TASK 7: Sensors and Motor Lab

Name:Rohit Dashrathi
Team :G (Robographers)
Teammates:
Jimit Gandhi
Gauri Gandhi
Rohit Dashrathi
Sida Wang
Tiffany May
ILR No.: # 1
Submission Date: Oct 16, 2015

1. Individual Progress

Responsibility: Interfacing the Servo Motor with Ultrasonic Sensor

Hardware Used (Figure 1): Hitech HS-225BB, Breadboard, Arduino Uno Microcontroller, Jumper Cables, 68uF Capacitor (2 Qty), 10kOhm Resistor, 100 Ohm Resistor, Maxbotix EZ Ultrasonic Sensor, USB 2.0 Serial Cable

Software Used: Arduino IDE

Task Description:

The task of interfacing servo motor with ultrasonic sensor was accomplished in following steps:

a. Checking & Calibrating the ultrasonic sensor
b. Implementation noise filtering for ultrasonic sensor
c. Checking & running the servo motor
d. Interfacing the servo motor with ultrasonic sensor and debounce enabled pushbutton switch.

a. Checking & calibrating the ultrasonic sensor[1]:

The maxbotix ultrasonic sensors[2] were interfaced with Arduino via AnalogPin 0 for Analog connection in following way:

`

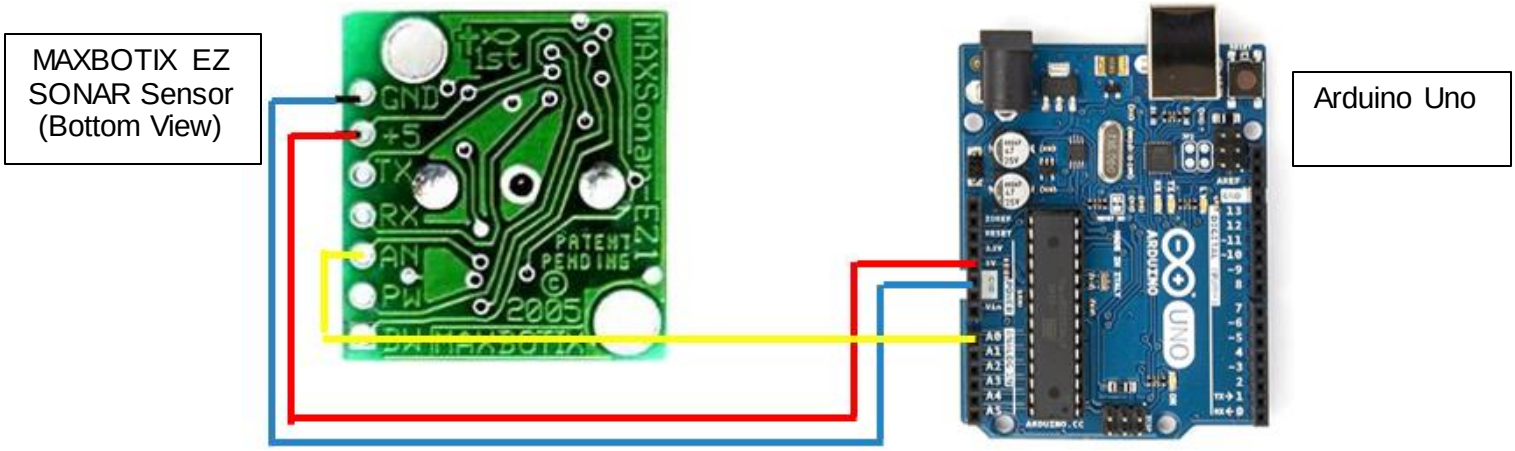

Figure 1: Checking and Calibrating Ultrasonic Sensor (Connection Diagram)

Code was written in IDE to check the readings given by sensor. It was observed that the sensor gave a count of 11 when any object was within 0 to 0.5 Ft distance from the sensor. While, the count was linearly increasing when the same object was moved in the range 0.5 ft to 20 ft which was confirmed with the given sensor range in its datasheet. Readings given by sensors were compared to equivalent distances measured by measuring tape. A set of 20 such readings was recorded and a calibration equation was formed as follows:

Reading_ft (Calibrated output) =Reading (Uncalibrated Sensor output)*0.046.

Hence, the sensor was now able to output real distances. However, the limitation was not being able to measure values between 0 to 0.5 Ft due to the inbuilt sensor error. All these values were overridden to 0 Ft programmatically.

b. Implementation noise filtering for ultrasonic sensor[3]:

It was observed that, the sensor reading were varying with a huge proportion over very short time intervals worth a few microseconds. It was noted that this happened due to the inbuilt sensor noise. To overcome this noise, a small filtration circuit was developed as follows:

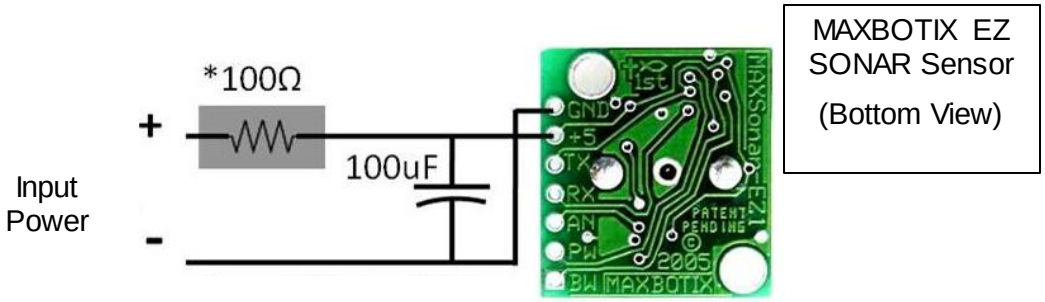

Figure 2: Noise filtering for ultrasonic sensor (Connection Diagram)

The noise filtering was done by using research material over internet. No cross check calculations were performed on my end. Also, due to absence of a 100uF, 6.3 V Capacitor in RI lab, 2 capacitors of 68 uF, 6.3V rating were used in parallel, accounting to a total capacitance of 136 uF.

Sensor readings were taken again and were found to be consistent over 500 ms of time.

c. Checking & running the servo motor

The given Hitech HS 225BB hobby servo motor was interfaced with Arduino via digital pin 10 while disconnecting the sensor connections done earlier. The motor was checked with a test program allowing the motor to rotate and reach its both ends while incrementing the angle by 1 deg at every 15 sec (delay). The motor was found to be working correctly. Connections were done as shown in following figure:

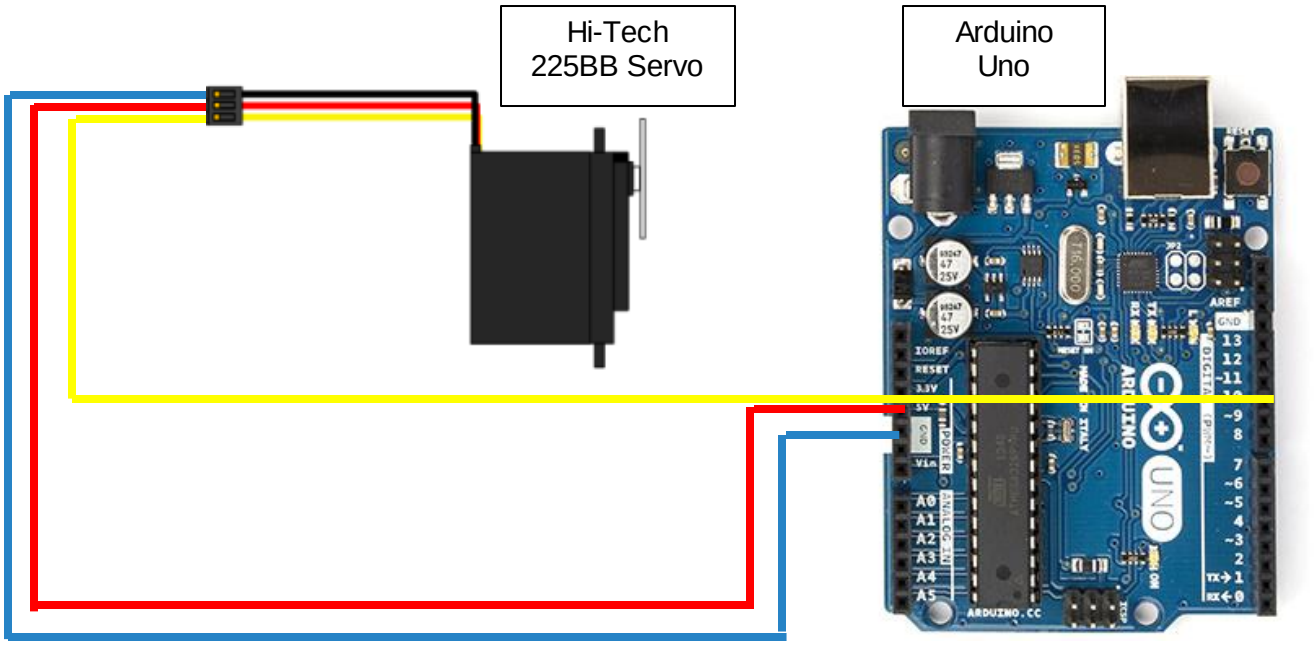

Figure 3: Servo Motor Checking (Connection Diagram)

d. Interfacing the servo motor with ultrasonic sensor and debounce enabled pushbutton switch:

Finally servo motor was implemented with ultrasonic sensor. This was done such that the input angle given to servo was determined by the distance in feet measured by ultrasonic sensor using following equation:

pos (Servo angle) =19.5*Reading_ft (Sensor Output)+0.5;

The servo angle was overridden to 180 deg for all the 'pos' values over 180 deg which is the maximum limit for Hitech HS 225BB servo.

A pushbutton switch was interfaced with Arduino using a pull down resistor of 100 ohms. The switch was programmed such that, for all even number of button presses, the servo would go to an initial position of 0 deg. While for all odd number of button presses, it would move according to corresponding sensor readings with respect to above equation. This was done by setting a counter variable for recording number of button pushes.
A debounce function (same as Task 2) was programmed to avoid the bouncing of switch and record correct button states, shown in following figure.

Final_Combined_Servo_Sensor_Button_Debounce

```
void debounce(int button) {
  // read the state of the switch into a local variable:
  int reading = digitalRead(button);

  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;
    }
  }

  lastButtonState = reading;
}
```

Figure 4: Button Debounce function

For debouncing, the button state was stored in a variable called 'reading'. Another global variable called 'lastButtonState' was declared and initiated to zero, which would record the previous button state after button is pushed. Global variables 'lastDebounceTime' and 'debounceDelay' were declared to record the the time lapse between button debouncing and a fixed delay time allotted for enabling debouncing in the button. DebounceDelay was set to 150 ms. The final code enabled good debouncing in the button.

The final circuit was developed as follows:

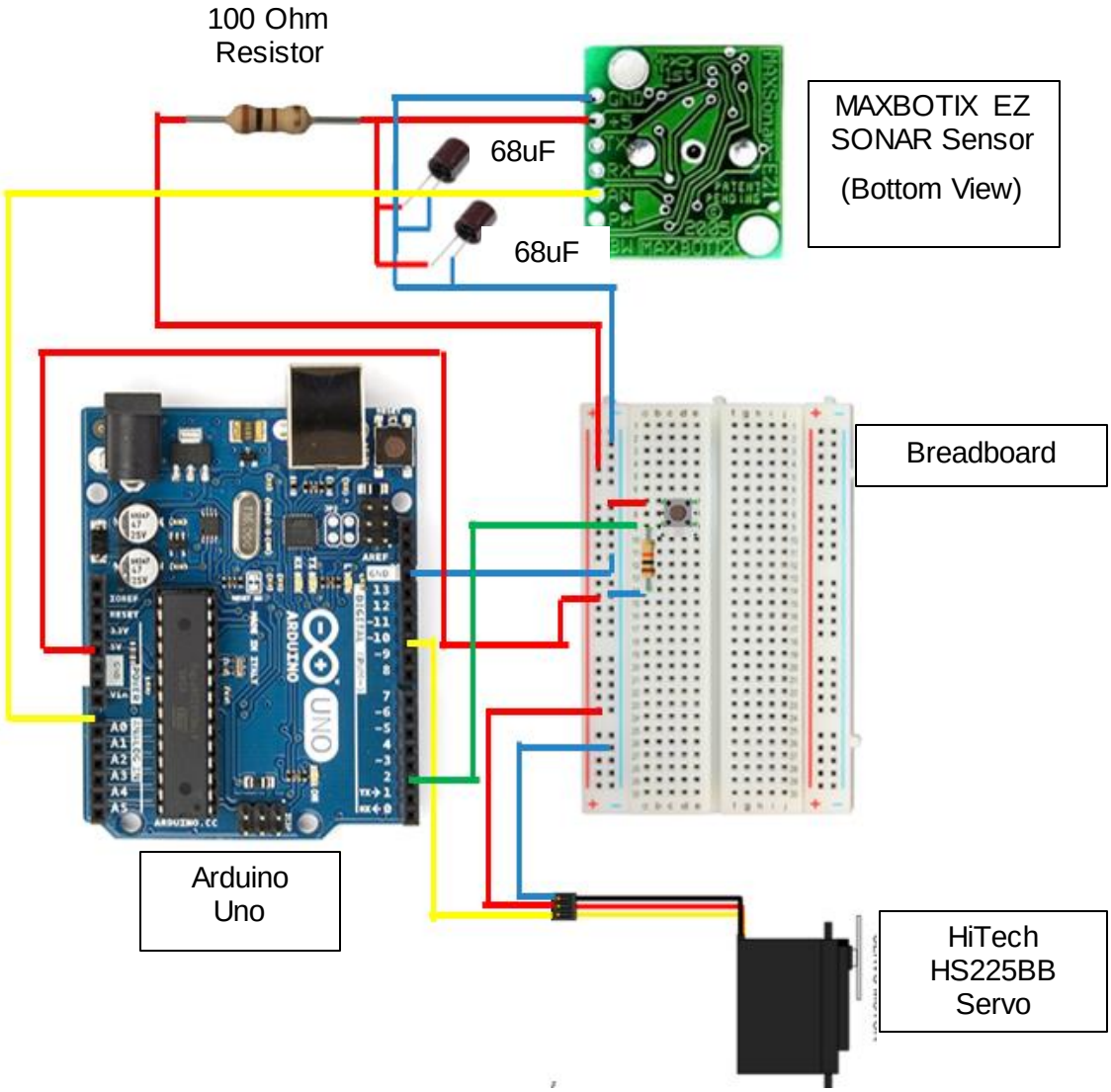

Figure 5:Hitech HS225BB Servo Motor Interfacing with MAXBOTIX SONAR Sensor (Connection Diagram)

e. Final Individual Code:

The final individual development code is attached at the end of document (Appendix I)

2. Challenges

The primary challenge during this lab assignment was the breakdown of Arduino. 2 Arduino uno boards got burned down during this assignment. Both these boards were broken within half of hour of circuit connections. Following troubleshooting steps were followed to figure out the reason for this breakdown:

1. **Checking the circuitry:** All the connections seemed fine during this step.
2. **Avoiding making connections while Arduino being connected:**

   This step was not considered while doing the connections for 1st and 2nd Arduino board which were burned out. However, it was taken in account while using the 3rd and final Arduino board which performed fine. It can be said that making connections while Arduino being connected might have generated excessive heat with the possibility of certain short/open connections.

3. **Life of Arduino:** The 2 Arduinos used before seemed old and worn out aesthetically as compared to the final board. Hence, age of device could also be a possible reason for breakdown.

   Another challenge faced was time elapsed to interface analog sensors. We had chosen all analog sensors to gain more knowledge about complexities in analog sensor interfacing instead of choosing simpler digital sensors. Hence, sensor calibration took almost 4 times the usual required time. Finally we had to compromise on our time for GUI-circuitry interfacing.

3. Teamwork

To complete the assignment in time, Team G split up the tasks evenly, taking individual skills into consideration. I took the responsibility to simplify project description and explain it during team meeting posted on blackboard, in form of 2-3 PPT slides. Sida Wang, Team G's main programmer and computer vision expert took the challenge to develop the GUI and integrated framework, combining the codes of other team members for motor and sensor interfaces. I, having good prior experience of Arduino programming and interfacing, was assigned to work with Servo Motors and Ultrasonic Sensor, while guiding others during their respective subtasks. Tiffany May being electronics developer of the team, worked with Rohit for Servo motor implementation with additional Thermo Sensor. Gauri Gandhi & Jimit Gandhi worked on integrating Force Sensor with Stepper motor and Slot Sensor with DC motor respectively. The team had set its target dates for each task and decided to complete it 4 days before the lab presentation, to allow sufficient time for integration. With this division of tasks, the team was able to achieve tasks in parallel to complete the lab with success.

4. Future Plans

Individual future plans before 1st progress review regarding the Robographer Project:

1. Bi-weekly plan for project till 26 Oct 2015
2. Completion of (PTZ) Pan Tilt Zoom unit design and 3D drawing.
3. Collecting and downloading CAD models of individual components in system
4. Completion of single turtlebot 3D model equipped with subsystem components such as camera, PTZ unit, and microcontroller.
5. Evaluation and planning of the strategy for planning and detection subsystem.

References

1 **SONAR sensor connection to Arduino:** http://www.maxbotix.com/articles/085.htm

2 **MAXBOTIX SONAR EZ datasheet:** https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCgQFjAAahUKEwiF_Z2DjrnIAhVIGj4KHbYxCyU&url=http%3A%2F%2Fwww.maxbotix.com%2Fdocuments%2FLV-MaxSonar-EZ_Datasheet.pdf&usg=AFQjCNGyGJuak70B13PTL3sFIAiT86M8YA&bvm=bv.104819420,d.cWw

3 **Sensor Noise Reduction Update:** http://diydrones.com/forum/topics/sonar-noise-reduction-update?commentId=705844%3AComment%3A676098

```
//                          APPENDIX I

// Rohit U Dashrathi (Team G)
// MRSD Project Task 7  ILR 1
// Individual Code: (Servo Motor + Ultrasonic Sensor + Debounced Pushbutton

#include <Servo.h>
Servo myservo;
int pos=0;
float Reading=0;
float Reading_ft=0;

const int  buttonPin = 2;    // the pin that the pushbutton is attached to

// Variables will change:
int buttonPushCounter = 0;   // counter for the number of button presses
int buttonState = 0;         // current state of the button
int lastButtonState = 0;     // previous state of the button

// For Debouncing
long lastDebounceTime = 0;  // the last time the output pin was toggled
long debounceDelay = 50;

void setup() {
  myservo.attach(10);
  pinMode(A0, INPUT);
  // initialize the button pin as a input:
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}


void loop()
{
  Reading=analogRead(A0);
 Reading_ft=Reading*0.046;
  if (Reading_ft<=0.51)
  {
    Reading_ft=0;
  }

  // read the pushbutton input pin:
  buttonState = digitalRead(buttonPin);
  // compare the buttonState to its previous state
  if (buttonState != lastButtonState)
  {
```

```
      if (buttonState == HIGH)
    {
     buttonPushCounter++;
      Serial.println("on");
      Serial.print("number of button pushes:  ");
      Serial.println(buttonPushCounter);
    }
    else
    {
          Serial.println("off");
    }
    // Delay a little bit to avoid bouncing
   //delay(50);

  }
debounce(buttonPin);
lastButtonState = buttonState;

pos=19.5*Reading_ft+0.5;
if (pos>180)
{
pos=180;
}

  if (buttonPushCounter % 2 == 0)
  {
    myservo.write(pos);
    Serial.println(pos);
    delay(150);
  }

  else {
    myservo.write(0);
  }

}

void debounce(int button) {
  // read the state of the switch into a local variable:
  int reading = digitalRead(button);

  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }
```

```
  if ((millis() - lastDebounceTime) > debounceDelay) {

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;
    }
  }

  lastButtonState = reading;
}
```