Daniel Komac

CPE 185: Introduction to Microprocessors

Lab Session: Wednesday

**Lab 7: Input and Stepper motor control**

Instructor: Chris Moyer

Date: 12/10/15

**Input to Control the Stepper Motor:**

To control a Robotic arm with user input through an Arduino Mega with two Joystick inputs. The inputs with control the rotation of the arm, the extension of the arm and raising of the arm. The intention was for the arm to be able to move an operating claw and pick objects up if need be, though on our final model the stepper motors we use do not have gear reduction and were not able to support the weight of the Servo Claw. In order to achieve this we would need to spend $50 on new Stepper Motors with gear reduction which was not an option with our time constraint.

First, I had to understand how the Arduino was registering the coordinates of the joystick so I made a small side program which read the Analog input of the X and Y pins and the digital read of the push switch. One problem that I encountered was Baud Rate being too low which affected the display screen. For a while all we could confirm is that it was registering the inputs but we could not display the X and Y coordinates, after a little bit of fine tuning in the serial window we found that the minimum value of X or Y was relatively close to 32 and the maximum was 1023. After calculating this I was able to find the dead zones for the joysticks on the Arduino.

//This is test code

const int VRxPin= 0; //VRx pin connected to arduino pin A0

const int VRyPin= 1; //VRy pin connected to arduino in A1

const int SwButtonPin= 8; //SW pin connected to arduino pin D8

int pressed= -1; //this variable will determine whether joystick has been pressed down (selected)

int x= -1;//this variable will hold the X-coordinate value

int y= -1; //this variable will hold the Y-coordinate value

void readJoystick() {

pressed= digitalRead(SwButtonPin);//reads whether joystick has been pressed down (selected) or not

x= analogRead(VRxPin);//reads the X-coordinate value

y= analogRead(VRyPin);//reads the Y-coordinate value

}

void setup() {

pinMode(SwButtonPin, INPUT);//sets the SW switch as input

digitalWrite(SwButtonPin, HIGH);//sets the SW button HIGH

Serial.begin(115200);//sets the baud rate

}

void loop() {

readJoystick();//calls this function which reads the digital input button SW, the X-coordinate and the Y-coordinate

Serial.print("X: ");

Serial.println(x);//prints the X-coordinate

Serial.print("Y: ");

Serial.println(y);//prints the Y-coordinate

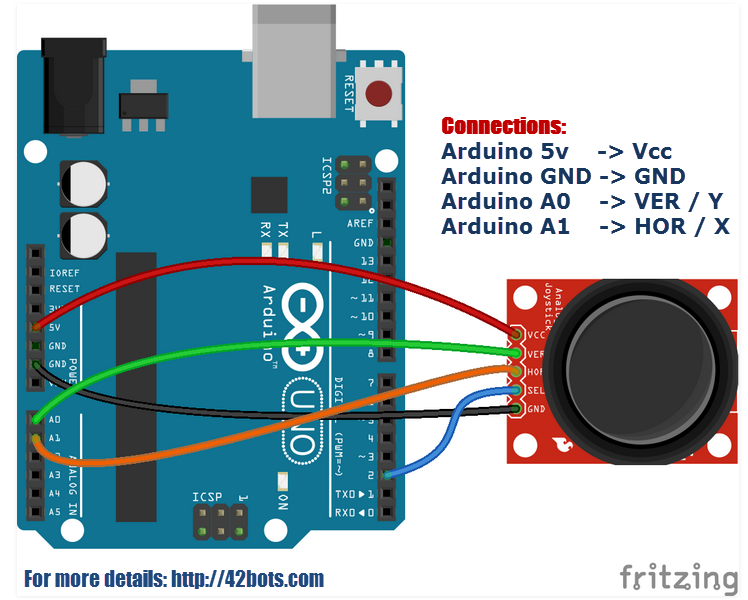
Serial.print (" Pressed: ");

Serial.println(pressed);//prints whether joystick knob has been pressed or not

delay(10);

}

Basic layout of joystick





After the testing we began to implement this code with controlling the stepper motors in incrementing the steps or decrementing it depending on whether we move it towards the relative minimum or maximum. This was simple to do because all it took was altering our dead zone code to implement stepper movements. Since stepper motors are continuous rotation they will continue to turn in the direction that we specify unless limited by limiters or by hardcoding a limit to the steps. For use of the bush buttons it took some time to get them to function how it was intended to. In the analysis of the buttons we saw that there was some floating voltage that was causing them to give a mixture of on/off inputs before going to the state that we needed, this was a simple fix by simply de-bouncing the switches with a 20k pull up resistors and shooting the wire straight from the input of the switch to voltage.

//Section of the code which used the joystick inputs.

if (analogRead(JOY\_3\_PIN) > 1012) { // If joystick is moved Left

if (!digitalRead(Y\_MAX\_PIN)) { // check if limit switch is activated

}

else { // if limit switch is not activated, move motor clockwise

stepper1.move(30);

stepper1.run();

}

}

if (analogRead(JOY\_3\_PIN) < 212) { // If joystick is moved right

stepper1.move(-30);

stepper1.run();

}

}

if (analogRead(JOY\_2\_PIN) > 1012) { // If joystick is moved Left

stepper3.move(30);

stepper3.run();

}

}

if (analogRead(JOY\_2\_PIN) < 212) { // If joystick is moved right

if (!digitalRead(Y\_MIN\_PIN)) { // check if limit switch is activated

// delay(5000);

}

else { // if limit switch is not activated, move motor counter clockwise

stepper3.move(-30);

stepper3.run();

}

}

if (analogRead(JOY\_1\_PIN) > 1012) { // If joystick is moved Left

if (!digitalRead(Y\_MAX\_PIN)) { // check if limit switch is activated

// delay(5000);

}

else { // if limit switch is not activated, move motor clockwise

stepper2.move(30);

stepper2.run();

}

}

if (analogRead(JOY\_1\_PIN) < 212) { // If joystick is moved right

if (!digitalRead(Y\_MIN\_PIN)) { // check if limit switch is activated

// delay(5000);

}

else { // if limit switch is not activated, move motor counter clockwise

stepper2.move(-30);

stepper2.run();

}

}

if (button2State == 1){

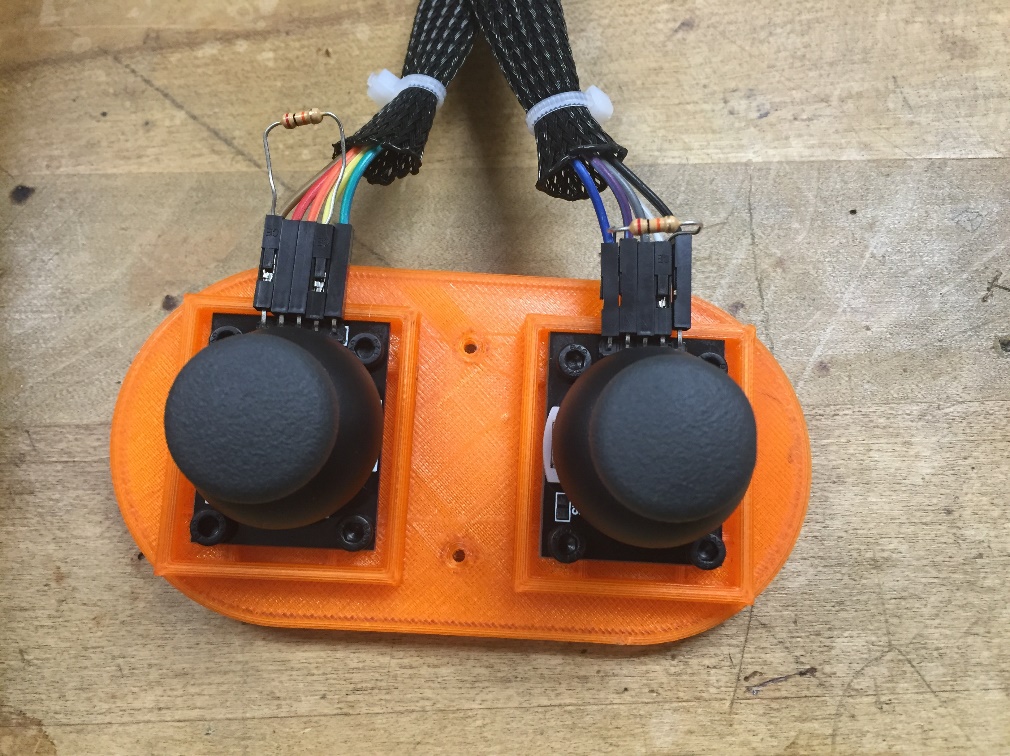
…

}

if (button1State == 1){

…

}



**Code:**

#include <AccelStepper.h>

#include <Servo.h>

#define X\_STEP\_PIN 54

#define X\_DIR\_PIN 55

#define X\_ENABLE\_PIN 38

#define X\_MIN\_PIN 3

#define X\_MAX\_PIN 2

#define Y\_STEP\_PIN 60

#define Y\_DIR\_PIN 61

#define Y\_ENABLE\_PIN 56

#define Y\_MIN\_PIN 14

#define Y\_MAX\_PIN 15

#define Z\_STEP\_PIN 46

#define Z\_DIR\_PIN 48

#define Z\_ENABLE\_PIN 62

#define Z\_MIN\_PIN 18

#define Z\_MAX\_PIN 19

#define JOY\_1\_PIN 13 // ANALOG NUMBERING

#define JOY\_2\_PIN 14 // ANALOG NUMBERING

#define JOY\_3\_PIN 15 // ANALOG NUMBERING

#define JOY\_1\_BTN 11

#define JOY\_2\_BTN 6

#define SERVO\_1\_PIN 4

#define SERVO\_2\_PIN 5

AccelStepper stepper1(AccelStepper::DRIVER, X\_STEP\_PIN, X\_DIR\_PIN);

AccelStepper stepper2(AccelStepper::DRIVER, Y\_STEP\_PIN, Y\_DIR\_PIN);

AccelStepper stepper3(AccelStepper::DRIVER, Z\_STEP\_PIN, Z\_DIR\_PIN);

Servo clawServo;

Servo armServo;

int button1State = -1;

int button2State = -1;

int clawClosed = 0;

int clawRot = 0;

void setup() {

Serial.begin(115200);//sets the baud rate

clawServo.attach(SERVO\_1\_PIN);

armServo.attach(SERVO\_2\_PIN);

pinMode(X\_ENABLE\_PIN, OUTPUT);

pinMode(Y\_ENABLE\_PIN, OUTPUT);

pinMode(Z\_ENABLE\_PIN, OUTPUT);

pinMode(Y\_MIN\_PIN, INPUT);

pinMode(Y\_MAX\_PIN, INPUT);

pinMode(JOY\_1\_BTN, INPUT);

pinMode(JOY\_2\_BTN, INPUT);

digitalWrite(X\_ENABLE\_PIN, LOW);

digitalWrite(Y\_ENABLE\_PIN, LOW);

digitalWrite(Z\_ENABLE\_PIN, LOW);

stepper1.setMaxSpeed(500);

stepper2.setMaxSpeed(500);

stepper3.setMaxSpeed(500);

stepper1.setAcceleration(2000);

stepper2.setAcceleration(2000);

stepper3.setAcceleration(2000);

stepper1.run();

stepper2.run();

stepper3.run();

clawServo.write(90);

armServo.write(90);

}

void loop() {

button1State = digitalRead(JOY\_1\_BTN);

button2State = digitalRead(JOY\_2\_BTN);

Serial.print(button1State);

Serial.println(button2State);

if (analogRead(JOY\_3\_PIN) > 1012) { // If joystick is moved Left

if (!digitalRead(Y\_MAX\_PIN)) { // check if limit switch is activated

}

else { // if limit switch is not activated, move motor clockwise

stepper1.move(30);

stepper1.run();

}

}

if (analogRead(JOY\_3\_PIN) < 212) { // If joystick is moved right

stepper1.move(-30);

stepper1.run();

}

}

if (analogRead(JOY\_2\_PIN) > 1012) { // If joystick is moved Left

stepper3.move(30);

stepper3.run();

}

}

if (analogRead(JOY\_2\_PIN) < 212) { // If joystick is moved right

if (!digitalRead(Y\_MIN\_PIN)) { // check if limit switch is activated

// delay(5000);

}

else { // if limit switch is not activated, move motor counter clockwise

stepper3.move(-30);

stepper3.run();

}

}

if (analogRead(JOY\_1\_PIN) > 1012) { // If joystick is moved Left

if (!digitalRead(Y\_MAX\_PIN)) { // check if limit switch is activated

// delay(5000);

}

else { // if limit switch is not activated, move motor clockwise

stepper2.move(30);

stepper2.run();

}

}

if (analogRead(JOY\_1\_PIN) < 212) { // If joystick is moved right

if (!digitalRead(Y\_MIN\_PIN)) { // check if limit switch is activated

// delay(5000);

}

else { // if limit switch is not activated, move motor counter clockwise

stepper2.move(-30);

stepper2.run();

}

}

if (button2State == 1){

switch (clawRot){

case 0:

clawRot = 1;

armServo.write(90);

delay(300);

break;

case 1:

clawRot = 0;

armServo.write(0);

delay(300);

break;

}

}

if (button1State == 1){

switch (clawClosed){

case 0:

clawClosed = 1;

clawServo.write(90);

delay(300);

break;

case 1:

clawClosed = 0;

clawServo.write(0);

delay(300);

break;

}

}

}

**Conclusion:**

Overall this was a fun experience and the lab taught me a lot about how to manual operate stepper motors. Applying this control can be very useful in more ways than just turning three stepper motors. The lab wasn’t very difficult as it was just reading inputs through analog pins but the difficult part was implementing it with the motors since it took a bit of trial and error to understand how to increment or decrement the motors efficiently. It took roughly a day to figure out input for the joysticks as there was small mechanical errors that ate up a lot of time, mainly coding issues such as lose of code or confusion. Overall the result we got was satisfactory and it was an enjoyable process.