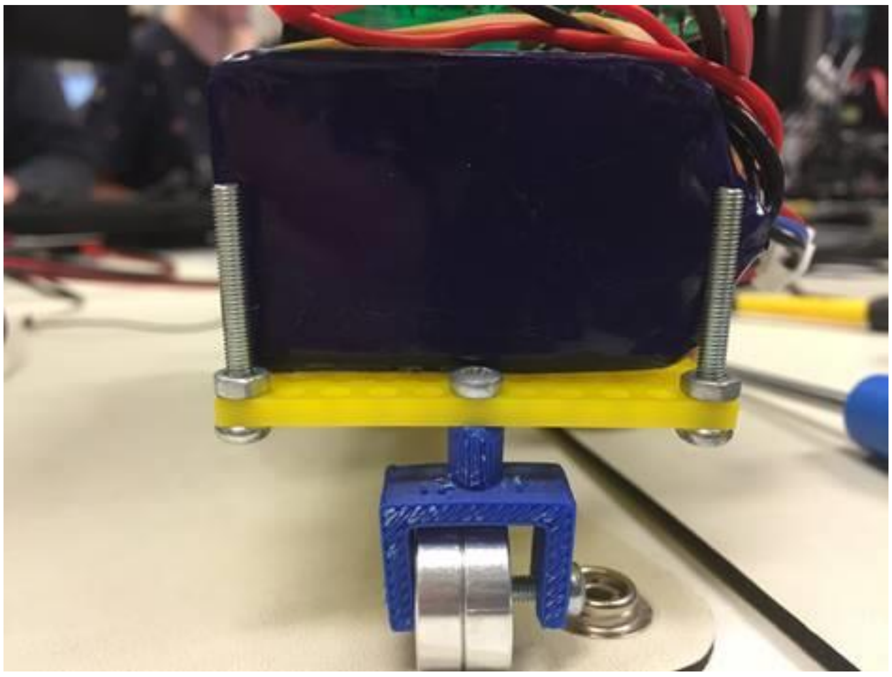
Team Epsilon started with the chassis provided by Arthur and the Raspberry Pi was placed on top of the chassis. It was placed at the front on the side of the two big wheels (5cm in diameter) so that there would be enough support for the weight, and so that there would be room for the battery to fit on the team's vehicle. The two big wheels were connected to the 5V brushed DC motor underneath the chassis. The motor was connected to the H-bridge upon the RPi. The boards were connected together by white standoffs as shown in the picture. The colour of the chassis was chosen because it was the one available.



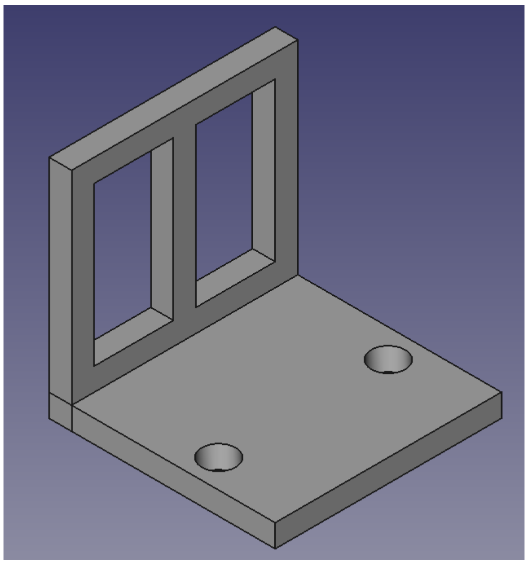
During week four, a camera holder was added which was chosen because its design was supplied and it was easy to change how much it was angled to the maze surface. The Raspberry Pi foundation PiCam version 1.3 was attached to the holder which therefore made it simple to manipulate how much light was able to enter the camera. The holder was placed in on the side with the big wheels, as that is the front of the robot, the team obviously wants it to see ahead to see where it should travel by recognising and following the white line. It stuck out from the main body of the vehicle as it took time for the pictures to be processed.



At the back of the autonomous vehicle, was where the “Nano-Tech: High Discharge LIFEP04 BATTERY 6.6V Receiver Pack” of dimensions: 5.5x3cm battery was placed when the vehicle manoeuvred the maze. The team decided to have it just above the back wheel (30mm in diameter) and it was initially supported by screws and a rubber band.

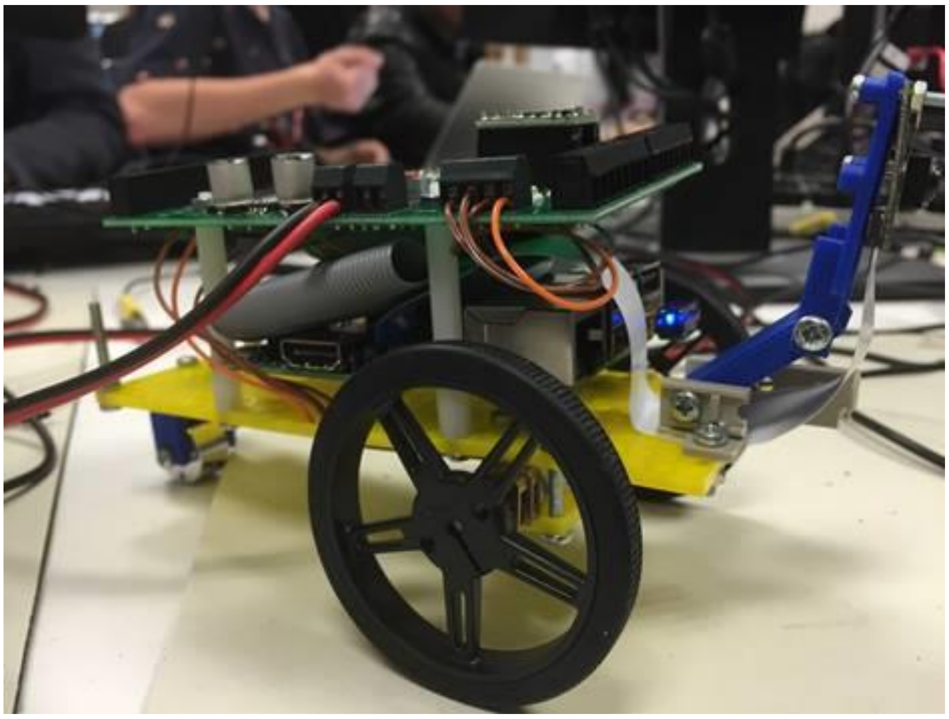


However, afterwards Team Epsilon printed off side brackets to support the battery and thus had no need for the rubber band anymore. These were placed on the sides of the chassis to the side of the laying battery.



During testing in week four, the robot lost the line and fell off the side of the maze, so one corner that held the screw broke off. The group fixed this by putting the screw back on, but weren't able to put the corner back on.

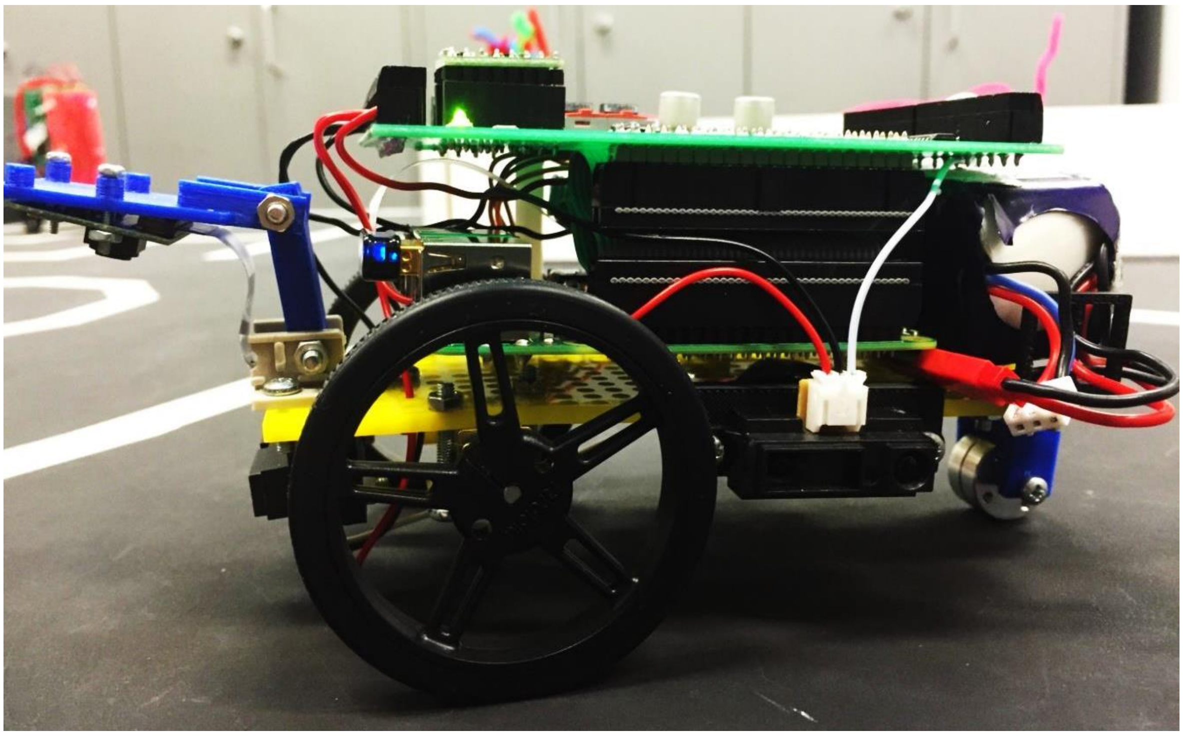
The fixed bigger wheels were chosen to be in the front as the vehicle would have more control over its movements. However, admittedly they were mainly chosen because of aesthetic reasons. But the big wheels were ideal to be chosen over the smaller ones, as bigger wheels have a greater circumference than smaller wheels so that there is more distance is travelled in one revolution. This was ideal for Team Epsilon as the collective group wanted the robot to travel the maze in a short period of time.



Three sensors were placed on the vehicle, one at the front and one on each side. These were analogue Sharp 0A41SK IR sensors and were used as they were the more sensitive ones.

The vehicle had wires looped between the boards for engineering aesthetic reasons; as loose wires aren't visually pleasing and can be easily damaged during testing by being caught on things or becoming detached as it falls off the table. Yet, it was informed after the final run that this could've been a potentially dangerous design as the RPi could overheat and burn through the wires. If this did occur, Team Epsilon would have had to near-to-fully dismantle the autonomous vehicle to rewire.

This was the final design.



Software

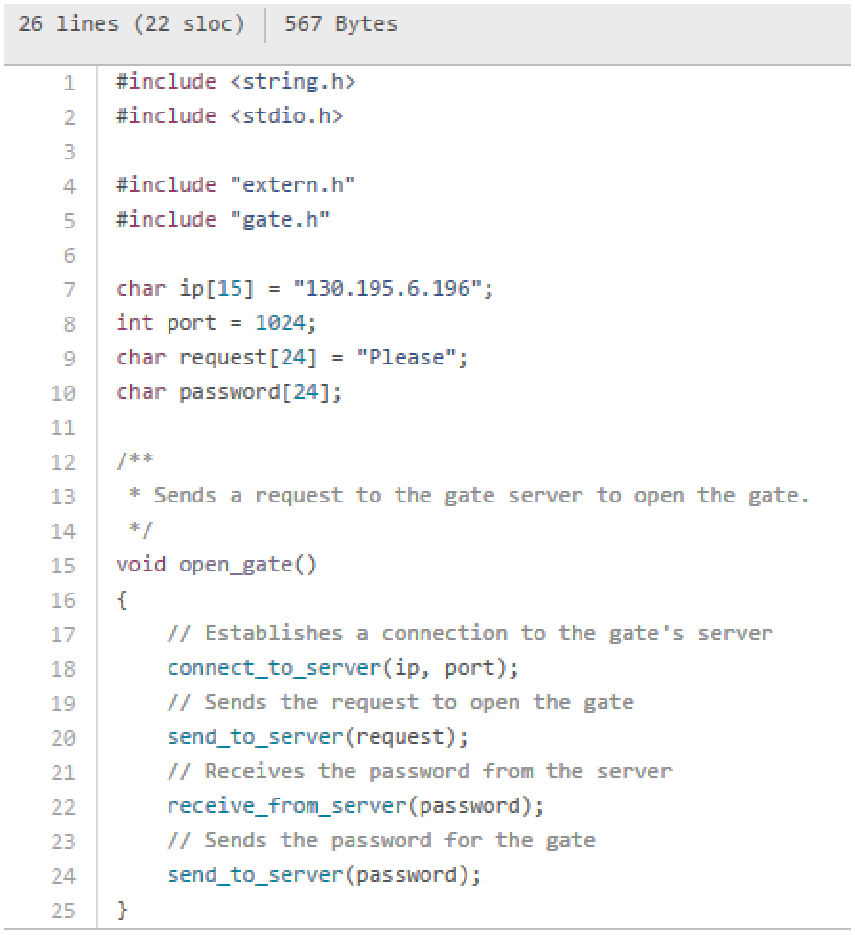
As briefly mentioned, Team Epsilon had been programming a Raspberry Pi and then using SSH. SSH let the vehicle connect to the IP address and team members were able to gain remote access to files within the Raspberry Pi so the code was able to be written and compiled without a wire connection to the computer *(Eldrige, last modified 12 April 2016*).

The linefollow.cpp folder includes the code in which the bot followed the line and this is what was first coded for because it was important to actually have the vehicle moving. Team Epsilon's vehicle oscillated a lot and this was because the KD wasn't tuned enough and it was very fast because the KI was quite high. Also, while turning, the vehicle didn't slow down one wheel, but instead sped up the other. Both quadrant one and two relied on this pattern of code.

When quadrant three was reached, the code switched to the follow_square_line method. This occurred when the robot detected that there were lines to the left, right and behind, but not to the front; indicating that it had reached the T-junction. The code was also manipulated to favour the left and had an offset to the left of the line as Team Epsilon's bot was consistent in its movements this way. Team Epsilons autonomous vehicle went much slower comparatively in this section as so this new method off set didn't get too large so that the robot completely lost the line. It also had to be drastically slowed down because the camera wasn't able to take fast enough photos to process and thus at the original speed the bot wasn't picking up that it had reached a 90 degree turn.

As the robot navigated the course, the team also coded the camera to take photos as hinted in the paragraph above. Every time the main loop looped, which was the main.cpp true loop, it took a picture. Therefore, when it had finished processing one picture it took another. This was essential so that the robot kept on moving on the white line.

Networking was the main focus of Team Epsilon's attention for the first three weeks, as if the team wasn't able to achieve a working networking code, the robot wouldn't have even been able to enter the first quadrant. However, this was successfully coded by the end of week three and tidied during week four. This code lets the vehicle establish and maintain a connection with the gate by connecting to the server, and asked the gate to "Please" send the password back to the vehicle, to which it then sends back to the server and the gate opens.


26 lines (22 sloc) | 567 Bytes

```
#include <string.h>
#include <stdio.h>

#include "extern.h"
#include "gate.h"

char ip[15] = "130.195.6.196";
int port = 1024;
char request[24] = "Please";
char password[24];

/**
 * Sends a request to the gate server to open the gate.
 */
void open_gate()
{
    // Establishes a connection to the gate's server
    connect_to_server(ip, port);
    // Sends the request to open the gate
    send_to_server(request);
    // Receives the password from the server
    receive_from_server(password);
    // Sends the password for the gate
    send_to_server(password);
}
```

The original pre-tidied networking code was much longer as it formatted the reply from the server. However, the team realised that this wasn't needed so the code was drastically shortened.

While the team had accounted and coded for Quadrant four, the group ran out of time to do testing and when it got into the maze it wouldn't turn. The team collectively realised this and the code was changed so that once the bot detected the loss of the white line to follow, by the red coloured tape being proceed by the camera, it went back to the original speed and made it burst forward so that it could hit the back wall as the group knew that this was the highest mark the team could possibly get.

Results

At the start of the challenge, during week one, the group had just started to code for the line follow in quadrant one. By week two, Team Epsilon had managed to piece together the code done during week one (coding for forward movement and turning of the vehicle), and the networking code so that the autonomous vehicle was able to have the code done for Quadrant one. During week three, testing for quadrant one had been completed and code had been written for quadrant two.

When Team Epsilon reached week four, the hardware and code for the first three sections of the course had been completed, however testing for quadrant three had to be done, which occurred during week 5. During that week the team also connected the sensors on.

For the last week of the challenge, the team had aimed to complete quadrant four by the final showing, but only managed to finish quadrant three and manage to just get into the maze and then hit the wall and overturned. However the robot managed to do this in the first go which made the team very happy. Team Epsilon is unsure of their final grade because they weren't able to have the meeting with the aesthetics judge after the robot had completed the maze as the lab time had ended and there was the last ENGR101 lecture occurring and catch up tests were going to occur within the lab in that time slot.

Discussion

When Team Epsilon started testing for quadrant one, the wait time for the gate was too low so the team's vehicle continuously rammed into the gate until it was high enough to pass through. To fix this, the wait time was increased to half a second so there was a lag before the vehicle would start. The robot also use to get confused and go back and forth before it went through the gate and would sometimes back right off the course. This was because there was too little light underneath the gate for the robot to process and pick up on the difference between black and white. This was solved by lowering the RGB value range for the white. During the very initial testing for the linefollow.cpp code, the team's autonomous vehicle turned left into the wall when it started to line follow. This was deduced that Team Epsilon had the values for left and right backwards.

During testing for quadrant two, the vehicle wasn't able to turn the corners as much as the group wanted too for it to continue following the line in order for it to move. This was fixed by adding a higher KP value. The vehicle also kept falling off the board and the team fixed this by reducing the speed as the vehicle was moving too fast in order to process the pictures being taken so was following the old line and thought that it was still on it while it was actually plunging to its death.

Occasionally, when testing for both the quadrants, the vehicle would veer into the wall or off the board even when Team Epsilon had accounted for and fixed the original mistakes. Team Epsilon concluded that this was because the camera holder was angled too far up and was processing the white of the wall or someone's hands as the line it had to follow and would promptly take that route found and would break itself. This was fixed by angling the camera holder lower so that it would only pick up on the white line on the floor of the course.

This is how the robot progressed in quadrant one and two by the end of week four. At this point in the challenge the autonomous vehicle completed both sections in about 27 seconds.
https://drive.google.com/file/d/0B-41OE09a9DweHdhMXROZDN2amM/view

As the vehicle reached section three of the course, it dramatically slowed down. This was because during the testing it wouldn't detect that it had reached the T-junction indicating the start of quadrant two because the camera wasn't able to take photos fast enough and process them so it would continue going straight forwards. When the initial method was used as the output for the vehicle during this section, the robot would turn right when it shouldn't and would sometimes just stopped following the line this was changed to a new method that was coded with an offset and a preference to the left and this worked during the first test. The speed remained slow when the robot entered quadrant three (not just approaching it to get through the T-junction), so that the robot completed this quadrant by having a suitable offset. If the speed was too great the offset would be too large and the robot would have once again fallen off the board which it did during that section once. Team Epsilon underestimated the time needed for this part of the course and it really set the team back.

As the team were putting on the sensors to get values to set within the code that had been put together, the sensors didn't work. However the team took a while to pick up on this and thought this was a fault in the code because the code itself hadn't been properly tested. This uncertainty set the team back even more and if this didn't happen, the team perhaps could've started testing earlier and had some of the maze code working.

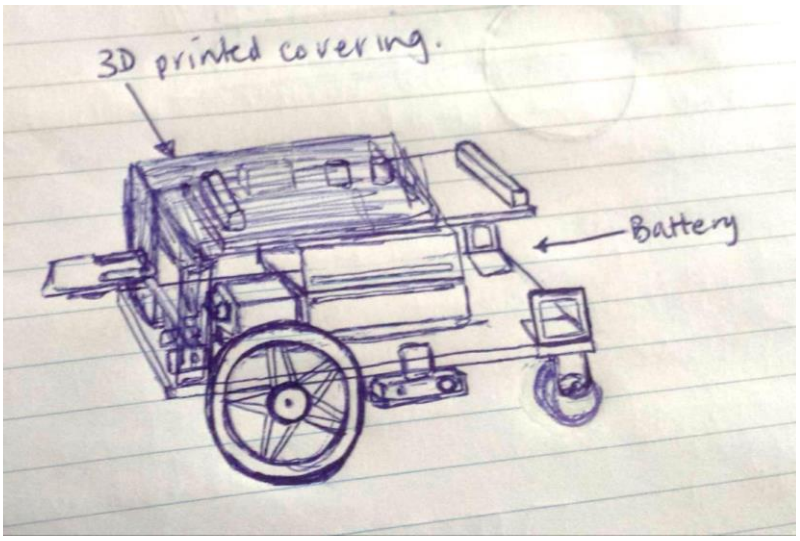
During the final run through, the robot was able to successfully complete quadrant one, two and three and entered the maze. As the group wasn't able to test the code of quadrant four a great deal before the showing of our vehicle, the code for that part of the course didn't work. Instead in our final run the robot reverted back to a fast speed and ran into the wall and overturned.
https://drive.google.com/file/d/0B-41OE09a9DwMG9OTGZYb3JoSW8/view

As seen in the video, Team Epsilon's vehicle was quite jerky also due to lack of time so tuning of the KD didn't occur. This didn't bother the team as this was just a toy sized robot being made. But if this were a life size autonomous car no one would've like to sit in it as the ride would be very unpleasant and would probably induce motion sickness.

When groups were initially assigned, the first task undertaken was to choose responsibilities for everyone. However, during week four, the team realised that responsibilities had to be more shared around and not strictly stuck to the assigned roles so that each member is able to develop on each skill stated in the introduction. Therefore, from then on, each member of the team tried to understand and immerse themselves in different positions.

While Team Epsilon was working on the networking, Quadrant one and Quadrant two coding, the team was running fairly on time with the weekly and personal goals and tasks. However, as predicted in my Progress Report, when coding started to occur for going further into Quadrants three and four, Team Epsilon didn't stick well with the original weekly goals as we ran out of time.

A lot of what the team wasn't able to complete and improve on was due to time constraints. If Team Epsilon was able to have another week of working I strongly believe that the team would've been able to complete the maze and would be able to improve the aesthetic of the robot more by adding some more 3D printed components onto the robot as the team had designs that the group wasn't able to print.



However, perhaps if the team did have more 3D printed objects on our vehicle; it may have not travelled as fast due to the increased weight of the components and would've led to increased pressure put on the motors. This may have not let the team achieve half of the aim. Testing would've needed to be done to see whether if the aesthetics would trump the speed.

Next time, if the team were to do this again, having a more compact design would be beneficial as the team had to put the camera holder and the battery supports at the very edge of the chassis. These areas are weaker and meant that bits of our chassis broke due to the pressure on the plastic and Team Epsilon then had to replace the camera holder once and the battery supports a few times. Also, having extra lab meetings earlier on would have been favourable for the team as it would've been more productive and would've actually been able to stay within the labs rather than being kicked out due to overcrowding. It would be effective to try other jobs earlier on when the jobs sharing got less strict and more interchangeable Team Epsilon understood more about the different processes being undertaken in this challenge as well as it improved the communication more.

Conclusion

The team has worked well and communication was effective. During this challenge, it had been great working with unfamiliar people as this is what a real life situation will be like. There will be situations where people will be working in a team with others they're completely unfamiliar with and in order to complete the task given, we will have to communicate effectively with each other to discuss ideas. Team Epsilon stuck closely to the client's expectations and while the team wasn't able to complete the general aim, the group was able to complete their personal aim to complete as much of the maze possible in the fastest time possible. This challenge was a great way of solidifying and understanding lecture ideas and applying them in real situations. Very few things are done independently in the real world and often tasks do need to cross over for cooperation between people and this was exhibited during this challenge. Members of the team now have the paramount knowledge of the amount of time and effort needed in a big assignment, as well as the knowledge to work with others which will help very much in the future.

Referencing

User: 122.15.201.20 (2016) *Autonomous car*
Retrieved from: https://en.wikipedia.org/wiki/Autonomous_car

Google Self-Driving Car Project
Retrieved from: https://www.google.com/selfdrivingcar/

ND (ND), *Raspberry Pi*
Retrieved from: https://www.raspberrypi.org/help/what-is-a-raspberry-pi/

Eldridge, E. (2016) *Wiki*
Retrieved from: https://github.com/kaiwhata/ENGR101-2016

Ghose, Tia (2015) *Self-Driving Cars: 5 Problems That Need Solutions*
Retrieved from: http://www.livescience.com/50841-future-of-driverless-cars.html

Rouse, Margaret (2014) *server*
Retrieved from: http://whatis.techtarget.com/definition/server

AerospaceControlsLab (2015) *Controlling Self Driving Cars*
Retrieved from: https://www.youtube.com/watch?v=4Y7zG48uHRo