Lab06 PreLab Report

Q1)

Steps per revolution of stepper motor:

Derivation of error propagation:

Q2)

#include <string.h>

#include <Adafruit\_MotorShield.h>

#include <string.h>

#include <Wire.h>

#include "utility/Adafruit\_PWMServoDriver.h"

// define your global variables here

int stepsPerRevolution = 200;

int portMotor1;

int portMotor2;

char direction[20];

// create a new object of Adafruit\_MotorShield, you can call it AFMS

Adafruit\_MotorShield AFMS = Adafruit\_MotorShield();

// create one instance for each motor, call them myMotor1 and myMotor2

Adafruit\_StepperMotor \*myMotor1 = AFMS.getStepper(stepsPerRevolution, portMotor1);

Adafruit\_StepperMotor \*myMotor2 = AFMS.getStepper(stepsPerRevolution, portMotor2);

//

// initialize pinouts for limit switches

int switch1 = 6;

int switch2 = 7;

int switch3 = 4;

int switch4 = 5;

// initialize step size (Number of steps in each iteration)

int stepSize = 20;

// Functions

void setup() {

// Start the serial communication at 115200 baud rate

Serial.begin(115200);

// Set serial communication timeout to 10

Serial.setTimeout(10);

// Start the Adafruit Motor Shield and set the maximum speed of the stepper

AFMS.begin();

// Set the input pins

pinMode(switch1, INPUT);

pinMode(switch2, INPUT);

pinMode(switch3, INPUT);

pinMode(switch4, INPUT);

}

int move\_steps (int steps, int dir, int motor) {

int switch\_check;

// Check the motor and direction of movement, and set the limiting switch accordingly

switch (motor) {

case (motor==1):

switch (dir) {

case (dir==1):

direction = "FORWARD";

switch\_check = switch2;

break;

case (dir==2):

direction = "BACKWARD";

switch\_check = switch1;

break;

}

case (motor==2):

switch (dir) {

case (dir==1):

direction = "FORWARD";

switch\_check = switch4;

break;

case(dir==2):

direction = "BACKWARD";

switch\_check = switch3;

break;

}

break;

}

// Limit the total number of steps to 999

if (steps > 999) steps = 999;

// Create a loop, which is executed if steps > 0 and the limit switch has not been reached,

// in the loop, move the desired motor in small steps (stepsize). Execute the loop until you moved

// the whole distance

while (steps > 0 && digitalRead(switch\_check) == LOW){

step(stepSize, direction, "Single");

steps -= stepSize;

}

// After running the loop, return ASCII for the switches

// If switch 1 is pressed, return '1'

if (digitalRead(switch1) == HIGH) return 49;

// If switch 2 is pressed, return '2'

else if(digitalRead(switch2) == HIGH) return 50;

// If switch 3 is pressed, return '3'

else if(digitalRead(switch3) == HIGH) return 51;

// If switch 4 is pressed, return '4'

else if(digitalRead(switch4) == HIGH) return 52;

// Else, return '0'

else return 48;

}

void loop() {

// Initialize parameters

char command[50];

char check;

byte read\_check;

int flag = 1; //if 1 command is proper, else not

int steps = 0;

int motor = 0;

int direction = 0;

// Check if there is a command on the serial port

if (Serial.available() > 0)

{

read\_check = Serial.readBytes(command,20);

// If the command is not 5 bytes long, discard it

if ((int)read\_check == 5)

{

command[5] = '\0';

check = '0';

}

else

{

command[0] = '\0';

check = '5';

Serial.print(check);

}

}

// Proper command contains 5 bytes:

// First byte is the stage number: 1 or 2

// Second byte is the direction: 1 or 2

// Third - Fifth bytes are number of steps: 000 - 999

if (command[0]!=0)

{

// Check the first byte and if it is not '1' or '2' discard it

// First byte determines the stage (stepper motor) that needs to be moved

if (command[0]!='1' || command[0]!= '2') flag = 0;

// Check the second byte and if it is not '1' or '2' discard it

// Second byte determines the direction

if (command[1]!='1' || command[1]!= '2') flag = 0;

// Check that third to fifth bytes are between '0' and '9'

// make sure to convert from chars to integers (subtract 48, the ASCII constant) and multiply accordingly

for (int i=2; i<5; i++){

if ( (command[i]-48) < 0 || (command[i]-48) > 9 ) flag = 0;

}

motor =command[0]-48;

direction = command[1]-48;

steps = (command[2]-48)\*100+(command[3]-48)\*10+(command[4]-48);

// If everything is fine, move the motors

if (flag){

check = move\_steps (command[0], command[1], steps);

}

// Check is sent over the serial back to microprocessor:

// '0' if motor moved

// '1' if switch 1 is pressed

// '2' if switch 2 is pressed

// '3' if switch 3 is pressed

// '4' if switch 4 is pressed

// '5' if command is bad

Serial.print(check);

// Reset the command

command[0] = '\0';

Serial.flush();

}

// Delay in ms

delay(5);

}