INDIVIDUAL LAB REPORT 1

Awadhut Thube
Team G - The Pit Crew
Team Members - Alex Withers, Justin Morris
ILR01
12th February 2020

Table of Contents

Individual Progress

Sensors and Motor Control Lab

For the sensors and motor control lab, my two main contributions are as follows:

1. **Interfacing a force sensor with the servo motor**
   I set up a voltage divider with the force sensor connected in one branch and a fixed resistor in the other. The choice of the fixed resistor was made after measuring the change in resistance of the force sensor upon application of external force. The servo was controlled to move through a fixed angle upon application of external force on the sensor. I also included a feature which allows us to control the movement angle from the GUI.
2. **Setting up the DC Motor for control through the GUI.**
   I was also responsible for the hardware and software for the DC Motor control. I have included functions to control the velocity and direction of the motor from the GUI. The program also includes a function which moves the motor through a specified number of degrees. Apart from the GUI, the motor can also be stopped or started through a tactile switch.

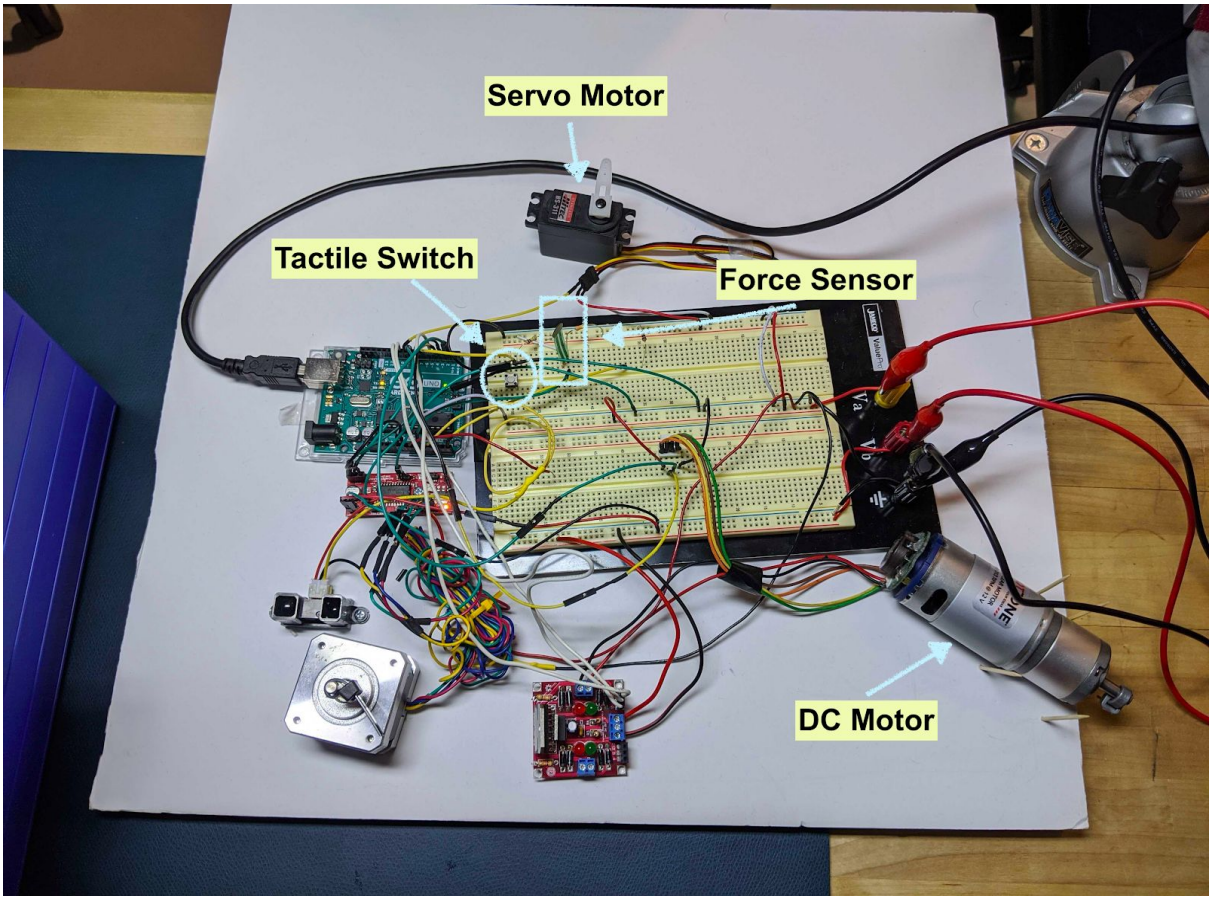


Fig 1. Elements I worked on

Sensor

The resistance between the terminals of the force sensor changes upon application of external force. I used a voltage divider to convert this change in resistance to change in voltage. This voltage output from the divider circuit is read as an analog input on the microcontroller which generates a quantized value in the range 0 to 1023. Then, a simple threshold is applied to this quantized value to achieve a binary output representing whether a force has been applied to the sensor. The way the voltage divider is set up, the motor is commanded to move when the quantized value is less than the threshold, i.e. when the binary value is 0.

To prevent multiple triggers (multiple movements of the servo motor), I use a flag to keep track of whether the sensor was released or not. Hence, even if the sensor is pressed continuously, the servo motor would move only once.

Servo Motor

The servo motor was controlled using the Servo Library of Arduino. The microcontroller is programmed to move the servo motor by a fixed angle. I have also added the functionality to change this fixed angle from the GUI. The servo motor is powered directly from a DC Power Source and only receives pwm signal from the microcontroller. The program is set up so the servo motor moves back to zero after exceeding the upper bound on its range which is 180 degrees.

DC Motor

The DC Motor was controlled through the GUI and could be started or stopped using a tactile switch. While writing the code for the DC Motor, the first thing that was implemented was a simple direction and speed control. An L298 motor driver was used for providing the required power to the motor. The DC motor was also equipped with a hall sensor encoder having a resolution of 322 pulses per rotation. The encoder count is maintained by the use of an interrupt. I have also implemented code to move the DC Motor by a specified angle in either direction. The angle and the direction was sent from the GUI. Additionally, the speed of the motor can be controlled as well. The speed control was achieved simply by sending a pwm signal from the GUI to the controller. Apart from this, the velocity of the motor is also calculated. This is done by timing the loop inside the microcontroller and then counting the number of encoder ticks in that particular time.

Pit Navigator Project

For the MRSD project, I have worked on the perception stack that will be used in our system. Initially, I acquired the Realsense D435i along with a Gigabyte Compact PC from the MRSD inventory. I set up the realsense drivers for use with python and the realsense-viewer. This allowed us to receive images from the camera. Moving ahead, I set up the realsense sdk so that the camera can be interfaced with the ROS framework. I wrote some basic code to capture images from the camera at regular intervals.
After setting up the camera, I wrote code to capture images at multiple camera exposure.

Apart from the camera operation, I also worked towards the pit detection problem using the dataset that our team received from the sponsor. The dataset includes images of a pit present at UTAH. I separated the images in the dataset based on their quality. This includes segregating the images based on angle of the camera, good exposure, good viewpoint, etc. I also tried some gradient based techniques to localize strong edges within the images. The output is as shown below. Based on the above tests, I arrived at the conclusion that manual determination of features and heuristics to detect the edge of a pit is a very hard problem and we will have to look at other methods for reliable detection.

Challenges

Sensors and Motor Control Lab

Initially for the lab, we considered building a pan tilt mount which would be controlled by one of our motors. However, after completing the hardware and software implementations for each of the motors individually, we ended up spending a substantial amount of time on the final integration. Hence, we had to drop the idea of developing a mount during the course of the lab. Control of the DC Motor was a major challenge. Getting the correct encoder readings via the interrupt and controlling the position and velocity of the motor required a large chunk of the total time I spent working on the lab. Justin helped me in writing the logic for the DC motor position control. Management of cables was a challenge because of the use of multiple components on one single breadboard. We tried to space things out as much as possible so that it would be easier for us to debug any hardware issues.

Pit Navigator Project

For the MRSD project, the biggest challenge was to be updated with all the work that has gone into the project before we started working on it. It was difficult to scope a portion of the project that would be valuable to our sponsor and also fit under the umbrella of an MRSD project.

My main focus in the project has been on the perception capabilities of the robot. Detection of pits from the available dataset was a very challenging problem. I am still working on the same but now I am looking into different methods to identify the edge of the pits. This has caused all of my prior work to be sidelined and I have to figure out and focus on different options which could be employed for pit detection.

Teamwork

Sensors and Motor Control Lab

Member	Motor	Sensor	Description of work
Alex Withers	Stepper Motor	Sharp IR Sensor	1. Converted physical quantity (distance), to its equivalent analog voltage. 2. Implemented an average filter for the sensor. 3. Controlled the direction and position of a stepper motor based on the sensor reading
Awadhut Thube	DC Motor, Servo Motor	Force Sensor	1. Interfaced the force sensor with a servo motor. 2. Approximated the physical force based on sensor reading. 3. Set up the DC motor for control through the GUI and a tactile switch.
Justin Morris	Graphical User Interface		1. Implemented a GUI using processing. 2. Set up a serial communication pipeline between the GUI and arduino.

Pit Navigator Project

Member	Description of work
Alex Withers	1. Alex has done an intensive study on the required camera specifications 2. He is also working closely with the previous MRSD Team to set up their simulation on our system.
Awadhut Thube	1. I have set up the Intel Realsense and integrated with ROS. 2. I have also written the baseline code for capturing stereo images and controlling the camera parameters.
Justin Morris	1. Justin has been responsible for acquiring previously written code for moonranger. 2. He also worked with one of the available skid steer robots (Blue) and tried setting it up for our project.

Plans

Pit Navigator Project

For the MRSD project, we plan to continue the work as it is previously distributed. We look forward to making progress on the overall brinkmanship aspect of our project. For the short term, it means that we will work to put together the individual components like camera operations, pit detection, navigation and planning. Personally, I will be working on stereo reconstruction for pit detection. We have also identified a pan-tilt mount for our system. We will be using that to demonstrate our data capture functionality from multiple viewpoints.

Servo Motor Code

```
#include <Servo.h>

Servo myservo;

int pos = 0;
int trigger = 0;
int moved_flag = 0;

void setup() {
  Serial.begin(9600);
  trigger = analogRead(A0);
  myservo.attach(9);
  myservo.write(10);
}

void move_servo(){
  if ((pos + 45) <= 180){
    pos = pos + 45;
    myservo.write(pos);
  }
  else{
    myservo.write(10);
    pos = 0;
  }
  moved_flag = 1;
  }

void loop() {
  trigger = analogRead(A0);
  if (trigger < 500)
  {
    if (moved_flag == 0)
    move_servo();
    }
  else{
    moved_flag = 0;
  }
  Serial.println(trigger);
}
```

DC Motor Code

```
const int m1 = 10;
const int m2 = 11;
const int enable = 5;
const int enc1 = 2;
const int enc2 = 3;
int pwm_val = 127;
unsigned long prev_debounce_time = 0;
unsigned long DebounceTime = 100;
int state_1 = 0;
int button = 4;
volatile long int ticks = 0;
String incoming_data = "";
long int deg = 0;
unsigned long prev_velocity_time = 0;
unsigned long prev_loop_time = 0;
long int prev_error = 0;
volatile long prev_ticks = 0;
int motor_dir = 1;
bool m1_dir = HIGH;
bool m2_dir = LOW;


void start(){
  digitalWrite(m1, m1_dir);
  digitalWrite(m2, m2_dir);
}
void change_direction(){
  if (motor_dir){
    m1_dir = HIGH;
    m2_dir = LOW;
  }
  else{
    m1_dir = LOW;
    m2_dir = HIGH;
  }
  start();
}
void count_ticks(){
  if (digitalRead(enc2) == 1)
    ticks++;
  else
    ticks--;
}

void setup() {
  // put your setup code here, to run once:
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
```

```
  pinMode(enable, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(enc1), count_ticks, RISING);
  digitalWrite(m1, HIGH);
  digitalWrite(m2, LOW);
  Serial.begin(9600);
}

void brake(){
  digitalWrite(m1, HIGH);
  digitalWrite(m2, HIGH);
  analogWrite(enable, 0);
}

void move_angle_positive(long int deg, long int curr_pos){
  long int req_count = (long int) (322*deg/360);
  start();
  while (abs(ticks) < (req_count + abs(curr_pos))){

  long int error = (req_count + abs(curr_pos)) - abs(ticks) ;
  Serial.print("p");
  Serial.print(error);
  Serial.println(";");
  float d_error = (prev_error - error)/(millis()-prev_loop_time+1);
  prev_loop_time = millis();
  int pwm = map(error, 1, req_count, 55,  100);

  analogWrite(enable, pwm - d_error*5);
  prev_error = error;
  }
  brake();
}

void move_angle_negative(long int deg, long int curr_pos){
  long int req_count = (long int) (-322.0*deg/360);
  start();
  while (abs(ticks) > (req_count + abs(curr_pos))){

  long int error = -(req_count + abs(curr_pos)) + abs(ticks) ;
  Serial.print("p");
  Serial.print(error);
  Serial.println(";");
  float d_error = (prev_error - error)/(millis()-prev_loop_time+1);
  prev_loop_time = millis();
  int pwm = map(error, 1, -req_count, 55,  100);
  analogWrite(enable, pwm);
  prev_error = error;
  }
  brake();
```

```
}


void switch_state_1(){
//  Serial.println("okay");
  if ((millis() - prev_debounce_time) > DebounceTime){
    if (state_1 == 1){
      state_1 = 0;
      start();
      analogWrite(enable, 0);
    }
    else{
      state_1 = 1;
      start();
      analogWrite(enable, pwm_val);
    }
  }
  prev_debounce_time = millis();
}

void readSer() {
  incoming_data = "";
  if (Serial.available() > 0){
    incoming_data = Serial.readStringUntil(";");
//    Serial.println(incoming_data);
  }
}


void loop() {
  readSer();
  float velocity =
float(ticks-prev_ticks)*(60*1000.0/322.0)/(millis()-prev_velocity_time+0.1);
  prev_velocity_time = millis();
  prev_ticks = ticks;
  if(incoming_data.substring(0,1).equals("p")){
    deg = incoming_data.substring(1).toInt();
    if (motor_dir)
    move_angle_positive(deg, ticks);
    else
    move_angle_negative(deg,ticks);
  }
  else if(incoming_data.substring(0,1).equals("v")){
    if (pwm_val == 0)
    pwm_val = incoming_data.substring(1).toInt();
    else{
    pwm_val = incoming_data.substring(1).toInt();
    analogWrite(enable, pwm_val);
```

```
    }
  }
  else if(incoming_data.substring(0,1).equals("d")){
    motor_dir = incoming_data.substring(1).toInt();
    change_direction();
  }
  int reading = digitalRead(button);
  if (!reading){
  switch_state_1();
  }


  Serial.print("v");
  Serial.print(int(velocity));
  Serial.println(";");
  delay(100);
}
```

Task 4 (Sensors and Motor Control Lab) Quiz

1. Reading a datasheet. Refer to the ADXL335 accelerometer datasheet (https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf) to answer the below questions.
   - What is the sensor's range?
     Answer – Min +/- 3g, Typ +/- 3.6g

   - What is the sensor's dynamic range?
     Answer – Min 6g, Type 7.2g

   - What is the purpose of the capacitor C_{DC} on the LHS of the functional block diagram on p. 1? How does it achieve this?
     Answer – The capacitor acts as a high pass filter to remove high frequency noise that can appear at the source. The capacitor is known as decoupling capacitor and it bypasses high frequency signal and protects the rest of the circuit.

   - Write an equation for the sensor's transfer function.
     Answer - Sensitivity = 300mV/g
     0g voltage at output = 1.5
     Transfer function = 1.5 + 0.3*acceleration

   - What is the largest expected nonlinearity error in g?
     Answer - The largest expected nonlinearity error is +/- 0.3% of 7.2g = +/- 0.0216g

   - How much noise do you expect in the X- and Y-axis sensor signals when the sensor is excited at 25 Hz?
     Answer - Noise = $150*10^{-6}*\text{sqrt}(25) = 750*10^{-6}$ g

   - How about at 0 Hz? If you can't get this from the datasheet, how would you determine it experimentally?
     Answer - Noise rms = Noise Density x sqrt(BW * 1.6)
     For x and y axis, the rms noise = $150*\text{sqrt}(1600*1.6) = 7590*10^{-6}$ g
     For z axis, the rms noise = $300*\text{sqrt}(550*1.6) = 8900*10^{-6}$ g

2. Signal conditioning
   - Filtering
     - Name at least two problems you might have in using a moving average filter.
       Answer –
       1. **A moving average filter will cause a delay in computation. (Especially when dealing with floats)**
       2. **Also, in the presence of large noise or a data point that is noisy, the average value can be different than the true average.**

     - Name at least two problems you might have in using a median filter.
       Answer –
       1. **It is computationally expensive to calculate the median (sorting is required in most cases)**
       2. **In certain cases, the median may not be representative of the true value or average value**
   - Opamps
     - In the following questions, you want to calibrate a linear sensor using the circuit in Fig. 1 so that its output range is 0 to 5V. Identify in each case:
       1) which of V1 and V2 will be the input voltage and which the reference voltage;
       2) the values of Rf/Ri and the reference voltage. If the calibration can't be done with this circuit, explain why.
       - Your uncalibrated sensor has a range of -1.5 to 1.0V (-1.5V should give a 0V output and 1.0V should give a 5V output).
         Answer –

       - Your uncalibrated sensor has a range of -2.5 to 2.5V (-2.5V should give a 0V output and 2.5V should give a 5V output).

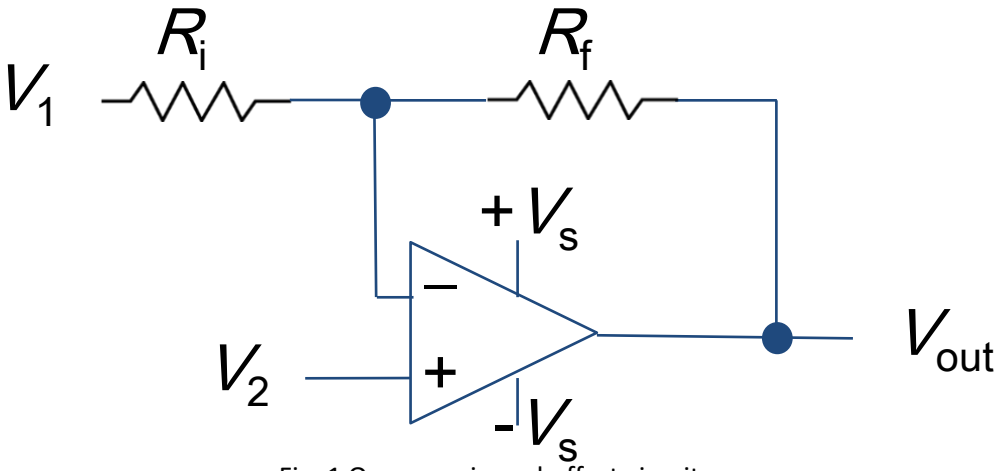
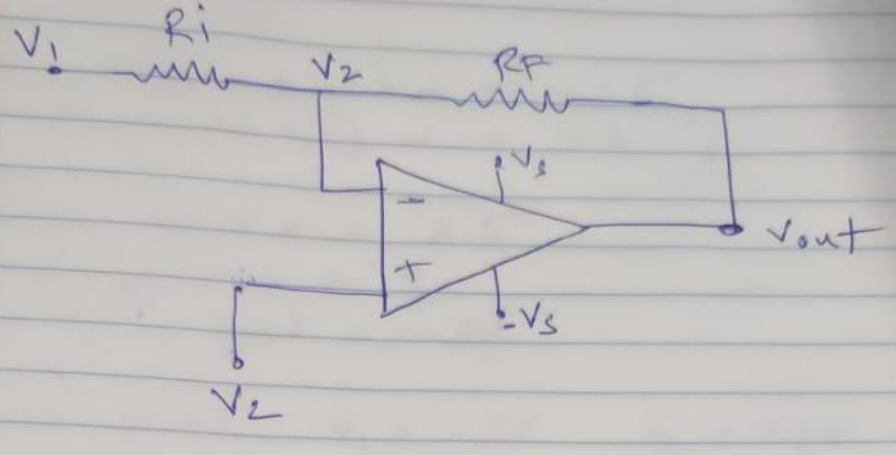


Fig. 1 Opamp gain and offset circuit



$$\frac{(V_1 - V_2)}{R_i} = -\frac{(V_{out} - V_2)}{R_F}$$

$$(V_1 - V_2)\frac{R_F}{R_i} = -(V_{out} - V_2)$$

for $\frac{R_F}{R_i} = \alpha$ and $V_1 = V_{ref}$

$$(V_{ref} - V_2)\alpha = V_2 - V_{out}$$

for $V_2 = -1.5\,V$ $V_{out} = 0\,V$

for $V_2 = 1\,V$ $V_{out} = 5\,V$

①

$$(V_{ref} + 1.5)\alpha = -1.5$$
$$-\{(V_{ref} - 1)\alpha = -4\}$$

$$2.5\alpha = 2.5$$
$$\alpha = 1$$
$$V_{ref} = -3V$$

$\therefore \alpha = 1 \quad V_1 = -3V \quad V_2 = \text{input}$

②

$$(V_{ref} + 2.5)\alpha = -2.5$$
$$(V_{ref} - 2.5)\alpha = -2.5$$

No solution for above equations.

Considering $V_2 = ref$ $\frac{R_F}{R_i} = \alpha$ V_1 as input

$$(V_1 - V_{ref})\alpha = V_{ref} - V_{out}$$

$$\therefore (-2.5 - V_{ref})\alpha = V_{ref}$$
$$(2.5 - V_{ref})\alpha = V_{ref} - 5$$

$\therefore$ No solution for the above equations

3. Control
    - If you want to control a DC motor to go to a desired position, describe how to form a digital input for each of the PID (Proportional, Integral, Derivative) terms.
      Answer – For the proportional term, we can compute the error between the desired angle and the actual angle and change our control in proportion to the error. For the derivative term, we compute the change in error over time. This is then used to counter the proportional term. For integral term we accumulate the error over time. If the accumulated error continues to grow because of steady state error, we increase our control input.

    - If the system you want to control is sluggish, which PID term(s) will you use and why?
      Answer - Increase the proportional gain so that it reaches the desired state faster.

    - After applying the control in the previous question, if the system still has significant steady-state error, which PID term(s) will you use and why?
      Answer – We will use the integral term to reduce the steady state error. By the final value theorem, the steady state error upon introduction of integral term goes to zero.

    - After applying the control in the previous question, if the system still has overshoot, which PID term(s) will you apply and why?
      Answer – To reduce the overshoot, we use the derivative term. It damps the response and reduces settling time.