UNITED STATES MILITARY ACADEMY

LAB REPORT

XE475: MECHATRONICS
SECTION B1

LTC LOWRANCE, DR. ROGERS, & MAJ LARKIN

BY

CDT LUIS AGUIRRE '18 F-1 & CDT SHARPSTEN TAYLOR '18, I-2

WEST POINT, NY 10996

30 OCT 2017

____ MY DOCUMENT IDENTIFIES ALL SOURCES USED AND ASSISTANCE RECEIVED IN COMPLETING THIS ASSIGNMENT.
____ I DID NOT USE ANY SOURCES OR ASSISTANCE REQUIRING DOCUMENTATION IN COMPLETING THIS ASSIGNMENT.

SIGNATURE: AGUIRRE FERNANDEZ.LUIS FERNANDO.1467628549 Digitally signed by AGUIRRE FERNANDEZ.LUIS FERNANDO.1467628549 DN: c=US, o=U.S. Government, ou=DoD, ou=PKI, ou=USA, cn=AGUIRRE FERNANDEZ.LUIS FERNANDO.1467628549 Date: 2017.10.30 23:25:12 -04'00'

SHARPSTEN.TAYLOR.ROSE.1022718823 Digitally signed by SHARPSTEN.TAYLOR.ROSE.1022718823 DN: c=US, o=U.S. Government, ou=DoD, ou=PKI, ou=USA, cn=SHARPSTEN.TAYLOR.ROSE.1022718823 Date: 2017.10.30 23:17:21 -04'00'

Contents

1. Introduction

1.1. Purpose

The purpose is to provide an introduction to PID Controllers.

The objectives of this lab consisted of surrounded closed-loop system, PID Controllers, and calculating error. By implementing a implementing a PID controller, we tune its gain values in order to improve the performance of the system. The objectives of this lab is applicable to the most closed-loop systems. Technology such as autonomous robotics, and self driving cars are examples that utilize PID controllers to calculate error and respond with the least overshoot, fastest rise time, and most stability.

1.2. Reference

Lowrance, Christopher. Pan-Servo Tracking System using PID Control (West Point, NY: Department of Electrical Engineering and Computer Science, 2017).

1.3. Audience

The goal of this lab is to build a system with a consisting of a motor shaft that is controlled by an Arduino UNO and receives feedback of the direction from a compass on top of the motor shaft. This system must correct its direction back to its original position upon a

2. Theory

Proportional integrative design (PID) is the leading theory behind the design and programming for this lab. Control law, closed-loop systems, and Pulse-Width Modulation (PWM) are additional theories that play a role but where learned in previous labs.

2.1. PID Controller

A proportional integrative derivative controller is a close loop-system that calculates the error of a system. Figure 1. shows the fundamental function used to sum the proportional, integrative, and derivative factors. In this lab, Proportional accounts for present values, Integrative accounts for past values, and Derivative accounts for predicted future values of error. In a control theory their are three fundamental objectives: Achieving Stability, Improving performance of the response, and reducing the steady state error. By adjusting these three factors, the system performance can be improved.

1. Increasing Proportional Gain: Decreases Rise Time, Increases Overshoot, Increases settling time, Decreases Steady-State Error

2. Increasing Integrative Gain: Decreases Rise Time, Increases Overshoot, Decreases then Increases Settling Time, Eliminates Steady-State Error

3. Increasing Derivative gain: increases Rise Time, decreases overshoot, decreases Settling Time, no change Steady-State Error

$$u(t) = K_p e(t) + K_i \int e(\tau)d\tau + K_d \frac{de}{dt}$$

Figure 1: Weighted Error Sum [1]

One can reference figure 2 in the document. The figure is on page 7.

2.2. Additional Theories

Control Law and PWM also play a role in this lab as it defines the speed at which the system will operated depending on the feedback received and the width of the pulses. a micro controller is a common way to create PWM.

Lab 3 describes in detail the theory behind closed-loop systems. overall, closed-loop systems provided a feedback mechanism in a system to account for error or any other form of event that a system needs to account for and react. An example of a

3. Experimental Setup and Procedure

For this lab, a compass, DC Motor controller, data logging shield, Arduino Uno, and motor shaft were the materials required to complete the lab.

1. Build an electrical system that will read and drive the motor shaft. Assure that the electrical system works.
2. Build a program that reads at a desired rate the location of the motor shaft and the changes in the motor shaft.
3. Add a button to the bread board as part of the electrical system.
4. Add to the program a function that records a starting position of the compass when the button is pressed.
5. Incorporate a code to the program that adds the error of the starting position to the current position.
6. Using PID, add code that returns the compass to its original position.
7. Adjust the PID coefficients, to test and experiment which values works the best in order to achieve desired results.
8. Success will be based on the reaction time, smoothness, and accuracy of the compass.
9. incorporate a data logger to the electrical system.
10. Add code that uses the data logger to record the behavior of the system.

4. Results and Discussion

4.1. Basic Operation and Assumptions

One of the biggest initial challenges to the successful operation of the controller was the wrap problem. On a range of 0 to 360 degrees, the controller needed to understand which direction to turn that would respond fastest, even when that turn wrapped over the 0, 360 degree point. We designed the controller to eliminate this problem in the beginning setup loop. When the button is pressed to set the desired direction, the controller records this direction as the new 0 degrees, and sets a variable offset. This variable is then used whenever a new heading measurement is taken, so that all measurements are relative to the desired direction 0, with a maximum of -180 degrees or 180 degrees to either turn direction from the 0. This eliminates the need for code to cover wrapping, speeding up operation of the controller.

4.2. PID Controller Performance

The controller successfully accomplished the lab goals, quickly adjusting to disturbances and settling on the desired direction, set by a pushbutton. We initially tuned Kp until the system responded to disturbances within a desired time and did not have a large overshoot. Following the tuning of Kp, Ki and Kd tuning was used to decrease steady state error and to further smooth the system response. Figure 2 depicts the controller's response to a single negative angle disturbance. The system settles with approximately -3 degrees of steady state error. We built in a tolerance of 5 degrees from the desired direction in order to reduce oscillatory behavior around the desired direction.

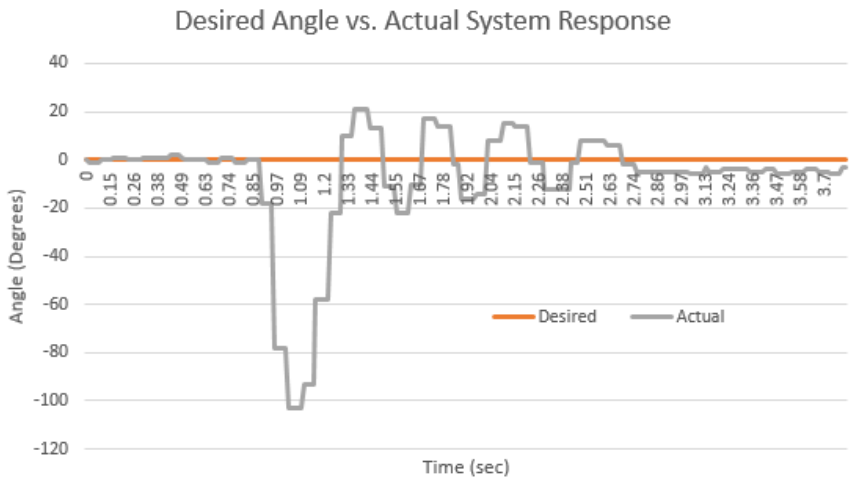

Figure 2: Controller response to disturbance

5. Conclusion and Recommendations

The PID Controller successfully responded to disturbances, correcting its heading to the desired direction. It is apparent that this type of controller can have many uses in robotics. However, it is also evident how it can be limited by the type of applications it can be applied to since it is fundamentally based off of error from a desired output. For future renditions of this lab, it may be helpful for the lab to be closer in time to the classroom discussion of PID control in order to reinforce the discussion on a practical level.

6. Documentation

Dos Santos Gama de Araujo, Francisco CDT H-4 '18. Assistance given to authors, verbal discussion. We were having trouble getting the motor to turn in the direction we wanted it to. Francisco assisted in troubleshooting by observing the motor behavior and mentioning a part of the code that we should double check. We made minor adjustments to variables and gains that didn't seem to work at first, but when we went through a second time in the same areas, the motor started functioning in the way we wanted. Francisco suggested areas that we should look at, but we made the changes in those areas as we saw fit. West Point, NY. 23 Oct. 2017.

7. References

[1] B.E.A. Saleh and M.C. Teich. *Fundamentals of Photonics*. Wiley Series in Pure and Applied Optics. Wiley, 2007.

Appendix A - Code

```
#include <Wire.h>
#include <HMC5883L.h>
#include <SPI.h>
#include <SD.h>
#define CLOCKWISE 1
#define COUNTERCLOCK 0
#define DIRA 2
#define PWMA 3

//Variables
HMC5883L compass;
static float error;
int reading;
static int offset;
int buttonPress=0;
int currentPos;
int relativePos;
int lastTime=0;
int currentTime;
bool dirWay;
int integral=0;
int last;
int P;
int I;
int D;
int drive;

//Tuning Variables
int Kp=10;
float Ki=.3;
int Kd=3;
int intThresh=30;
int scaleFactor=1;

// data logging variables
File logfile;
const int chipSelect=10;
static float start;
float timeNowMil;
float timeNowSec;
float desiredDir=0;

//Define Functions
float getHeading();  //Reads heading and returns in degrees
float rad2deg();     //Change radians into degrees
int whichWay(int);   //Determines current heading relative to a setpoint direction of 0 degrees
void dataLog();      //Records time from start in seconds, desired angle, and current heading

void setup() {
  Wire.begin();
  compass.SetMeasurementMode(Measurement_Continuous);
  pinMode(5,OUTPUT);
  pinMode(DIRA,OUTPUT);
  pinMode(PWMA,OUTPUT);
  digitalWrite(DIRA,LOW);
  digitalWrite(PWMA,LOW);
  Serial.begin(9600);

  //Datalog setup
  while(!Serial){
    ;   //wait for serial port to connect.
  }
  if(!SD.begin(chipSelect)){
    Serial.println("Card failed, or not present");
```

```
65      }
66      Serial.println("Card initialized.");
67      File logfile = SD.open("Logger82.CSV",FILE_WRITE);
68      if(!logfile){
69        Serial.println("Error: Could not create file");
70      }
71      logfile.close();
72
73      //Record desired position
74      while(buttonPress==0){
75        Serial.println("Waiting for the desired direction to be set");
76        buttonPress=digitalRead(5);
77      }
78      reading = getHeading();
79      //Set offset relative to the desired direction at 0
80      if(reading<=180){
81        offset=0-reading;
82      }
83      else{
84        offset=360-reading;
85      }
86      start=millis();       //Starts timer for datalogging purposes
87    }
88
89    void loop() {
90      currentTime=millis();
91      if((currentTime-lastTime)>=14){       //pulls heading data at ~71.4Hz (every 14 ms)
92        error=whichWay(getHeading());       //pulls current heading realtive to desired
          direction at 0
93        dataLog();                          //records data on csv file
94        if(abs(error)<=5){      //within 5 degrees of desired = "achieved direction"
95          error=0;
96        }
97        if(abs(error)<intThresh){
98          integral=integral+error;
99        }
100       else{
101         integral=0;   //reset integral
102       }
103       if(error>0){
104         P=error*Kp+75;    //75 offset to allow for motor to function according to small
            movements
105       }
106       else{
107         P=error*Kp-75;    //75 offset to allow for motor to function according to small
            movements
108       }
109       I=integral*Ki;
110       D=(last-error)*Kd;
111       drive=P+I+D;
112       drive=drive*scaleFactor;
113       if(drive<0){        //sets direction of turn
114         dirWay=CLOCKWISE;
115       }
116       else{
117         dirWay=COUNTERCLOCK;
118       }
119       if(abs(drive)>255){     //sets maximum motor speed
120         drive=255;
121       }
122       spinMotor(dirWay,abs(drive));
123       lastTime=currentTime;   //reset timer for next pull
124     }
125   }
126
127   int whichWay(int heading){
```

```
  //reads current direction and translates it to a degree b/wn -180 and 180
  //relative to a 0 at the fixed direction determined with the button press
  currentPos=heading+offset;
  if(currentPos>180){
    relativePos=currentPos-360;
  }
  else{
    relativePos=currentPos;
  }
  if(relativePos>=0){
  }
  else{
  }
  return relativePos;
}

float getHeading(){
  MagnetometerScaled sensor=compass.ReadScaledAxis();
  float heading=atan2(sensor.YAxis,sensor.XAxis);
  return rad2deg(heading);
}

float rad2deg(float heading){
  if(heading<0) heading+=2*PI;
  if(heading>2*PI)heading-=2*PI;
  return heading*180/PI;
}

void spinMotor(byte dir,byte speeed){
  digitalWrite(DIRA,dir);
  analogWrite(PWMA,speeed);
}

void dataLog(){
  timeNowMil=millis()-start;
  timeNowSec=timeNowMil/1000;

  File logfile=SD.open("Logger82.CSV",FILE_WRITE);
  if(logfile){
    logfile.print(timeNowSec);
    logfile.print(", ");
    logfile.print(desiredDir);
    logfile.print(", ");
    logfile.println(error);
  }
  else{
    Serial.println("Error:Could not open file");
  }
  logfile.close();
}
```