Lab 04 Report

Vy Nguyen, Jamaal Wairegi, Di Wu

17 February 2023

In this lab, we will explore the methods of Arduino communications with the external environment. To build a proximity detector that emits a sound whose pitch correlates with the distance to a nearby object, we will use sensors, such as an ultrasonic sensor to measure distance, and actuators, such as a buzzer to introduce a sound.

We gain an understanding of these communication devices through two exploratory exercises, and demonstrate our proficiency in organizing code and handling these devices to our liking through a design challenge.

Exploration 1: Buzzer.

In this exploration, we use a buzzer to test frequency levels.

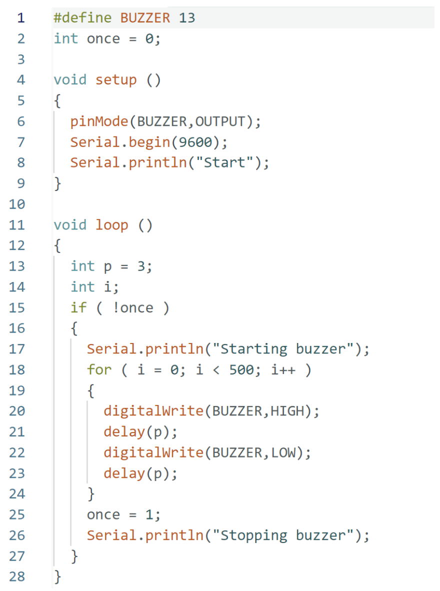
To set up, we need to plug the buzzer onto the breadboard so that the two pins span two different rows. The + pin is connected to the power source while the - pin is connected to ground.

There are in total four tests we need to elaborate for this section.

The first one is to listen to the “click" sound each time we touch or release the +5V source. The struggle in this test is that the volume of the “click" sound is so low that we could not hear it; thus, we spend a bit too much time on this experiment.

In the next experiment, we plug the power connection of the buzzer into the frequency generator of a breadboard. The buzzer makes a sound briefly before stopping. Using the frequency knob and buttons, we can change the frequency of the sound and the time it takes to stop before sounding again. There is not so much of a difference in the outcome of the buzzer when we change the frequency generator but we could barely notice that as the higher the frequency is, the louder and higher pitch of the sound the buzzer makes.

In the two last experiments with the buzzer, we connect to the Arduino and modify the delay time as well as the frequency of the buzzer, using the following code:



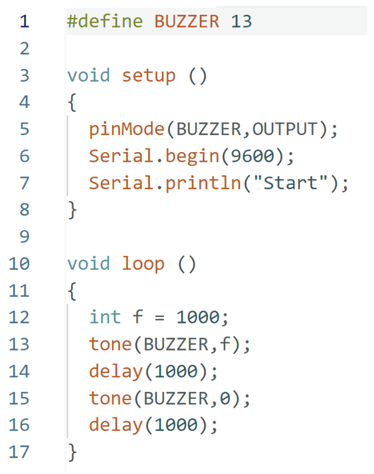
```
#define BUZZER 13
int once = 0;

void setup ()
{
  pinMode(BUZZER,OUTPUT);
  Serial.begin(9600);
  Serial.println("Start");
}

void loop ()
{
  int p = 3;
  int i;
  if ( !once )
  {
    Serial.println("Starting buzzer");
    for ( i = 0; i < 500; i++ )
    {
      digitalWrite(BUZZER,HIGH);
      delay(p);
      digitalWrite(BUZZER,LOW);
      delay(p);
    }
    once = 1;
    Serial.println("Stopping buzzer");
  }
}
```

We make the buzzer sound two different sounds: a high frequency and a low frequency, one after the other. The variable ***p*** controls the time it takes for one sound to transition to the other–a higher value for ***p*** will mean each sound plays for longer, and vice versa.

Not only can we change the time it takes for a tone to play, but we can also change the frequency. Using the code below:



```
#define BUZZER 13

void setup ()
{
  pinMode(BUZZER,OUTPUT);
  Serial.begin(9600);
  Serial.println("Start");
}

void loop ()
{
  int f = 1000;
  tone(BUZZER,f);
  delay(1000);
  tone(BUZZER,0);
  delay(1000);
}
```

We can configure a frequency for the buzzer to play. The variable f controls the frequency. Below is a table of values for f and a description of the tone the buzzer plays respectively:

Frequency	**Tone Description**

50 Hz	A series of low frequency clicks
100 Hz	Low, droning sound
500 Hz	A medium buzzing sound
1,000 Hz	A loud buzzing sound
3,000 Hz	Similar to a police or ambulance siren
10,000 Hz	The sound becomes very high and irritating
20,000 Hz	The sound begins to go out of range

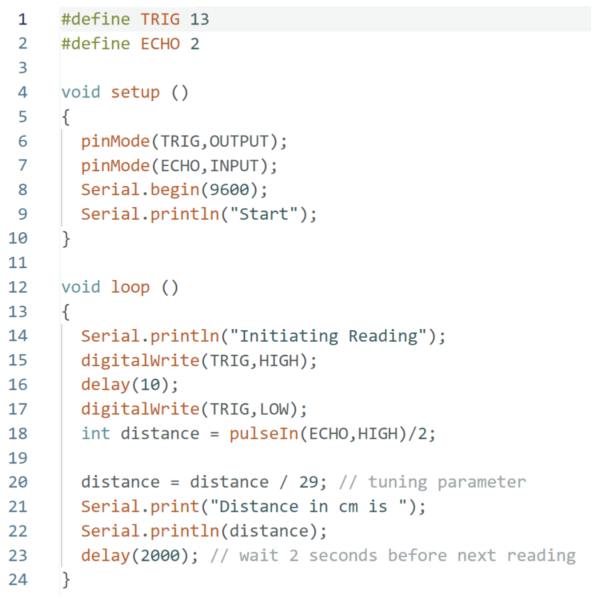
This exploration showcased the capabilities of an Ardunio and a buzzer. Using certain parameters in our Arduino code to manipulate the frequency and timing of the buzzer's sounds.

In this experiment, we encountered a difficulty such that we could not tell much about the tone of the buzzer once we changed the frequency in the program. If we had known more about the tone and pitch, we could have given a better description.

<u>**Exploration 2: Ultrasonic Sensor.**</u>

In this exploration, we use an ultrasonic sensor, a device that measures distance through sonar methods.

Connecting a sensor to an Arduino and running the following code:



```
#define TRIG 13
#define ECHO 2

void setup ()
{
  pinMode(TRIG,OUTPUT);
  pinMode(ECHO,INPUT);
  Serial.begin(9600);
  Serial.println("Start");
}

void loop ()
{
  Serial.println("Initiating Reading");
  digitalWrite(TRIG,HIGH);
  delay(10);
  digitalWrite(TRIG,LOW);
  int distance = pulseIn(ECHO,HIGH)/2;

  distance = distance / 29; // tuning parameter
  Serial.print("Distance in cm is ");
  Serial.println(distance);
  delay(2000); // wait 2 seconds before next reading
}
```

We are able to see in the serial monitor output display what distance the sensor gets. In the range of about 5 to 30 centimeters, the sensor was accurate. However, any distance greater or less than that range becomes less accurate. This is due to the sensor picking up other objects as we move an object away from it.

There is one parameter we can change in our code. The ***delay()*** function, which takes in an integer and delays the actions of the Arduino by the given amount. Decreasing the value will give us results for the distance more often, and vice versa.

This exploration demonstrates the abilities and limits the Arduino has when using sensors.

Exploration 2: Ultrasonic Sensor.

In this exploration, we use a photoresistor, a device that measures light brightness.

For the first part of this experiment, we need to use the multimeter to read the value of the photoresistor when we darken it or add extra light onto it. We ended up with a table below.

Lighting Conditions	**Measurement (kΩ)**
Completely Covered	

Partially Covered (cupped hand over top)	
Pointing to the floor	
Ambient lit room	
With flash light on face (I used my iPhone light)	

In the next section, we need to connect the photoresistor to the ground and a 1k resistor, while the other end of the resistor is connected to the +5V power source. We still read it from the multimeter by connecting the black lead to ground and the red lead on the positive side of the photoresistor. We end up with a table of values recorded below.

Lighting Conditions	**Measurement (kΩ)**
Completely Covered	
Partially Covered (cupped hand over top)	
Pointing to the floor	
Ambient lit room	
With flash light on face (I used my iPhone light)	

The one problem came up during this experiment is that the 1k resistor is a bit too small for us to read the value. In order to move forward with the experiment, we need to change to a resistor with a higher value such as 5kΩ.

In the last test, we connect the positive end of the photoresistor (where it connects to the 1k resistor) to the Arduino analog input 0 and run the following program to observe the results in the serial window as we light on the photoresistor.

```
#define V_PIN 0
void setup ()

{

Serial.begin(9600);
pinMode(V_PIN,INPUT);

}

void loop ()

{

int val = analogRead(V_PIN);

Serial.println(val);

delay(1000);

}
```

Lighting Conditions	**Measurement (kΩ)**
Completely Covered	
Partially Covered (cupped hand over top)	
Pointing to the floor	
Ambient lit room	
With flash light on face (I used my iPhone light)	

<u>Design Challenge: Distance Detector.</u>

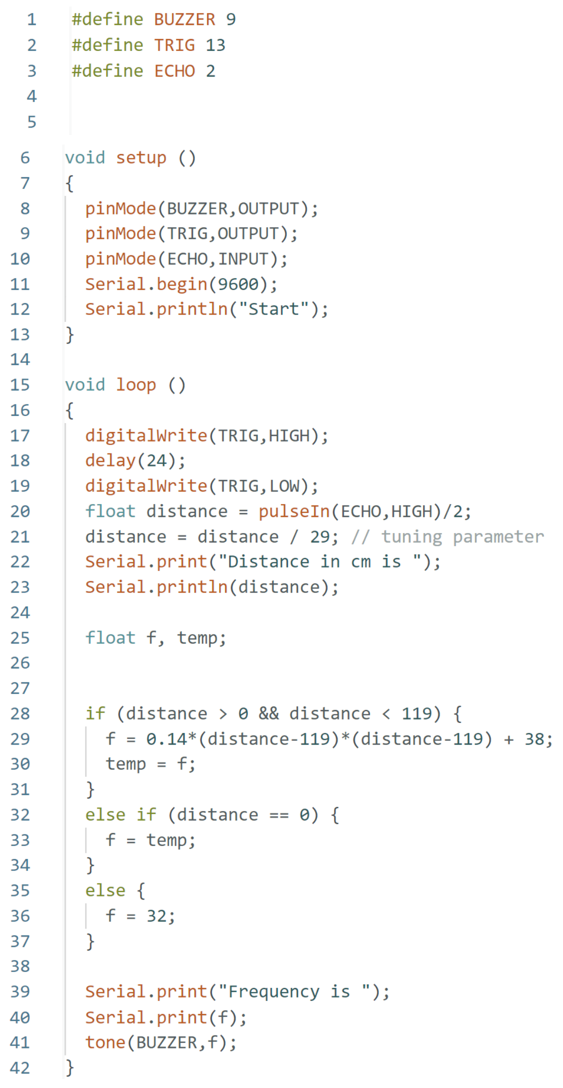
Using what we learned in the two explorations, we will design a distance detector. It will consist of an ultrasonic sensor and a buzzer.

For this design, we chose the distance detector because we think that it is so difficult for us to control the amount of light in a room.

The ultrasonic sensor will read an object's distance, then the buzzer will use that distance as an input for a function we have created ourselves to output a frequency for an appropriate tone to play. When an object is close, the frequency will become higher, and vice versa.

The goal of our design challenge is to create a distance detector that immediately responds to a given distance with a tone using a frequency function that allows for a range of tones.

Here is the code we used to implement these features:

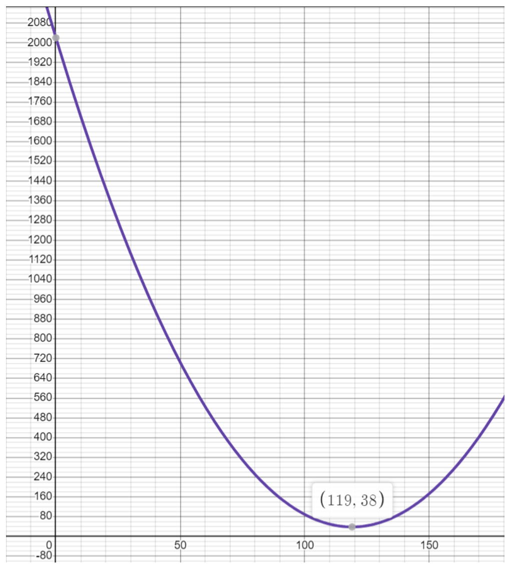


```
 1  #define BUZZER 9
 2  #define TRIG 13
 3  #define ECHO 2
 4
 5

 6  void setup ()
 7  {
 8    pinMode(BUZZER,OUTPUT);
 9    pinMode(TRIG,OUTPUT);
10    pinMode(ECHO,INPUT);
11    Serial.begin(9600);
12    Serial.println("Start");
13  }
14
15  void loop ()
16  {
17    digitalWrite(TRIG,HIGH);
18    delay(24);
19    digitalWrite(TRIG,LOW);
20    float distance = pulseIn(ECHO,HIGH)/2;
21    distance = distance / 29; // tuning parameter
22    Serial.print("Distance in cm is ");
23    Serial.println(distance);
24
25    float f, temp;
26
27
28    if (distance > 0 && distance < 119) {
29      f = 0.14*(distance-119)*(distance-119) + 38;
30      temp = f;
31    }
32    else if (distance == 0) {
33      f = temp;
34    }
35    else {
36      f = 32;
37    }
38
39    Serial.print("Frequency is ");
40    Serial.print(f);
41    tone(BUZZER,f);
42  }
```

In lines 17 to 23, we retrieve and print the distance read from the ultrasonic sensor. Then, in lines 25 to 41, we process this distance into a function for the frequency of the buzzer, then sound the buzzer.

The function is $0.14(distance\ -\ 119)^2\ +\ 38$. This function gives us a good range of values that will allow us to create distinct sounds for each distance measured. Below is a visualization of the graph:



Although the graph shows that the frequency seems to begin to rise as the distance increases, the if-else statements in lines 28 to 37 help to mitigate that.

If the distance is between 0 and 119 (centimeters), the frequency function will be used to compute the tone. If the distance is zero, no tone is played, as no object is found on the sensor. And if the distance is any other value than 0 through 119, a frequency of 32 will be played on the buzzer, which is just a very low sound.

We also did not include any use of the ***delay()*** function. This way, we would have immediate feedback from the sensor and buzzer

The end result was a distance detector with a wide range of tones and immediate feedback. The detector was very accurate at smaller distances; however, larger distances were not very accurate due to the sensor picking up other stray objects. We mitigated this “noise” picked up by the sensor by using the aforementioned if-else statements to set the frequency to a constant, so that our function could handle the more accurate measurements.

In this challenge, we encountered more difficulty than we thought.

First, we could not figure out a proper function between the distance and frequency of sound as we wanted. Thus, we decided to use a bunch of conditional statements to modify this. However, this results in an abrupt change in pitch when we move the object back and forth. Thus, we came up with a new function that allows us to modify the outcome sound as we wanted.

Second, we did not pay attention at the delay time in the program so the sound responding to the distance of the object is not so rapid. We then decided to remove the delay time and it worked as a charm!

<u>**Conclusion.**</u>

In this lab, we explored the abilities of two different communication devices, a buzzer and an ultrasonic sensor, and used them to design a distance detector that responds to distances picked up by a sensor with tones from a buzzer with frequencies derived from the given distance.

Not only have we demonstrated understanding in the devices we researched, we have also shown proficiency in designing and testing a complex device that receives external data, manipulates that data for internal processes, then uses that internal result to produce another external activity.