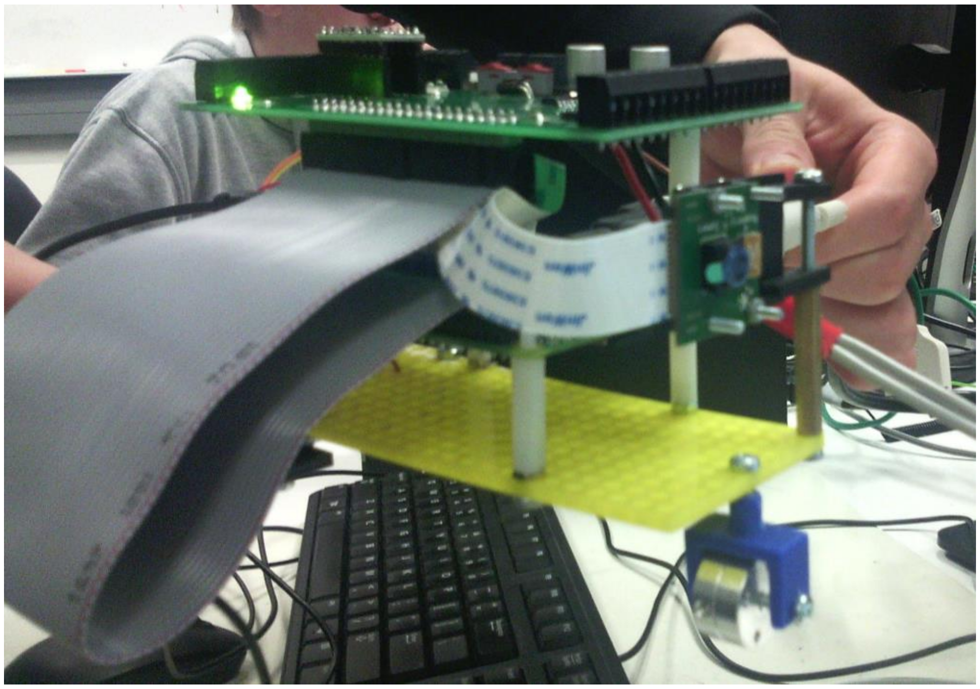
ENGR 101 AVC Robot Progress Report

Name: Christopher Straight
Student Number: 300363269
Course Code: ENGR101
Lecturers: Elf Eldridge and Bryan Ng
Assignment: AVC progress report
Date:16/5/2016

Team Name: Group 5
Team Members: Caitlin, Jacob, Christopher, Brandon, Pai Zhou
Github Repository: https://github.com/thrand-Antharo/AVC-2016-group5

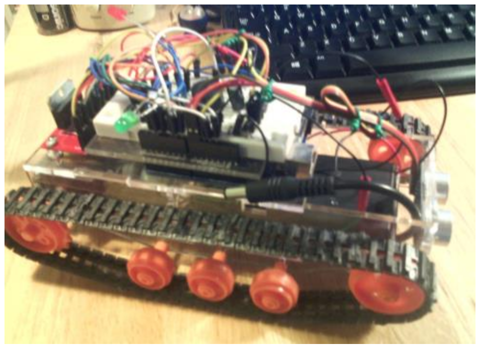
Roles:
Christopher: project lead and hardware support (organising team meetings, reporting regularly on progress, CAD designing components)
Jacob, Pai: software development (writing core code and extending functionality)
Caitlin: software testing and documentation (debugging software and committing to git, writing test cases and documenting performance against milestones)
Brandon, Christopher: hardware (building the chassis, testing components, connecting sensors, debugging hardware)



Introduction

The aim of the Project is to Create a robot that can autonomously navigate through the world. The robot will need to follow a white line on a black background, open a gate with network commands and navigate a maze.

Background



An autonomous robot is similar to but different, from an automated system. automated systems will perform specific actions in very specific situations, however if the conditions are wrong an automated system will stop working and be unable to correct the conditions. An example of an automated systems is this robot (picture right) which I made in 2014. The robot uses an ultrasonic sensor to find the distance of objects in front of it, when it gets to close it turns right. The problem with the robot's automated system is that if it got stuck in a corner it would have no way to get out without outside intervention.

Autonomous means having the ability to act independently, thus an autonomous robot should be able to choose how respond to changing conditions without outside intervention.

An example of an advanced of autonomous robot is Google's self-driving car. The self-driving cars use $70,000 LIDAR system and Google's Google Chauffeur software. Using a range finder on the roof of the car it creates a 3D map of its environment, then GPS data and real world maps are used alongside the 3D map to decide how the car needs to move.

Method

To follow the line, the robot uses the raspberry pi camera and a PID control system. At regular intervals the camera takes a photo, the centre row of pixels is then analysed to find how off centre the line is. If no line is detected ten, the robot turns until it finds the line or enters the maze (when there are walls within a certain distance on both sides). When the line is in the image the is used PID to determine how much the robot needs to turn, to centre the line. The angle is then passed to the motor control method; motor control calculates what speed is needed on each motor for the specified turn.

To open the network gate, the robot will connect to the gate wirelessly, via a Wi-Fi dongle plugged into the raspberry pi. The robot the calls then command to open the gate, allowing it through.

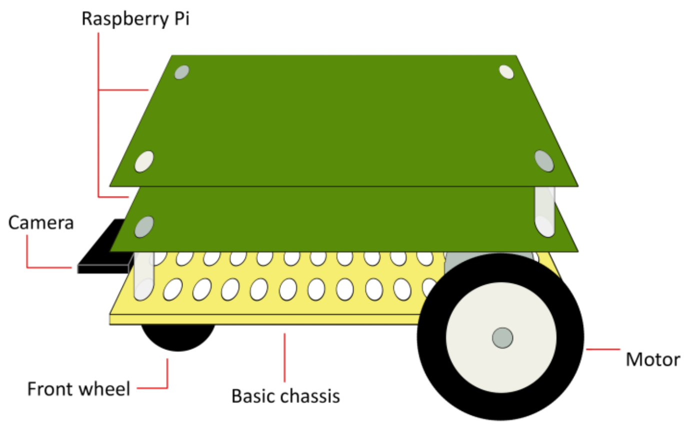
To navigate the maze, the robot will use IR sensors to find the distance of the walls. There will be two side and one front IR sensor. The side IR sensors will provide an error signal to keep the robot centred in between the walls. The front IR sensor will tell the robot to turn on the spot if it gets too close to a wall. By turning on the spot the robot will be able to find a new direction to move.

I have designed a new chassis, Because the basic chassis is too large to turn around in the maze. The new chassis takes up less horizontal space by holding the raspberry pi vertically

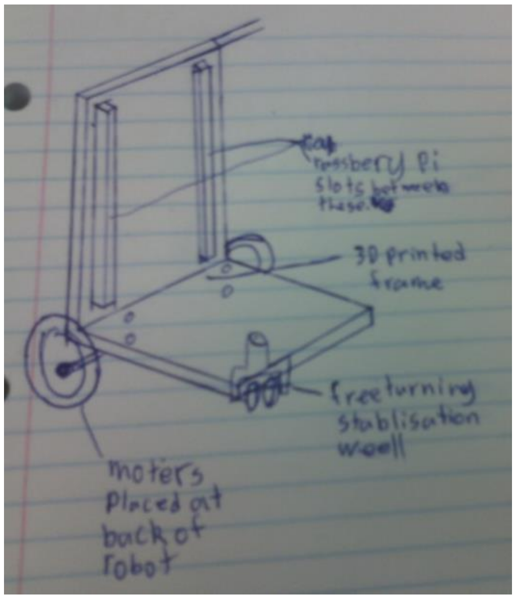
Results

Chassis Design

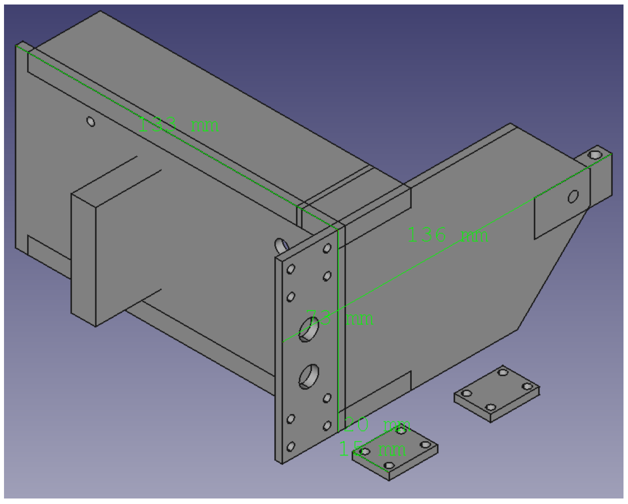
Original design:

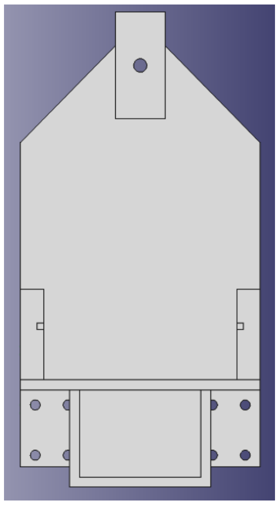


Design sketch:



CAD design (for 3D printing):





- The motors are attached behind the raspberry pi so that the robot will be balanced.
- There is a single free-turning front wheel to hold up the front without impeding the robot’s motion.
- The raspberry pi stots into grooves in the horizontal part of the chassis. Using grooves to hold the chassis in place allows it to be easily removed. Holding the pi vertical means the robot can be more compact for the maze.
- At the back of the robot there is a place to hold the battery.
- There are holes for cable management spread around the chassi.
- The front of the robot has a location to attach the camera mount.

Software

- the robot takes pictures with the camera.
- Scans the centre row of pixels for the line.
- If it finds no line, then it turns right.
- When it finds the it should use the PID to centre itself on the line, however it is currently never detecting the line.
- We also have code that should open the network gate but that needs to be tested.
- The robot can use IR sensors to get distances at the front and sides, but we don't have any IR sensors to use yet.

Discussion

To Finish the chassis design I need to get measurements for the IR sensors. Once I know how to space the screws I can add attachment points for the IR sensors.
For the code to work we will need fix whatever isn't working with the line detection. Once the line detection is working we will be able to tune the PID, that should get us to the end of the second quadrant. We will then need to add in the IR sensor code which is currently in another file. When both the side IR sensors detect a wall nearby the program will need to switch to a maze mode. While in the maze mode the side IR sensors should be used for the error signal. The front IR sensor should be constantly reporting the distance in front of the robot during maze mode. When that distance is small enough the robot should turn until it finds a direction it can move in

Conclusions

To conclude, the robot currently uses the camera to search for the line. When it finds the line the robot will use the PID to determine what angle it needs to turn. However, when it was tested the robot could not find the line, as a result the robot turned in circles. To complete the robot we need to; fix the line detection, tune the PID and setup the IR sensors for the maze.

References

- Google Self-Driving Car Project. (n.d.). Retrieved May 15, 2016, from https://www.google.com/selfdrivingcar/
- Self-Driving Car Technology and Computing Requirements. (n.d.). Retrieved May 15, 2016, from http://www.intel.com/content/www/us/en/automotive/driving-safety-advanced-driver-assistance-systems-self-driving-technology-paper.html
- Kaiwhata/ENGR101-2016. (n.d.). Retrieved May 15, 2016, from https://github.com/kaiwhata/ENGR101-2016