William Clemons
Nathan Nhek
Instructor Heckathorn
CS 440-03

Goal

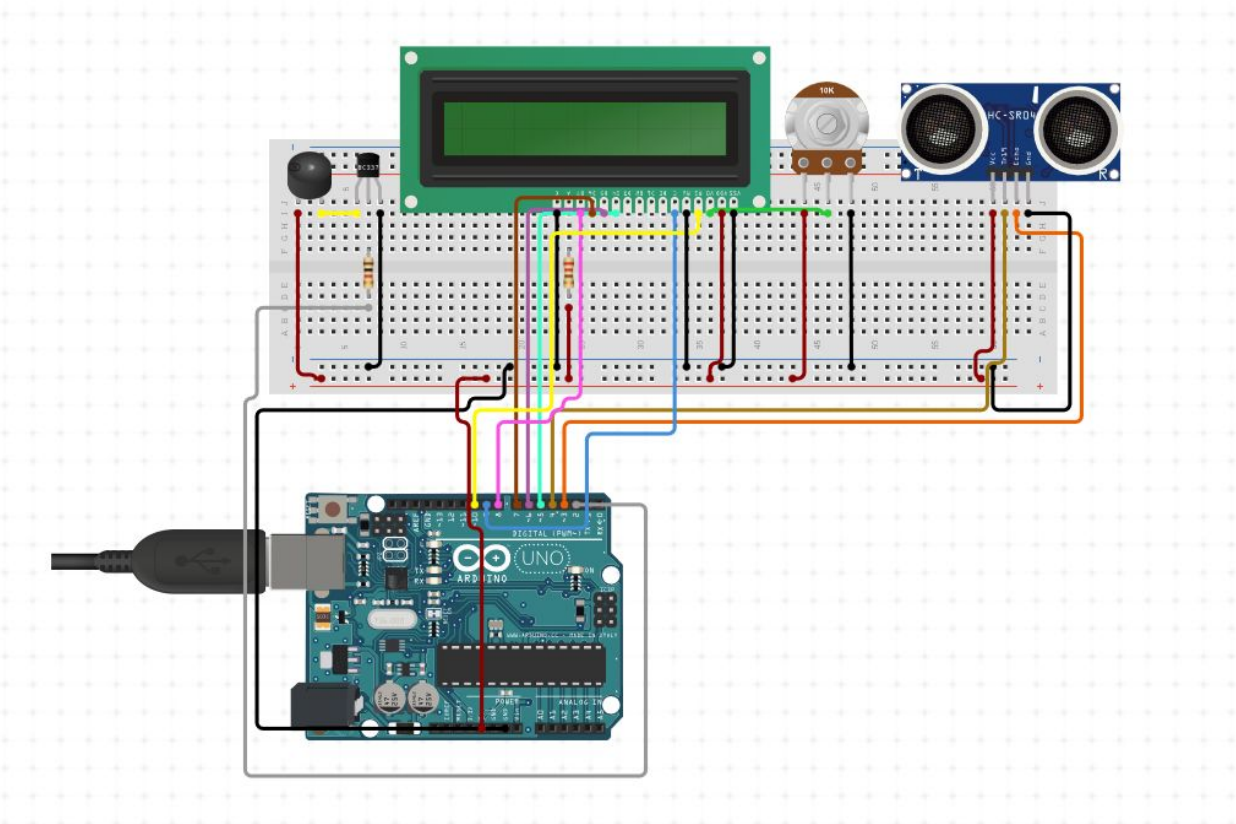
Our goal with this project was to create an alarm that will beep when it detects something within a specified range of the arduino. The idea behind this is that the arduino can act as an alarm that can alert the user when someone approaches a protected object/place, near the arduino.

Hardware Components

- (1) x Elegoo Uno R3
- (1) x USB cable
- (1) x LCD1602 module
- (1) x Potentiometer (10k)
- (1) x 830 tie-points Breadboard
- (1) x Ultrasonic sensor module
- (1) x Passive buzzer
- (1) x Transistor - PN2222
- (1) x 1k ohm resistor
- (1) x 220 ohm resistor
- (26) x M-M wires (Male to Male jumper wires)

Elegoo Uno R3 is the arduino model used and a USB cable is connected to power everything.
LCD1602 is the LCD screen used to display the distance.
A potentiometer is used to provide varying contrast levels (lower = less, higher = more) to the LCD screen.
A 830 tie-points breadboard is used to run wires through, in order to be efficient about space usage and not need to split and daisy chain wires. This allows current to flow easily between wires set up in columns of the breadboard.
An ultrasonic sensor is used to detect the distance of objects it is pointed at, converted into inches for practical measuring.
A passive buzzer is used to generate the alarm once the ultrasonic sensor detects an object too close.
The transistor - PN2222 can allow for a switch to the passive buzzer for current to flow when needed.
The 1k and 220 ohm resistors limit the current flowing to certain parts, as shown in the circuit diagram, in order to keep from having the parts overcharge.
M-M wires are used to connect each component either to the breadboard, or the arduino to get power.

Wiring Setup



NOTE The wiring is actually similar to this, as some modifications were made that cannot be reflected in the diagram above. The overall order of placement is the same but specific row and column locations are slightly different. A big difference is the pin connections to the arduino. In the diagram the yellow wire (RS on the LCD) connects to pin 7, The dark blue wire (EN on the LCD) connects to pin 8. The cyan wire (D4 on the LCD) connects to pin 9. The dark purple wire (D5 on the LCD) connects to pin 10. The dark brown wire (D6 on the LCD) connects to pin 11. The pink wire (D7 on the LCD) connects to pin 12.

Libraries

<LiquidCrystal.h> -- Liquid crystal is used to run the LCD screen.

< HC-SR04.h> -- This library is used to run with the echo pin, the trig pin, and get the distance using the ultrasonic sensor.

“Pitches.h” -- Contains public constants for all the tones used by the passive buzzer.

Code

ultrasensor_console | pitches.h

```
/* William Clemons
 * Nathan Nhek
 *
 * Proximity Alarm
 * Detect nearby objects and set off a buzzing noise when under a certain distance.
 * Set to display the distance to a LCD screen.
 */
// include the library that contains the #defines for all passive buzzer pitches
#include "pitches.h"
// include the library for the LCD screen
#include <LiquidCrystal.h>

// define where the lcd connects to the arduino pins
const int rs = 7, en = 8, d4 = 9, d5 = 10, d6 = 11, d7 = 12;
// init the display
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// defines pins numbers
// trig pin generates the ultrasonic burst
// echo pin recieves waves after bouncing back
const int trigPin = 4;
const int echoPin = 3;

// defines variables
long duration;
int distance;

//array to hold all the needed notes to make the alarm noise
int alarm[] = {
  // note scheme = 1, 2, 2, 3, 2, -, 4, 1
   NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  //set the columns, rows to display on
  lcd.begin(16,1);
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
}
```

```
void loop() {
  //by default, have no tone emitting
  noTone(2);

  // Clears the trigPin by setting it to 0
  digitalWrite(trigPin, LOW);
  // set a delay of 2 ms
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds (v = 340 m/s -> .034cm/s)
  duration = pulseIn(echoPin, HIGH);

  // Calculating the distance (time = distance / speed = [d/.034cm/s])
  // solve for distance; distance = time*.034/2 (divide by 2 because the echo pin doubles the sound wave
  // the echo pin doubles the sound wave because it travels out from the arduino, and then back to the arduino)
  // distance here is in inches (1 cm = 0.393701 inches)
  distance = duration*0.034/2*0.393701;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.print(" inches");
  Serial.println();

  // print to the LCD screen and add a delay after so it's not overwhelmed
  lcd.clear();
  lcd.print("Distance: ");
  lcd.print(distance);
  lcd.print("in");
  delay(100);

  // If something is detected within 12 inches, start the alarm
  if (distance < 12) {
    // iterate over the notes of the alarm:
    for (int thisNote = 0; thisNote < 8; thisNote++) {

      // to calculate the note duration, take one second
      // divided by the note type.
      //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
      int noteDuration = 1000 / noteDurations[thisNote];
      // tone takes in pin #, alarm noise pitch, and duration of note
      tone(2, alarm[thisNote], noteDuration);

      // to distinguish the notes, set a minimum time between them.
      // the note's duration + 30% seems to work well:
      int pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);
      // stop the tone playing:
      noTone(2);
    }

  }
}
```

ultrasensor_console | pitches.h

```
//contains all the notes needed for pitches used by passive buzzer, defined as public constants
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
```

```
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

Limitations

Originally, the idea behind this project was to include a lighting system that would cause a light to turn on when something approaches it. However, as time went on, our idea for the project changed. With the current hardware we have on the breadboard, there is not enough room to fit an LED light, nor on the pins on the arduino.

Furthermore, due to the nature of the ultrasonic sensor, it only detects what is in front of it, so it can be circumvented if something were to come up to it from behind the sensor. This means the alarm system will only protect from the angle to ultrasonic sensor is pointing at.

Future Work

In the future, an additional breadboard could be attached to the existing one, to allow for more modifications, and at the very least, the addition of an LED light, so that the alarm can give a visual representation of an alert.