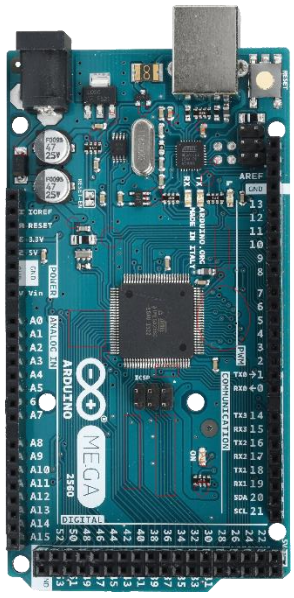
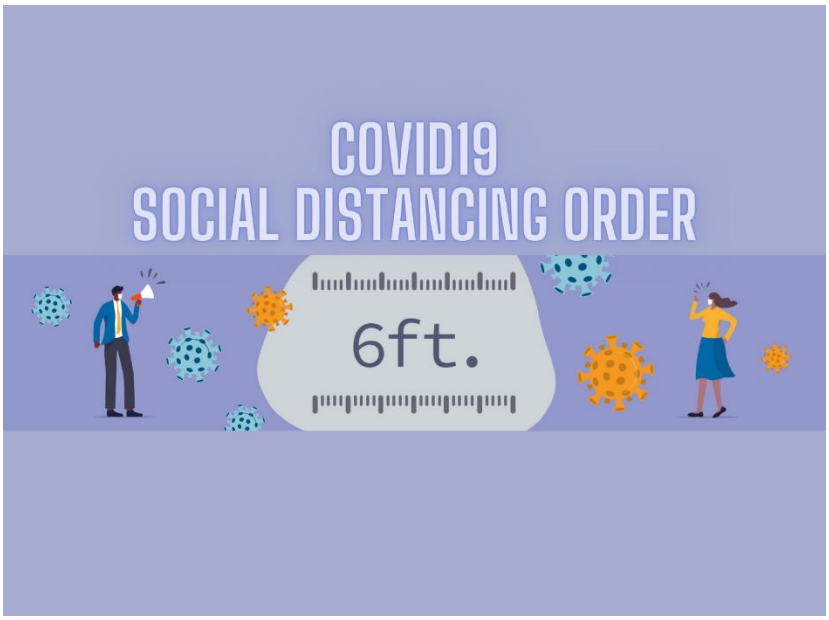
Final Project | Team 24





Electrical and Computer Engineering 5

Coronavirus Social Distancing Simulation using Arduino

Developed by Muhammad Arsani, Nada Ali, Ankeen Arestakesyan

What You Will Need

Materials:

- 1 Arduino Board
- 10 Jumper Wires
- 1 Ultrasonic Sensor
- 1 Piezo Buzzer
- 1 I^2C LCD screen (comes with potentiometer)
- 1 Breadboard
- 1 Button
- 1 LED
- 1 Resistor (330 Ω)

Machinery:

- Computer/Laptop

Software:

- Arduino IDE
- LiquidCrystal_ PCF8574.h Library

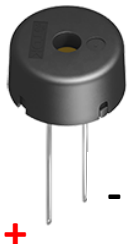
Challenge #1: Piezo Buzzer and Button

Objective

In this challenge, you will build a circuit that triggers the piezo buzzer to beep every time the button is pushed.

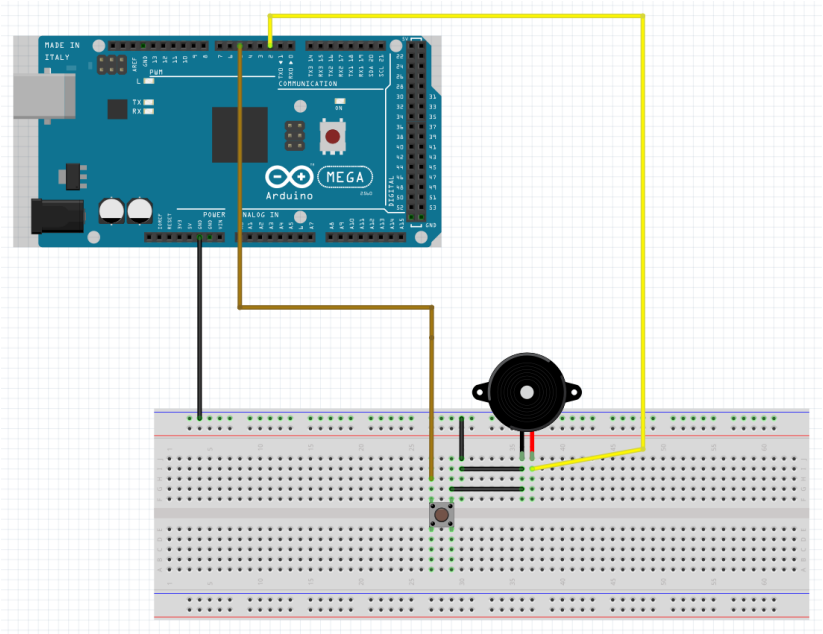
Components

- Arduino
- 1 Piezo Buzzer
- 5 (Approx.) Jumper Wires
- 1 Button
- Breadboard



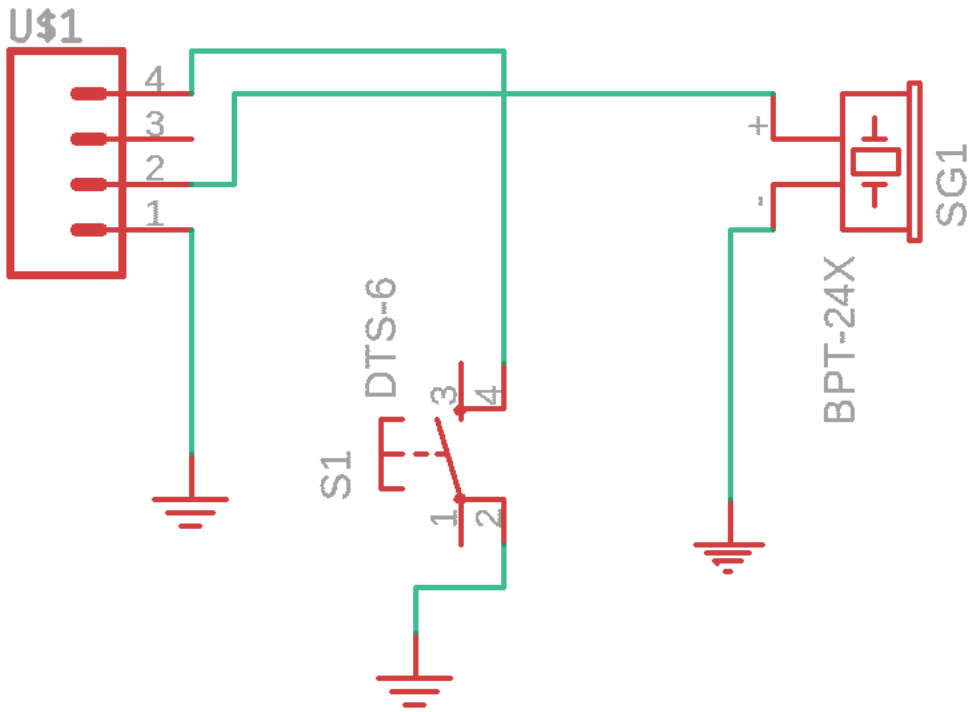
In this challenge, you will work with INPUT_PULLUP in the pinMode() function instead of the INPUT function we used to work with. It monitors the state of a switch by establishing serial communication between your Arduino and your computer over USB.

Circuit Diagram

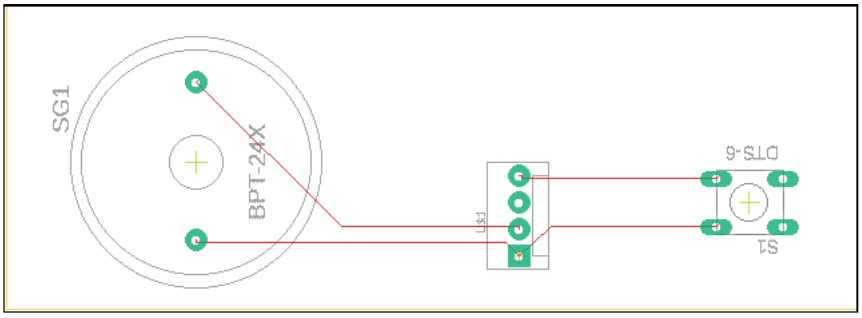


Note: Make sure to check the connections for the button, it's very easy to overlook!

Schematics



PCB



Task

- Make sure the buzzer will only beep when the button is pushed
- Make sure the buzzer does not beep when the button is not pushed.

Code

```
const int Button_Pin = _; // the arduino pin that is connected to the Button
const int Buzzer_Pin = _; // the arduino pin that is connected to the Buzzer

void setup()
{
  Serial.begin(9600);
  pinMode(Button_Pin, _____);
  // As you can see, we use the INPUT_PULLUP as the argument of the pinMode function instead of INPUT
  // Also, remember about the input pullup function mentioned in this challenge's explanation
  pinMode(Buzzer_Pin, _____);
  // What about the buzzer, should it be input or output?
}

void loop()
{
  int _____ = digitalRead(Button_Pin); // declare a variable for the state of our button

  if (_____ == LOW) // what should happen to the button state if we are pressing the button?
  {
    _____("You are pressing the button, sir");
    // print a message on the serial monitor to indicate that you are pressing the button
    digitalWrite(_____, HIGH);
  }
  else if (_____ == HIGH) // what should happen to the button state if we are not pressing the button?
  {
    _____("You have lifted your finger from the button, sir.");
    // print a message on the serial monitor to indicate that you are not pressing the button
    digitalWrite(_____, LOW);
  }
}
```

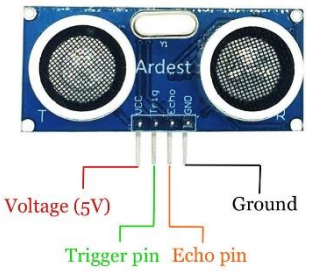
Challenge #2: Piezo Buzzer and Ultrasonic Sensor

Objective

In this challenge, you will build a circuit from Challenge 1, replacing the button with an ultrasonic sensor. Your buzzer should beep when there is an object within the threshold distance of the Ultrasonic Sensor You will learn how Ultrasonic sensor works and how do we program it using Arduino built in functions.

Components

- Arduino
- 1 Piezo Buzzer
- 8 (Approx.) Jumper Wires
- 1 Ultrasonic Sensor
- Breadboard

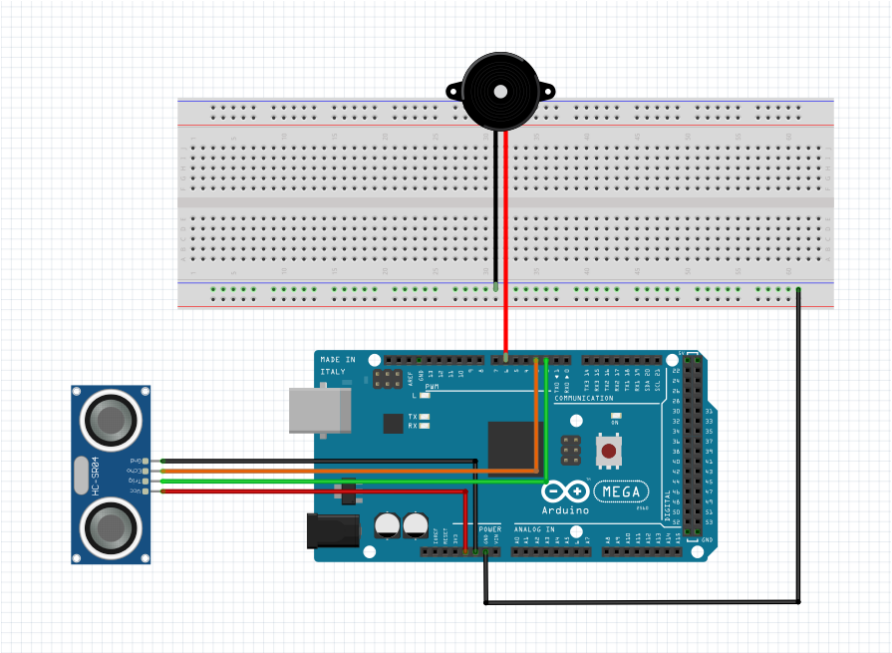


The Ultrasonic sensor has 4 pins, the voltage input (5V), the Trigger pin, Echo pin, and the Ground pin (0V). Ultrasonic sensors work by emitting sound waves at a frequency too high for humans to hear. They then wait for the sound to be reflected back, calculating distance based on the time required. This is similar to how radar measures the time it takes a radio wave to return after hitting an object. Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave, and we do this in our code by using the *delayMicroseconds(us)* function. Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor, and we achieve this using the *pulseIn(pin, value, timeout) function.*

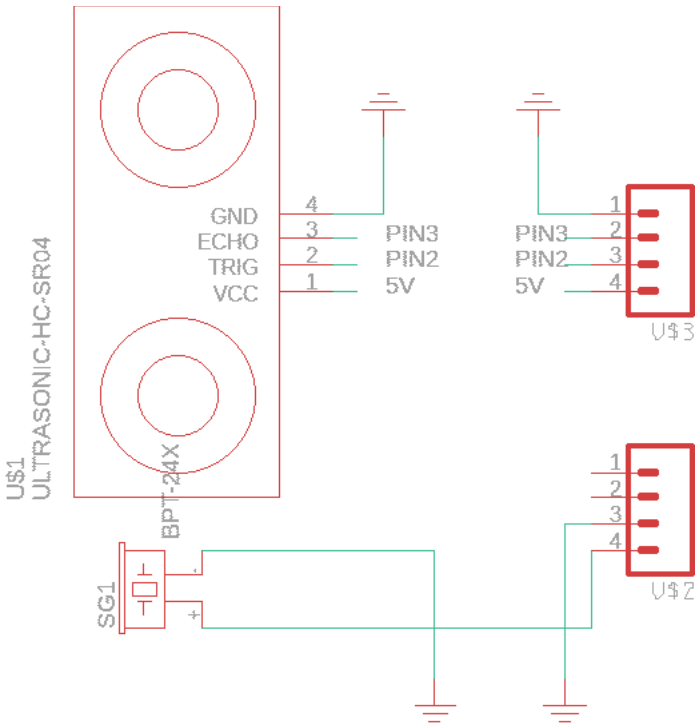
delayMicroseconds(us) function (same as *delay* function) pauses the program for the amount of time (in microseconds) specified by the parameter. As you might notice, it has "*us*" as the argument in the parameter (recall what you learned in Stepik, Intro to Python module, on what it means by parameters and arguments in a function). The argument "*us*" specifies the number of microseconds to pause. We will set this to 10us to initialize measurement by sending wave.

pulseIn(pin, value, timeout) function reads the pulse on a *pin*, *HIGH* or *LOW*. If the *value* is *HIGH*, this function waits for the *pin* to go from *LOW* to *HIGH*, starts timing, then waits for the pin to go *LOW* and stops timing. It will return the length of pulse in microseconds or it will return 0 if no complete pulse was received within the timeout. The *timeout* argument is optional, but you need to specify the pin on which you want to read the pulse, and the value (HIGH or LOW). The *timeout* argument is the number of microseconds to wait for the pulse to start. If you don't specify anything, it will take the value 1 second by default.

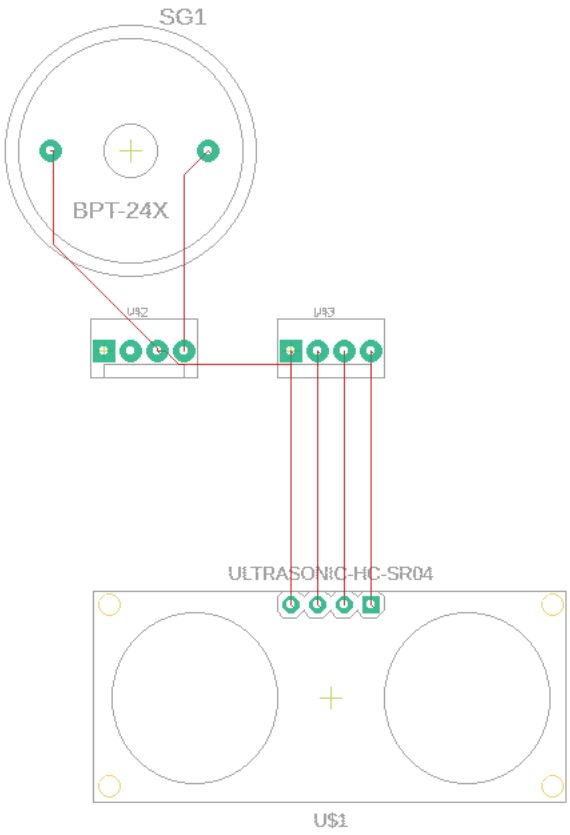
Circuit Diagram



Schematics



PCB



Task

- Make sure the buzzer will only beep when the ultrasonic sensor detects an object within the distance threshold
- Make sure the buzzer does not beep when there is no object within the distance threshold
- Keep track of the distance of your object from the ultrasonic sensor by printing the values to the serial monitor

Code

```
const int Trig_Pin    = _; // the arduino pin that is connected to the Ultrasonic Sensor's TRIG pin
const int Echo_Pin    = _; // the arduino pin that is connected to the Ultrasonic Sensor's ECHO pin
const int Buzzer_Pin = _; // the arduino pin that is connected to the Piezo Buzzer's pin
const int Distance = __; // this is the minimum/threshold distance, in centimeters, that will trigger the buzzer

float duration_us, distance_cm;
//duration of the pulse measured by the ultrasonic sensor
//distance of object from the ultrasonic sensor

void setup()
{
  Serial.begin (9600);          // initialize serial port
  pinMode(Trig_Pin, _______);   // think about whether the Trig pin of the Ultrasonic Sensor should be input or output
  pinMode(Echo_Pin,_______);    // think about whether the Echo pin of the Ultrasonic Sensor should be input or output
  pinMode(Buzzer_Pin, _______); // think about whether the Buzzer pin of the Ultrasonic Sensor should be input or output
}

void loop()
{
  // generate 10-microsecond pulse to TRIG pin
  digitalWrite(Trig_Pin, ____);
  delayMicroseconds(10);
  digitalWrite(Trig_Pin,____);

  // measure duration of pulse from ECHO pin
  duration_us = pulseIn(Echo_Pin, HIGH);
  // calculate the distance
  distance_cm = 0.017 * duration_us;

  if(distance_cm < Distance)
  {
    digitalWrite(Buzzer_Pin, ____); // here, what should be the condition of the Piezo Buzzer, on or off?
  }
  else
  {
    digitalWrite(Buzzer_Pin,____);  // here, what should be the condition of the Piezo Buzzer, on or off?
  }

  // print the distance values to Serial Monitor
  Serial.print("distance: ");
  Serial.print(____________);
  Serial.println(" cm");

  delay(500);
}
```

Check out these links if you want to learn more about the functions we used in this challenge!
pulseIn() - https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/?setlang=it
delayMicroseconds() - https://www.arduino.cc/reference/en/language/functions/time/delaymicroseconds/

Challenge #3: Piezo Buzzer, LED, Ultrasonic Sensor

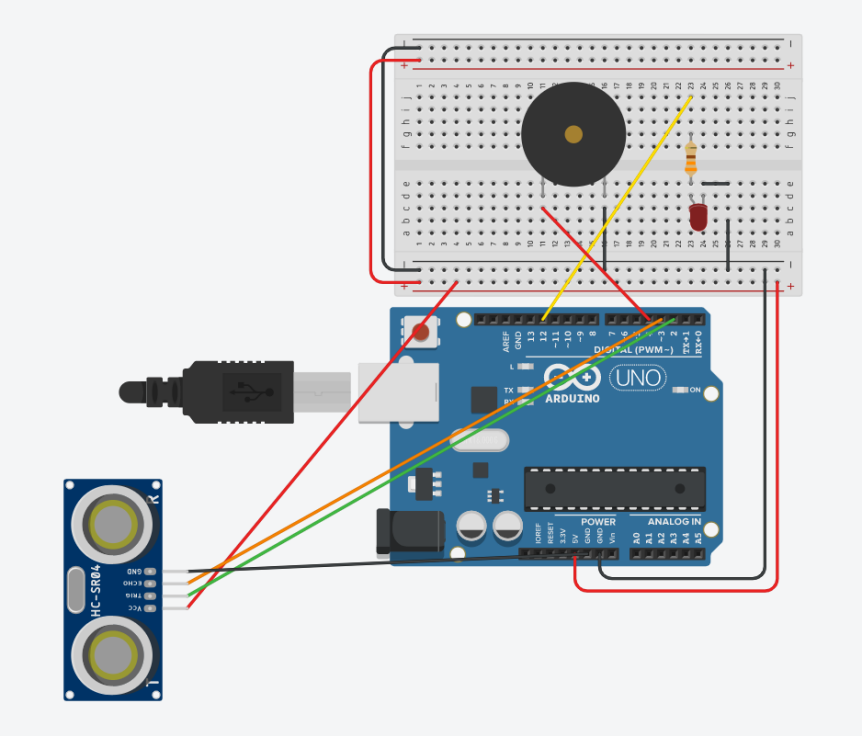
Objective

Yes, you guessed it right! In this challenge, you will build a circuit that turns on the LED when the buzzer beeps due to the Ultrasonic Sensor sensing an object within the distance threshold.

Components

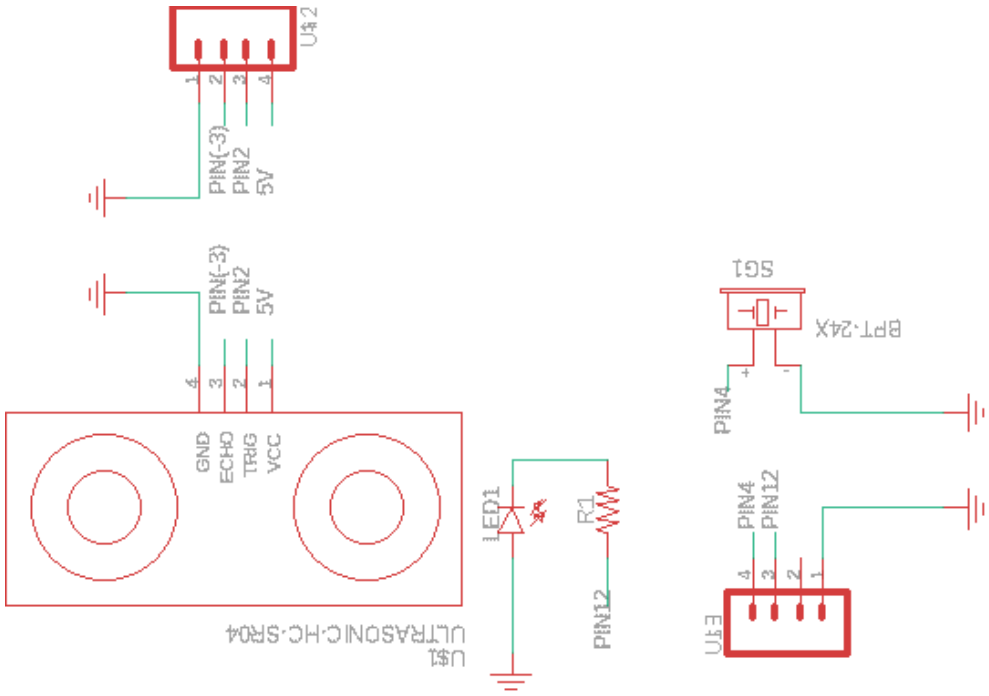
- Arduino
- 1 Piezo Buzzer
- 8 (Approx.) Jumper Wires
- 1 Ultrasonic Sensor
- 1 LED
- 1 Resistor (330 Ω)
- Breadboard

Circuit Diagram

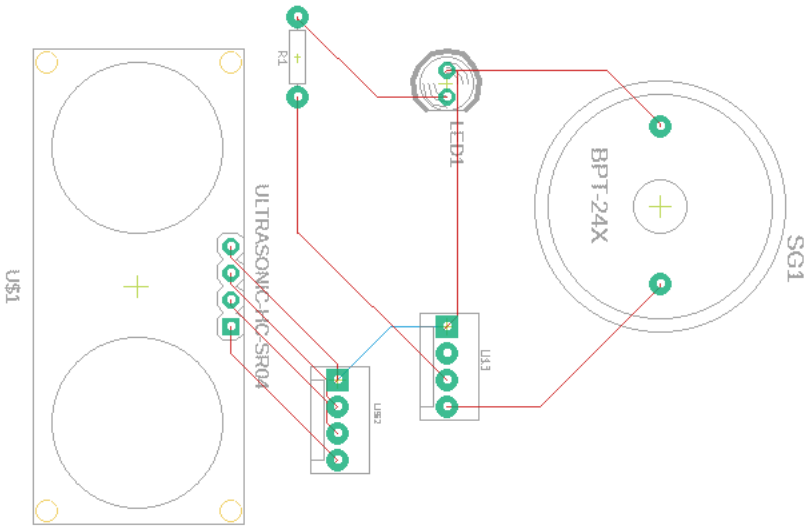


Note: Make sure your LED is properly connected to the circuit. Remember, longer leg is the positive terminal!

Schematics



PCB



Task

- Make sure the buzzer will only beep when the ultrasonic sensor detects an object within the distance threshold
- Make sure the buzzer does not beep when there is no object within the distance threshold
- Keep track of the distance of your object from the ultrasonic sensor by printing the values to the serial monitor
- The LED should light up every time the buzzer is beeping, and it should turn off when the buzzer is not beeping.

Code

```
const int Trig_Pin    = __; // the arduino pin that is connected to the Ultrasonic Sensor's TRIG pin
const int Echo_Pin    = __; // the arduino pin that is connected to the Ultrasonic Sensor's ECHO pin
const int Buzzer_Pin = __; // the arduino pin that is connected to the Piezo Buzzer's pin
const int Distance = __; // this is the minimum/threshold distance, in centimeters, that will trigger the buzzer
const int LED_Red = __; // the arduino pin that is connected to the LED pin

float duration_us, distance_cm;
//duration of the pulse measured by the ultrasonic sensor
//distance of object from the ultrasonic sensor

void setup()
{
  Serial.begin (9600);          // initialize serial port
  pinMode(Trig_Pin, _______);    // think about whether the Trig pin of the Ultrasonic Sensor should be input or output
  pinMode(Echo_Pin,_______);     // think about whether the Echo pin of the Ultrasonic Sensor should be input or outpu
  pinMode(Buzzer_Pin, _______); // think about whether the Buzzer pin of the Ultrasonic Sensor should be input or outpu
}

void loop()
{
  // generate 10-microsecond pulse to TRIG pin
  digitalWrite(Trig_Pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig_Pin, LOW);

  // measure duration of pulse from ECHO pin
  duration_us = pulseIn(Echo_Pin, HIGH);
  // calculate the distance
  distance_cm = 0.017 * duration_us;

  if(distance_cm < Distance)
  {
    digitalWrite(Buzzer_Pin, ____); // here, what should be the condition of the Piezo Buzzer, on or off?
    digitalWrite(LED_Red, ____); // what about the LED? should it turn on when the buzzer beebs? Read the task carefully ;p
    // p/s: make sure you are using the right resistor, or the light from the LED won't be that visible.
  }
  else
  {
    digitalWrite(Buzzer_Pin,____);  // here, what should be the condition of the Piezo Buzzer, on or off?
    digitalWrite(LED_Red,____); // Sometimes, you just need to try your luck and see if it works ;)
  }
  // print the distance values to Serial Monitor
  Serial.print("distance: ");
  Serial.print(____________);
  Serial.println(" cm");

  delay(500);
}
```

Challenge #4: LCD Screen

Objective

In this challenge, you will build a circuit that prints out anything you want on the LCD screen display. You will learn about the LiquidCrystal_PCF8574.h library and the built-in Wire.h library, how LCD screen works, and how to set it up before you are able to program it. You will also learn about the SDA and SCL pins on our Arduino as you will be working with these pins from now on.

Components

- Arduino
- 4 (Approx.) Jumper Wires
- 1 I^2C LCD screen
- Breadboard

Software:

- LiquidCrystal_PCF8574.h Library
- Wire.h Library
- Matheretel LCD screen test code from GitHub to initialize the LCD https://github.com/mathertel/LiquidCrystal_PCF8574/blob/master/examples/LiquidCrystal_PCF8574_Test/LiquidCrystal_PCF8574_Test.ino

You have learned in Lab 1 about libraries and how we can download it, enabling us to use the functions from the libraries in our code. Again, this is what Lab 5 is all about: Integrating all the knowledge you have gained in previous Lab assignments! Nevertheless, don't worry, this challenge will still walk you through the steps!

First, run Arduino IDE. You will need to go to Sketch > Manage Library. You should be able to see a small window that says Library Manager. Next, go to the search bar and search LiquidCrystal_PCF8574 (you can just copy and paste this on the search bar too). Then, simply click install and choose the latest version.

Before we can begin experimenting the LCD screen, we first need to know how it works. There are tons of LCD screen that works great with Arduino you can find online on websites like Amazon. However, not all of these LCD's will work under the same code/program. Here's why: Different LCD has different drivers in it, meaning that it operates using different libraries. Some might even need different test code than what we are using. So, our code will not be compatible with all LCD screens, alterations to the code might be needed. In this challenge, we are using the 12C LCD screen which you can find here: https://www.amazon.com/gp/product/B071FGZX8G/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1

Now let's learn about the working principles of our LCD screen. Most LCD screens have a number of pins: **Register Select (RS) pin** that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which

is where the LCD's controller looks for instructions on what to do next. **Read/Write (R/W) pin** that selects reading mode or writing mode. **Enable Pin** that enables writing to the registers. **8 Data Pins (D0 -D7)**. The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read. **Display Contrast Pin (Vo)**, **Power Supply Pins (+5V and Gnd)** and **LED Backlight (Bklt+ and BKlt-)** pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively. *(you can read more here: https://www.arduino.cc/en/Tutorial/LibraryExamples/HelloWorld)*

Well, THOSE ARE A LOT OF PINS RIGHT!? That is why we use I^2C LCD in this project to make things easier and simpler. With this LCD, you will only need to work with 4 pins: **Input Pin (5V), Ground Pin (0V), SDA Pin, SCL Pin.** The SDA pin is an I2C pin. It handles all the data pins in our LCD, so we don't have to work with 8 data pins and get our breadboard messy. SCL is the clock line, it is used to synchronize all data transfers in all circuits that use I^2C bus protocols.

We have been talking about I^2C a lot in this challenge, but what is I^2C actually? I^2C, created by Philips Semiconductor in 1982, stands for "Inter-integrated circuit," or "inter-IC," and is a simple, 8-bit, serial communication bus protocol that uses just two bus wires; a **serial data wire (SDA)** and **a serial clock wire (SCL)**. I^2C is integrated into many ICs and allows devices to communicate directly with each other, avoiding CPU cycles. I^2C operates on a master-slave basis, and all devices on an I^2C bus have a unique address. I^2C is used both internally in integrated chips to communicate between areas in the chip-based circuit, as well as externally, from chip to chip. You can read more about I^2C here: https://www.microcontrollertips.com/i2c-k-squared-c/

Now let's talk about the coding part of this challenge. We will be using the *Wire.begin()* function which serves the same purpose as the regular *Serial.begin()* function. *Wire.begin()* initiate the Wire library and join the I2C bus as a master or slave. We need to include the *Wire.h* library in order to access this function call, and you don't need to download anything since Arduino IDE already has it installed. But just in case it does not work for you, simply install the library using the same steps mentioned previously. Read more about the *Wire.h* library here: https://www.arduino.cc/en/reference/wire Next, from the LiquidCrystal_PCF8574 library, we will use these four functions:

lcd.clear()
Clears the LCD screen and positions the cursor in the upper-left corner.

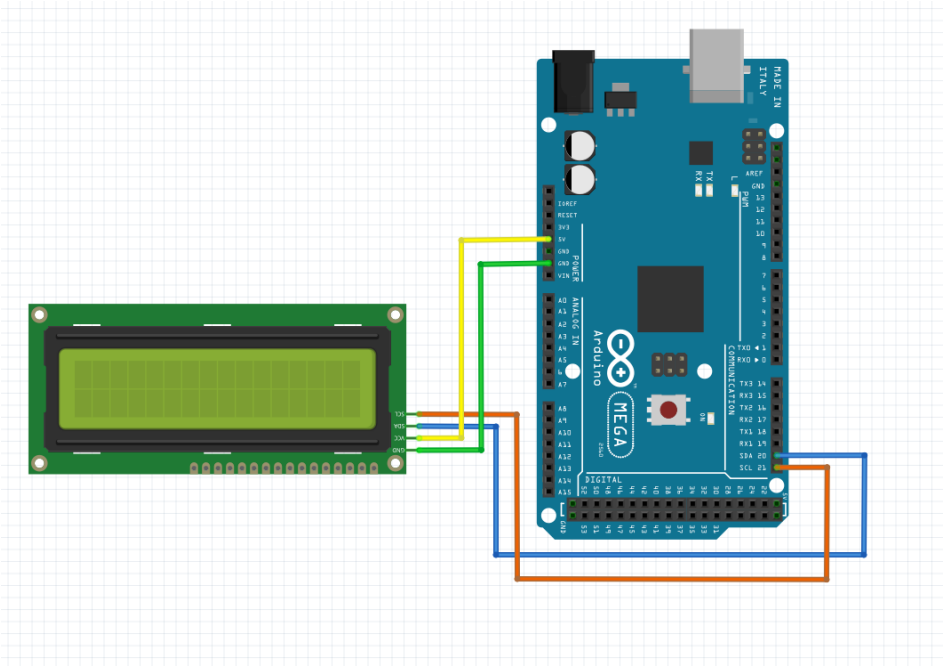
lcd.setBacklight(color)
change the backlight color assuming you have an RGB LCD on.
color: specify the color code you want

lcd.setCursor(col, row)
Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.
col: the column at which to position the cursor (with 0 being the first column)
row: the row at which to position the cursor (with 0 being the first row)

lcd.print()
Prints text to the LCD

You're all set! Now begin building the circuit! Make sure to follow the process carefully as you will need to initialize the LCD screen first before it can work.

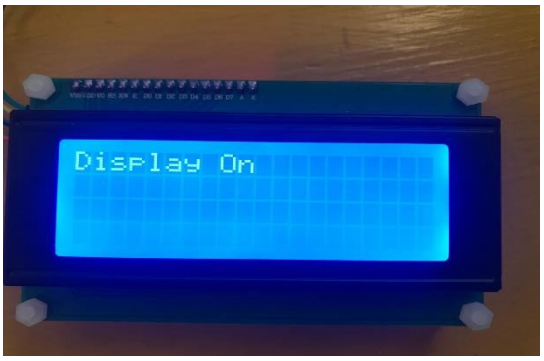
Circuit Diagram



Note: Make sure the SDA pin and SCL pin of your LCD screen is connected to the SCL and SDA pin of the Arduino

Now that your circuit has been set up, we first need to initialize the LCD screen before we are able to run the code the prints “hello world” to the screen. Here’s what you need to do:

1. Create a new empty Arduino file, then copy and paste the code from this link https://github.com/mathertel/LiquidCrystal_PCF8574/blob/master/examples/LiquidCrystal_PCF8574_Test/LiquidCrystal_PCF8574_Test.ino
2. Compile and run the code. You should see something like this (refer diagram below) on your LCD screen. If you are not seeing anything, try changing the potentiometer value at the back of the LCD PCB board until you see something.



There will also be other messages besides the one shown in the image above, you don’t have to worry about that.

3. Your LCD is ready to go! Now you can proceed to completing this challenge.

Task

- Print “hello world” (or anything else) on the LCD screen.

Code

```
#include <LiquidCrystal_PCF8574.h>
#include <Wire.h>

LiquidCrystal_PCF8574 lcd(0x27); // set the LCD address to 0x27 for a 16 chars and 2 line display

void setup()
{

  Wire.begin();
  lcd.clear();
  lcd.setBacklight(255); // Make sure backlight is on

  lcd.setCursor(0,0); // Set cursor to col 0 on line 0
  lcd.print("Hello World"); // Print a message on the LCD.
}

void loop()
{

}
```

Challenge #5: COVID19 Social Distancing Simulation

Objective

HOORAYYY!! You made it to our last and ultimate challenge: COVID19 Social Distancing Simulation. The Coronavirus pandemic has made detrimental impacts to everyone in the world. It destroys businesses, limits educations to online learning, and in the worst case, it takes away people's life. Besides putting our hopes and faith to scientists and researches in finding a cure or vaccine to this virus, we can do them and the frontliners a favor by maintaining social distancing, one of the easiest yet neglected way to fight this virus. In this challenge, you will build a circuit that simulates social distancing practice. You will code a program that tells the Buzzer to beep and the LCD screen to print "You're too close, maintain social distancing!" whenever the Ultrasonic Sensor detects an object within 6 cm (to simulate the 6 feet social distancing order). If the object is between 6 cm to 15 cm, you will command the LCD screen to print the message: "You're good. You're 6 feet away!". If the object is beyond 15 cm, don't print any message. You will incorporate all the things you learned in the previous 4 challenges to execute this last challenge successfully. Good luck!

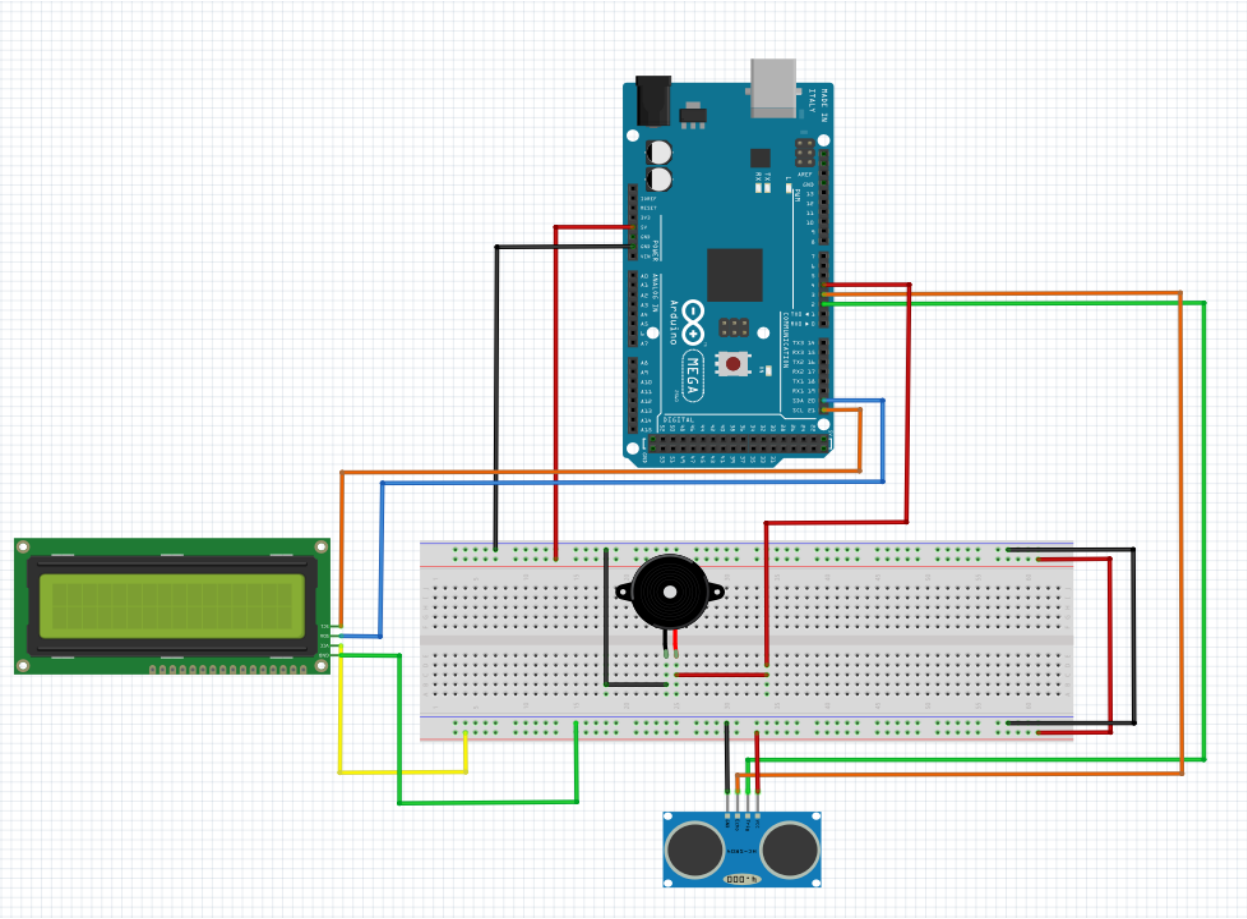
Components:

- 1 Arduino Board
- 10 Jumper Wires
- 1 Ultrasonic Sensor
- 1 Piezo Buzzer
- 1 I^2C LCD screen (comes with potentiometer)
- 1 Breadboard

Software:

- Arduino IDE
- LiquidCrystal_PCF8574.h Library

Circuit Diagram



Note: If you are having errors with the display on your LCD screen, try to run the code to initialize the LCD screen (from challenge 4) again. Sometimes you have to initialize it every time you unplug and plug it back in. After that, you should be able to run the code for this challenge.

Task

- The buzzer must beep when there is an object **within** 6 cm detected by the ultrasonic sensor.
- A message “You’re too close, maintain social distancing!” should be printed on the screen every time the buzzer beeps.
- The buzzer should not beep when the object is between 6 cm and 15 cm and the LCD screen should print the message “You’re good! You are 6 ft away!”.
- Beyond 15 cm, no message should be printed on the LCD screen and the buzzer should not be beeping.
- Keep track of the distance of your object from the ultrasonic sensor by printing the values to the serial monitor

Code

```
#include <LiquidCrystal_PCF8574.h>
#include <Wire.h>

LiquidCrystal_PCF8574 lcd(0x27); // set the LCD address to 0x27 for a 16 chars and 2 line display

const int Trig_Pin    = __; // the arduino pin that is connected to the Ultrasonic Sensor's TRIG pin
const int Echo_Pin    = __; // the arduino pin that is connected to the Ultrasonic Sensor's ECHO pin
const int Buzzer_Pin = __; // the arduino pin that is connected to the Piezo Buzzer's pin
const int Distance = __; // this is the minimum/threshold distance, in centimeters, that will trigger the buzzer

float duration_us, distance_cm;
//duration of the pulse measured by the ultrasonic sensor
//distance of object from the ultrasonic sensor

void setup()
{
  Serial.begin (9600);
  Wire.begin();

  pinMode(Trig_Pin, ______);
  pinMode(Echo_Pin, ______);
  pinMode(Buzzer_Pin, ______);
  lcd.clear();
  lcd.setBacklight(255);
  lcd.setCursor(0,0);
}

void loop()
{
  // generate 10-microsecond pulse to TRIG pin
  digitalWrite(Trig_Pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig_Pin, LOW);

  // measure duration of pulse from ECHO pin
  duration_us = pulseIn(Echo_Pin, HIGH);
  // calculate the distance
  distance_cm = 0.017 * duration_us;

  if(distance_cm < _________)
  {
    digitalWrite(Buzzer_Pin, ____); // here, what should be the condition of the Piezo Buzzer, on or off?
    lcd.print(_____________________________________); // What do you want to print on the LCD screen
  }

  else if (_______________)
  {
    digitalWrite(Buzzer_Pin,____); // here, what should be the condition of the Piezo Buzzer, on or off?
    lcd.print("You're 6ft away, good!"); // What do you want to print on the LCD screen
  }

  // print the distance values to Serial Monitor
  Serial.print("distance: ");
  Serial.print(___________);
  Serial.println(" cm");

  delay(500);
}
```

Challenge #6: A Case Using CAD - OnShape

Objective

You made it to the final challenge!! It is time to pull everything you built together with a touch of CAD. In this challenge you will be designing a case for your device in OnShape. The best way to do this is to stack up your breadboard, LCD screen, Arduino board and all other components to see how it would fit in the case. Experiment with different arrangements and see what works best! Take measurements of the dimensions of the overall size for the case body once you have finalized the arrangement of your components. Then take the dimensions of the sensor, the LCD screen, and the buzzer to create cut-outs in your case.

Software:

- OnShape

Task

- Take measurements of the dimensions of each of the components: LCD screen, buzzer, and ultrasonic sensor.
- Create a case using OnShape.

Citations and References:

Tutorials/Theories

LCD2004 20x4 12C LCD Display articles
https://protosupplies.com/product/lcd2004-20x4-i2c-blue-lcd-display/

What is I^2C?
https://www.microcontrollertips.com/i2c-k-squared-c/

LCD simple hello world code
https://protosupplies.com/product/lcd2004-20x4-i2c-blue-lcd-display/

Ultrasonic Sensor & LED
https://arduinogetstarted.com/tutorials/arduino-ultrasonic-sensor-lcd

Ultrasonic Sensor & Piezo Buzzer
https://arduinogetstarted.com/tutorials/arduino-ultrasonic-sensor-piezo-buzzer

Concepts/Theory on Ultrasonic Sensor and how it works
https://www.arrow.com/en/research-and-events/articles/ultrasonic-sensors-how-they-work-and-how-to-use-them-with-arduino#:~:text=Ultrasonic%20sensors%20work%20by%20emitting,return%20after%20hitting%20an%20object

LCD Screen
https://arduinogetstarted.com/tutorials/arduino-lcd

About LCD screen
https://www.arduino.cc/en/Tutorial/LibraryExamples/HelloWorld
https://core-electronics.com.au/tutorials/use-lcd-arduino-uno.html

Functions and Libraries

LiquidCrystal_PCF8574.h library
https://github.com/mathertel/LiquidCrystal_PCF8574/blob/master/examples/LiquidCrystal_PCF8574_Test/LiquidCrystal_PCF8574_Test.ino

Wire.h library
https://www.arduino.cc/en/Reference/Wire

https://components101.com/ultrasonic-sensor-working-pinout-datasheet

pulseIn() function
https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/?setlang=it

delayMicroseconds() function
https://www.arduino.cc/reference/en/language/functions/time/delaymicroseconds/

display() and *noDisplay()* methods
https://www.arduino.cc/en/Tutorial/LibraryExamples/LiquidCrystalDisplay

Wire.begin() function
https://www.arduino.cc/en/Reference/WireBegin

lcd.clear() function
https://www.arduino.cc/en/Reference/LiquidCrystalClear

lcd.setBacklight() function
https://learn.adafruit.com/rgb-lcd-shield/using-the-rgb-lcd-shield

lcd.setCursor() function
https://www.arduino.cc/en/Reference/LiquidCrystalSetCursor

lcd.print() function
https://www.arduino.cc/en/Reference/LiquidCrystalPrint

Initialize LCD screen
https://github.com/mathertel/LiquidCrystal_PCF8574/blob/master/examples/LiquidCrystal_PCF8574_Test/LiquidCrystal_PCF8574_Test.ino

Eagle Library

Arduino Mega
https://www.diymodules.org/eagle-show-object?type=shr&id=15&device=BOARD_ARDUINO-MEGA#variant-BOARD_ARDUINO-MEGA

Ultrasonic Sensor
https://www.diymodules.org/eagle-show-object?type=dm&file=diy-modules.lbr&symbol=ULTRASONIC-HC-SR04

I^{2}C LCD
http://www.diymodules.org/eagle-show-object?type=usr&id=2550&device=2X16LCDTOP#variant-2X16LCDTOP

I^{2}C LCD adapter
https://www.diymodules.org/eagle-show-object?type=dm&file=diy-modules.lbr&device=LCD-I2C-CONVERTER

Buzzer
https://www.diymodules.org/eagle-show-library?type=std&file=buzzer.lbr

Fritzing:

I²C LCD part
https://forum.fritzing.org/t/16x2-i2c-lcd-part/2041/4

Images:

Piezo Buzzer
https://product.tdk.com/en/search/sw_piezo/sw_piezo/piezo-buzzer/list#_l=20&_p=1&_c=part_no-part_no&_d=0

Images on Page 1 link
https://www.vdh.virginia.gov/coronavirus/what-is-social-distancing-and-how-can-i-do-my-part-to-slow-the-spread-of-covid-19/

Ultrasonic Sensor
https://www.amazon.com/Measuring-Ultrasonic-Distance-Avoidance-Raspberry/dp/B01N1KVIOD

LCD image on slide/presentation
https://www.arduino.cc/en/Tutorial/LibraryExamples/HelloWorld

LED image on slide/presentation
http://www.pngmart.com/image/83883

Resistor image on slide/presentation
https://webstockreview.net/pict/getfirst

Jumper wires image on slide/presentation
https://voltspace.in/product/jumper-wires-1p-1p-male-to-female-40pcs/