Laavanye Bahl

Team D: CuBi

Teammates:
Jorge Anton, Paulo Camasmie, Changshen Shen, Nithin Subbiah Meganathan

ILR01

Feb. 14, 2019

Individual Progress

Sensors and Motor Control Lab

I worked on:

1. Wiring/soldering the ultrasonic sensor and writing the code to get readings from it.
2. Wiring/soldering the stepper motor driver and writing the code to control it and set it at a certain angle.
3. Combine ultrasonic and stepper motor: Read distance from ultrasonic and move stepper motor to a corresponding angle in both directions.

Ultrasonic Sensor

I checked out the website of the ultrasonic sensor product which gave me information about how to solder the wires. It was also stated that it requires a 2.5V to 5.5V supply with 2mA typical current draw. Therefore, I used the arduino to power it up, giving it 5V. I initially used analog signals to receive the input from the ultrasonic sensor, but then shifted to PWM for more accurate readings. With the pulse in function of arduino, I calculated the time difference of signal. From the data-sheet I noted down the value of pulse width = 147uS/inch. I finally got the distance in mm by the following calculations:

```
deltaT = pulseIn(pwPinUltra, HIGH);     // raw readings  // microsecond
inches = deltaT/pulse_width;
mm_curr = inches*25.4;
```

I also implemented a moving average window filter to remove noise. After trying a couple of different values I used an average over 20 readings for smooth readings.

Stepper Motor

I looked up the data-sheet to check the current and voltage requirements. Then I saw the data-sheet and manual of the motor driver that came with the stepper motor. I found how to solder the wires and connectors and followed the procedure. The stepper had a step size of 1.8 degrees. So, I initially started with the normal step wise code for control. Then, I came across micro stepping which motived at one eight of the step size. I used this to precisely control the motor. I made a function which could take as input any angle in degrees and then move the motor to that angle. I maintained the current angle to get the relative position.

Ultrasonic + Stepper Motor

I had to handle three things. Make a general function to receive any angle, enable stepper motor to be controlled both with GUI and sensor and define a range of operation for the ultrasonic sensor. A range of 0 to 360mm was chosen for the demonstration.
So, as the distance of the object from the ultrasonic increases the the stepper motors moves accordingly in positive direction and as the distance decreases the motor moves in the opposite direction. I always maintain my current angle for this purpose and look for the difference between the desired and current angle.

Cubi Project

- I worked on senor trade study and selection of final sensors for different applications like object detection and SLAM.
- Have been working on figuring out the right places to mount the sensors.
- Research work for object detection such as PCL clustering.
- Coded initial programs to get an output as shown below. I would continue to work on this, improve and try other methods. This activity will simultaneousl validate our sensor placement and object sizes.

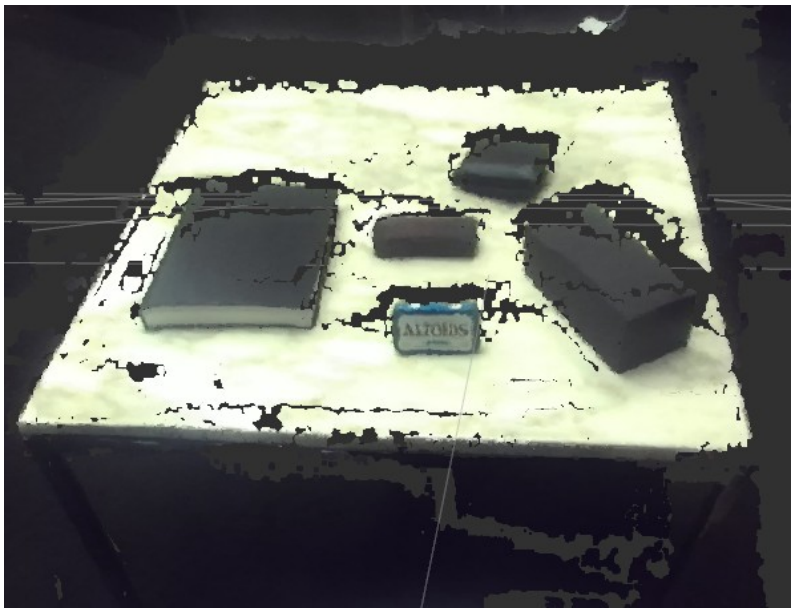


Fig 1)a) 3D point cloud output

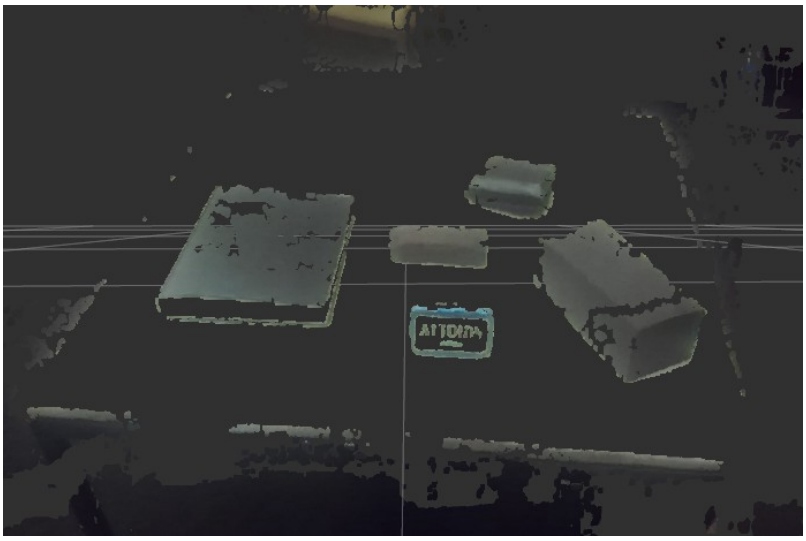


Fig 1)b) Segmented Objects

Challenges

Sensors and Motor Control Lab

- I had not done a lot of soldering before. So, that was a bit frustrating.
- The ultrasonic readings were noisy. I implemented smoothing to filter out noise.
- Integration was very challenging as it would break individual components.

CuBi Project

- Sensor placement in accordance with the final design of the manipulator as it could potentially block its view.
- Complete manipulator design as soon as possible, to have good progress with controlling it.

Teamwork

Sensors and Motor Control Lab

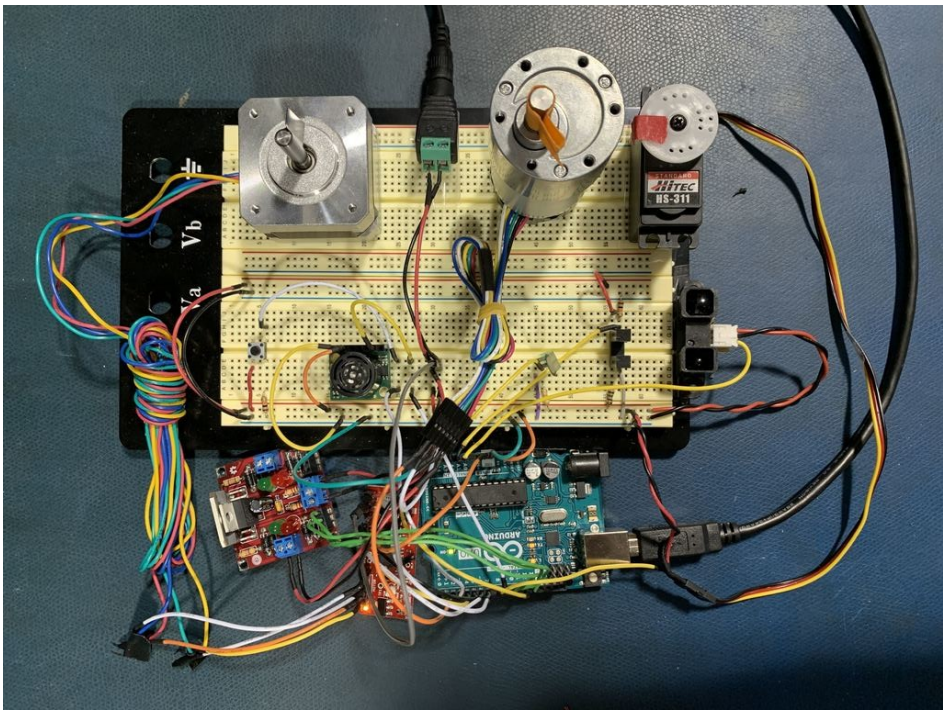


Fig 2) Our integrated circuit for the task

- Paulo worked on the controlling the DC motor with PID.
- Nithin mapped voltage values to real outputs and made electric circuits.
- Bobby did a soldering for the DC motor, made the GUI and worked with Jorge for communication between ROS and arduino and sensors.
- Jorge worked on integration, ROS communication and micro sensor.

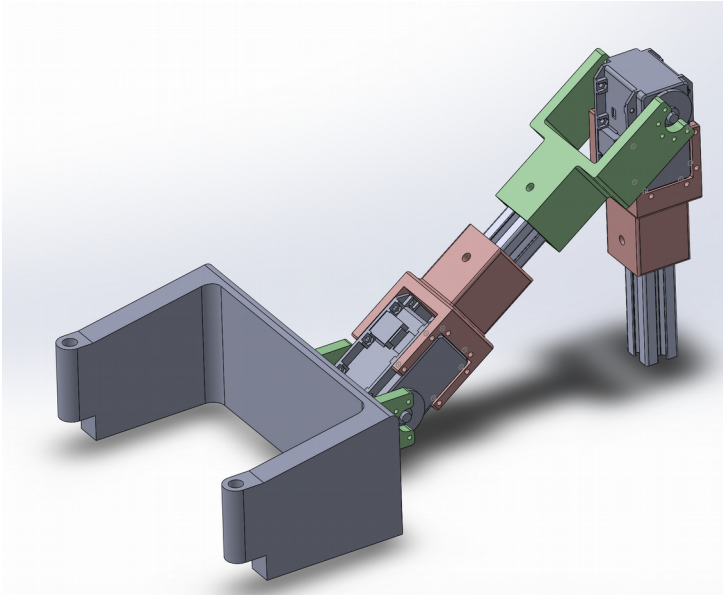


Fig 3) Manipulator prototype

- Paulo mainly worked on the fast prototyping of the design. The whole team met and discussed multiple times to validate different designs and improve it .
- Bobby set up the mobile base turtle-bot and did a lot of work on purchasing various components.
- Jorge worked on finalizing the schedule and did great work as the project manager. He is also working on controlling dynamixels.
- Nithin helped Paulo with manipulator and Bobby with assembling the turtle-bot, ordering of dynamixels and its initial control.

The whole team had multiple meetings with Prof. Nancy Pollard and Prof David Held for the progress reviews.

Future Plans

We want to quickly finalize the design of our first prototype for the gripper and validate picking of objects. We are working on developing the controller for it. Once the design is finalized we will have more clarity on the sensor placement. We will work on improving object detection algorithm and research more in that area. After that we will work on trying to get a 3d bounding box around our object.

Appendix A:

Task 4 (Sensors and Motor Control Lab) Quiz

4. **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?**
     ±3g (Minimum), ±3.6g (Typical)
   - **What is the sensor's dynamic range?**
     6g (Minimum), 7.2g (Typical)
   - **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?**
     The capacitor behaves like the device's energy storage to eliminate the noise of power supply. When there is a drop in supply voltage, then current is drawn out of the capacitors for compensation.
   - **Write an equation for the sensor's transfer function.**
     $$V_{out} = 1.5 + \left(\frac{300}{1000}\right) * a$$
   - **What is the largest expected nonlinearity error in g?**
     $$0.3 * 6g = 0.018g$$
   - **How much noise do you expect in the X- and Y-axis sensor signals when the sensor is excited at 25 Hz?**
     $$150 * \sqrt{25} = 750 \mu g$$
   - **How about at 0 Hz? If you can't get this from the datasheet, how would you determine it experimentally?**
     To determine this experimentally, we can place the accelerometer still (0 value, 0 Hz) and measure the output signal. Whatever is the out put is noise.

5. **Signal conditioning**
   - **Filtering**
     - **What problem(s) might you have in applying a moving average?**
       One of the major problems is that very high data can affect the moving average and force it towards a high value.
       Also, equal and opposite data does not have effect as a whole.
     - **What problem(s) might you have in applying a median filter?**
       If the window is big, it would result in latency in the readings.

If the window is small, then the noisy data would be prevalent and have its effects.

- **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: 1) which of V1 and V2 will be the input voltage and which the reference voltage; 2) the value of the reference voltage; and 3) 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 (-1.5V should give a 0V output and 1.0V should give a 5V output).**
    - **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).**

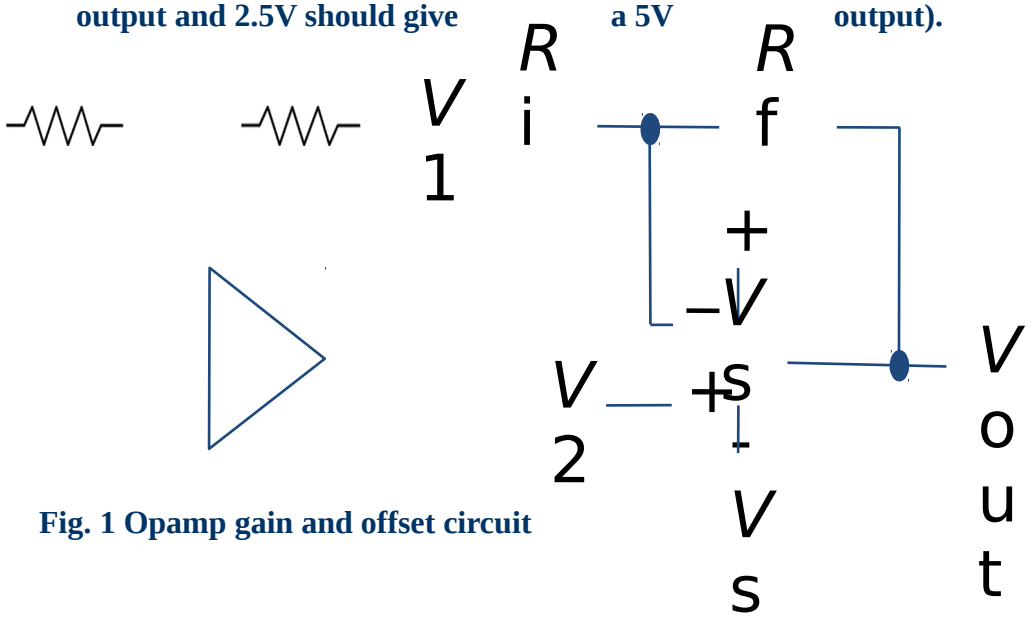


Fig. 1 Opamp gain and offset circuit

Ans) $V_{out}=\left(V_{in}-V_1\right)\frac{R_f}{R_i}+V_{in}$

1) Uncalibrated sensor has a range of -1.5 to 1.0V.

We take $V_{in}=V_2$

Minimum output $0=\left(-1.5-V_1\right)\frac{R_f}{R_i}-1.5$

Maximum output $5=\left(1.0-V_1\right)\frac{R_f}{R_i}+1.0$

By solving the above two we get,

$V_1=V_{ref}=-3, \frac{R_f}{R_i}=1$

2) **Uncalibrated sensor has a range of -2.5 to 2.5V.**

We take $V_{in} = V_2$

Minimum output $0 = (-2.5 - V_1)\frac{R_f}{R_i} - 2.5$

Maximum output $5 = (2.5 - V_1)\frac{R_f}{R_i} + 2.5$

By solving the above two we get,

$-2.5 = 2.5$

Hence, no solution exists for this case.

6. 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 keep getting the readings and calculate the difference between the target and current position to get the error. We keep on storing these errors and accumulate them for the integral term. For the derivate term we see the difference of the current and previous error over the time step.

  Control signal is given by $u = K_p * error + K_d * derivative_{error} + K_i * total_{error}$

  We tune the gains Kp , Kd and Ki

- **If the system you want to control is sluggish, which PID term(s) will you use and why?**

  We will apply proportional control as it will make the response time faster as the error term will have more effect.

- **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?**

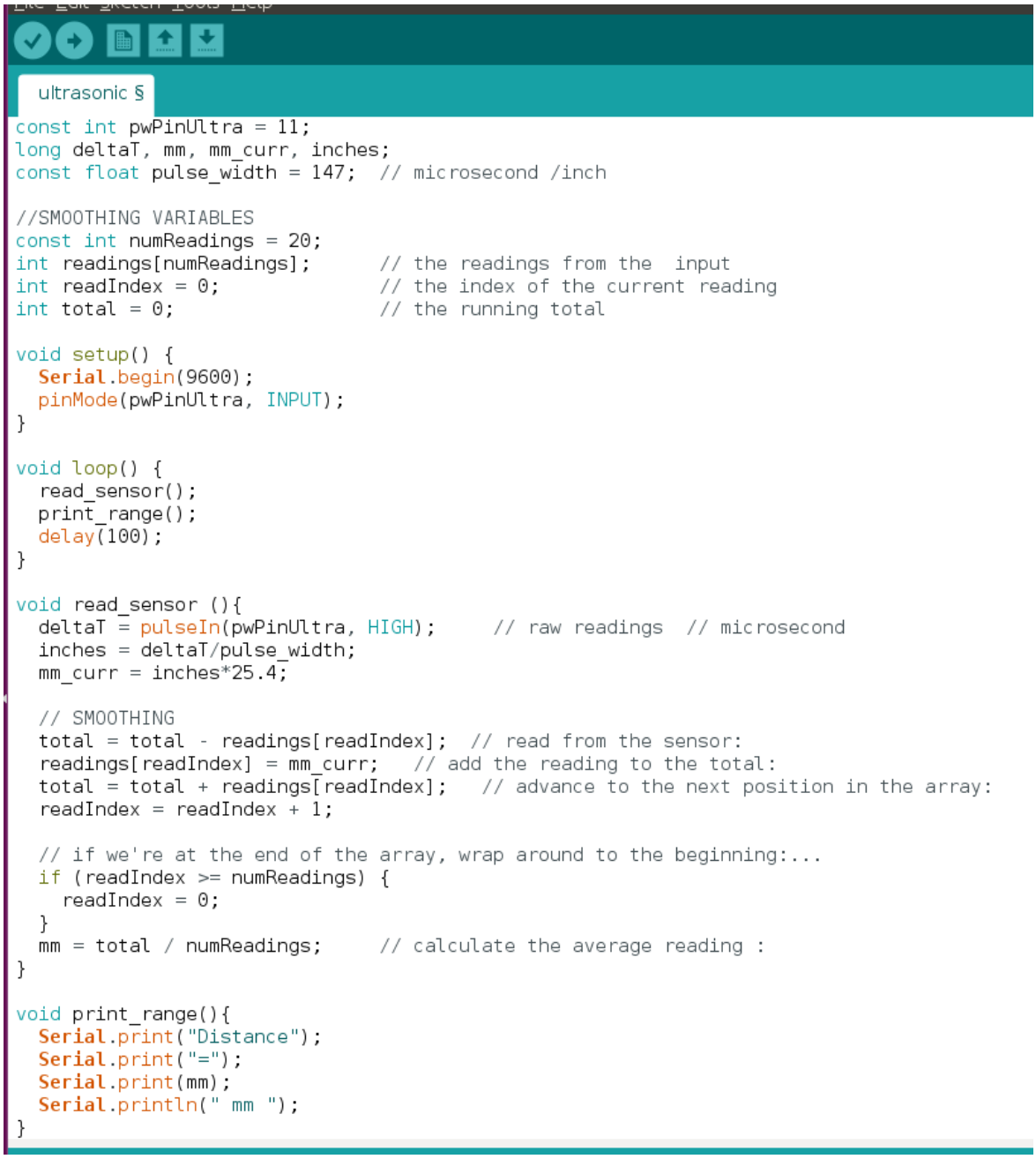
  We will apply integral control to increase the effect of the accumulated history of errors. In this way the steady state error will get accumulated and be removed.

- **After applying the control in the previous question, if the system still has overshoot, which PID term(s) will you apply and why?**

  We will apply derivative control to damp the response and remove overshooting by adjusting for the change in error.

Appendix B:

Code for reading from ultrasonic sensor



ultrasonic §

```
const int pwPinUltra = 11;
long deltaT, mm, mm_curr, inches;
const float pulse_width = 147;  // microsecond /inch

//SMOOTHING VARIABLES
const int numReadings = 20;
int readings[numReadings];      // the readings from the  input
int readIndex = 0;              // the index of the current reading
int total = 0;                  // the running total

void setup() {
  Serial.begin(9600);
  pinMode(pwPinUltra, INPUT);
}

void loop() {
  read_sensor();
  print_range();
  delay(100);
}

void read_sensor (){
  deltaT = pulseIn(pwPinUltra, HIGH);     // raw readings  // microsecond
  inches = deltaT/pulse_width;
  mm_curr = inches*25.4;

  // SMOOTHING
  total = total - readings[readIndex];  // read from the sensor:
  readings[readIndex] = mm_curr;   // add the reading to the total:
  total = total + readings[readIndex];   // advance to the next position in the array:
  readIndex = readIndex + 1;

  // if we're at the end of the array, wrap around to the beginning:...
  if (readIndex >= numReadings) {
    readIndex = 0;
  }
  mm = total / numReadings;     // calculate the average reading :
}

void print_range(){
  Serial.print("Distance");
  Serial.print("=");
  Serial.print(mm);
  Serial.println(" mm ");
}
```

Appendix C:

Code for controlling stepper motor with readings from ultrasonic sensor and GUI

```
#define stp 2
#define dir 3
#define MS1 4
#define MS2 5
#define EN  6
#define pwPinUltra 11

long deltaT, mm, inches, mm_curr;
const float pulse_width = 147;  // microsecond /inch
float resolution = 1.8;

//Declare variables for functions
int x;
int numsteps;
float curr_angle=0;
float angle_diff = 0;

//Smoothing
const int numReadings = 20;
int readings[numReadings];      // the readings from the analog input
int readIndex = 0;              // the index of the current reading
int total = 0;                  // the running total

//Reset Easy Driver pins to default states
void resetEDPins()
{
  digitalWrite(stp, LOW);
  digitalWrite(dir, LOW);
  digitalWrite(MS1, LOW);
  digitalWrite(MS2, LOW);
  digitalWrite(EN, HIGH);
}

void setup() {
  pinMode(stp, OUTPUT);
  pinMode(dir, OUTPUT);
```

```
  pinMode(MS1, OUTPUT);
  pinMode(MS2, OUTPUT);
  pinMode(EN, OUTPUT);

  pinMode(pwPinUltra, INPUT);

  resetEDPins(); //Set step, direction, microstep and enable pins to default states
  Serial.begin(9600); //Open Serial connection for debugging
}

void loop(){
  digitalWrite(EN, LOW); //Pull enable pin low to allow motor control
  move_stepper_from_ultrasound(calcDiff(););
  resetEDPins();
  delay(1);
}

long read_ultrasound (){

  // raw readings
  deltaT = pulseIn(pwPinUltra, HIGH);    // microsecond
  inches = deltaT/pulse_width;
  mm_curr = inches*25.4;

  // SMOOTHING
  total = total - readings[readIndex];
  // read from the sensor:
  readings[readIndex] = mm_curr;
  // add the reading to the total:
  total = total + readings[readIndex];
  // advance to the next position in the array:
  readIndex = readIndex + 1;
  // if we're at the end of the array...
  if (readIndex >= numReadings) {
    // ...wrap around to the beginning:
    readIndex = 0;
  }
  // calculate the average reading :
  mm = total / numReadings;
  return mm;
}
```

```
// 1/8th microstep foward mode function
void SmallStepModeForward(float angle){
   int numsteps = (int) (angle/resolution)*8;
   digitalWrite(dir, LOW); //Pull direction pin low to move "forward"
   digitalWrite(MS1, HIGH); //Pull MS1, and MS2 high to set logic to 1/8th microstep resolution
   digitalWrite(MS2, HIGH);
   for(x= 1; x<numsteps; x++)  //Loop the forward stepping enough times for motion to be visible
   {
     digitalWrite(stp,HIGH); //Trigger one step forward
     delay(1);
     digitalWrite(stp,LOW); //Pull step pin low so it can be triggered again
     delay(1);
     curr_angle += resolution/8;
   }
}

// 1/8th microstep reverse mode function
void SmallStepModeReverse(float angle){

   int numsteps = (int) (angle/resolution)*8;
   digitalWrite(dir, HIGH); //Pull direction pin low to move "forward"
   digitalWrite(MS1, HIGH); //Pull MS1, and MS2 high to set logic to 1/8th microstep resolution
   digitalWrite(MS2, HIGH);
   for(x= 1; x<numsteps; x++)  //Loop the forward stepping enough times for motion to be visible
   {
     digitalWrite(stp,HIGH); //Trigger one step forward
     delay(1);
     digitalWrite(stp,LOW); //Pull step pin low so it can be triggered again
     delay(1);
     curr_angle -= resolution/8;
   }
}

float calcDiff(){
  read_ultrasound();
  //print_range();
  float angle = 0;
  if(mm<360 && mm>0){
      angle = mm;
  }else{
```

```
    angle = curr_angle;
  }
  return angle;
}

void print_range(){
  Serial.print("sensor distance");
  Serial.print("=");
  Serial.print(mm);
  Serial.print(" mm, ");
  Serial.print(" current Angle: ");
  Serial.print(curr_angle);
  Serial.print(" diff Angle: ");
  Serial.println(angle_diff);
}

void moveStepper(float angle){
  angle_diff = curr_angle - angle;
  digitalWrite(EN, LOW);              //Pull enable pin low to allow motor control
  if (angle_diff<0 && abs(angle_diff)>0.5){
     SmallStepModeForward(abs(angle_diff));
  }else{
     SmallStepModeReverse(abs(angle_diff));
  }
  resetEDPins();
}

void move_stepper_from_ultrasound(){
  moveStepper(calcDiff());
}
```