VICTORIA UNIVERSITY, ENGR101

AVC Project Report

Jacob Beal

June 12, 2016

Lab: Tuesday 9am—1pm

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 Aims

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 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 PCB, motors and Raspberry Pi.

2.3 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

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.(Eldridge, 2016)
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

h

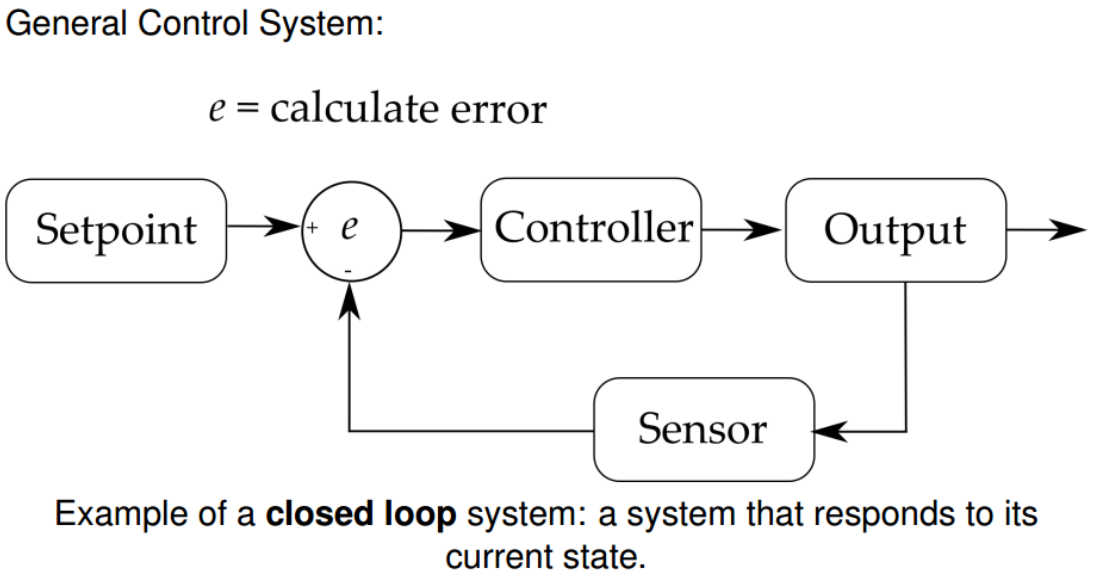

Figure 3.1: A Closed loop diagram (Eldridge, 2016)

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 Tuning a PID system

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(Metso Minerals Industries, 2016)(Douglas, 2012).

3.1.4 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.
A Raspberry Pi is a small credit card sized computer that is booted from an SD card and gen-

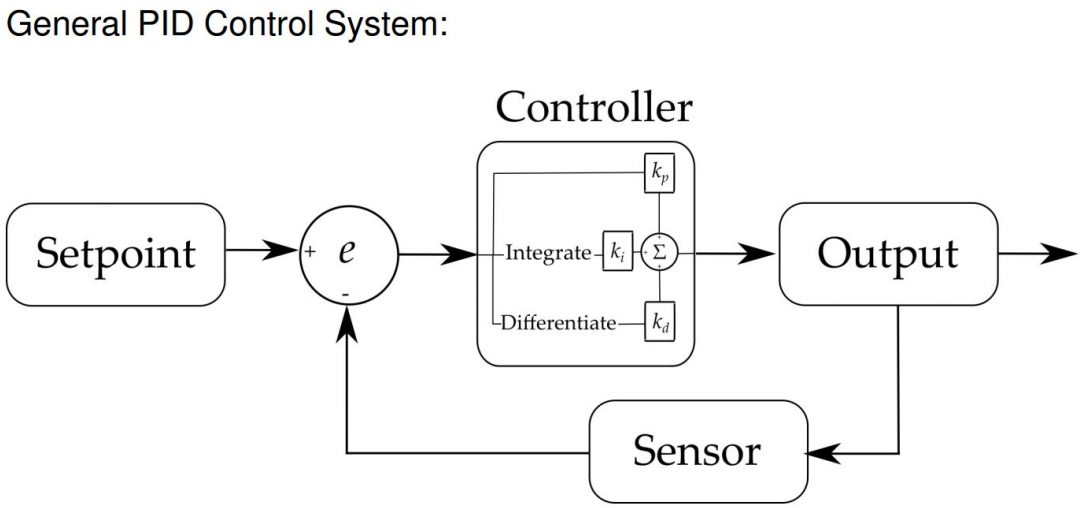

Figure 3.2: A PID system diagram(Eldridge, 2016)

erally runs a UNIX based operating system. For this challenge the team used Raspberry Pi 2 Model Bs.

3.2 The Maze

Figure 3.3: The Maze used to test the Autonomous vehicle

The Maze is (See fig 3.3) normally split into 4 discrete quadrants as follows:

3.2.1 Quadrant 1

Quadrant 1 contains a *portcullis* like gate, which is controlled via a network response, and following in a straight line along a white strip of masking tape

3.2.2 Quadrant 2

Quadrant 2 contains a twisting path of white marking tape. Some of which are sharp bends and others are more gentle.

3.2.3 Quadrant 3

Quadrant 3 contains a 'maze' of white masking tape.
It consists of straight lines with intersections and junctions and dead-ends with one path that leads out of the maze, and into quadrant 4.

3.2.4 Quadrant 4

Quadrant 4 starts inside a physical maze which will change between the testing phase of the challenge and the assessment.
This maze has physical walls above it, and must be navigated until it reaches a designated 'finish line'

4 Methods

4.1 Equipment

The AVC Project was based on a Raspberry Pi and the PCB.
The Autonomous vehicle was built with a 'PiCam' camera (provided), a pair of motors (provided), a LiFe battery, 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 vehicle wouldn't have to be rewritten because of hardware decisions to be made later.

The chosen design is primarily based on the initial plastic board and building a design which would allow both the PCB and the Raspberry Pi, with enough space left over

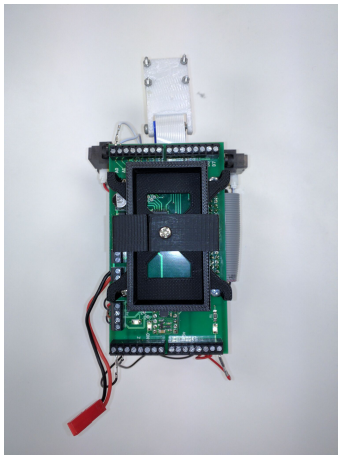

Figure 4.1: A top-down view of the design chosen for the AVC

such that the battery could be safely held without risking damage to it.

As such the design was chosen to hold the battery above the PCB and the Raspberry Pi, such that it was easy to remove, as it needed to be removed between testing periods, and on a stable base attached to the PCB, this was aesthetically appeasing and allowed easy access to the power switch. While also being smaller than the raised portcullis vertically.

4.2.2 SOFTWARE

The network code was first to be created, to deal with the *portcullis* found in quadrant 1. The code in question sends a predetermined message to the server using the provided API. The server would return a message which would be sent in a message to the server, this would cause the portcullis to raise and allow the vehicle to pass beneath. The code in question can be found in Appendix (9.2.1).

The quadrant 2 code works by taking a picture with the 'PiCam' and scanning the vertical middle of the picture taken. The average 'whiteness' of the pixels is found, this was used as a threshold for whether a pixel was 'white' or 'black'. (See Appendix 9.2.3)

The vertical middle pixels were read, and checked against the threshold value to see if they were 'black' or 'white'. Using the position of the pixel against the center. A value of in which direction the 'white' pixels were. This value was multiplied by `CONSTANT_PROPORTIONAL` to find which direction the line was in.

A case was also added in this section for if the entire screen was black, and to reverse back until the line was found again.

For quadrant 3, at predefined vertical columns it checks across the image to find whether the average is greater than the average of the entire image.
From this it derives whether there is a junction, and if it encounters a junction it carries out a predefined action. See Appendix 9.2.5.

If the analogue sensors were below a specified value it assumes that there are walls around it and hence it is within the maze.

Due to time constraints quadrant 4 (the maze), was not completed.

5 Results

The vehicle only made it to half way through quadrant 3. In a time of 3 minutes, 12 seconds. The value for $K_p = \frac{1}{500}$ (although in the code it is referenced as 500, but the operation is divided, thus the coefficient is $\frac{1}{500}$).

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 semi-sphere 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, which had tight bends, and other vehicles had been seen leaving the line and the team felt that this was a method of sticking to the line and prevent the vehicle from leaving the line.

Late into the project the battery clip was redesigned such that the battery was more securely held without the screw touching the battery, this was so that in the case of a failure or in the case the vehicle is dropped, the battery would be largely unaffected and to prevent the Li-Fe battery from leaking.

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 up to date and this would ensure 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. Provided this experience a better response would have been to not have lost the code in the first place, but the system functioned well enough to complete the course

7 Conclusion

The team worked well, despite numerous setbacks, in the implementation of the AVC vehicle.

8 Bibliography

References

Douglas, B. (2012, December 13). Pid control - a brief introduction. Retrieved from https://www.youtube.com/watch?v=UR0hOmjaHp0

Eldridge, J. (2016, May 15). Control systems and image processing for avc. Retrieved May 15, 2016, from http://ecs.victoria.ac.nz/foswiki/pub/Courses/ENGR101_2016T1/LectureSchedule/ENGR101_Lecture20.pdf

Metso Minerals Industries, I. (2016, May 14). Pid tuning tutorial. Retrieved May 14, 2016, from http://www.expertune.com/tutor.aspx

9 Appendix

9.1 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

9.2 CODE EXCERPTS

9.2.1 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);
}
```

9.2.2 STRAIGHT LINE

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

9.2.3 AVERAGE 'WHITENESS' OF A PICTURE

```
int determine_average(){
```

```
  int max = 0;
  int min = 255;
  int average;

  take_picture(); // Takes initial picture.
    for(int i = 0; i<320; i++){

    if(get_pixel(i, 120, 3)>max){ // Establishes Max.
      max = get_pixel(i, 120, 3);
    }

    if(get_pixel(i, 120, 3)<min){ // Establishes Min.
      min = get_pixel(i, 120, 3);
    }
    }

  average = (max+min)/2;
  return average;
}
```

9.2.4 Finding the white bias and hence direction of travel

```
for(int i=0; i<320; i++){
int error = average_error(i);

if(error >= threshold){ // If RPi sees 'white'
  error = 1; // Converts to binary represenation
  seeLine = true; // The Line can be seen
  num_of_white++;
}
else{ // If RPi sees 'black'
  error = 0; // Converts to binary representation
}

current_error = current_error + error*(i-160);
}
```

Where `average_error` is as follows:

```
int average_error(int i){
  int error1 = get_pixel(i, 116, 3);
  int error2 = get_pixel(i, 118, 3);
  int error3 = get_pixel(i, 120, 3);
  int error4 = get_pixel(i, 122, 3);
```

```
  int error5 = get_pixel(i, 124, 3);

  int average_error = (int) (error5+error4+error3+error2+error1)/5;

  return average_error;
}
```

9.2.5 Finding the junction type

```
bool seeLine = false; // Whether or not the line can be seen.
int current_error = 0;
num_of_white = 0;
take_picture();
for(int i=0; i<320; i++){
  int error = average_error(i);
    if(error >= threshold){ // If RPi sees 'white'
      error = 1; // Converts to binary represenation
      seeLine = true; // The Line can be seen
      num_of_white++;
    }
   else{ // If RPi sees 'black'
     error = 0;      // Converts to binary representation
   }
  current_error = current_error + error*(i-160);
}
int proportional_signal = (int) (current_error/PROPORTIONAL); //Sets proportional signal
// Print checks:
//printf("Current Error: %d\n", current_error);
//printf("Proportional Signal: %d\n", proportional_signal);
//printf("Number of White Pixels: %d\n", num_of_white);

//Turn
if(num_of_white > 120){
  if(average_error(20) >= threshold){
    leftpixel = 1;
  }
  else{
    leftpixel = 0;
  }

  if(average_error(300) >= threshold) {
    rightpixel = 1;
  }
  else{
```

```
    rightpixel = 0;
  }
  if(get_pixel(160, 0, 3) >= threshold) {
    frontpixel = 1;
  }
  else{
    frontpixel = 0;
  }


  if(leftpixel == 1 && rightpixel == 0 && frontpixel == 1){
    printf("Bend: -|, Turn: Front\n");
    proportional_signal = 0;
  }
  else if(leftpixel == 0 && rightpixel == 1 && frontpixel == 1){
    printf("Bend: |-, Turn: Front\n");
    proportional_signal = 0;
  }
  else if(leftpixel == 1 && rightpixel == 0 && frontpixel == 0){
    printf("Bend: L, Turn: Left\n");
    proportional_signal = -25;
  }
  else if(leftpixel == 0 && rightpixel == 1 && frontpixel == 0){
    printf("Bend: L, Turn: Right\n");
    proportional_signal = 25;
  }

}
if(leftpixel == 1 && rightpixel == 1 && frontpixel == 0){
  printf("Bend: T, Turn: Left\n");
  proportional_signal = -25;
}

if(seeLine){
  set_motor(1, 35+proportional_signal);
  set_motor(2, 35-proportional_signal);
  proportional_signal_previous = proportional_signal;
  Sleep(0,100000);
}
```

9.3 Team Agreement

9.3.1 ROLES

Rhaz	Project Lead and Hardware/Software support (organising team meetings, reporting regularly on progress, debugging software committing to Git)
Jacob	Software Development + Updating Git (writing core code and extending functionality)
Mitchell	Software Testing (debugging software and committing to git, writing test cases and documenting performance against milestones)
Theo	Hardware (building the chassis, testing components, connecting sensors, debugging hardware)
Andrew McManaway	Software help (Commiting to Git. Documentation.)
Julius Tilles	Hardware. Wasn't present for the first lab.

9.3.2 AGREEMENT

By signing below, all team members are acknowledging that they have read and committed to their part in the AVC. They acknowledge that they will attempt to complete the tasks agreed on by the group each week and document this on the team github account. They acknowledge that failure to meet these goals can result in the team recommending any member receives a lesser grade for their AVC report. In the event that a team member is unable to complete their task due to circumstances beyond their control (i.e. sickness, bereavement etc) that they will inform the team at the earliest possible time. Finally, the team acknowledges that a member going a week without contact with other team members (except when discussed with the team in advance) will constitute the member in question being considered AWOL. In this instance the team agrees to inform the ENGR101 course co-ordinator immediately. The penalty this for this can range from a reduction in the final grade to immediate failure of the AVC (and thus the ENGR101 course). Should the team unanimously agree that a member (or members) have failed to contribute to the AVC sufficiently for other reasons, on the day of robot testing the team will be given the opportunity to anonymously vote for a team member to receive 0% for the robot part of the AVC. Should the team choose this option they MUST be able to show that the member in question had been assigned tasks that they failed to complete and that the team had afforded them an opportunity to make up for past mistakes.