IRIS PROJECT

Objective

Creating a detector that uses ultrasonic sensor and LED to assist with motion detection when an obstacle is nearby. The ultrasonic sensor emits high frequency sound waves and measures the time taken for the waves to bounce back from an obstacle, using this the distance is calculated based on the speed of sound in air. The LED main functionality is for showing the proximity by adjusting its brightness. In this example, if the object is closer than 60cm then the LED turns on.

Real-Life application usage

Smart parking assistance system aims to assist drivers in parking by providing real-time distance feedback using light indicators to show proximity. The system also allows drivers to adjust the desired distance threshold for triggering alerts.

Overview

The Arduino Uno R3 is connected to the ultrasonic sensor, which continuously measures the distance between the sensor and any nearby obstacles. The Arduino processes this distance data and adjusts the LED's brightness accordingly, based on the proximity of the obstacle. A Python script communicates with the Arduino via serial connection, retrieving the distance measurements and plotting them in real-time using the matplotlib library.

Components used

- Arduino UNo R3
- Ultrasonic sensor
- LED
- Resistor (220 ohm)
- Breadboard
- Wires

Setup

Link to Tinkercad:

https://www.tinkercad.com/things/acenuf8iovh-grand-jaban-lahdi/editel?sharecode=-aNC0jRgZNpqzCp7smDcZdQvaJTVKaTflm5zpGWTOO0

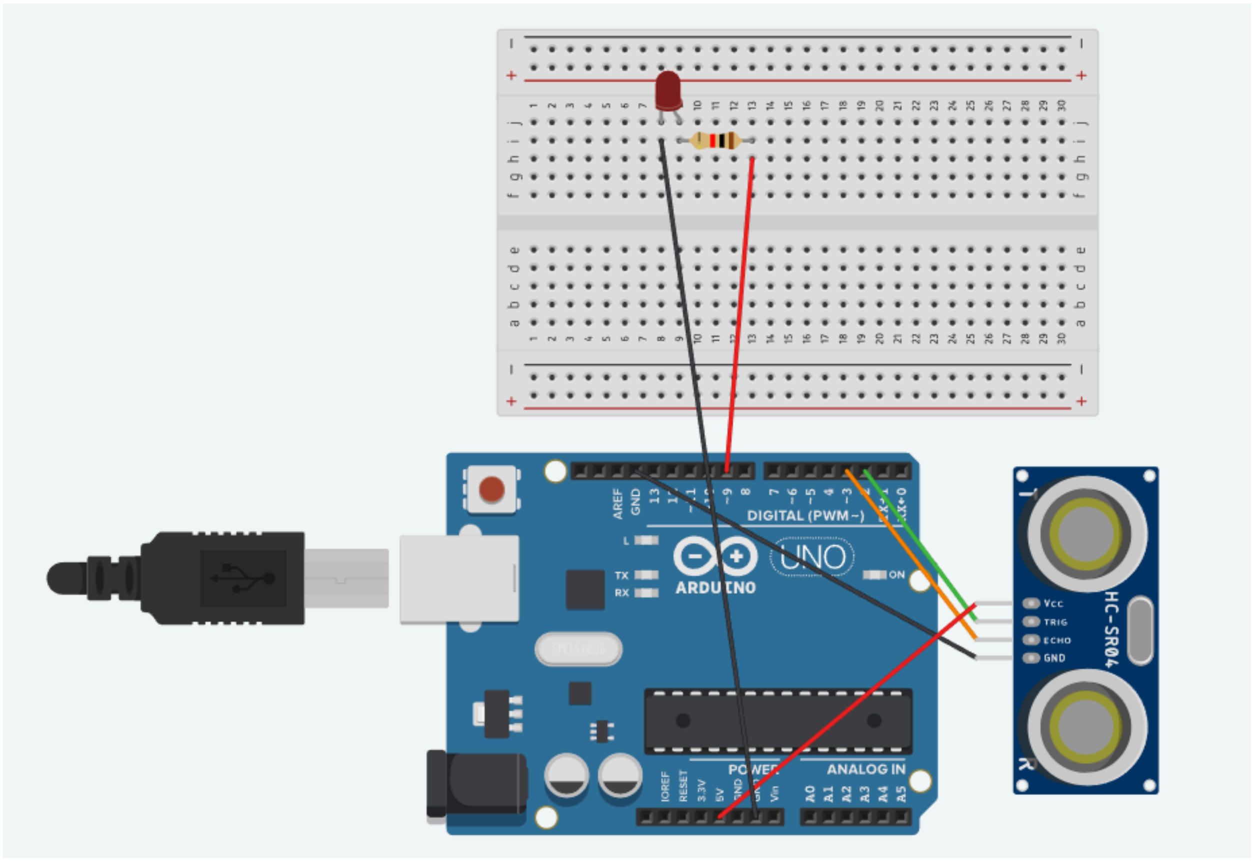


Fig 1.1

For ultrasonic sensor:

Connect the ultrasonic sensor to the Arduino Uno R3 as follows:

- VCC to 5V
- Trig to Digital Pin 2
- Echo to Digital Pin 3
- GND to GND

For LED:

Place the LED into the Breadboard as shown above in Fig 1.1. The longer side of the LED wire is the anode and the shorter side is the cathode

For cathode:
Connect a wire from the cathode to GND
For anode:
Connect the 220 Ohm Resistor to the anode and on the other side of the Resistor to Digital Pin 9 as shown above in Fig 1.1

Code Implementation

- Write Arduino code to read the distance from the ultrasonic sensor and control the LED based on the distance.
- When the distance detected by the ultrasonic sensor is above a certain threshold (ex: 60cm), turn off the LED.
- When the distance is below the threshold, turn on the LED and adjust its brightness based on the distance to the obstacle.
- You can use the analogWrite() function to control the brightness of the LED based on the distance.

Code part:

```
// Define constants for ultrasonic sensor pins
const int trigPin = 2;
const int echoPin = 3;

// Define LED pin
const int ledPin = 9;

// Define variables for distance calculation
long duration;
int distance;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Set pin modes
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Clear the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Send a pulse to the sensor to trigger measurement
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Measure the duration of the echo pulse
  duration = pulseIn(echoPin, HIGH);

  // Calculate distance in centimeters
  distance = duration * 0.034 / 2;

  // Send distance data to serial port
  Serial.println(distance);

  // Control LED based on distance
  if (distance < 60) {
    analogWrite(ledPin, map(distance, 0, 60, 255, 0)); // Adjust LED brightness based on distance
  } else {
    analogWrite(ledPin, 0); // Turn off LED if distance is greater than 60cm
  }

  // Delay before next measurement
  delay(100);
}
```

Python Implementation

- Write a Python script to read the serial data from the Arduino.
- Parse the distance data and plot it using matplotlib.
- You can continuously update the plot as new data is received from the Arduino.

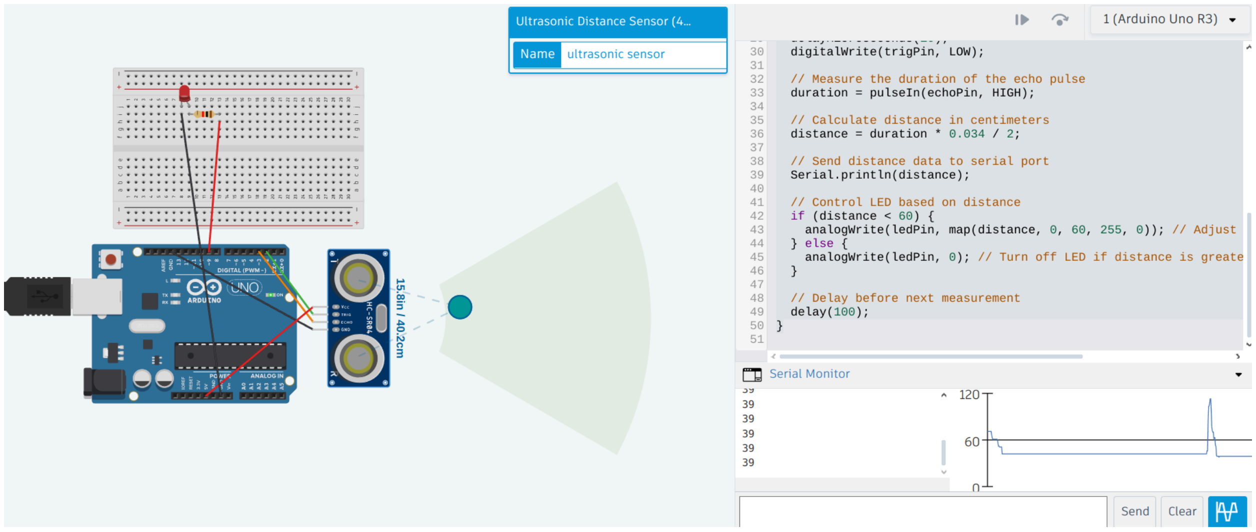
Code part:

```
import serial
import matplotlib.pyplot as plt

# Set up serial communication
ser = serial.Serial('COM3', 9600)  # Change 'COM3' to the port connected to Arduino

# Initialize plot
plt.ion()  # Turn on interactive mode
fig, ax = plt.subplots()
line, = ax.plot([], [])  # Create an empty plot
ax.set_xlabel('Time (s)')
ax.set_ylabel('Distance (cm)')
ax.set_title('Real-time Distance Measurement')

# Initialize empty lists to store data
x_data = []
y_data = []

# Main loop to read data and update plot
try:
    while True:
        # Read distance data from serial port
        data = ser.readline().decode().strip()
        if data:
            distance = int(data)

            # Append data to lists
            x_data.append(len(x_data) + 1)
            y_data.append(distance)

            # Update plot
            line.set_data(x_data, y_data)
            ax.relim()
            ax.autoscale_view()
            plt.draw()
            plt.pause(0.01)  # Create a 0.01 second delay

# Handle KeyboardInterrupt to gracefully exit
except KeyboardInterrupt:
    print("Exiting...")
    ser.close()
    plt.close()
```

Instructions:

- Upload the Arduino code to your Arduino Uno.
- Connect the Arduino to your computer via USB.
- Run the Python code on your computer (ensure you have matplotlib installed).
- Adjust the serial port in the Python code (COM3 in this example) to match the port connected to your Arduino.
- You should see a real-time plot displaying the distance measurements from the ultrasonic sensor.

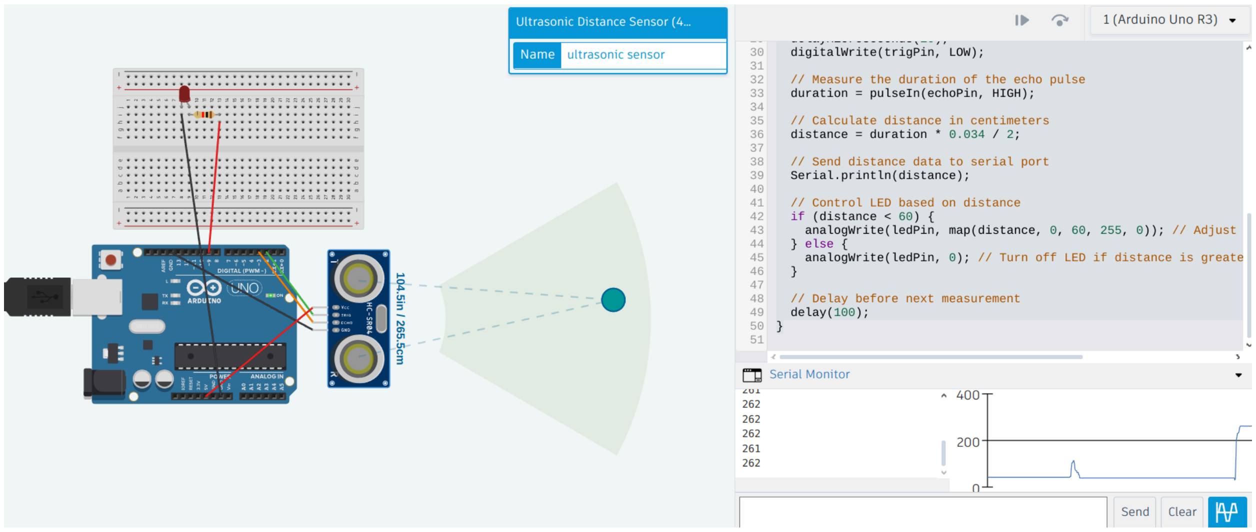
Sample of working model



```
  digitalWrite(trigPin, LOW);

  // Measure the duration of the echo pulse
  duration = pulseIn(echoPin, HIGH);

  // Calculate distance in centimeters
  distance = duration * 0.034 / 2;

  // Send distance data to serial port
  Serial.println(distance);

  // Control LED based on distance
  if (distance < 60) {
    analogWrite(ledPin, map(distance, 0, 60, 255, 0)); // Adjust
  } else {
    analogWrite(ledPin, 0); // Turn off LED if distance is greate
  }

  // Delay before next measurement
  delay(100);
}
```

```
39
39
39
39
39
```

Fig 1.2



```
  digitalWrite(trigPin, LOW);

  // Measure the duration of the echo pulse
  duration = pulseIn(echoPin, HIGH);

  // Calculate distance in centimeters
  distance = duration * 0.034 / 2;

  // Send distance data to serial port
  Serial.println(distance);

  // Control LED based on distance
  if (distance < 60) {
    analogWrite(ledPin, map(distance, 0, 60, 255, 0)); // Adjust
  } else {
    analogWrite(ledPin, 0); // Turn off LED if distance is greate
  }

  // Delay before next measurement
  delay(100);
}
```

```
262
262
262
261
262
```

Fig 1.3

- Fig 1.2 shows when the object is less than 60cm from the Ultrasonic sensor. LED turns on

- Fig 1.3 shows when the object is more than 60cm from the Ultrasonic sensor. LED turns off