Title: Interfacing the Arduino with an external sensor using serial communication protocol for implementing an obstacle detection system.

1. Cover page:

2. Objective:

The purpose of this experiment is to interface the Arduino microcontroller with an external sensor using serial communication protocol for implementing an obstacle detection system. The experiment involves using a sonar sensor (HCS04) to detect the distance of an obstacle and based on the distance between the sensor and the object being detected, one or more LEDs will glow as soon as it detects the obstacle. The ultrasound transmitter (Trigger pin) emits a high-frequency ultrasonic sound wave (40 kHz) that travels through the air, and if it finds an object, it bounces back to the module. The experiment includes writing code for a simple obstacle detection system in Arduino IDE and implementing the system using an Arduino microcontroller.

3. The apparatus and software name:

- Arduino UNO
- Arduino IDE (any version)
- LED
- Sonar Sensor (HCSR04)
- Breadboard
- Connecting Wires

4. Theory and programs:

The open-source Arduino platform is used to build interactive electronics projects. In order to write and upload computer code to the microcontroller board, the Arduino system comprises of a programmable microcontroller and an application known as the IDE (Integrated Development Environment) that runs on your computer. Moreover, the Arduino Uno does not require a hardware component (programmer or burner) to upload code to the board. With simply a USB cord and the Arduino IDE (which utilizes a simpler version of C++ to develop programming), we can quickly load a code onto the board. To measure the distance of an obstacle, we will use a sonar sensor (HCS04) in this experiment. One or more LEDs will begin to illuminate as soon as the obstacle is recognized, depending on the distance between the sensor and the object being detected.

The HCSR04 ultrasonic sensor measures the distance to an object using a sonar wave. This sensor has a 0.3 cm accuracy range of 2 cm to 400 cm (0.8 inches to 157 inches) (0.1 inches). A transmitter, receiver, and control circuit make up the HCSR04 module. The pins are labeled VCC, GND, Trigger, and Echo.

5. A brief procedure:

1. Gather the necessary components, including an Arduino board, an HCSR04 ultrasonic sensor, connecting wires, and LEDs.
2. Connect the HCSR04 sensor to the Arduino board by connecting the VCC pin to +5V, the GND pin to GND, the Trigger pin to digital pin 11, and the Echo pin to digital pin 12.
3. Connect the LEDs to the Arduino board by connecting the anode (positive) pin of each LED to digital pins 2, 3, and 4, and the cathode (negative) pin of each LED to a 100 ohm resistor connected to GND.
4. Open the Arduino IDE on your computer and write the code for the obstacle detection system, including initializing the pins, sending a trigger signal to the sensor, calculating the distance based on the echo signal, and turning on the appropriate LED based on the distance threshold.
5. Compile the code and upload it to the Arduino board.
6. Open the serial monitor in the Arduino IDE to observe the distance readings and LED activations.

7. Test the system by placing objects at varying distances from the sensor and observing the LED activations in response.

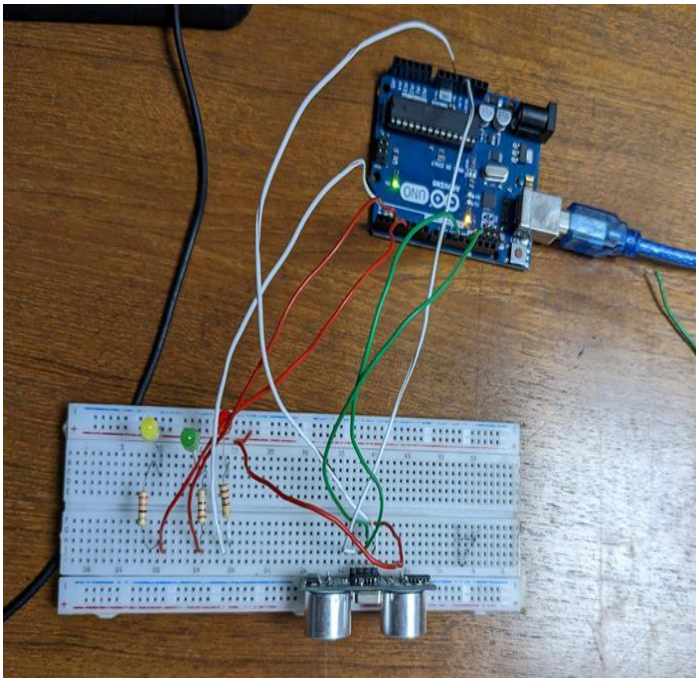
Code:

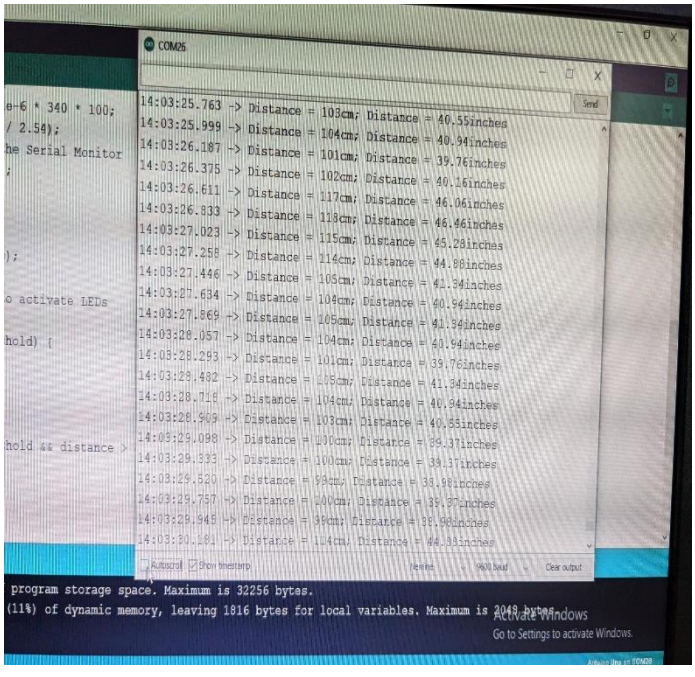
```
// define the pin numbers
const int trigPin = 11;
const int echoPin = 12;

// define variables
long duration;
int distance;
float distance, distanceInches, distanceThreshold;

void setup() {
   Serial.begin(9600); // Starts the serial communication
   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
   pinMode(2, OUTPUT); // Sets pins 2, 3, and 4 as the Output pin
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
}

void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
 // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = (duration/2) * 1e6 * 340 * 100;
  distanceinches = (distance / 2.54);
  // Prints the distance on the Serial Monitor
  Serial.print("Distance = ");
```

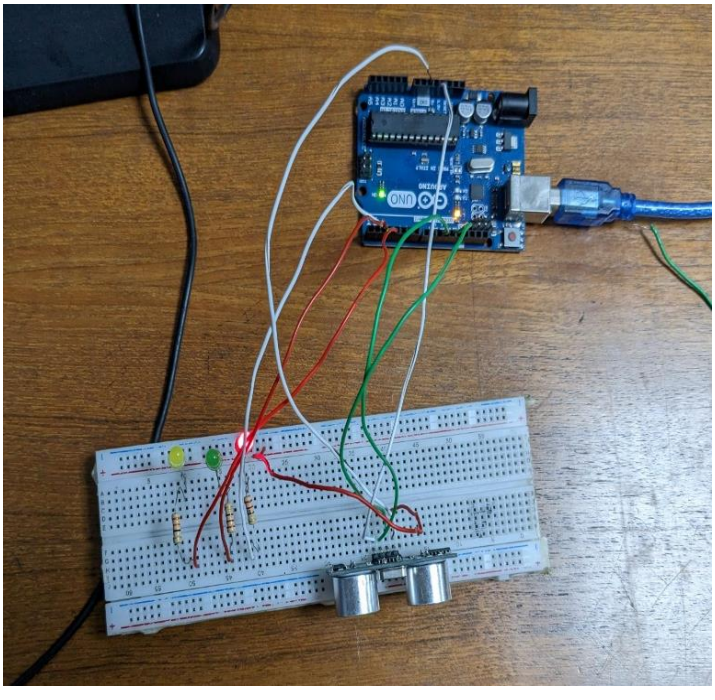
```
  Serial.print(distance);
  Serial.print(“cm;  ”);
  Serial.print("Distance = ");
  Serial.print(distanceinches);
  Serial.println(“inches”);

  // set threshold distance to activate LEDs
distanceThreshold = 80;

if (distance > distanceThreshold) {
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
}

if (distance < distanceThreshold && distance > distanceThreshold-30) {
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
}

 if (distance < distanceThreshold-30 && distance > distanceThreshold-50) {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
}

  if (distance < distanceThreshold-50 && distance > distanceThreshold-70 ) {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
}
delay(200); // Wait for 200 millisecond(s)
}
```

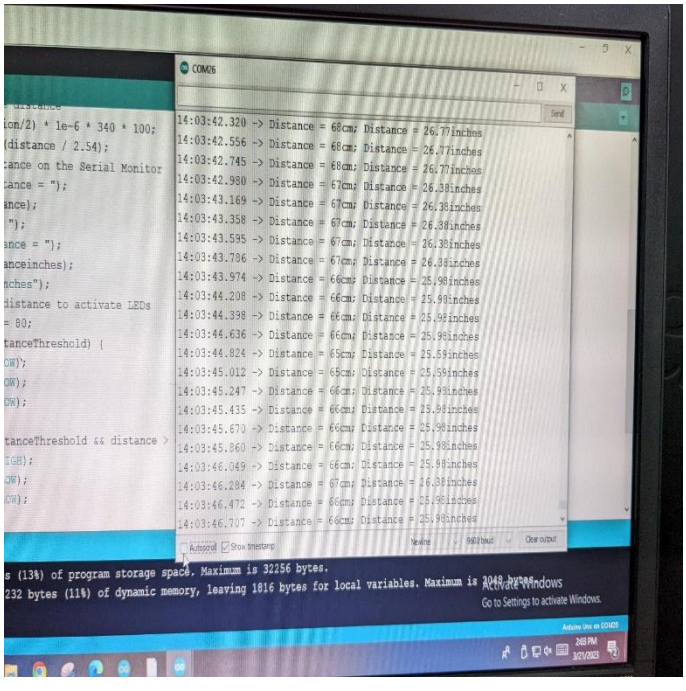
Simulation:

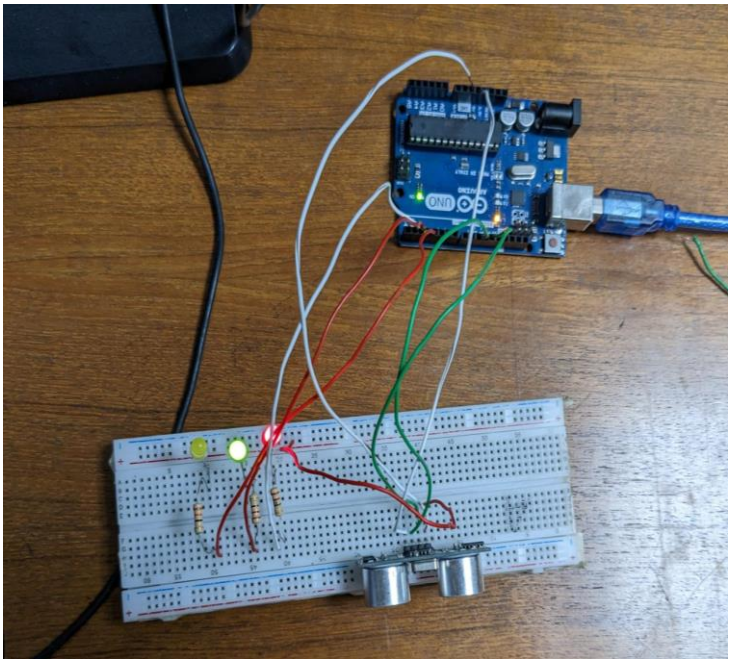
Lab Simulation:

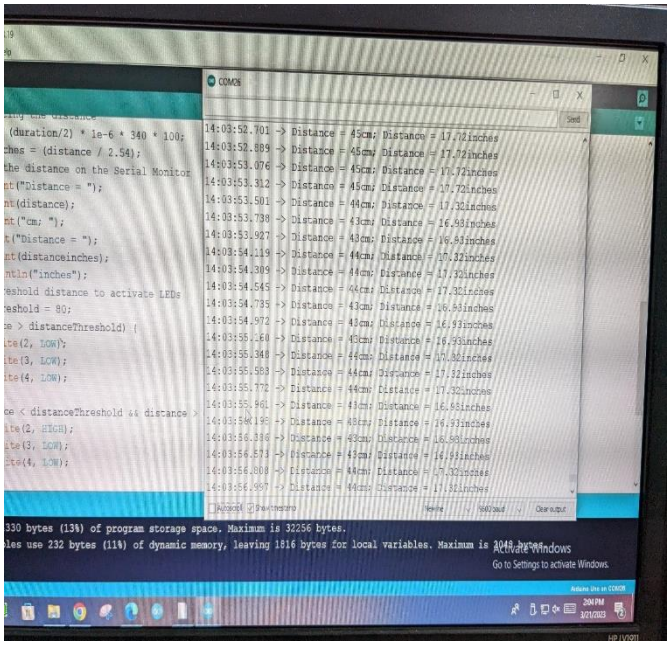


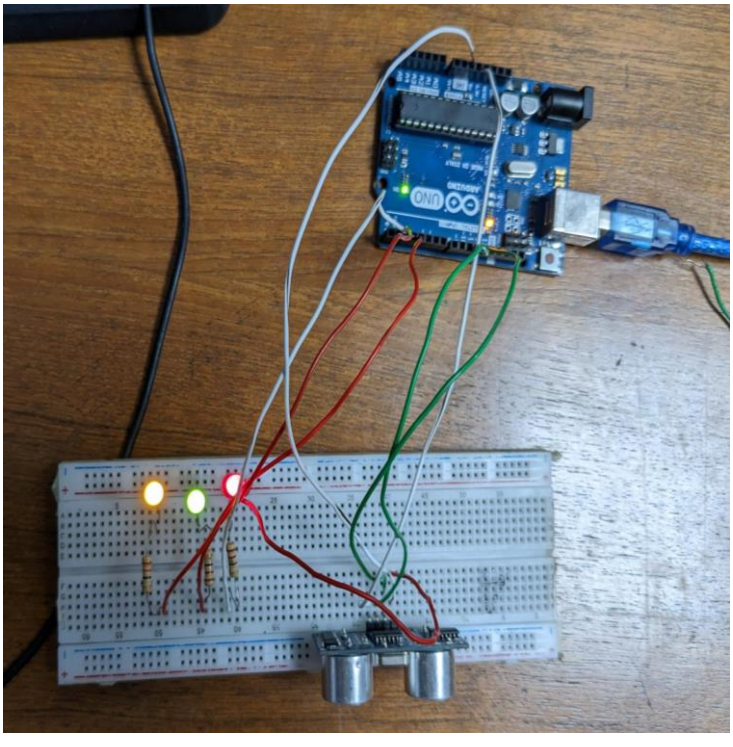


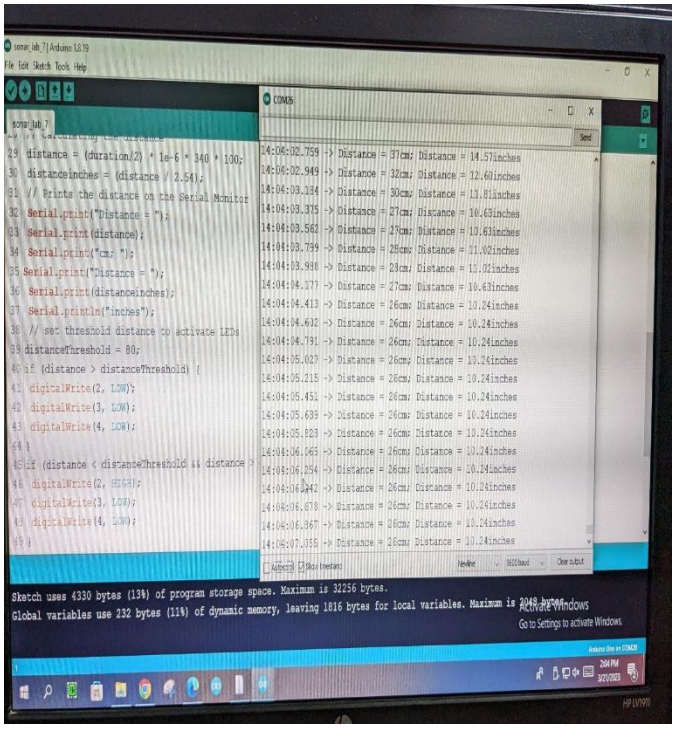


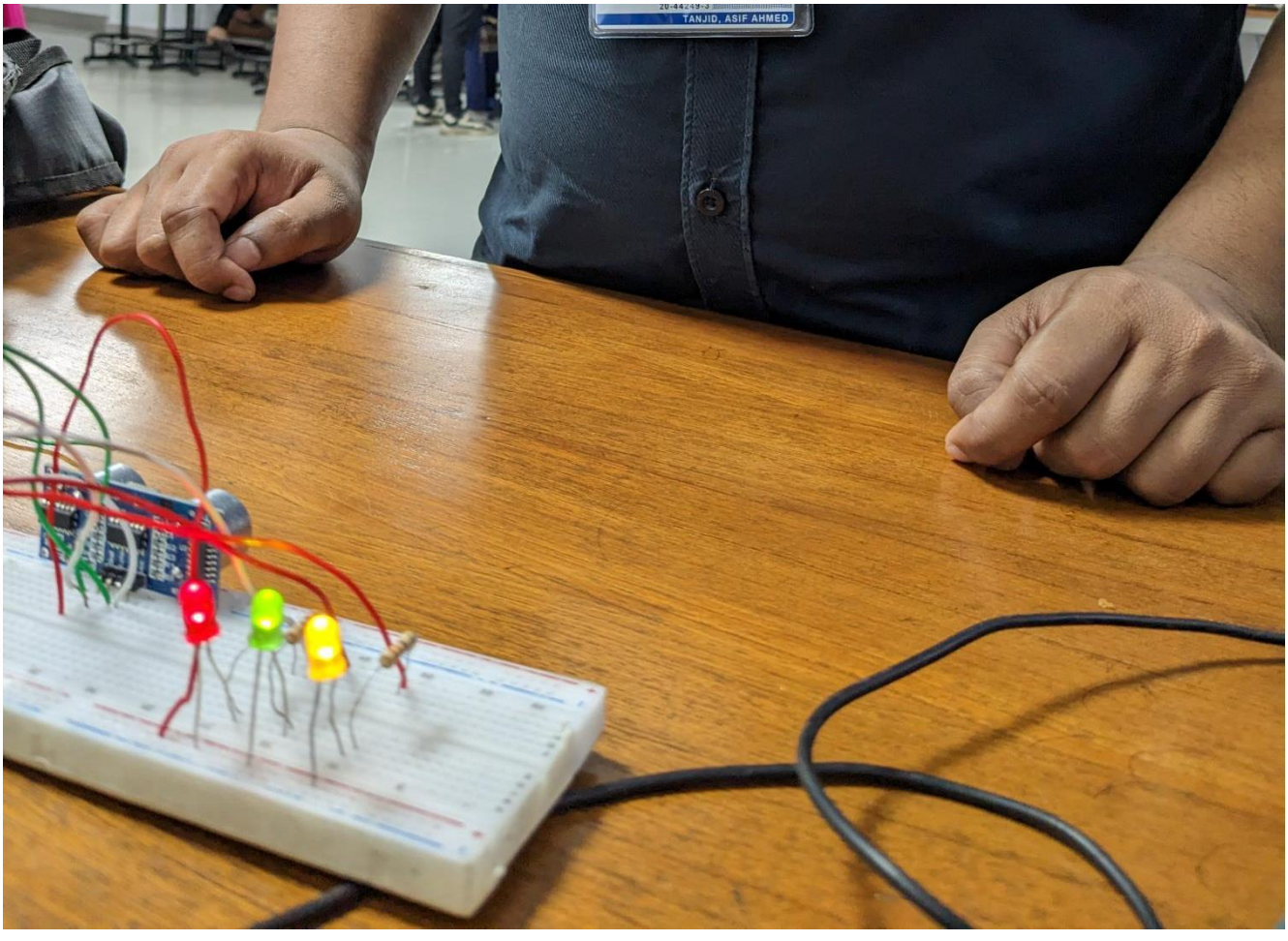





COM26
14:03:52.701 -> Distance = 45cm; Distance = 17.72inches
14:03:52.889 -> Distance = 45cm; Distance = 17.72inches
14:03:53.076 -> Distance = 45cm; Distance = 17.72inches
14:03:53.312 -> Distance = 45cm; Distance = 17.72inches
14:03:53.501 -> Distance = 44cm; Distance = 17.32inches
14:03:53.738 -> Distance = 43cm; Distance = 16.93inches
14:03:53.927 -> Distance = 43cm; Distance = 16.93inches
14:03:54.119 -> Distance = 44cm; Distance = 17.32inches
14:03:54.309 -> Distance = 44cm; Distance = 17.32inches
14:03:54.545 -> Distance = 44cm; Distance = 17.32inches
14:03:54.735 -> Distance = 43cm; Distance = 16.93inches
14:03:54.972 -> Distance = 43cm; Distance = 16.93inches
14:03:55.160 -> Distance = 43cm; Distance = 16.93inches
14:03:55.348 -> Distance = 44cm; Distance = 17.32inches
14:03:55.583 -> Distance = 44cm; Distance = 17.32inches
14:03:55.772 -> Distance = 44cm; Distance = 17.32inches
14:03:55.961 -> Distance = 43cm; Distance = 16.93inches
14:03:56.196 -> Distance = 43cm; Distance = 16.93inches
14:03:56.386 -> Distance = 43cm; Distance = 16.93inches
14:03:56.573 -> Distance = 43cm; Distance = 16.93inches
14:03:56.808 -> Distance = 44cm; Distance = 17.32inches
14:03:56.997 -> Distance = 44cm; Distance = 17.32inches
330 bytes (13%) of program storage space. Maximum is 32256 bytes.
les use 232 bytes (11%) of dynamic memory, leaving 1816 bytes for local variables. Maximum is 2048 bytes.
Activate Windows
Go to Settings to activate Windows.




COM26
14:06:02.759 -> Distance = 37cm; Distance = 14.57inches
14:06:02.949 -> Distance = 32cm; Distance = 12.60inches
14:04:03.134 -> Distance = 30cm; Distance = 11.81inches
14:04:03.375 -> Distance = 27cm; Distance = 10.63inches
14:04:03.562 -> Distance = 27cm; Distance = 10.63inches
14:04:03.799 -> Distance = 28cm; Distance = 11.02inches
14:04:03.930 -> Distance = 28cm; Distance = 11.02inches
14:04:04.177 -> Distance = 27cm; Distance = 10.63inches
14:04:04.413 -> Distance = 26cm; Distance = 10.24inches
14:04:04.602 -> Distance = 26cm; Distance = 10.24inches
14:04:04.791 -> Distance = 26cm; Distance = 10.24inches
14:04:05.027 -> Distance = 26cm; Distance = 10.24inches
14:04:05.215 -> Distance = 26cm; Distance = 10.24inches
14:04:05.451 -> Distance = 26cm; Distance = 10.24inches
14:04:05.639 -> Distance = 26cm; Distance = 10.24inches
14:04:05.828 -> Distance = 26cm; Distance = 10.24inches
14:04:06.065 -> Distance = 26cm; Distance = 10.24inches
14:04:06.254 -> Distance = 26cm; Distance = 10.24inches
14:04:06.442 -> Distance = 26cm; Distance = 10.24inches
14:04:06.678 -> Distance = 26cm; Distance = 10.24inches
14:04:06.867 -> Distance = 26cm; Distance = 10.24inches
14:04:07.055 -> Distance = 26cm; Distance = 10.24inches
Sketch uses 4330 bytes (13%) of program storage space. Maximum is 32256 bytes.
Global variables use 232 bytes (11%) of dynamic memory, leaving 1816 bytes for local variables. Maximum is 2048 bytes.
Activate Windows
Go to Settings to activate Windows.



Report:

All codes, scripts and proteus simulation of the blink program and traffic light system is attached above.

8. Discussions:

In this experiment, we are implementing an obstacle detection system using an Arduino microcontroller and an HCSR04 ultrasonic sensor. The HCSR04 sensor is a popular and relatively inexpensive sensor that can measure distances from 2 cm to 400 cm with an accuracy of 0.3 cm. It uses sonar signals to determine the distance to an object, and consists of a transmitter, receiver, and control circuit. This experiment demonstrates the use of an Arduino microcontroller and a sensor to implement a simple obstacle detection system. It shows how easy it is to interface external sensors with the Arduino board using serial communication and highlights the flexibility and versatility of the Arduino platform for creating interactive electronics projects.

9. Conclusions:

In conclusion, the implementation of an obstacle detection system using an Arduino microcontroller and an HCSR04 ultrasonic sensor was successfully demonstrated in this experiment. The code was written in the Arduino IDE, and the sensor was interfaced with the microcontroller using serial communication protocol. The experiment showed that the sensor could accurately detect the distance of an obstacle within its range and trigger the appropriate LED based on the distance threshold. This project can be further developed and used in various real-world applications, such as robotics, automated systems, and security systems, to detect obstacles and avoid collisions.