VICTORIA UNIVERSITY, ENGR101

AVC Project Report

Jacob Beal

June 10, 2016

Contents

1 Abstract

The Following Lab Report details how the application of fundamental engineering concepts could be used in such a way to construct and operate an Autonomous Vehicle. This report will detail the design, construction, and programming decision that we made, the final outcome of the project and the end results, and in particular issues that arose, and how this was resolved.

The Robot in question did not complete the entire maze, and did not complete the third quadrant.

2 Introduction

2.1 Motivation

The purpose of the AVC is to demonstrate engineering skills taught in the ENGR101 Labs; and to work as a team to create an autonomous vehicle. This is used to solidify the teams understanding of fundamental Engineering concepts.

2.2 Problem

The challenge of a self-driving car is not immediately intuitive; the act of simply "following a line". This involves computing the relative position of the vehicle compared to the line. The team employed other methods, by tracking the relative position over time, that will be detailed later in the report.

2.3 Scope

The scope of this project was using C/C++, the ENGR101 C Library and skills we have learned in the Labs for the project. This includes *PID* Error Correction (Proportional, Integral and Derivative Responses) to follow the white line and other techniques taught in lectures.
We are also limited by our budget of V\$100 for use at the *"part bazzar"* and the equipment that were provided at the beginning of the project, the Roboshield, motors and Raspberry Pi.

2.4 Motivation

Understanding how an AVC works and solving a problem is important to the future, with autonomous vehicles increasing in prevalence, and being developed at break neck speed, it is of paramount importance that we understand how they work, and their limitations.

3 Background

3.1 Theory

h

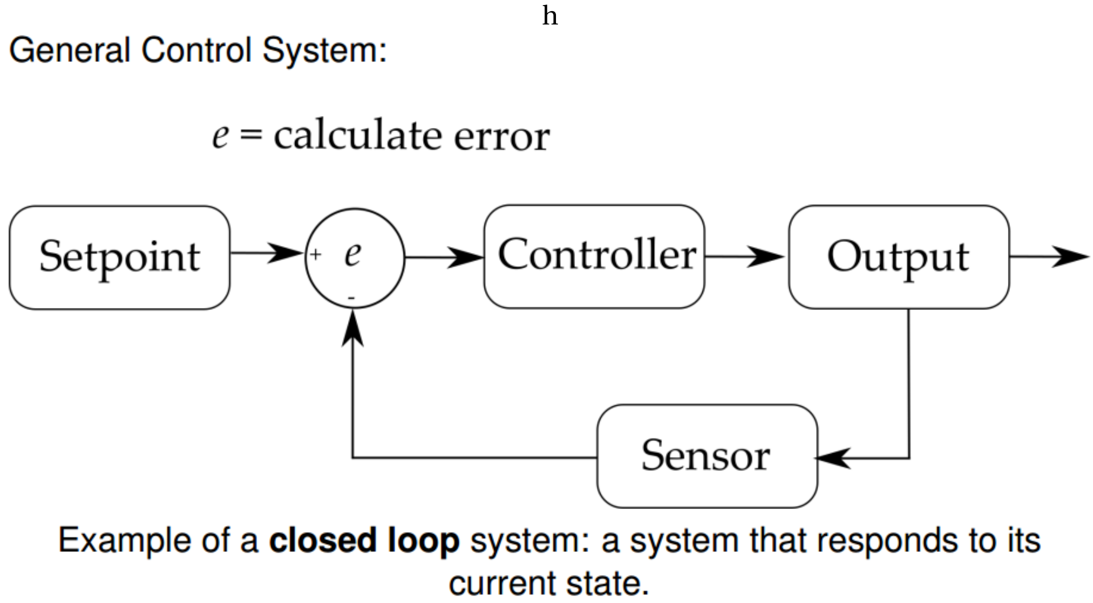

Figure 3.1: A Closed loop diagram [2]

3.1.1 Closed Loops

A closed loop system is a system which constantly loops, checking a physical variable. This variable is then computed in such a fashion as to deliver an output which is used in such a method to influence the physical variable the next time the loop is checked.[2]
A closed loop system is useful in the "follow the line" section of the AVC where a difference in position of the line and the centre of the camera, which can be used to generate a difference in wheel speed. This difference in wheel speed can cause a turn in the vehicle

3.1.2 PID

PID stands for *Proportional, Integral and Derivative* response to a change in the conditions in a closed loop.
A proportional response is the current conditions and how it responds to them immediately. i.e. the further the object is from the line, the stronger corrective response is.
A integral response checks over time to see whether the car is on one side too often, and compensates accordingly.
A derivative response is related to how quickly the proportional response changes, and therefore turns the car the other way when it approaches the line.

3.1.3 C and Raspberry Pi

The project will be written using C++, with an imported C library for the functionality to control the motors and receive input from sensors.

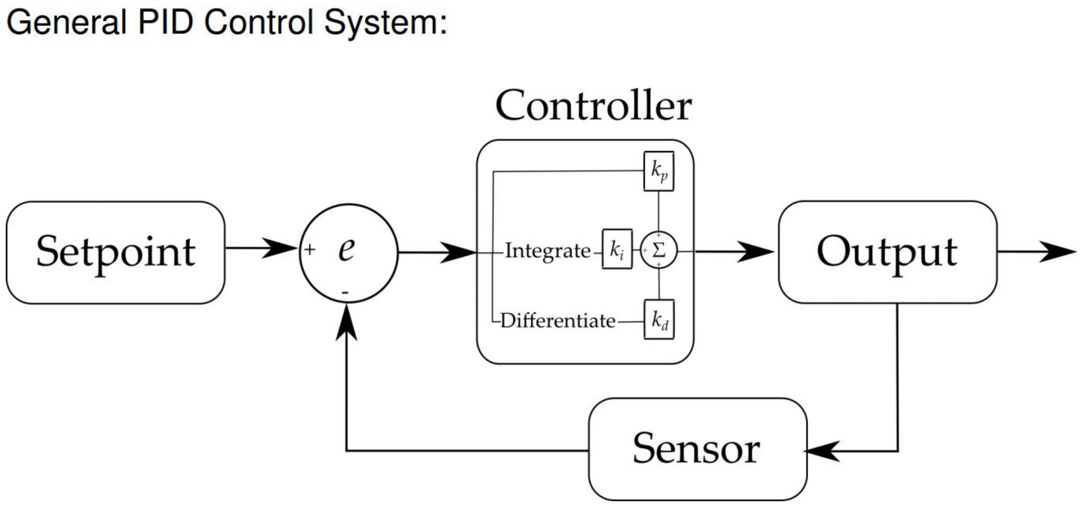

Figure 3.2: A PID system diagram[2]

3.2 RESEARCH

Tuning the PID system should be done in the specific order of Proportional, Integral then derivative responses. This is because the integral response can only be turned following the proportional response, so the offset at the end of the sinusoidal activity, and the derivative response merely acts as a dampener[3] [1]

3.3 THE MAZE

Figure 3.3: The Maze used to test the Autonomous vehicle

4 Methods

4.1 Equipment

The AVC Project was based on a Raspberry Pi and the Roboshield.
The Autonomous vehicle was built with a 'PiCam' camera (provided), a pair of motors (provided), a LiFe , 2 'Sharp GP2Y0A41SK0F Analog Distance' sensors, a 3D printed chassis, and half a ping pong ball.

4.2 Procedure

4.2.1 Hardware

The team first designed the layout of the vehicle. Such that any decisions made in the code for the The chosen design is based primarily on the given initial board. From this the motors were mounted at the front (See Fig 4.1), and the rear wheel, a mounted semi-sphere.
From there the network code to deal with the *portcullis*, found in Appendix .

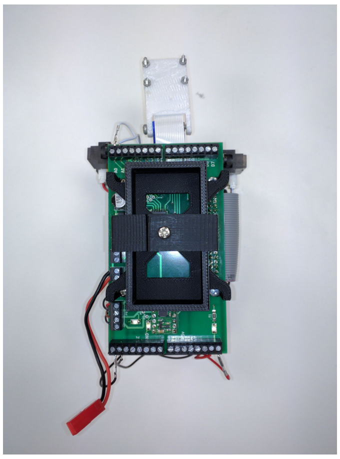

Figure 4.1: A top-down view of the design chosen for the AVC

4.2.2 Software

The team then worked on a *PID* system to model the behaviour of the motor for quadrant 2. Given a
From here the team worked to make the 'line maze' (quadrant 3), navigable. The team completed the code to make this work, however the numerous bugs caused it to fail at this stage.

5 Results

The vehicle was unable to complete the course and ended half way through the third quadrant. Upon reaching a terminus of a line, instead of doing a 180 deg turn. The vehicle went onto the 'correct path'.
The `CONSTANT_PROPORTIONAL` was set to 500. This reflects the fact that the

6 Discussion

6.1 Team

The team had a good general cohesion on the whole, and worked well cooperatively together during lab sessions. This had a positive effect overall as it led to better communication of ideas and improved the quality of the vehicle overall.
The lead developer had a general lack of availability due to them taking *PHYS114*, which consumed alot of their time. This meant that meetings were less frequent than desired and that bursts of work were sporadic and more long-winded. Overall this had a negative impact upon the project as the team had less time and less updates on the progress of individual aspects of the project. This led to increased stress in the team, and a slower development cycle overall. To resolve this in the future more frequent 15 minute meetings, via a VOIP service, such as google hangouts or skype, would better keep everyone informed, and allow swifter and more productive use of time

The loss of a team member immediately set the team back, and meant that the amount of time each member had to contribute was greater than other teams. This decreased team productivity, and led to more stress within the team itself. The increased workload led to a lower quality of work overall and had a negative impact upon the project itself. In the future the team needed to better manage time together to keep development on track.

6.2 Hardware

The decision to not have a rear wheel and instead have a semisphere from a ping pong ball. This was made because it would give a greater ability to turn the vehicle and make the AVC more responsive to the change in environment. This was beneficial in the second quadrant

The camera was not correctly plugged in upon initial construction. As a result the team fell behind their schedule and was unable to complete the project. As a result of this, the PID system that was developed during that period was ineffective and incorrect as it caused issues that otherwise would not have existed

6.3 Software

The co-efficient of proportionality is much higher then derivative which is higher than the integral response, to better respond to the present situation.
Following a mistake by members of the team in the final weeks, an incorrectly merged git commit caused the loss of the weeks work, this resulted in the team being left even further behind then previously. From this the team decided to only make changes to the code via the github.com UI such that it is always upto date and so that no merge commits would have to be done

Because of time restraints and loss of functioning PID code in the git merge, the teams took the decision to make a Proportional system as opposed to a PID one. The constant of proportionality was set to much higher than previously set and the the speed to be much lower. This would mean that less damping would be needed (via the differential component of PID) and it freed up development time to complete the third quadrant.

7 Conclusion

References

[1] Brian Douglas. *PID Control - A brief Introduction*. Dec. 13, 2012. URL: `https://www.youtube.com/watch?v=UR0hOmjaHp0`.

[2] James Eldridge. *Control Systems and Image processing for AVC*. Ed. by Victoria University of Wellington. May 15, 2016. URL: `http://ecs.victoria.ac.nz/foswiki/pub/Courses/ENGR101_2016T1/LectureSchedule/ENGR101_Lecture20.pdf` (visited on 05/15/2016).

[3] Inc Metso Minerals Industries. *PID Tuning Tutorial*. May 14, 2016. URL: `http://www.expertune.com/tutor.aspx` (visited on 05/14/2016).

8 Appendix

8.1 Weekly Log

Week 1

✓	ALL	Construct a Project Plan and Contact Julius
✓	Rhaz	Organise Meeting
✓	Jacob	Create GitHub and Establish Facebook Chat
✓	Andrew	Study SSH for Thursday Meeting
✓	Mitchell	Study Unit Testing
✓	Theo	Make a general plan for the chassis
✓	Julius	Get in contact with the group

Week 2

✓	ALL	Discuss ideas and start on the AVC Code
✓	Rhaz	Keep up to date on weekly tasks
	Jacob	Robot moving in straight line — appears to work, but haven't tested with a battery yet
✓	Andrew	Complete the README.md
✓	Mitchell	Look into the networking code, and how it works
✓	Theo	Figure out how to use the CAD software
	Julius	Show Up — Not turned up — Postponed

Week 3

✓	ALL	Progress update with team members and begin writing progress report
✓	Rhaz	Robot Opens Gate
✓	Jacob	Robot Opens Gate
✓	Andrew	Robot Opens Gate
✓	Mitchell	Update the README create Hardware updates and assist in the hardware efforts.
✓	Theo	Create a non-powered wheel case and battery case for the robot
	Julius	Show up and contribute to team meetings

Week 4

✓	ALL	Progress update with team members
✓	Rhaz	quadrant 1 completed
✓	Jacob	quadrant 1 completed
✓	Andrew	quadrant 1 completed
✓	Mitchell	Assist in hardware and pre-reading for quadrant 2
	Theo	Attach sensors to robot so it can detect the maze and obstructions — postponed as code not up to that section yet.
	Julius	Contribute in some way

Week 5

✓	ALL	Progress update with team members
✓	Rhaz	PID system functioning
✓	Jacob	PID system functioning
✓	Andrew	PID system functioning
✓	Mitchell	Bug fixing and tinker with PID constants
✓	Theo	Make battery clip easier
	Julius	Contribute in some way

Week 6

	ALL	quadrant 3 complete
	Rhaz	quadrant 3 complete
	Jacob	quadrant 3 complete
	Andrew	quadrant 3 complete
	Mitchell	quadrant 3 complete
✓	Theo	Attaching sensors to navigate quadrant 4
	Julius	Contribute in some way

8.2 Code Excerpts

8.2.1 Networking for Portcullis

```
void network(){
```

```
char message[24]; // Assigns memory to password.

connect_to_server("130.195.6.196", 1024); // Connects to Gate.
send_to_server("Please"); // Requests permission.

receive_from_server(message); // Assigns the password to 'message'.
send_to_server(message); // Sends password to server

printf(message);
}
```

8.2.2 STRAIGHT LINE

```
set_motor(1, 100);
set_motor(2, 100);
Sleep(1,0);
set_motor(1, 0);
set_motor(2, 0);
```