Wesley Myers
18-578
Team F
Individual Lab Report
Week 4

Individual Progress

Since I had the most experience with the Arduino, I took the lead with the programming and assembly aspect. I designed a wiring chart so that when we assembled everything, we knew where all the wires needed to go on assembly. I also encouraged the team to make our system modular, rather than just plugging in single strands of wire. So initially we spent a huge amount of time making wires for our system.

I was able to easily get the servo working rather quickly. I used the Arduino Servo library to operate it. I decided to create a 5V rail on the breadboard from the 5V pin on the Arduino. The spec sheets for the encoder, servo, motor driver, and stepper driver wanted a 5V input voltage; so doing this was ideal. The current draw from all the devices without a load meant that it was possible for the Arduino to supply the 5 volts. Otherwise we had to power it from an external power supply to support the current draw.

Next was the stepper motor, which didn't take too much time to figure out. I spent the bulk of the time on that part figuring out how to wire it properly. I was able to program it rather quickly to get it to move in either direction.

I set up the DC motor so that we can control the direction of the motor and read encoder values. This took a little bit of work since I needed to figure out the pins for the DC Driver. I was pretty sure about the pin out was for the encoder, however I double-checked with a TA which took an extra day. Through experimentation, research, and assistance from my teammates, I was able to get it to work. I passed the PID control to my teammates to finish.

The next day I was told that the PID control wasn't working, so I set out to complete it. I had to restructure the code to take advantage of the loop() function of the Arduino. I properly set up the proportional, derivative, and integral control systems. During testing I noticed that there seemed to be a problem with the encoder. It seemed unclear since the system sometimes performed consistently, while other times erratically. With most control systems, it is really hard to narrow the error down to zero. So I put a plus or minus three ticks buffer. In the grand scheme of things, that is an error of 1.43 degrees. This allowed the system to come to a halt. Another reason I did this is because the system often got close, but it couldn't apply a strong enough PWM signal to make the motor move. When it did reach that threshold due to the integral term, it often jumped pass the zero point,

thus often causing more error rather than less. Thus with this new system, I was able to get the motor to point in the correct direction most of the time.

Another feature I added to the board were LEDs to tell when different power supplies were on. I added one to the 5V rail so that we knew that the Arduino was on. I also added one to the 12V rail, which was supplying power to the drivers. The reason I did this is because often with testing, many people forget to power things on. We'll look at a board and wonder why it isn't doing anything, when in reality the power isn't on. Most of the time, looking at a power supply isn't the first thing people check. Since we're usually staring at the board, the LEDs give us a visual cue for the system. Figure 3 illustrates what the LEDs looked like on the breadboard. Figure 1 also shows where they are in relation to the system.

I also did the debouncing circuit. It was rather simple. I just added a switch with a 15K ohm resistor with a wire going to a digital pin on the Arduino. I did the debouncing on the software side. I made sure that the pin was high for a while before "saying that the switch is close." I essentially had the loop poll every ten milliseconds. If the pin was high for 50 milliseconds (5 cycles), then the switch must be on, thus make the servo move. Figure 2 illustrates the simple circuitry.

During demo day, we decided to investigate even further as to why the motor didn't seem to point consistently. Within 5 minutes before the demo, I got a TA to give us a new motor. It performed consistently, which was great! However, I didn't get a chance to tune the system properly before the demo. Figure 1 illustrates the overall system that we demoed.

As for the mockup, Richard did most of the work. I added what will be my significant portion to this project, a picture of the camera. A picture of the mockup is shown in figure 4.

Challenges

The main challenges I faced were getting used to the Arduino software architecture. Though I have used it before, it has been a while. It is clear that it is set up for a finite state machine, which is definitely an advantage. It was a bit of a challenge to figure out the semantics for some things, such as using analogWrite on a digital pin to control PWM.

One specific challenge we ran into was the use of the quadrature encoder. We hooked up the output lines of the encoder to interrupt pins on the Arduino. The information we receiving didn't make sense. So we tried swapping the motor and got the same result, thus eliminating the encoder having a problem. Next we replaced the cable with an entirely brand new cable. We thought that maybe the wire could have been shorted since it was one of the first attempts at soldering for someone on our team who constructed it. The next option was to look at the code. Sure enough it was the code. We had used the wrong pins. Though the wires are

hooked up to pins 2 and 3, it is interrupt 0 and 1 in the code. Next the logic for reading of the encoder in the software was incorrect. We corrected all of these things and it worked. This took a team effort in debugging and searching the internet for how to properly do it.

Teamwork

I took the role of getting most of the bulk programming out of the way. I am the only person on my team who has experience with the Arduino, so I wrote most of the code and described to them how it worked with the Arduino, etc.

The team members were mostly responsible for the assembly of the system. So this involved assembly of the Motor Driver, Stepper Driver, and wiring for the system. Rick was new to programming and wrote a little code. Soohyun and David took responsibility for writing the User Interface for the Servo, Stepper, and DC Encoder code.

We all cooperated in the testing of the system for demo. In future labs, we hope for more collaboration since we now know how we all work and what our strengths and weaknesses are.

Future Work

Within our design, we plan on using two stepper motors. We are using one for the rack and pinion for the ball shooting and the other for panning the robot. We found a stepper motor online that will give us 0.9 degree per step precision. This lab provided us with the information we need to operate this.

As for the DC motor, we now have a greater understanding on how to operate a DC motor and vary the speed of the shaft. Before this I had no clue how this was done. This will be important for the project.

As for work for next week, I have experience with some of the sensors. We haven't quite discussed as a team how we are splitting up the work for that one. I imagine that I will be doing the infrared red and range finding sensors, given my experience with those.

Figures

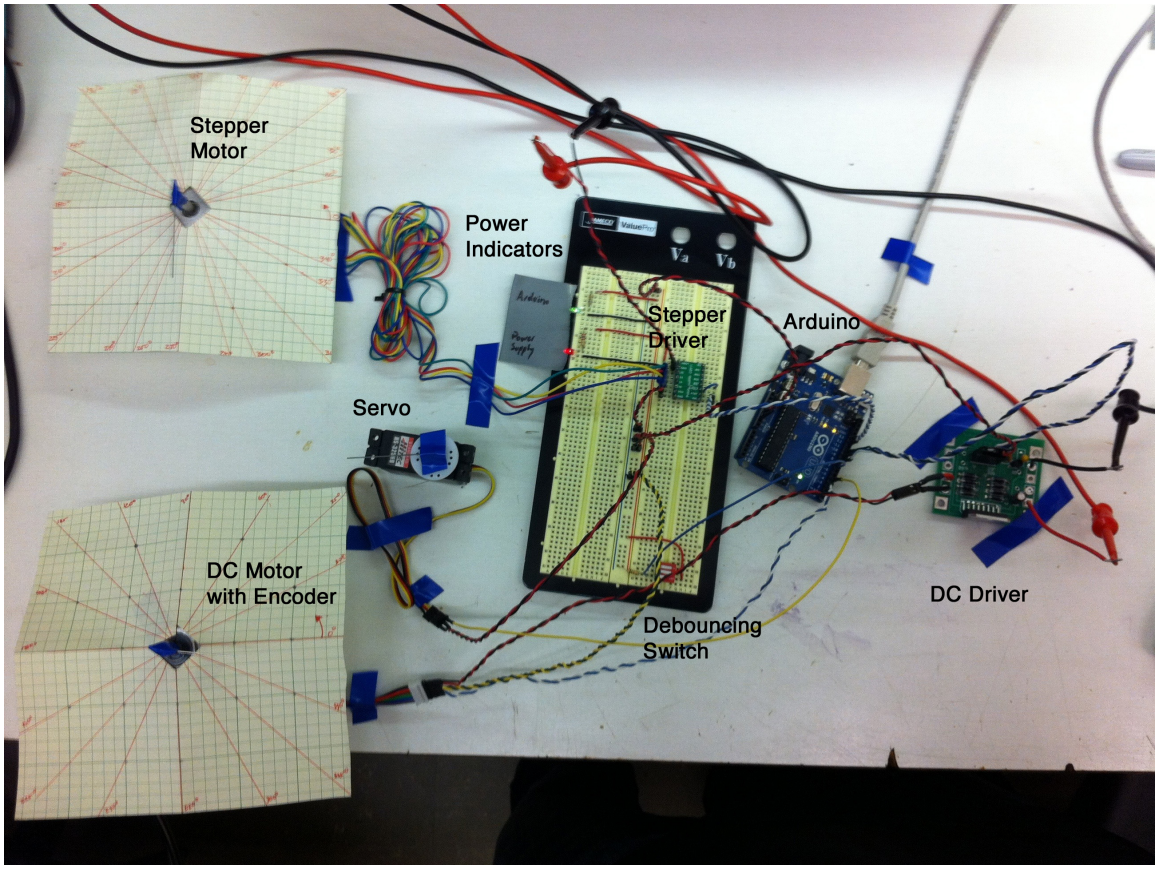

Figure 1. Overall System

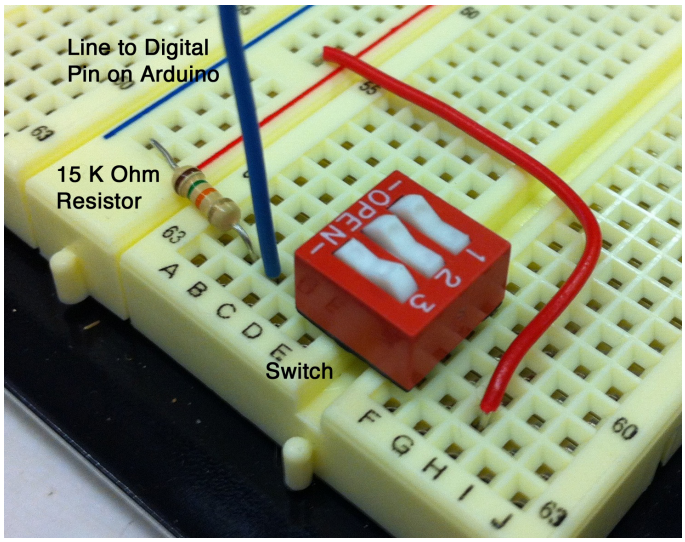

Figure 2. Denouncing Example Circuit

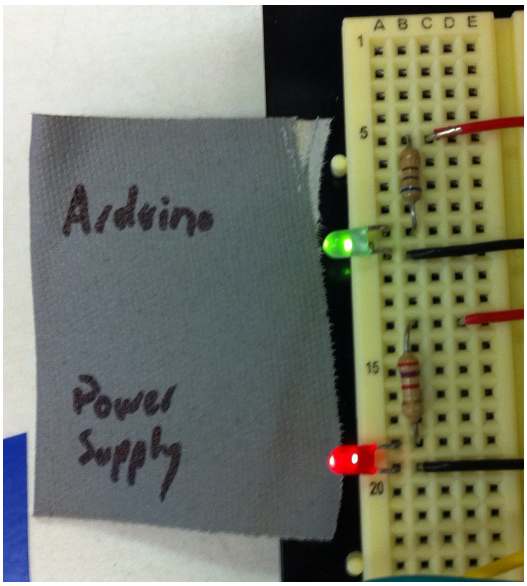

Figure 3. Power Supply Visual Indicators

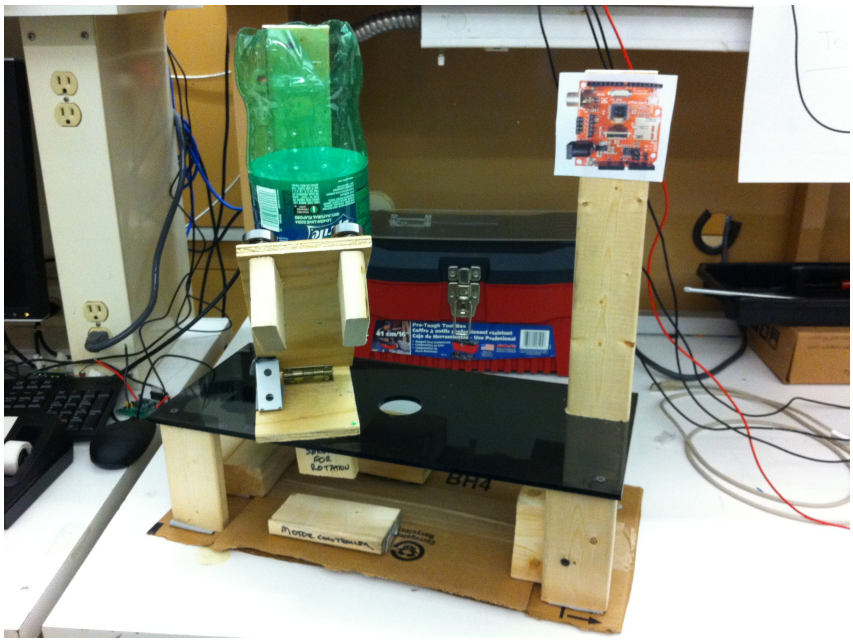

Figure 4. Mockup

Code

The following contains code from this lab. Code highlighted in red signifies that I personally wrote it, though as a team we did consult together on some parts.

PID Control:

```
#include <stdio.h>

int encoderA = 2;
int encoderB = 3;

int driverIn2 = 8;
int driverEn = 9;
int driverIn1 = 10;

volatile int encoderPos = 0;

int error;
int sumError;
int firstEnter;  //for when we first reach error range
int last_error;
int Kp = 2;
int Kd = 2;
int KiNUM = 10;
int KiDEN = 100;
int motorSpeed;
int desiredPos;

int input;

int run;

int ninetyDegrees = 170;

unsigned long prevTime;

unsigned long LOOP_TIME = 10;

void setup()
{
 //configure pins
 pinMode(encoderA, INPUT);
 pinMode(encoderB, INPUT);
 digitalWrite(encoderA, HIGH);
 digitalWrite(encoderB, HIGH);

 pinMode(driverIn2, OUTPUT);
 pinMode(driverEn, OUTPUT);
 pinMode(driverIn1, OUTPUT);

 Serial.begin(9600);

 //get input from user
 input = -1;
 Serial.println("Enter in desired angle (0-9) to go to degree.");
 Serial.println("DC motor will rotate to 90*input degrees.");
 Serial.println("Enter . for CW");
 Serial.println("Enter - for CCW");
```

```
 while(input == -1)
 {
  input = Serial.read();

 }

 input = input - 48;  //ascii offset

 if(input > 0)
  desiredPos = input * ninetyDegrees;

 attachInterrupt(0, doEncoderA, CHANGE);
 attachInterrupt(1, doEncoderB, CHANGE);

 encoderPos = 0;
 firstEnter = 0;
 sumError = 0;

 run = 0;
}

void loop()
{
 if(input > 0)
 {
  if((millis()-prevTime) > LOOP_TIME)
  {
   prevTime = millis();

   int pidTerm = 0;

   error = desiredPos - encoderPos;

   //reached threshold, keep track of integral term
   if(error < (desiredPos*.1) && firstEnter == 0)
    firstEnter = 1;

   //summing integral term
   if(firstEnter == 1)
    sumError += error;

   //within acceptable error range
   if(abs(error) <= 3)
    sumError = 0;

   pidTerm = abs(int((Kp * abs(error)) + (Kd * abs(error - last_error))) + (KiNUM*abs(sumError)/KiDEN));

   if(pidTerm > 150)
    motorSpeed = 150;
   else
    motorSpeed = pidTerm;

   analogWrite(driverEn, motorSpeed);

   last_error = error;

   //debug prints
   Serial.print("Error: ");
   Serial.print(error);
   Serial.print(" ");
   Serial.print("Pid: ");
```

```
    Serial.print(pidTerm);
    Serial.print(" ");
    Serial.print("motorSpeed: ");
    Serial.print(motorSpeed);
    Serial.print(" ");
    Serial.print("desiredPos: ");
    Serial.print(desiredPos);
    Serial.println("");

    if(error > 0 )
    {
     digitalWrite(driverIn1, LOW);
     digitalWrite(driverIn2, HIGH);
    }

    else if (error <= 0)
    {
     digitalWrite(driverIn1, HIGH);
     digitalWrite(driverIn2, LOW);
    }
   }
  }
  else if(input == -2)
  {
   digitalWrite(driverEn, HIGH);
   digitalWrite(driverIn1, LOW);
   digitalWrite(driverIn2, HIGH);
  }
  else if(input == -3)
  {
   digitalWrite(driverEn, HIGH);
   digitalWrite(driverIn1, HIGH);
   digitalWrite(driverIn2, LOW);
  }
}

/*encoder interrupt handlers*/

void doEncoderA()
{
 if(digitalRead(encoderA) == HIGH)
 {
  if(digitalRead(encoderB) == LOW)
  {
   encoderPos++;  //CW
  }
  else
  {
   encoderPos--;  //CCW
  }
 }
 else
 {
  if(digitalRead(encoderB) == HIGH)
  {
   encoderPos++; //CW
  }
  else
  {
   encoderPos--; //CCW
  }
```

```
  }
}

void doEncoderB()
{
 if(digitalRead(encoderB) == HIGH)
 {
  if(digitalRead(encoderA) == HIGH)
  {
   encoderPos++;  //CW
  }
  else
  {
   encoderPos--;  //CCW
  }

 }
 else
 {
  if(digitalRead(encoderA) == LOW)
  {
   encoderPos++;  //CW
  }
  else
  {
   encoderPos--;  //CCW
  }
 }
}
```

Servo Code:

```
#include <Servo.h>

Servo s;

int servo_pin = 4;
int pos = -1;

void setup()
{
  s.attach(servo_pin);

  Serial.begin(9600);
  pos = -1;
}

void loop()
{
  Serial.println("Enter in position (0-9) to go to in increments of 20
degrees");
  Serial.println("Enter . for demo");
  while(pos == -1)
  {
    pos = Serial.read();
  }

  pos -= 48;  //subtract ascii
  pos *= 20;  //scale

  if(pos >= 0)
  {
    s.write(pos);
    delay(1000);
    pos = -1;
  }
  else
  {
    while(1)
    {
      s.write(0);
      delay(1000);

      s.write(90);
      delay(1000);

      s.write(180);
      delay(1000);
    }
  }

}
```

Stepper Code:

```
int dir_pin = 13;
int step_pin = 12;

int counter;
int i;

int pos;
int dir;

void setup()
{
  pinMode(dir_pin, OUTPUT);
  pinMode(step_pin, OUTPUT);

  counter = 1;

  Serial.begin(9600);
  pos = -1;
  Serial.println("Enter in position (1-9) to go to in increment by 90 degrees");
  Serial.println("Enter in 0 for other options");
  while(pos == -1)
  //while(Serial.available() == 0)
  {
    pos = Serial.read();
  }

  pos -= 48;  //subtract ascii
  pos *= 50;  //scale

  Serial.println(pos);
}

void loop()
{
  if(pos == 0)
  {
    getDecision();

    if(dir == 3)
    {
      funDemo();
    }
    else if(dir == 2)
    {
      digitalWrite(dir_pin, HIGH);  //CCW

      while(1)
      {
        digitalWrite(step_pin, LOW);
        delay(5);
        digitalWrite(step_pin, HIGH);
        delay(5);
      }
    }
    else if(dir == 1)
    {
      digitalWrite(dir_pin, LOW);  //CW

      while(1)
      {
        digitalWrite(step_pin, LOW);
        delay(5);
        digitalWrite(step_pin, HIGH);
        delay(5);
      }
    }
  }
  else
  {
    counter = 0;
```

```
    while(counter < pos)
    {
      digitalWrite(step_pin, LOW);
      delay(5);
      digitalWrite(step_pin, HIGH);
      delay(5);

      counter++;
    }

    while(1)
    {
      digitalWrite(step_pin, LOW);
    }
  }
}

void getDecision()
{
  dir = -1;
  Serial.println("Enter 1 for CW");
  Serial.println("Enter 2 for CCW");
  Serial.println("Enter 3 for demo");

  while(dir == -1)
  {
    dir = Serial.read();
  }

  dir -= 48;  //subtract ascii

  Serial.println(dir);
}

void funDemo()
{
  while(1)
  {
    if(counter == 1)
    {
      digitalWrite(dir_pin, LOW);
    }
    else
    {
      digitalWrite(dir_pin, HIGH);
    }

    for(i = 0; i < 200; i++)
    {
      digitalWrite(step_pin, LOW);
      delay(5);
      digitalWrite(step_pin, HIGH);
      delay(5);
    }

    counter *= -1;
  }
}
```

Switch Code:

```
#include <Servo.h>

Servo s;

int servo_pin = 4;
int button_pin = 7;
int pos = -1;

int sum;

void setup()
{
  s.attach(servo_pin);

  Serial.begin(9600);
  pos = -1;
}

void loop()
{
  if(digitalRead(button_pin) == LOW)
    sum = 0;

  if(digitalRead(button_pin) == HIGH)
    sum++;

  if(sum > 10)
  {
      s.write(0);
      delay(1000);

      s.write(90);
      delay(1000);

      s.write(180);
      delay(1000);

  }

  delay(10);

}
```