Timothy Blake: Assignment 3

October 27, 2024 6:03 PM

Student ID: 2442745

This is my COMP 444 Assignment #3 submission. It is a log of my Chapter 10 and 11 readings, definitions I learned during this time, and responses to food for thought.

This submission also includes my experiences with SIK exercises 4C Who am I, 5A Motors, and 5B Remote Controlled Robot, and 5c autonomous robot.

As usual, all source code and materials are available on my git:
https://github.com/malcolmsdad/COMP444/

Unit 5 - Introduction

October 17, 2024 8:10 PM

Unit 5: Robotic Control 1 – Feedback and Architectures

Actuators, affectors, and Sensors are all critical components of a functional robot, but we need a way to bring it all together and control the robot. In this unit you will begin to examine robotic control mechanisms.

Learning Objectives

After completing this unit, you should be able to

- describe various robotic control mechanisms, including feedback control.
- discuss the use of error in feedback control
- discuss and differentiate between types of feedback control including proportional control, derivative control, integral control, and combinations of these control mechanisms.
- describe the building blocks of robotic control, including control architectures.
- list several languages for programming robots.
- discuss the creation of a robot that uses feedback control.
- employ the Arduino and the SparkFun Inventor's Kit to create circuits which demonstrate the use of feedback control.

Readings

Please read the following chapters in the textbook:

- Stay in Control (Chapter 10)
- The Building Blocks of Control (Chapter 11)

Questions to Ponder

At the end of each chapter in the assigned readings there are questions labelled "Food for Thought." Please answer these questions as best you can in your weblog, which will become part of your portfolio of competence submitted for marking during this course.

Exercises

Exercises for this unit can be found in the *Instructor's Weblog* on the Landing. Please follow along with the exercises and programs using your own Arduino kit, and keep a record of your explorations in your own weblog.

Further Readings

At the end of each chapter in the assigned readings you will find a section titled "Looking for More." While the links and readings mentioned in this section are not assigned, please feel free to examine them if you are interested, or ask questions on the Landing.

Assignment 3

It is time to submit your weblog for Assignment 3.

From <https://comp.athabascau.ca/444/r1/unit05.html>

Chapter 10

October 17, 2024 8:17 PM

Readings

Feedback control is the next chapter of the book, and this is a topic I've been looking forward to. I'm very interested in learning different techniques and problems with confirming a systems actions. It's hard to know what to tell a robot what to do, if you don't know if those instructions work.

Several new terms are introduced immediately, including how a robot has a desired state, and how feedback is used to confirm its error, and accomplishment goals.
Maintenance goals are then discussed, which means activities that robot must maintain at all times. These could be keeping a robot balanced on two legs, or driving along a wall. These goals can be both interior and exterior.

Error is the difference between current and expect, and magnitude is how far off they are. This can help reduce error by identifying how far to the goal the system is.

The chapter then proposes how a feedback control robot executes. The goal is to walk along a wall, the error is how far the robot is from the wall. If the robot is within the desired distance to the wall, then its at goal state. The chapter then discusses how the controller would be written to (instructions), which sensors to use, and pseudo code on how the robot could operate. Software must be written carefully or oscillations can occur. This can be prevented by calculating error often. Increasing sampling rate can also help, sampling rate is how often data is retrieved for sensor for feedback analysis.

The book analyzes three types of feedback controls: proportional, proportional derivative and proportional integral derivative. P, PDF, PID.

Proportional Control

Proportional control systems react to the error in proportion to the magnitude of error. The factor of response can be altered by gain, which can also be altered based on proportion of change, this is called proportional gain. Damping is the process of decreasing oscillations and approach and end goal. Adjusting gains involves analyzing feedback and adjusting appropriately. Actuator uncertainty can make this difficult, if we don't have accurate actuator information.

Derivate control

Derivative control is more similar to using momentum to determine the gain of response. Starts slowly and increases speed, until it gets closer to its end state, then slows. Gets its name, because velocity is derivative of position. Start slowly to wall, gets faster, until it gets close to end goal, then slows down.

Integral Control

By tracking previous errors and their associated inputs, we can build an expectation of response. This can help adjust gain based on previous actions, and also some level of invalid data identification.

All types of control systems can be used together, and in fact PD is often uses in industrial plants for process control.

Finally we learn of closed loop (where the state of the system is considered before action is taken)

and open loop, where the inputs are estimated. Open loop systems rely on much more controlled environment and their state is well defined.

Chapter 10 - Definition

October 17, 2024 8:17 PM

Feedback control: is a means of getting a system to achieve and maintain a desired state.

Set point: The desired state the robot wishes to achieve.

Feedback: the information that is sent back into the system's controller.

Desired state / Goal state: where the system wants to be

Achievement goals: are states that the system tries to reach, such as location or end of a maze.

Maintenance goal: goals that require ongoing work, such as staying balanced

Error: The difference between the current and desired states.

Direction of Error: Confirms that system direction towards or away of error

Magnitude of Error: the distance to the goal state.

Sampling Rate: The rate that the sensor is checked and computed

Proportional Control: is to have the system respond in proportion to the error, using both the direction and the magnitude of the error.

Gain: The parameters that determine the magnitude of the system's response

Proportional Gain: The value of a particular gain is proportional to that of the error

Damping: Process of systematically decreasing oscillations.

Actuator Uncertainty: Makes it impossible for a robot to know the exact outcome of an action ahead of time.

Derivative Term: The proportionate amount that the velocity is reduced by.

Integral Term: Tracks previous errors and smooths out discrepancies and adjusting input controls.

Steady state error: Small lingering error, when a system has reached stable state.

Closed Loop Control: A system that continuously monitors feedback and adjusts its input.

Open Loop Control / Feedforward control: a system that does not use sensor feedback, and state is not fed back into system.

Chapter 10 - Food for Thought

October 17, 2024 8:17 PM

What happens when you have sensor error in your system? What if your sensor incorrectly tells you that the robot is far from a wall but in fact it is not? What about vice versa? How might you address these issues?

Sensors are the eyes, ears, touch of the robot. They provide a wealth of information for processing, and much of the robots actions are based these input. If the input is wrong, then the responses will also be wrong. In the programming world, we have an expression: "garbage in, garbage out", I suppose in the robotics world, its more like "garbage in, garbage bot." This I because if there isn't proper error correction, or input validation / sanitation than the robot will act completely incorrectly or unexpectedly. This could lead to damage or injury to anyone in the environment or damage to the robot.
The perfect example is if a sensor incorrectly tells the robot its far from an wall, when in fact its next to it because the robot would drive full speed right into the wall. Damaging the robot and possibly the wall. The proper reaction would be to slow down, stop or change direction - avoiding contact. There are several ways to prevent sensor error. First would be to increase sampling rate of the sensor. Sensors rarely fail, so the larger the data set, the more likely the data will be accurate. Increasing sampling rate, provides more data, and increases accuracy. Applying error control, or input validation could also work. Using trending, or averages, or even percentage change over previous readings could lower the impact or eliminate erroneous or outlier readings. Finally, sanitizing an environment could help, as objects within an environment could have an impact on sensor readings, such as ambient light, textured surfaces etc.

What can you do with open loop control in your robot? When might it be useful?

Open loop control is great for robots that have well defined and controlled environments. They can reduce cost, are more reliable, improve efficiency and speed of operation. A good example of one could be a dishwasher. The dishes are always in the same place, the robot knows the cycles to run and what order, as well as which arms to oscillate. In this case open loop would reduce the cost of making the robot, as it does not need complicated sensors (like a shaft position sensor) or weigh scales (to detect number of dishes), it just runs its preprogrammed process.

Same with large industrial manufacturing, the inputs are strongly controlled (i.e.: sheet metal) and the actions of slicing the metal, placing on a press, stamping a license plate number.

Chapter 11

October 17, 2024 8:17 PM

Reading

The next chapter focuses on the various control systems within computer. First architectures are analyzed.
These describe the principal and constraints for a robots brain. Architectures are the building blocks that are available to the robot. That includes motors, processors, sensors. The brain of a robot has a lot going on. Robots must be robust to failure, including being able to validate input from other processes.
Hardware is good for fast specialized uses, software is more robust.
Algorithms are used to solve problems that could stand in the way of achieving a robots goal. Comp Sci devotes a great deal of research to develop and analyze algorithms.

Robots use a variety of programming languages. Choosing or designing software systems I a difficult task. Control architectures are a good way to determine how a robot can be programmed. Modern programming languages are "Turing universal" meaning it can be used to write any program. As robots develop, programming languages are evolving to match.

Next are the various architectures used for programming languages. These including deliberative where reponses are considered for long term, reactive, where reactions are immediate based on input, hybrid which combines the two, or behaviour-based where processes are broken down further. When choosing a control architecture several things must be considered. Innlcuding, sensor noise, does the robot have enough information, how quickly can robot sense changes in environment, how quickly can the robot? Is there any noise in the response (actuator), how did pass actions go?

Control architecutres differ fundamentally including how they treat the following: Time (how fast do things happen) Modularity, what are the componets of the system. Reprenstation: what does the robot know and keep in its brain.

The term TIME means how quickly the robot has to respond to environment and how quickly it responds.

Modularity is how the robots brain is broken down into functions including sensing, acting, planning.

Representation is complicated and will have its own chapter.

Chapter 11 - Definition

October 17, 2024 8:17 PM

Control Architecture: The architecture provides guiding principals and constraints for organizing a robot's control system (it's brain)

Algorithm: Process of solving a problem using a finite set of step by step instructions

Time-scale: How quickly the robot has to respond to the environment, compared to how quickly it can sense and think.

Deliberative Control: Long term time-scale consideration for response.

Reactive Control: Response to the immediate real-time demands of the environment.

Hybrid Control: Combines the long time-scale of deliberative control and the short time-scale of reactive.

Behavior Based Control: Multiple moduiles, including perception, planning, acting and do their work in sequence, with output of previous sequence the input for the next

Modularity: refers to the way the control system is broken into pieces of components.

Chapter 11 - Food for Thought

October 17, 2024 8:17 PM

How important is the programming language? Could it make or break a particular gadget, device, or robot?

According to Materic, nearly any programming language can be used to write a program for a Robot, as long as its "Turing universal". Turing universal means that it can be used to write any program. Although this could be true, I would argue that they shouldn't. Programming languages come in a million different formats: procedural, functional, object oriented and scripting. Each types has their own benefits and fall backs. The type of control type would ultimately be the best criteria for determining the type of programming language.
Beyond the type of languages, there are also environmental issues to consider: Some languages are pre-compiled and can execute using machine code, others must be read and compiled at runtime (such as scripting). If the time-scale isn't sufficient for the robot to respond to the environment in time, then this would not be a good selection.
Just because any language can be used, doesn't mean it should - control architectures, timing, developmental practices and libraries must be considered first.

With the constant development of new technologies that use computation, do you think there will be increasingly more or increasingly fewer programming languages?

I absolutely believe that new programming languages will be created as new technologies and robots evolve. In the 40 years that I've been developing, I've seen thousands of languages rise and fall, even learned and professional worked with some of these languages. Artificial intelligence is rapidly changing all aspects of the world, including technology. Perhaps AI will introduce new language syntax or protocols for interacting with robots? Perhaps programming languages may even be made obsolete.

SIK Excercises

October 24, 2024 6:41 PM

Starting with Circuit 4C and finishing with 5C, the last circuit

4C - DIY Who am I?

October 25, 2024 5:55 PM

That was fun! It was super easy to setup, and I played the game with my daughter. This programming circuit didn't introduce any new devices or sensors, but it did increase programming skills. I'm a pretty skilled programmer so for this one, I'm going to add a new category with new words, then randomize the category. This will involve using a dimensional array.

The new category will be: **video games**
Words are: "Super Mario Bros", "Luigi Mansion", "Minecraft", "Grand Theft Auto", "Tetris", "Donkey Kong", "Mario Kart", "Mario Party", "Jumpman", "Kirby", "Sonic", "Animal Crossings", "Mega Man", "Teenage Mutant Ninja Turtles", "Double Dragon", "Metroid", "Legend of Zelda", "Just Dance", "Wii Sports", "Contra", "Harry Potter", "Yoshi Island", "Dance Dance Revolution", "Fortnight", "Pacman", "Excite Bike"

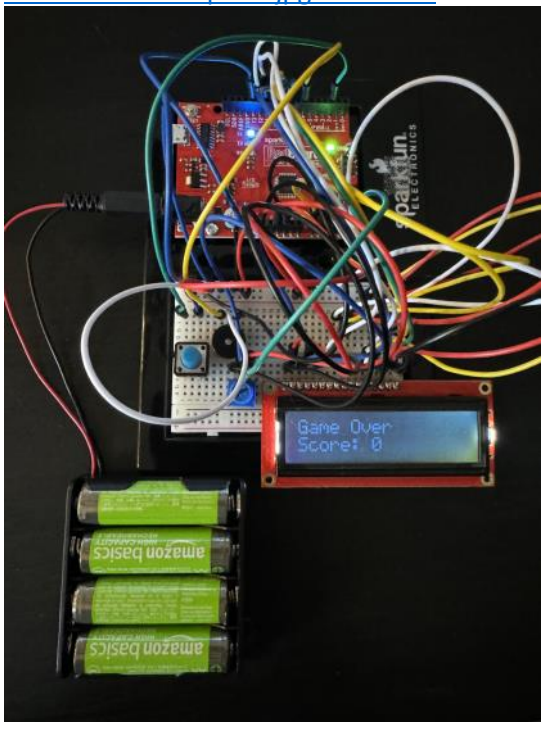
The source code is available here:
https://github.com/malcolmsdad/COMP444/tree/b65177f4241edb80e1ba653ad73f1b903677e104/Unit%205/C4C%20-%20DIY%20Who%20am%20I/SIK_Circuit_4C_DIYWhoAmI_Challenge

Here is a demonstration of the circuit:
https://youtu.be/bvZUzU31mxc

Notes:

- You will notice that there is now a random category generation (I had to hit reset a few extra times for the random number to generate)
- As well as a list of 25 new words to guess (Video Games).
- I also found the game too easy, so I reduced the amount of time per word
- Finally, there are new winning and losing game tones.

The board with battery pack:
https://github.com/malcolmsdad/COMP444/blob/main/Excercises/C4C%20-%20Who%20am%20I%20-%20Complete.jpg?raw=true



5A - Motors

October 26, 2024 11:08 AM

This was a pretty simple circuit to set up, I was initially initiminated by the motor controller, but after reviewing the inputs pins, its pretty straightforward.

Here is the image of the completed board and motor:
https://github.com/malcolmsdad/COMP444/blob/main/Excercises/C4D%20-%20Motors.jpg?raw=true

Initially the board didn't run well. I typed a speed and hit enter, it'd set the speed my input, then a minute later it'd set the speed to 0. I suspected it was a parseInt issue, but I didn't know where the 0 was coming from. After a quick google I found that the Serial Monitor was sending a carriage return along with the Enter, meaning after the speed value was parsed, an enter was parsed (as a zero). This was easily fixed. Here is the link where I found the solution:
https://forum.arduino.cc/t/serial-parseint-set-0-error/1145174

Coding Challenge

- For this challenge, I decided to add a photoresistor that would only move the motor when no light was detected (ie: a hand over the sensors)
- I also added a button, with a release handler, so that any time the button was pressed, speed would increase by 50, and roll back to zero, when over speed is reached.

Here is a gif of the circuit in action. Notice the speed increases with button press, or the motor stops when I remove my hand:

https://github.com/malcolmsdad/COMP444/blob/main/Excercises/C4D%20-%20Motors%20Challenge.GIF?raw=true

The code is available here:
https://github.com/malcolmsdad/COMP444/tree/69a919f94534ccac7fd30a15471c809bbcaac453/Unit%205/C4D%20-%20Motors/SIK_Circuit_5A_MotorBasics_Challenge

5B - Remote Controlled Robot

October 27, 2024 11:43 AM

This robot introduced ASCII characters, parsing of strings, and the setup involves assembling some wheels motor and robotic body.

I finished the setup and programming of the robot with no major difficulties. The robot worked quite well, although I found a bit of a difference between the SIK textbook and the actual code.

Coding Challenge #1 asks to enhance the circuit by inputting the distance that should be driven, however the sample code already has this functionality! Here is an animated gif of the action:

https://github.com/malcolmsdad/COMP444/blob/main/Excercises/C5B%20-%20Remote%20Control%20Robot%20-%20Complete.GIF?raw=true

These are the commands that I ran:
12:39:55.434 -> f 200
12:40:04.067 -> l 50
12:40:12.099 -> r 100
12:40:18.175 -> b 50

Coding Challenge #2:
For this challenge, I have decided to introduce a new manoeuvre: the avoidance zig-zag. I'm hoping that I can build a manoeuvre that will be able to take a pre-programmed avoidance manoeuvre around an object. The manoeuvre would consist of a turn to the left, drive a short distance, turn to the right, resuming a trajectory parallel to the original, continue along that path. Then pivot to the right, and continue until meeting with the original path, then pivot left, to continue the original trajectory.

In order to do this, I had to have a consistent driving behaviour, which this robot didn't have. Loose wires on the breadboard are a constant problem, week connection leads to poor performance from the motors. I had to disconnect and swap wires several times.
The next problem I ran into was the paperclip arm was getting caught in seems of my hardwood floor. I found an old wheel from an old office shelf, that, with the help of a quick tie, made a very functional pivoting third wheel.

It worked extremely well! You can see the groove of the wheel axel fit perfectly with the paper clip, and the tension that the blue pull tie puts keeps the axel very stationary. The plastic wheel pivots about the metal axel making it very maneuverable, with no drag. It also has a built-in parking brake.

Attempt #1:
The first attempt was mixed success. The routine followed the correct steps, however the duration of the turns and driving was not correct. The robot did not end on the same path.

Aligning of wheels: The biggest problem I noticed during this exercise was the robot tending to veer left while driving straight. This meant the right wheel is larger, or rotating at faster speed. I will use a gain on each motor to adjust the speed. I completed this with the variable:

```
const double gainRightWheel = 0.95;  // reduce or gain by a factor of ...
const double gainLeftWheel = 1;      // reduce or gain by a factor of...
```

https://github.com/malcolmsdad/COMP444/blob/main/Excercises/C5B%20-%20Remote%20Control%20Robot%20-%20Gain%20code.png

I found the code in the sample very messy, especially having so much procedural code in the main functional loop, so I moved the driving functions into dedicated functions, and refactored the loop:

```
if (botDirection == "f") driveForward(driveTime * distance.toInt());
else if (botDirection == "b") driveBackward(driveTime * distance.toInt());
else if (botDirection == "r") driveRight(driveTime * distance.toInt());
else if (botDirection == "l") driveLeft(driveTime * distance.toInt());
else if (botDirection == "z") driveZigZag(driveTime * distance.toInt());
```

https://github.com/malcolmsdad/COMP444/blob/main/Excercises/C5B%20-%20Remote%20Control%20Robot%20-%20Loop%20code.png?raw=true

The results were mixed, I appear to have poor connections within the breadboard. Sometimes, the robot moves in controlled consistent movements, other times it zips around uncontrollably. Following the instructions in the SIK, I tried powering from both battery as well as USB. Battery pack was most useful for testing of the maneuver.

The source code is here:
https://github.com/malcolmsdad/COMP444/tree/994716a68bd1c7348358d05e00161ec3df68c20f/Unit%205/C5B%20-%20Remote%20Control/SIK_Circuit_5B_RemoteControlRobot_Challenge

5C - Autonomous Robot

October 27, 2024 6:08 PM

This robot will enhance the previous robot and give it some smarts to be able to drive around on its own. It will include a distance sensor to detect obstacles.

It was a really simple setup, just add the distance sensor, power, ground, and data wires. Here it is:

https://github.com/malcolmsdad/COMP444/blob/main/Excercises/C5C%20-%20Autonomous%20-%20Complete.jpg?raw=true

And here it is meeting "Grandpa" Roomba:
https://github.com/malcolmsdad/COMP444/blob/main/Excercises/C5C%20-%20Autonomous%20-%20Met%20Grandpa.GIF?raw=true

Coding Challenge:
For the challenge, I'm going to increase the distance for when the robot reacts to objects. This is because the robot often hits obstacles because it can't respond in time. I will also make the robot do a little spin, before doing another direction.

Here is the code for this challenge:
https://github.com/malcolmsdad/COMP444/tree/4f8f4ced3e8d557d9c132cecae374af11d4bbf10/Unit%205/C5C%20-%20Autonomous/SIK_Circuit_5C_AutonomousRobot_Challenge

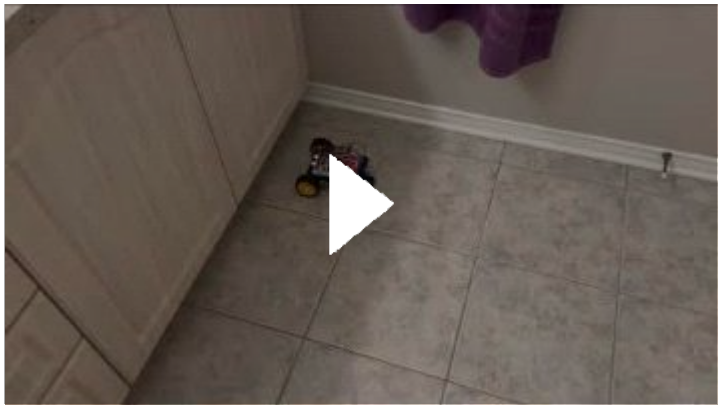
I still wasn't happen with the performance of the robot, so I decided to apply proportional control, and have the robot slow its speed as it approaches an obstacle. This greatly improved the performance of the robot, you can see it slowing before obstacles in the video here:

Coding Challenge Part Deux: Proportional response

I suspect if I installed two additional sensors facing 30 degrees forward, I could greatly improve the performance of the robot. Adding extra sensors would provide additional context of the environment (turn left or right when an obstacle is detected) as well as redundancy for faulty readings, which occasional happen. It's funny, but in my opinion, adding the proportional response was almost biomimetic, in that the robot is hesitant or afraid of the obstacle.

Here is the video of it running in a controlled environment, my bathroom. I moved from the previous area after an incident falling downstairs. NOTE: My HVAC is running, and I'm quite sure it causes errors in the distance sensor.

C5C - Autonomous Robot - Proportional Response



As it approaches an object it proportionally slows the motor speed, until it hits the distance limit. At this point it backs up, does a little spin, and resumes its driving at full speed. At one point in the middle my furnace turns on, and the robot goes haywire, it recovers shortly after. This is due to a irregular sensor reading, which I notice often while connected via USB.

Code for this circuit is here:
https://github.com/malcolmsdad/COMP444/tree/4f8f4ced3e8d557d9c132cecae374af11d4bbf10/Unit%205/C5C%20-%20Autonomous/SIK_Circuit_5C_AutonomousRobot_Challenge_Proportional