Sensor and Motors Lab

Individual Lab Report-1

By Danendra Singh

Team F: Falcon Eye

Danendra Singh

Yuchi Wang

Pulkit Goyal

Pratibha Tripathi

Rahul Ramakrishnan

13th October 13, 2017

Individual Progress:

As my contribution to the Sensors and Motors lab, I programmed the slot sensor and the stepper motor. Also, I contributed with Pratibha for overall electrical layout and hardware integration.

Omron Slot Sensor:

EE-SX1042

The slot sensor used was Omron EE-SX1042 transmissive photomicrosensor. It is a PCB mounting type sensor, 14.5 mm tall with a deep slot. It has a high resolution with 0.5 mm wide aperture. The sensor is shown in Figure 1.

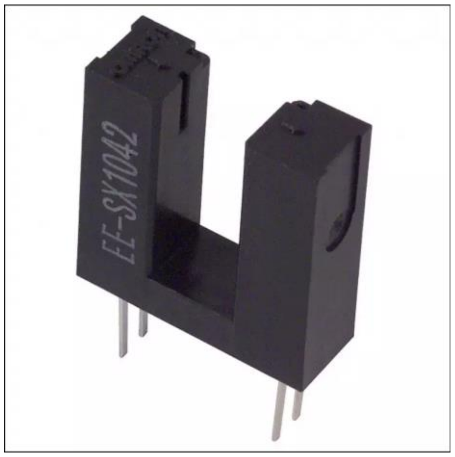


Figure 1: Omron EE-SX1042 Slot Sensor
www.omron.com/ecb/products/pdf/en-ee_sx1042.pdf

The Sensor is basically consisting of two parts, a LED transmitter and a Bipolar Junction Transistor (BJT) , one in each leg of the sensor. The LED transmitter emits light on being subjected to forward voltage. This light transmits through the slot of the sensor and reaches the other leg containing a BJT in common base configuration. This causes the base-emitter junction become forward biased. Hence now current can flow between the collector-emitter junction which was previously blocked by the reverse biased base-emitter junction. Hence, here the BJT is working as a switch. The internal circuit of the slot sensor is shown in Figure 2.

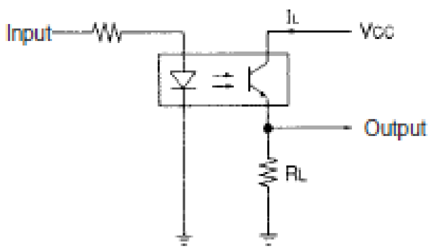


Figure 2: Internal Circuit of slot sensor

The Slot sensor is an analog sensor and returns a value between 0-1023 depending upon the amount of obstruction of the slot by an object. However, I wanted to use it as a digital sensor which actuates a Stepper motor when the slot is blocked. Hence, I connected the slot sensor to the digital pin ____ of the Arduino to receive TRUE or FALSE value from the sensor. The operation of my senor-motor module is as follows:

```
if (Slot_sensor==1)
        Actuate Stepper Motor;
else
        Keep Stepper Motor off;
```

To limit the current flowing through the LED leg of the slot sensor I used a 200 ohm resistor in series with the supply voltage to the LED. Also, to receive the output from the BJT I used a pull down resistor of 10 kohm, across which I took input voltage to the Arduino. This is clear by the circuit diagram shown in Figure 3.

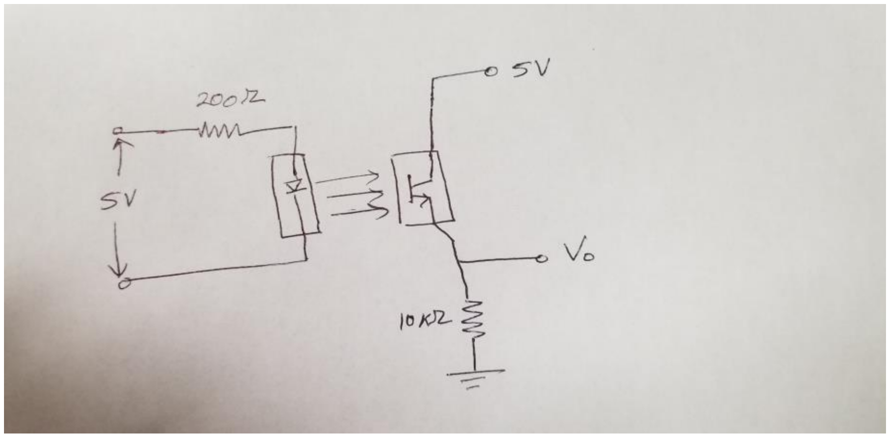


Figure 3: Circuit diagram for Slot Sensor

Stepper Motor Control

The stepper motor we used for the laboratory was Mercury Motor SM-42BYG011-25 along with Pololu A4988 Stepper motor driver as shown in Figure 4 and Figure 5 respectively. The stepper motor had each step corresponding to 1.8^o and could have been moved in half step as well (0.9^o) but since we didn't need such precision, it was used at full step intervals. The degrees moved by stepper motor was accurately mapped to angle between 0-360^o. So for rotating a full circle of 360^o , we needed 200 steps (360/1.8) to be moved by the stepper motor.

Figure 4: Stepper Motor SM-42BYG011-25
https://www.sparkfun.com/products/9238

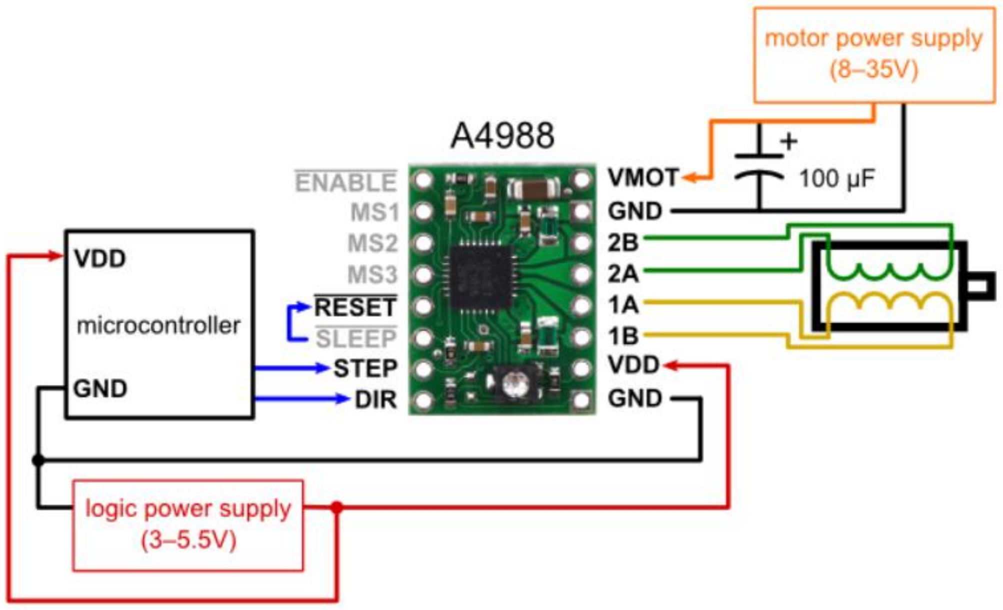


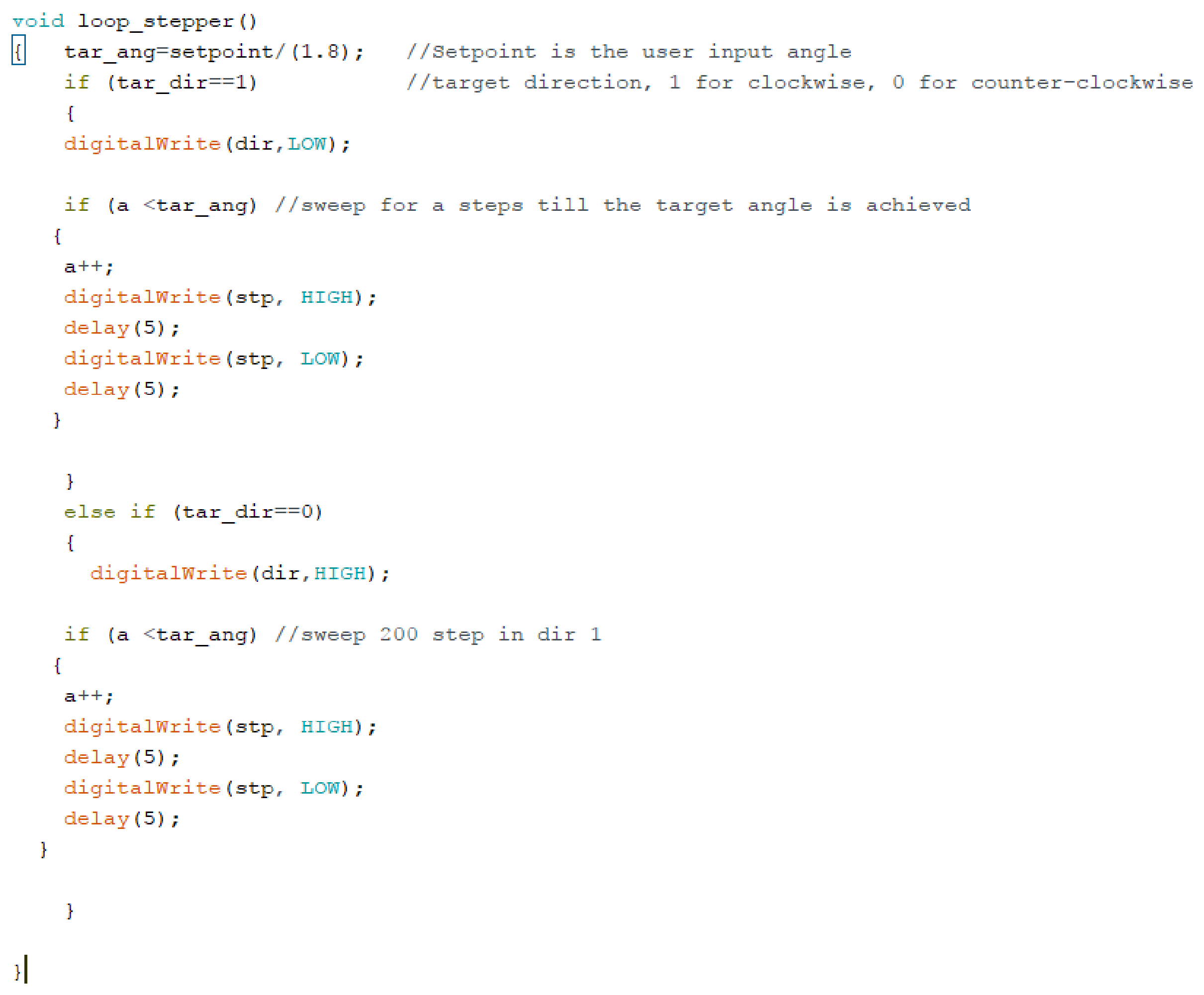
Figure 5: Pololu Stepper Motor Driver with its connections to a microcontroller
https://www.pololu.com/product/1182

The movement of the stepper was controlled from the GUI having an Arduino interface. The user inputs the number of degrees to be moved by the stepper motor which was then converted to the number of steps moved by the stepper and hence used to switch on and off a voltage step at fixed intervals to rotate the stepper. The number of steps are counted inside a loop that continuously switches the stepper on and off to generate voltage pulses.

The direction of the stepper was controlled by a DIR pin which determines the clockwise or anti-clockwise direction. This acts as an output pin from the Arduino and writing logical HIGH makes the motor turn clockwise, while writing LOW makes it move counter-clockwise.

Arduino Code:

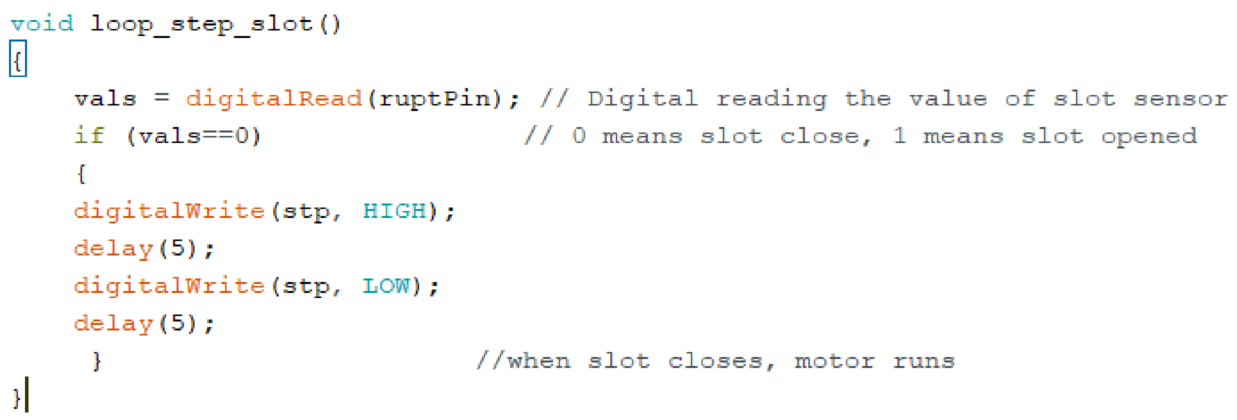
Figure 6 shows the code to control the stepper motor direction and angle from the user input value via GUI.



```
void loop_stepper()
{   tar_ang=setpoint/(1.8);    //Setpoint is the user input angle
    if (tar_dir==1)            //target direction, 1 for clockwise, 0 for counter-clockwise
    {
    digitalWrite(dir,LOW);

    if (a <tar_ang) //sweep for a steps till the target angle is achieved
   {
    a++;
    digitalWrite(stp, HIGH);
    delay(5);
    digitalWrite(stp, LOW);
    delay(5);
   }

    }
    else if (tar_dir==0)
    {
      digitalWrite(dir,HIGH);

    if (a <tar_ang) //sweep 200 step in dir 1
   {
    a++;
    digitalWrite(stp, HIGH);
    delay(5);
    digitalWrite(stp, LOW);
    delay(5);
  }

    }

}
```

Figure 6: Code to control stepper motor angle with user input

Figure 7 shows the code to control the stepper motor with the slot sensor.



```
void loop_step_slot()
{
    vals = digitalRead(ruptPin); // Digital reading the value of slot sensor
    if (vals==0)                  // 0 means slot close, 1 means slot opened
    {
    digitalWrite(stp, HIGH);
    delay(5);
    digitalWrite(stp, LOW);
    delay(5);
     }                         //when slot closes, motor runs
}
```

Figure 7: Code to control the stepper motor with the slot sensor

Hardware Integration:

Hardware Integration was carried out by me and Pratibha. I soldered the Pot pins so that they can be reliably used with the breadboard. Taking inputs form Rahul, Yuchi and Pratibha, I wired all the connections of the Arduino, while at the same time placing them at appropriate locations to avoid electromagnetic interference from the motors and reducing the overall lengths of the wires to and from the microcontroller. To place the motors on their assigned places, we drilled holes on a wooden board frame and secured the motors with wire ties. For sensors, we used Velcro tapes to attach them on the wooden base. The connection of the slot sensor and Pololu driver is shown in Figure 8.

The overall systems is shown in Figure 9.

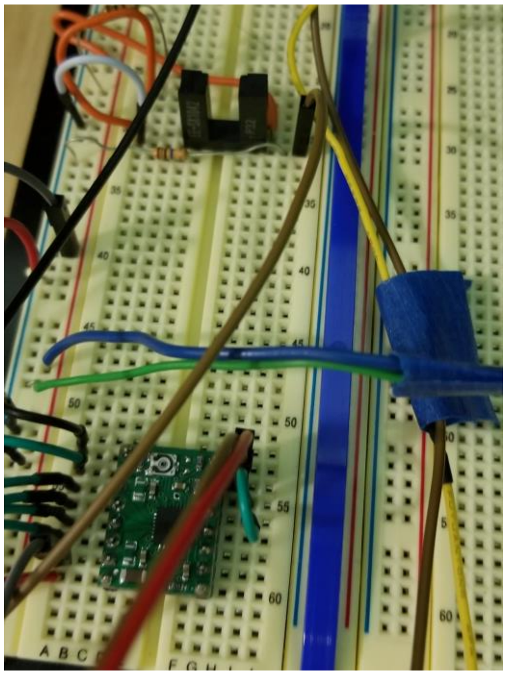


Figure 8: Pololu driver and Slot Sensor

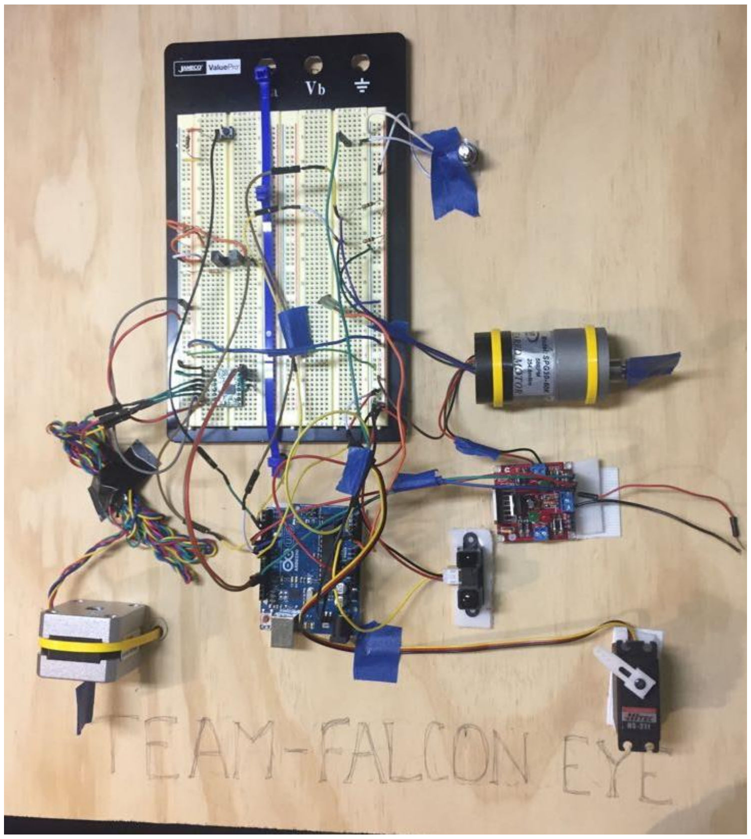


Figure 9: Overall System

<u>Challenges Faced:</u>

Although I encountered many challenges during the preparation for lab demonstration, but by effective communication with other team members and support from other peers helped me overcome them.

Firstly, since I was reading the connection layout of a different Pololu stepper driver which looked exactly the same as the A4988 driver but had one connection different, I had to spend some time to figure out the right connections.

Secondly, the datasheet of the Omron slot sensor was a bit misleading as per the connections of the LED leg. So initially after checking the connections, I thought that the sensor might be broken and figured out later that the connections have to be reversed at the LED side after searching over the internet.

Thirdly, the stepper motor had some issues interfacing from the GUI after overall system code was integrated together. We had to do some mapping to enable the stepper motor accurately figure out the rotation angle and direction input from the user.

Finally, when we integrated all the hardware together, we figured out that stepper was causing interference with DC motor functioning. We figured out that when using the SERVO.h library, the interrupt pins 9 and 10 were disabled for any other use and we were using pin 10 for DC motor driver control. So, we changed the interrupt pin for motor from 10 to 11 and the problem was solved.

<u>Teamwork:</u>

Danendra Singh: Stepper Motor and Slot Sensor

I was responsible for the stepper motor control and slot sensor. I also integrated all the circuit together and mounted the parts on the wooden frame. At the same time of final software integration, I also helped in debugging the system code.

Yuchi Wang: Force sensor, PID control, Code integration

Yuchi worked to develop code to read force value from the force sensor and make the DC motor run at a variable speed with PID control according to the force which the user exerts on the force sensor. He also integrated all the code together and helped in debugging.

Pulkit Goyal: Graphical User Interface

Pulkit developed a GUI for the system and figured out the communication interface between the Arduino and the GUI running in Linux.

Pratibha Tripathi: Servo motor with IR Sensor

Pratibha developed the code for reading values from IR range sensor and control the servo motor with the IR sensor. She also developed the mapping for the IR sensor distance with the output voltage.

Rahul Ramakrishnan: DC motor PID and Servo with Potentiometer

Rahul worked to develop the PID control of the DC motor and also developed the Arduino code for potentiometer control for the Servo motor.

We worked very efficiently for this task and the responsibilities of each team member were divided fairly. The team worked together for this lab assignment at our workbench which made possible the resolving of queries and conflicts at one place instantaneously which wouldn't have been possible if we had worked independently. Yuchi was in-charge of software integration combining codes from all the members. Pulkit had been working on the entire GUI part and made it ready for interfacing with the Arduino. Pratibha helped combine all the sensors and motors on one board. Rahul worked to develop DC motor PID control and Servo motor control.

<u>Current Project Progress and Future Plan:</u>

We got a Clearpath Husky mobile platform from Prof. George Kantor. The platform did not have live batteries and hence we ordered and got Li-Fe-Po batteries. To communicate with Robot PC and Husky we ordered USB to serial cable.

Currently, we are working on the husky to get it mobile. We are having a lot of driver related issues and are trying to resolve them.

We have also received Parrot Beebop2 drones from Prof. Katia and have tested the vehicle by remote controlled flight and also test its stability and camera output.

We have to figure out the support for the drone's SDK. We are waiting for a Velodyne puck to get started with obstacle avoidance with the Husky. Studies are being done on the various algorithms used for SLAM and path planning. Our next milestone is to autonomously move the husky in a user specified direction.

Task 7 (Sensors and Motor Control Lab) Quiz

Danendra Singh

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?
     - +/- 3.6g (typical), +/- 3g(minimum)
   - What is the sensor's dynamic range?
     - 20log(0.707*3.6/150ug)= 84.59 micro
   - 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?
     - C_{DC} is a decoupling capacitor of value 0.1uF. It is used to decouple ADXL335 from noise from supply and hence its placed close to the input pins. It absorbs high-frequency input surges and hence helps maintains constant supply.
   - Write an equation for the sensor's transfer function.
     - Y=1.5+ 0.3*X
   - What is the largest expected nonlinearity error in g?
     - ±0.3% of full scale
   - How much noise do you expect in the X- and Y-axis sensor signals when the sensor is excited at 25 Hz?
     - 750ug/rms
   - How about at 0 Hz? If you can't get this from the datasheet, how would you determine it experimentally?
     - Measure deviation from nominal voltage at 0 Hz and find the RMS value of this deviation.

2. Signal conditioning
   - Filtering
     - What problem(s) might you have in applying a moving average?
       - It does not categorize different frequencies. Loses some data at the beginning and/or at the end of the time series .The response time may be slow when we are filtering high frequencies.
     - What problem(s) might you have in applying a median filter?
       - It is computationally expensive. It has boundary issues- misses boundary values of the signal.
   - 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 which of V1 and V2 will be the input voltage and which the reference voltage, the value of the reference voltage, and the value of Rf/Ri in each case. If the calibration can't be done with this circuit, explain why.

- Your uncalibrated sensor has a range of -1.5 to 1.0V.
  - Input Voltage: V1
  - Reference Voltage: V2
  - Solve the equation: $(V_1-V_2)/R_i = (V_2-V_{out})/R_f$
  - We get: $R_f/R_i = 1$ and $V_{ref} = -3V$
- Your uncalibrated sensor has a range of -2.5 to 2.5V.
  - Since we have two equations and 3 variables, we get no solution.
  - Hence no solution exists.

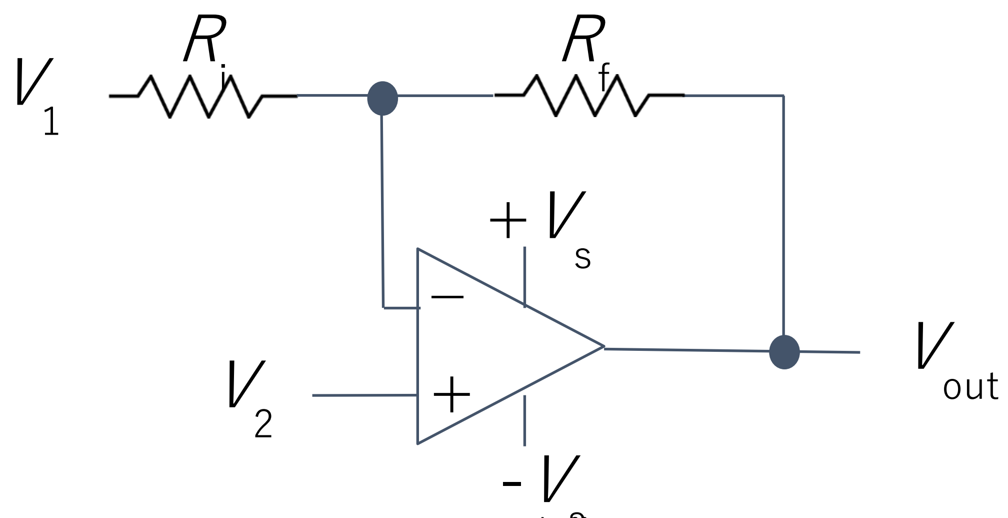


Fig. 1 Opamp gain and offset circuit

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.
     - We first calculate the error term by subtracting the current position with the desired position, then we input this to the proportional term. To do the Integral control, we simply sum up the past values. To approximately calculate the Derivative term we take the difference between the past values.
   - If the system you want to control is sluggish, which PID term(s) will you use and why?
     - Proportional term is used to get rid of the sluggishness as it multiplies the error term with a fixed constant value and overcomes the sluggishness by increasing the rise time.
   - 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?
     - We use the Integral Control to eliminate the steady-state error as the I term increases as long as there is a deviation from required position.

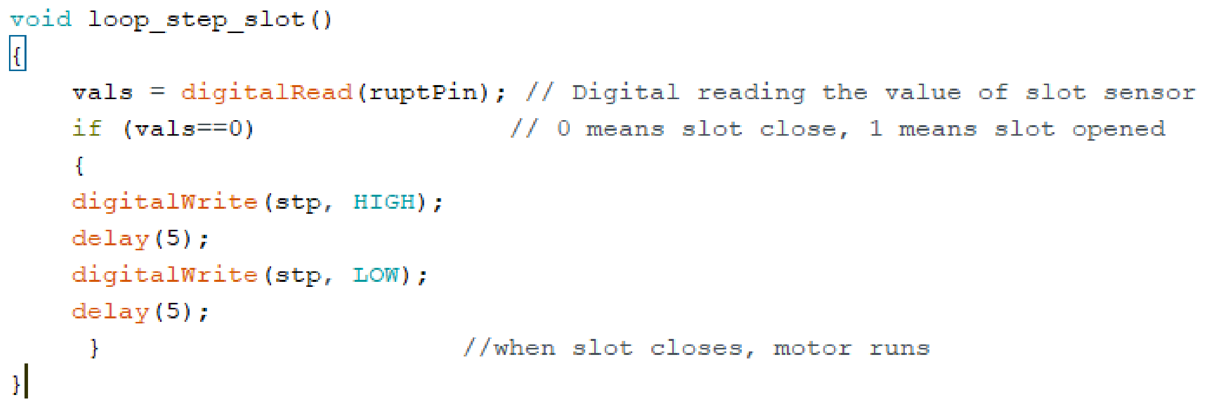
- After applying the control in the previous question, if the system still has overshoot, which PID term(s) will you apply and why?
    - To overcome the overshoot, we use the Derivative control as it doesn’t allow rapid changes in the error function. It damps the system and brings down the rate of change of error fucntion to zero.

Code Layout:

1. Code to control the stepper motor in the user specified Direction by a user specified angle:

```
void loop_stepper()
{   tar_ang=setpoint/(1.8);    //Setpoint is the user input angle
    if (tar_dir==1)            //target direction, 1 for clockwise, 0 for counter-clockwise
    {
    digitalWrite(dir,LOW);

    if (a <tar_ang) //sweep for a steps till the target angle is achieved
   {
    a++;
    digitalWrite(stp, HIGH);
    delay(5);
    digitalWrite(stp, LOW);
    delay(5);
   }

    }
    else if (tar_dir==0)
    {
      digitalWrite(dir,HIGH);

    if (a <tar_ang) //sweep 200 step in dir 1
   {
    a++;
    digitalWrite(stp, HIGH);
    delay(5);
    digitalWrite(stp, LOW);
    delay(5);
  }

    }

}
```

2. Code to control the Stepper motor with Slot Sensor



```
void loop_step_slot()
{
    vals = digitalRead(ruptPin); // Digital reading the value of slot sensor
    if (vals==0)                 // 0 means slot close, 1 means slot opened
    {
    digitalWrite(stp, HIGH);
    delay(5);
    digitalWrite(stp, LOW);
    delay(5);
     }                         //when slot closes, motor runs
}
```