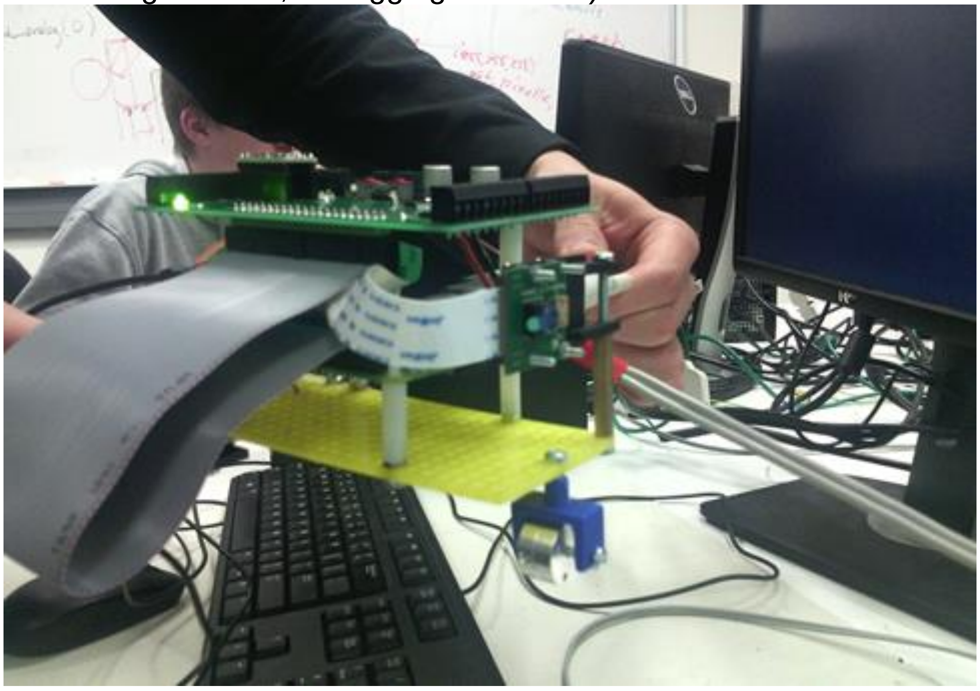
ENGR 101 AVC Robot Progress Report

Name: Christopher Straight
Student Number: 300363269
Course Code: ENGR101
Lecturers: Elf Eldridge and Bryan Ng
Assignment: AVC progress report
Due Date:16/5/2016
Team Name: Group 5
Team Members: Caitlin, Jacob, Christopher, Brandon, Pai Zhou
Github Repository: 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)



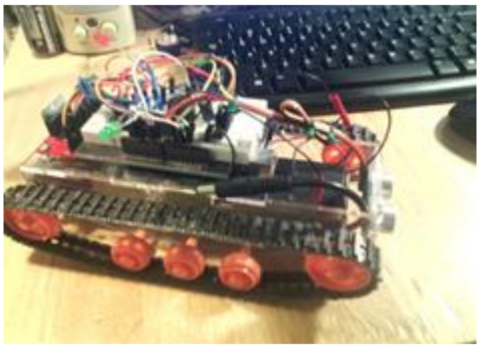
Abstract

In this project we built a robot to autonomously navigate along a line and through a maze, as well as opening a gate. We were able to make the robot follow the line on curves but not 90 degree turns. The robot can also open a gate but we didn't have enough time to implement navigating the maze.

Introduction

In this project the team was trying to create a robot using a raspberry Pi, we did this because we wanted to learn how to use a raspberry Pi as well as code an autonomous system. We only had 6 weeks to complete the project. Our resources were also limited. we had the basic parts that we were given such as the Pi and battery, 100$ for sensors and any 3D printed parts we could make. 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

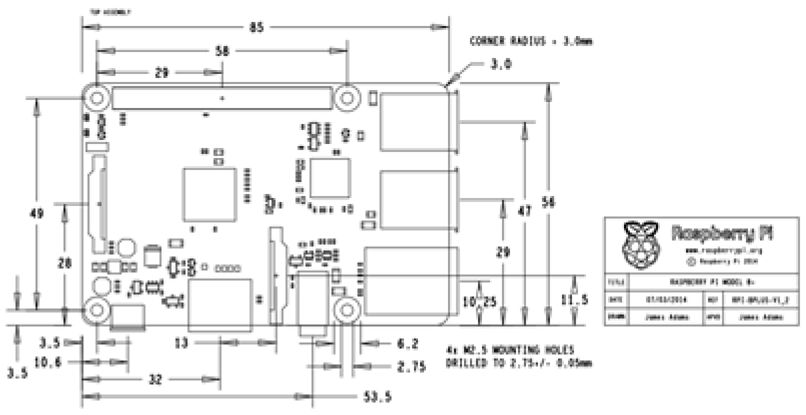


An autonomous robot is similar to but different, from an automated system. automated systems will perform specific actions in very specific situations, however if the conditions are wrong an automated system will stop working and be unable to correct the conditions. An example of an automated systems is this robot (picture right) which a team member 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.

Components:



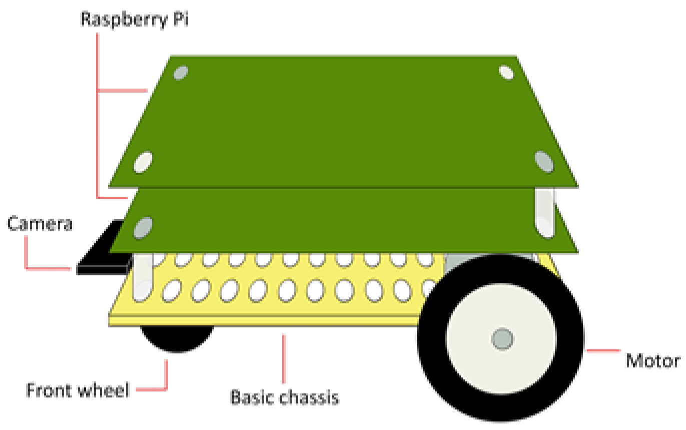
The robot was controlled using a Raspberry Pi and a PCB with an H-bridge and 10-bit ADC. The robot is powered by a 6.6-volt High discharge Life p04 battery. The Raspberry Pi Camera Rev 1.3 used in the robot is a five megapixel fixed-focus camera. The camera attaches to the Raspberry Pi via a 15 cm ribbon cable to the CSI port. The wheels used on the motors have a diameter of 33 mm.

The Raspberry Pi is credit-card sized computer; it can do pretty much anything that a normal computer can. The Pi can run programing languages such as Python, or as we used C++. The Pi is frequently in projects similar to our own. It was chosen instead of simpler systems like Arduino because can be connected to the internet via a wireless dongle. This internet connection means that we could SSH into the Pi and control it from a separate computer while it was being tested.

Method

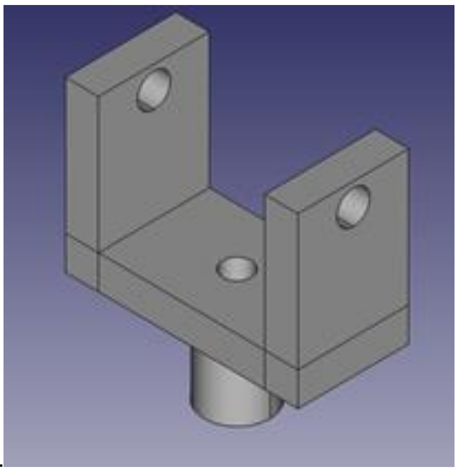
Chassis Design

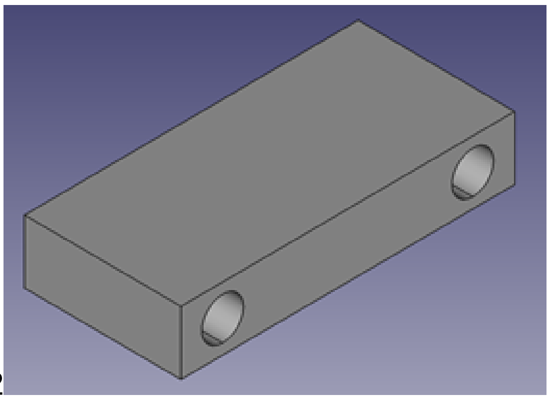
Original design:

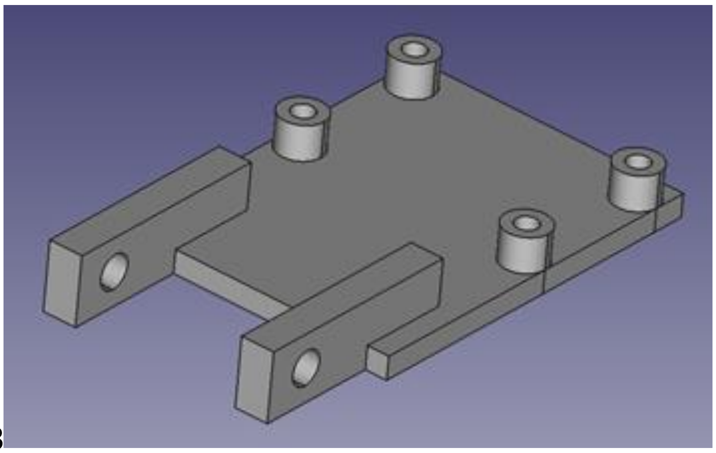


In the initial design of the robot a flat acrylic board is used as the base. The Raspberry Pi is attached to the base with plastic standoffs, then the PCB is attached above that with further standoffs. The RC motors are sandwiched between the bottom of the base and separate acrylic pieces screwed to the robot. The camera is attached to the base via a 3D printed mounting system. The battery is stored below the Raspberry Pi where it won't fall off during operation of the robot.

Camera mounting system:

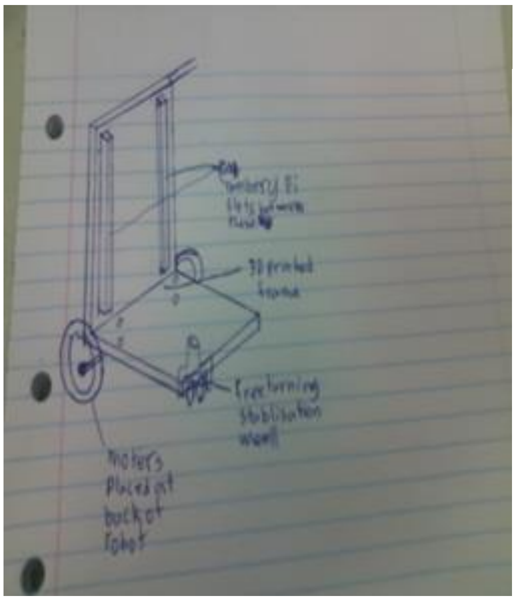
1


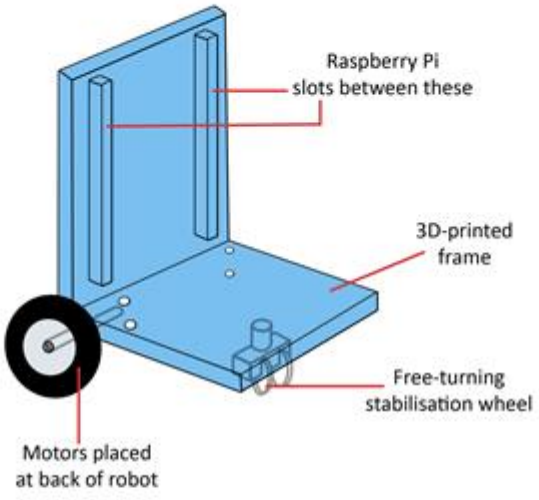
2


3


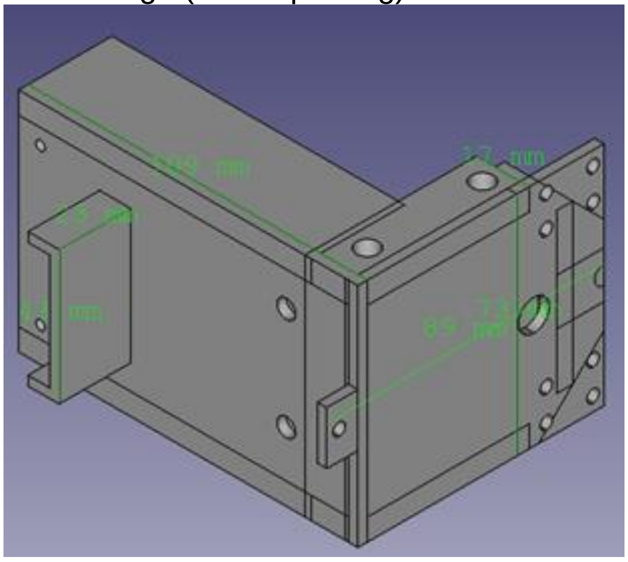
Part 1 attaches to the base with a single screw at the bottom. A screw through the arms of part 1 and an end of part 2 connects part 2. part 3 is connected in a similar fashion. The camera is then screwed onto part 3.

Design sketch:


Raspberry Pi slots between these
3D printed frame
Free-turning stabilisation wheel
Motors placed at back of robot


Raspberry Pi slots between these
3D-printed frame
Free-turning stabilisation wheel
Motors placed at back of robot

CAD design (for 3D printing):



The motors are attached at front of the raspberry pi, with a free-turning wheel to hold up the back without impeding the robot's motion. This means that the centre of mass is between the motors and back wheels, which will keep the robot stable. In the original design the motors were at the back, through testing we found that they worked better at the front. 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 chassis. The front of the robot has a location to attach the camera mount.
Due to time constraints and other people using the 3D printers we were unable to print the new chassis design. As a result, we decided to use the original design instead.
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, then the robot moves backwards until it finds the line or enters. When the line is in the image the PID is used to determine how much the robot needs to turn, to centre itself on the line. 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 then commands the gate to open, allowing it through.

The new chassis was designed, 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

Testing. Each week in our lab time the team tested that any alterations to the code worked and did not break any previously working sections. We also used the testing to determine what changes needed to be made to add functionality to the code. In week 1 we were able to get the robot to move forward and take pictures. By the end of week 2 we had the robot creating an error signal from the pictures, however it did not yet change its movement based on the error signal. Week 3 the robot was reacting to the line but it would turn the wrong way then start spinning. Week 4 we the P of the PID was completed and the robot mostly followed the line, although it was always turning left then correcting. The robot was able to successfully turn left but not right. This was the week we finished the code to open the gate. Week 5 we realized that the left motor was always slower than the right, this was why the robot was always turning. We replaced the left motor and the problem went away. Week 6 we tested everything on the maze and tried to finish implementing the I and D of the PID, however we were unable to do so.

Testing grids:

Week 1

test	**result**
Does the robot move	Yes, but only forwards
Does the robot take pictures	Yes
Does the robot produce an error signal	No (not implemented)
Does the robot follow on a straight line	No (not implemented)
Does the robot turn with the line	No (not implemented)
Does the robot open the gate	No (not implemented)
Does the robot detect the walls of the maze	No (not implemented)
Does the robot turn the corners in the maze	No (not implemented)
Does the robot navigate the maze	No (not implemented)

Week 2

test	**result**
Does the robot move	Yes, but only forwards
Does the robot take pictures	Yes
Does the robot produce an error signal	Yes, it scans the centre row of pixels to create the error signal.
Does the robot follow on a straight line	No (not implemented)
Does the robot turn with the line	No (not implemented)
Does the robot open the gate	No (not implemented)
Does the robot detect the walls of the maze	No (not implemented)
Does the robot turn the corners in the maze	No (not implemented)
Does the robot navigate the maze	No (not implemented)

Week 3

test	**result**
Does the robot move	Yes
Does the robot take pictures	Yes
Does the robot produce an error signal	Yes, it scans the centre row of pixels to create the error signal.
Does the robot follow on a straight line	Mostly, although it slowly moves to the left and then starts spinning when it losses the line.
Does the robot turn with the line	No (not implemented)
Does the robot open the gate	It opens the gate but the gate closes before it would have time to get through.
Does the robot detect the walls of the maze	No (not implemented)

Does the robot turn the corners in the maze	No (not implemented)
Does the robot navigate the maze	No (not implemented)

Week 4

test	**result**
Does the robot move	Yes
Does the robot take pictures	Yes
Does the robot produce an error signal	Yes, it scans the several rows of pixels to create the error signals. The robot then averages those error signals and uses the result.
Does the robot follow on a straight line	Mostly, although it slowly moves to the left, but that gets corrected when it goes too far.
Does the robot turn with the line	The robot turns left but continues mostly straight when it should turn right
Does the robot open the gate	Yes, the robot opens the gate with enough time for it to get through.
Does the robot detect the walls of the maze	No (not implemented)
Does the robot turn the corners in the maze	No (not implemented)
Does the robot navigate the maze	No (not implemented)

Week 5

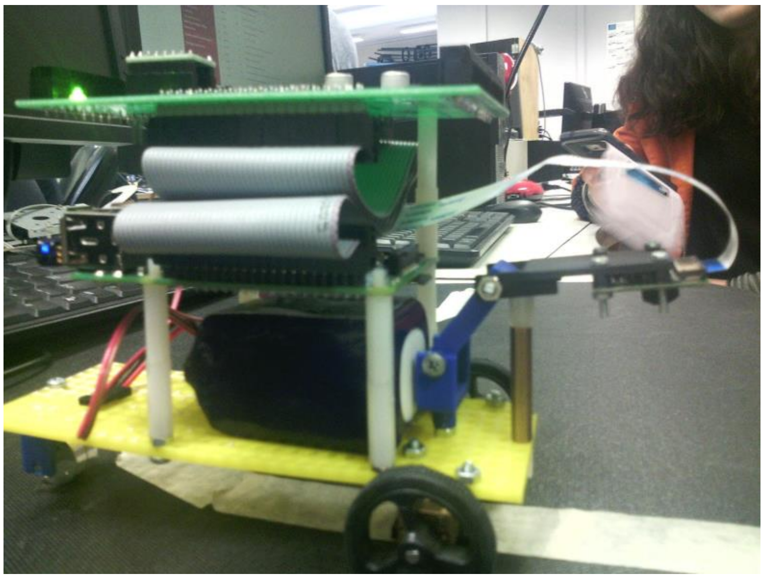
test	**result**
Does the robot move	Yes
Does the robot take pictures	Yes

Does the robot produce an error signal	Yes, it scans the several rows of pixels to create the error signals. The robot then averages those error signals and uses the result.
Does the robot follow on a straight line	Mostly, although it slowly moves to the left, but that gets corrected when it goes too far.
Does the robot turn with the line	The robot will follow curves in the line but not 90 degree turns.
Does the robot open the gate	Yes, the robot opens the gate with enough time for it to get through.
Does the robot detect the walls of the maze	No (not implemented)
Does the robot turn the corners in the maze	No (not implemented)
Does the robot navigate the maze	No (not implemented)

Week 6

test	**result**
Does the robot move	Yes
Does the robot take pictures	Yes
Does the robot produce an error signal	Yes, it scans the several rows of pixels to create the error signals. The robot then averages those error signals and uses the result.
Does the robot follow on a straight line	Mostly, although it slowly moves to the left, but that gets corrected when it goes too far.
Does the robot turn with the line	The robot will follow curves in the line but not 90 degree turns.
Does the robot open the gate	Yes, the robot opens the gate with enough time for it to get through.
Does the robot detect the walls of the maze	No (not implemented)
Does the robot turn the corners in the maze	No (not implemented)

Does the robot navigate the maze	No (not implemented)



Github Repository: github.com/thrand-Antharo/AVC-2016-group5

Final results:

The robot passed quadrant A without any problems. The robot successfully opened the gate long enough for it to pass then followed the line to quadrant B. In quadrant B the robot had some difficulty, it took a lot of time losing then re-finding the line on the tight curves. However, it was eventually able to complete the quadrant. The robot was only able to get part way through quadrant C as it was unable to complete the 90 degree turns.

Discussion

In the first week the team organized into roles although we did not know who was best at what so the assignments were largely arbitrary. If we had had a better Idea of who could do what we may have been able to get properly started sooner. In week 2 meetings were organised for the following weeks, although not everyone was able to attend meetings at the same time. To overcome the problem of timing meetings we found a time when most of the team could attend, then later emailed important information to the one member unable to be there. Throughout development of the robot we were unsure of how to implement the PID, as a result some team members were left without tasks while other were overloaded. If we did the project again we would want to research how to implement a PID before had so that we could divide it into steps and tasks. Because the I and D weren't working by the time of the final test they were removed from the code. In week 7 the robot had its final test. We were un able to print 3D print the new chassis for the robot due to time constraints. However, this did not affect the performance of the robot so it is not a major issue, although it would have been preferable to have it printed.

When the time came for the final test we had been unable to implement more than the P of the PID. This is probably why the robot had so much trouble with the turns. If we were to try again from the beginning, we may have been able to make a working PID in the time we had. The main cause for the PID's incompletion was that we did not know how to start work on it initially which delayed us.

Conclusions

The robot can successfully follow a white line on a black background however it has difficulty with tight turns and can't follow the line if it turns 90 degrees. The robot is also able to use network commands to open the gate, however it could not complete the maze because we realized that the robot would never make it that far so we didn't even include IR sensors. Despite the lack of full functionality, we did learn about using the raspberry Pi and creating autonomous systems. If we were to have another go at making the robot we would likely do much better.

References

Google. (n.d.). Google Self-Driving Car Project. Retrieved June 5, 2016, from https://www.google.com/selfdrivingcar/

Google. (n.d.). Google Self-Driving Car Project Monthly Report May 2015. Retrieved June 5, 2016, from https://assets.documentcloud.org/documents/2094029/report-0515.pdf

Google Self-Driving Car Project. (n.d.). Retrieved May 15, 2016, from https://www.google.com/selfdrivingcar/

Kaiwhata/ENGR101-2016. (n.d.). Retrieved May 15, 2016, from https://github.com/kaiwhata/ENGR101-2016

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

The Raspberry Pi Foundation. (n.d.). Camera Module - Raspberry Pi. Retrieved June 4, 2016, from https://www.raspberrypi.org/products/camera-module/

The Raspberry Pi Foundation. (n.d.). Raspberry Pi Documentation. Retrieved June 4, 2016, from https://www.raspberrypi.org/documentation/