Brian Sheridan

Lab 2: Collision Avoidance

Performed on 4/16/15

Submitted on 4/23/15

Objective:

The objective of the lab was to learn how to use the ultrasonic sensor to detect distance from an object. The distance must be calculated through sending a pulse from the Arduino and waiting for the pulse to return to determine the elapsed time, which allows one to determine the distance. Also to use control statements to control the flow of the code using while, if, else if, and for.

Introduction:

The Arduino Uno microcontroller is an Atmel ATmega328 integrated onto a developer board to allow for easy prototyping with the chips many pins. The ATmeaga328 has 6 analog inputs, 14 digital inputs/outputs along with 3.3v, 5v and ground pins. Arduino also has a integrated development environment (IDE) for quick prototyping and coding which allows one to quickly design and test a project.

A three LED shield was used with the Arduino to investigate how control statements and flow works. The LED was powered by the 5v pin and also contacted to 3 analog pins, A0, A1 and A2. Through the use of the digitalWrite library, the code was able to selectively turn on each LED through setting the pin to High or Low.

The ultrasonic sensor is a 4 pin board with 2 ultrasonic speakers. It has a ground, trigger, echo and vcc pin. The trigger pin is powered to emit a sound, which is sent out the speakers. The echo pin will be driven high when the pulse is bounced back to the sensor. This is used to calculate the elapsed time it took to reach the object. This time can then be transformed into distance to determine how far away the object was based on the speed of sound and distance traveled.

Procedure:

Part 1 of the lab consisted of setting up the ultrasonic sensor and wiring it into the Arduino. The VCC pin was wired to the 5V pin on the ISP header, trigger was wired to pin 2, echo was wired to pin 3, and ground was wired to the ground pin on the Arduino. The lab sheet contained code which was copuied into the Arduino Development Enviroment. The given code was modified to display in inches instead of millimeters through dividing by 25.4 to get inches. The code is displayed below:

```
int trigger = 2;
int echo = 3;

double getDistance(unsigned char triggerPin, unsigned char echoPin);

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  double targetDistance;
  targetDistance = getDistance(trigger,echo);
  Serial.println(targetDistance/25.4);
  delay(500);

}

double getDistance(unsigned char triggerPin, unsigned char echoPin){
   unsigned long int startTime, endTime;
   double mm;
  digitalWrite(triggerPin,HIGH);
  delayMicroseconds(20);
  digitalWrite(triggerPin,LOW);
  startTime = micros();

  while(digitalRead(echoPin)==LOW && micros()-startTime<100000){
    delayMicroseconds(20);
  }

  while(digitalRead(echoPin)==HIGH && micros()-startTime<100000){
  delayMicroseconds(1);
  }
  endTime = micros();

  if(endTime>startTime){
  mm = (endTime - startTime)/6.496;
  }

  delay(100);
  return(mm);

}
```

Part 2 of the lab consisted of modifying part 1 of the lab. The goal is to control the LED breakout board and change it to a specific color based on the rangefinders distance from an object. If the object is greater than 12 inches, the light should be green, if the range is between 12 inches and 8 inches the light should be yellow, and if the object is closer than 8 inches, the color should be red. This was achieved through using if else statements to check the status of the rangefinder every time it is returned. The code from the previous lab used to change the color of the light, which is similar to a traffic light. The code is displayed below:

```
int trigger = 2;
int echo = 3;

double getDistance(unsigned char triggerPin, unsigned char echoPin);

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(A0, OUTPUT);
  pinMode(A1, OUTPUT);
  pinMode(A2, OUTPUT);

  digitalWrite(A0, LOW);
  digitalWrite(A1, LOW);
  digitalWrite(A2, LOW);
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  double targetDistance;
  targetDistance = getDistance(trigger,echo);
  Serial.println(targetDistance/25.4);
  targetDistance = targetDistance/25.4;


  if(targetDistance>12){
      digitalWrite(A1, LOW);
      digitalWrite(A0, HIGH);
      digitalWrite(A2, HIGH);
  }
  else if(targetDistance < 12 && targetDistance > 8){
      digitalWrite(A0, LOW);
      digitalWrite(A1, LOW);
      digitalWrite(A2, HIGH);
  }
  else{
      digitalWrite(A0, LOW);
      digitalWrite(A1, HIGH);
      digitalWrite(A2, HIGH);
  }

  delay(500);

}

double getDistance(unsigned char triggerPin, unsigned char echoPin){
   unsigned long int startTime, endTime;
   double mm;
  digitalWrite(triggerPin,HIGH);
  delayMicroseconds(20);
  digitalWrite(triggerPin,LOW);
  startTime = micros();

  while(digitalRead(echoPin)==LOW && micros()-startTime<100000){
    delayMicroseconds(20);
  }

  while(digitalRead(echoPin)==HIGH && micros()-startTime<100000){
  delayMicroseconds(1);
  }
  endTime = micros();

  if(endTime>startTime){
  mm = (endTime - startTime)/6.496;
  }

  delay(100);
  return(mm);

}
```

Results:

Part 1 of consisted of creating the driver code to run the ultrasonic sensor. The ultrasonic sensor was wired up to pins 2, and 3, and VCC and Ground. The trigger pin would be written high, and then a 20 microsecond delay, which it would then be driven low. The code then waits for the echo pin to be driven high to indicate the pulse has bounced off an object and returned. This gives an elapsed time which allows us to calculate the distance. The intial reading was returned in millimeters. The final reading converted to inches through divinding the returned value by 25.4 which is the ratio of millimeters to inches. The results were displayed through the serial port which was accessed with PuTTY.

Part 2 of the lab consisted of modifying the previous part of the lab. The control statements were added to change the LED's color based on the distance returned from the range finder. The code worked as expected, as when something is pretty close to the ranger finder, under 8 inches, the light turns red. When the object was moved farther away from the finder, it would turn yellow. Finally the object was moved farther away than 12 inches and the light would turn green. Overall the code works exactly like the stop light code from the previous lab, though it is driven by the distance returned from the range finder and not a timer.

Conclusion:

Overall this lab has shown how to write driver code to run an ultrasonic sensor on the Arduino Uno microcontroller. It has also shown how to use control statements such as if, else if, while, and for to control the flow of code on the microcontoller. In this case, the control statements changed the color of the LED based on the range finders reading. This lab has achieved its goal of illustrating how to write driver code to run an ultrasonic sensors and how to effectively use control statements to control the LED shield based on the values returned by the range finder.