Sensor Based Systems Project Report - Characterization of Ultrasonic transducers

Team:

Divya Kalkotwar : divyak@kth.se

Krishnaswamy Kannan : kkannan@kth.se

Sagar Sharma : sagarsh@kth.se

Shivaram singh Rajput : shivaram@kth.se

Application Idea:

The project focuses on building an ultrasonic sensor to detect obstacles using two ultrasonic transducers with one acting as a transmitter sending ultrasound waves, while the other acting as a receiver receiving the ultrasound waves. This ultrasonic sensor can be embedded in a walking stick used by blind people to alert them when they approach an obstacle. In this way, blind people can also be alerted about obstacles which are way beyond the reach of their sticks, i.e. the obstacles at different heights from the ground level and the obstacles far from the reach of their sticks. This application will be very useful in helping the blind people avoid accidental falls due to obstacles in their path. Figure 1 illustrates the idea behind the project.

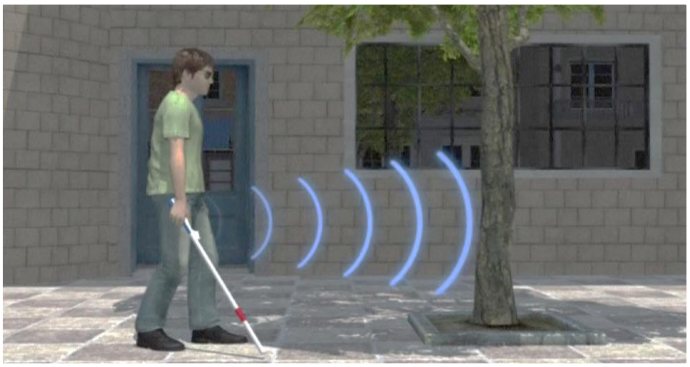

Figure 1: Obstacle detection for Blind using Smart walking stick

How does the project use the sensors?

An ultrasonic transducer is a device which converts alternating current (AC) into ultrasound waves or vice versa. There are two types of ultrasonic transducers, such as the ultrasonic transmitter which generates ultrasound waves when an AC voltage is applied at its input, and the ultrasonic receiver which senses ultrasound waves and converts them into electrical signals. We used the Ultrasonic transducers operating at 40 KHz in our sensor design. These transducers are in turn connected to the ATmega328P microcontroller on the Arduino Uno hardware platform via dedicated interfacing circuits. The sensor generates and senses ultrasound waves using the transmitter and receiver respectively. So, when an object appears in its vicinity, these sound waves are reflected back and depending on the strength of the received signal, the distance to the obstacle is obtained.

System Description:

The block diagram of the sensor is shown in Figure 2.

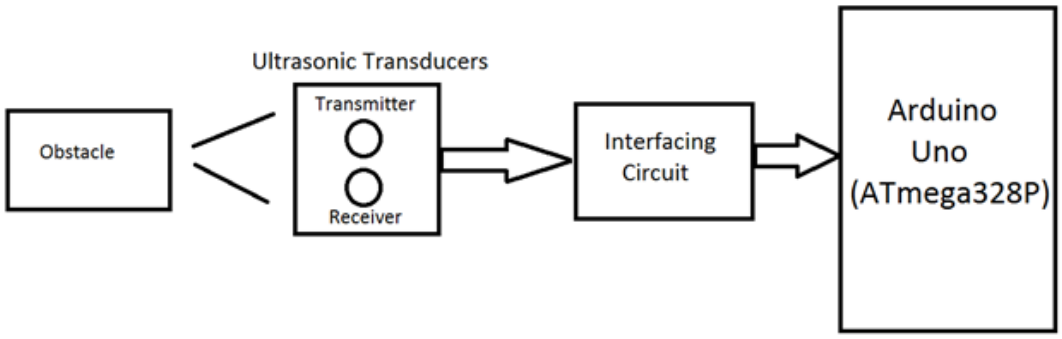

Figure 2: Ultrasonic sensor

The ultrasonic sensor comprises of the 40 KHz ultrasonic transmitter and receiver along with the circuits required to interface them with the microcontroller. The transmitter circuit is a voltage amplifier that amplifies the 5V from the ATmega328P to 10V in order to achieve a better range than that achievable by the transmitter transducer. The receiver circuit is a band pass amplifier which is used to amplify the 40 KHz frequencies of the electrical signal obtained from its output terminals.

A. Components used:

The components used to build the interfacing circuits for the ultrasonic sensor design are;

- ATmega328P microcontroller (Arduino Uno)
- 40 KHz ultrasonic transducers
- Transmitter circuit (Voltage amplifier):
  - 2N3906 PNP Transistors
  - 2N3904 NPN Transistors
  - 40 KHz ultrasonic transmitter
- Receiver circuit (three stage passive amplifier):
  - 2N4401 NPN Transistors
  - 40 KHz ultrasonic receiver
  - LM386 Audio Power Amplifier

B. Hardware Architecture:

a) Transmitter:

The transmitter H-bridge circuit used to interface the 40 KHz ultrasonic transmitter to the ATmega328P microcontroller is shown in Figure 3.

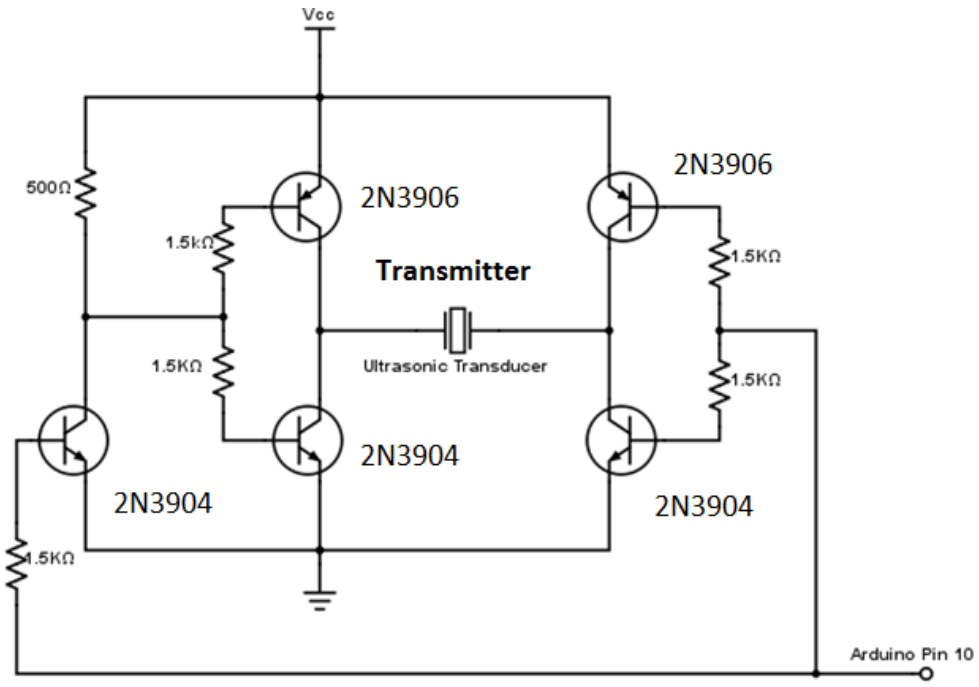

Figure 3: Ultrasonic Sensor Transmitter circuit

In order to increase the output power of the ultrasonic transmitter for a given supply voltage, a bridged circuit consisting of two PNP transistors (2N3906) and three NPN transistors (2N3904) is used. This circuit doubles the input voltage, Vcc (5v) from the Arduino Uno board to 10V Peak to Peak in turn enabling the ultrasonic transmitter to achieve a maximum range of 1 meter (m). The 40 KHz AC voltage required at the input terminals of the ultrasonic transmitter is provided by the Pulse Width Modulation (PWM) signal (Arduino pin 10) generated using Arduino Uno. This signal is then amplified to 10V by the bridge circuit and applied to the input terminals of the ultrasonic transmitter connected across the Collector terminals of the PNP-NPN pair. The applied input voltage causes the transmitter to generate 40 KHz ultrasound waves.

b) Receiver:

The receiver circuit used to interface the 40 KHz ultrasonic receiver to the ATmega328P microcontroller is shown in Figure 4.

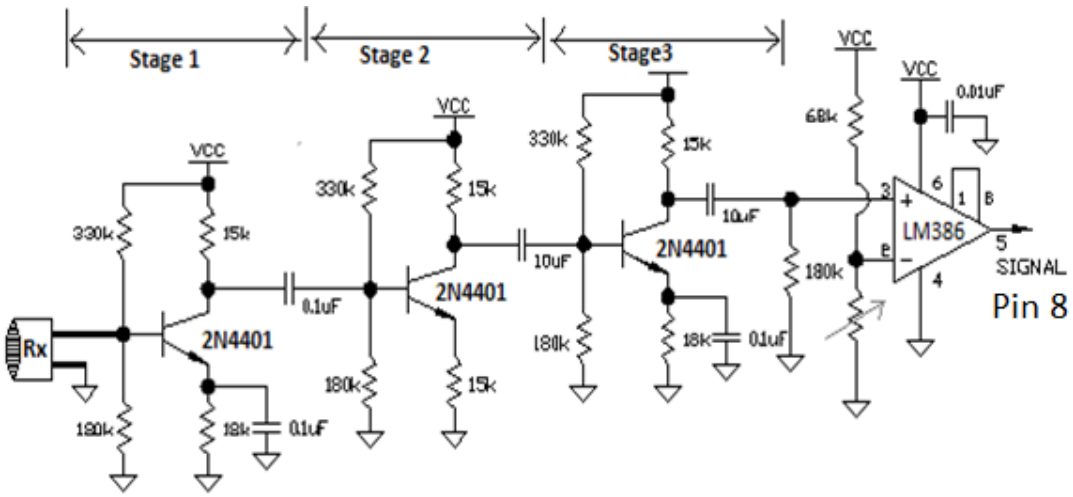

Figure 4: Ultrasonic Sensor Receiver circuit

The receiver circuit is a three stage passive amplifier that is used to amplify the signal from ultrasonic receiver. Each stage is coupled by a capacitor to get rid of the DC component. In the third stage, the amplified signal is coupled through a 10 µF capacitor, and fed into the positive input of the comparator (LM386). This generates a square wave TTL output when the received signal strength is high. The voltage divider at the negative input of the comparator ensures that the

square wave is not generated when the signal strength is low. When 40 KHz ultrasound waves are sensed by the receiver, the signal from the receiver's terminals is amplified by the receiver circuit and a corresponding square wave TTL output is generated at the digital pin 8 of Arduino Uno.

The transmitter and receiver circuits are shown in Figure 5.

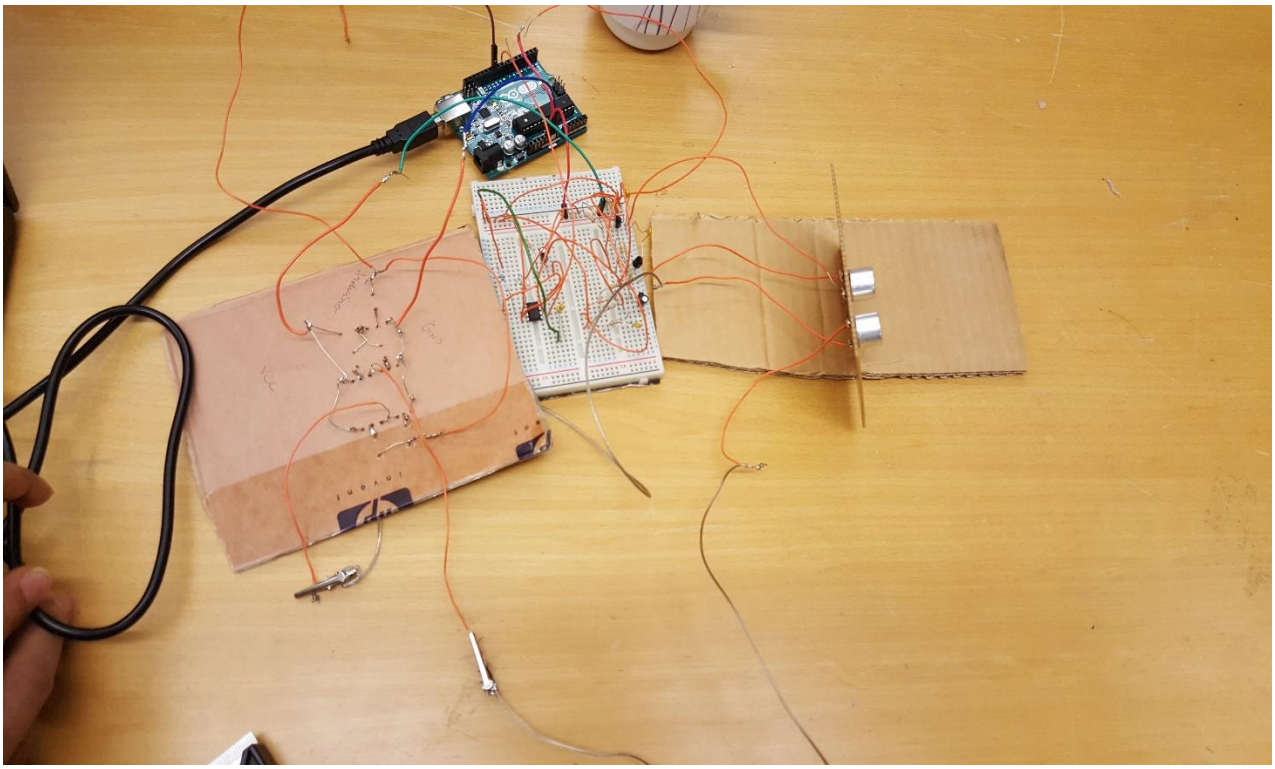

Figure 5: Transmitter and Receiver circuits of the Ultrasonic sensor

C. Software Architecture:

The software architecture of our ultrasonic sensor is illustrated by the flow chart in Figure 6.

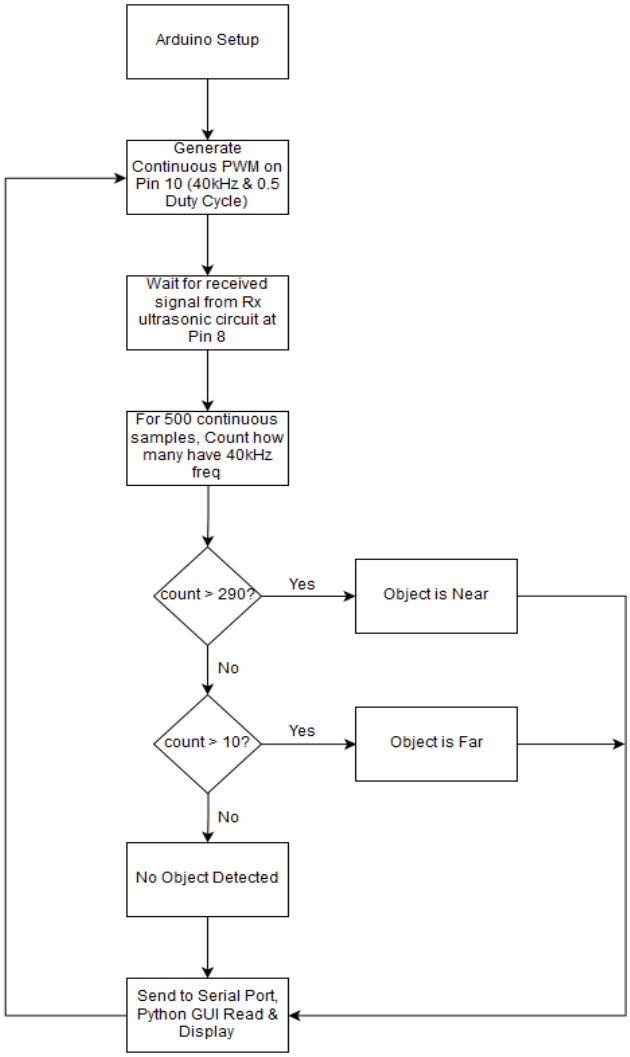

Figure 6: Software Algorithm of Ultrasonic sensor

The first step from the software point of view was to generate the 40 KHz PWM signals using the Arduino Uno. The initial hardware testing was done using a function generator for this purpose. The PWM signal with a duty cycle of 0.5 was generated using the timer/counter 1 of the ATmega328P microcontroller. The 40 KHz square wave from pin 10 of Arduino Uno is amplified and applied to the ultrasonic transmitter which generates 40 KHz ultrasound waves. When an object is placed in front of the transducers, these waves are reflected back and sensed by the receiver. The signal from the output terminals of the ultrasonic receiver are

amplified and the resulting square wave TTL from pin 8 of the Arduino Uno is monitored continuously. About 500 samples are captured continuously and the number of samples around the 40 KHz range are counted. Based on this count value, different messages are printed in the Python GUI on the computer screen. The Arduino code snippet for the object detection logic is shown in Figure 7.

```
void loop()
{
  startTransducer(40000.0, 0.5);

  while (1){
  int cnt500 = 0;
  int cnt40k = 0;
  while(cnt500 < 500) {
  ++cnt500;

  durationH = pulseIn(pin, HIGH);
  durationL = pulseIn(pin, LOW);

  if((durationH+durationL)==0)
    Serial.println("3");//Object not detected
  else
    frq =(1000000/(durationH+durationL));

  if (frq>=37500 && frq<=42000)
    cnt40k++;
    }

  if (cnt40k>290){
  Serial.println("1");}//- Object is near
  else if( ((cnt40k<=290) && cnt40k>=0)){
  Serial.println("2");}//Object is Far
  else{
  Serial.println("3"); //Object not detected
  }
  }
  Serial.flush();
}
```

Figure 7: Arduino code snippet for Object Detection algorithm

The startTransducer function is responsible for generating the PWM using Timer 1 of the ATmega328P microcontroller. The durationH and durationL variables store the duration of square wave at pin 8 of Arduino Uno by using the in-built pulseIn function. From this time period, the frequency is calculated from which the number of 40 KHz samples are obtained and stored in the cnt40k variable. If the count exceeds 290, then a message saying “Object is near” is displayed in the Python GUI on the computer screen, if the count is between 10 and 290, then a message saying “Object is Far is displayed, and if the count is less than 10, then a message saying “Object not detected” is displayed. The output messages are displayed in a GUI designed using Python. So, they are encoded as 1 (near), 2 (far),

and 3 (not detected) and sent to the computer through the serial port where a Python script decodes the messages and prints the text in a GUI window.

D. Testing

The testing of the sensor was done in different stages. Initially, we tested the ultrasonic transducers without any interfacing circuit to test their condition. We connected the ultrasonic transmitter to a Function generator (FG) and the ultrasonic receiver to an oscilloscope. We applied a 40KHz AC using the FG to the transmitter, placed a cardboard in front of the transducers at different distances and observed the output of the receiver in the oscilloscope. The amplitude of the received signal varied with respect to the distance of the object. Then, the transmitter bridge amplifier circuit was tested. The circuit doubled the input voltage from the PWM signal generated by Arduino Uno (from 5V to 10V) as expected. Then, we tested the three stage amplifier receiver circuit and this part consumed time as the circuit worked well when tested with the sinusoidal input from the FG, but failed to work when tested with the PWM input from the Arduino Uno. So, in order to make it work, we had to change the potentiometer resistor value connected to the negative terminal of the comparator (refer to Figure 4) to 3.9Kohms.

After verifying the functionality of the transmitter and receiver circuits separately, the entire sensor that we built was tested with objects of different materials like a bag, a metal sheet and a human being. The test environments containing the module and the objects are shown in Figure 8 and 9.

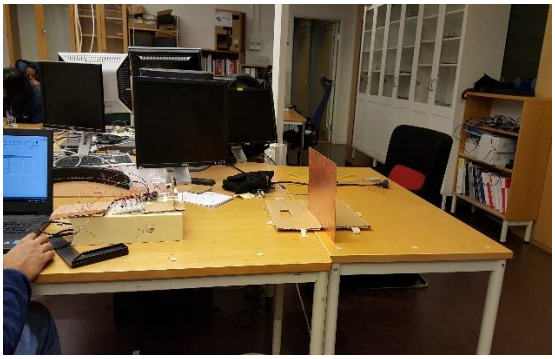

Figure 8: Test with a metal sheet

Figure 9: Test with a bag

For all the three types of objects tested, the accuracy, standard deviation and standard error were calculated. The maximum range achieved by our design was

1m beyond which the accuracy reduces tremendously. The data that describes the performance of our design is shown in Table 1, 2 and 3.

Bag	25cm	50cm	75cm	100cm
Std. Deviation	27,2282533	81,95771267	94,65310936	37,7335689
Std. Error	8,610329714	25,91730439	29,93194132	11,9324022
Accuracy	100%	90%	70%	100%
Error in %	0%	10%	30%	0%

Table 1: Sensor performance when the object is a bag

Metal Sheet	25cm	50cm	75cm	100cm
Std. Deviation	8,126773311	18,9279335	114,0939671	141,3910653
Std. Error	2,569911369	5,985538127	36,07968034	44,7117807
Accuracy	100%	100%	90%	80%
Error in %	0%	0%	10%	20%

Table 2: Sensor performance when the object is a metal sheet

Human	25cm	50cm	75cm	100cm
Std. Deviation	31,28080704	49,10351877	38,08178334	30,4514185
Std. Error	9,891859729	15,52789604	12,04251727	9,629584045
Accuracy	100%	100%	100%	100%
Error in %	0%	0%	0%	0%

Table 3: Sensor performance when the object is a human being

E. Challenges Faced

We faced many challenges during the course of the project. Some of them are;

- The lack of required components during the initial stages. The NPN and PNP transistors that we used intially failed to meet the requirements of our design. After testing, we found that they do not provide the amplification required by our design. Eventually, we bought the components that were needed for our design from Elfa.
- As any electronic circuit building involves the danger of malfunctioning, we ran into problems with our transmitter design that corrupted the Arduino Uno board we started working with, and we had to re-flash the bootloader which consumed some time.
- The band pass amplifier receiver circuit that we tested initially did not work as expected and hence we shifted to a three stage passive amplifier and used the Arduino pulseIn function to capture only the 40 KHz frequencies.
- We soldered the transmitter circuit on a general purpose board, but the receiver circuit was fairly complex and hence we decided to use the breadboard connections for the demo. As a result we were unable to mount the sensor on a walking stick and integrate the vibration sensor for haptic feedback. Instead, we wrote a simple Python GUI script to display a message on the computer screen.

- Characterizing the sensor was a challenging task due to the disturbances and reflections from other objects in the lab.
- There were plenty of tutorials covering the programming (Arduino libraries) of off-the-shelf ultrasonic sensor modules like Ping, HC-SR04 used by the other teams. However, we found very few circuit tutorials for the design of the interfacing circuits.