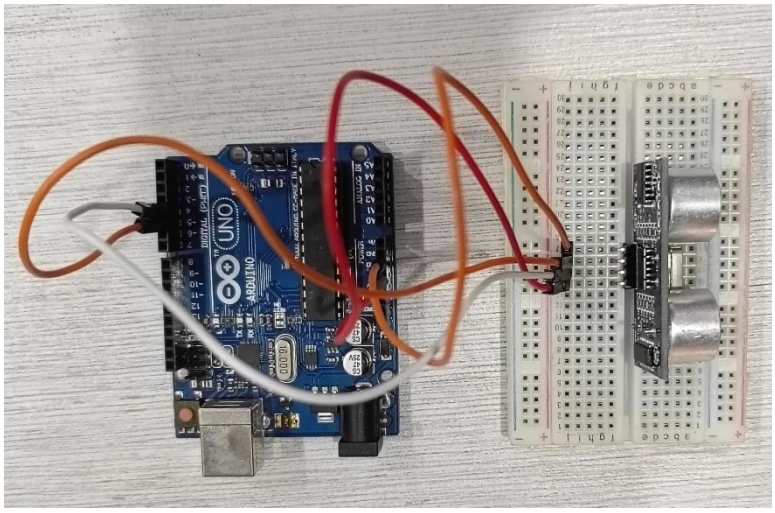
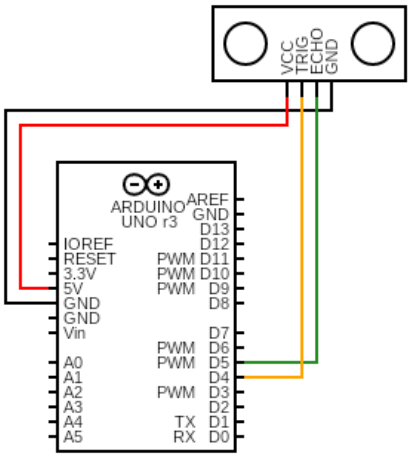
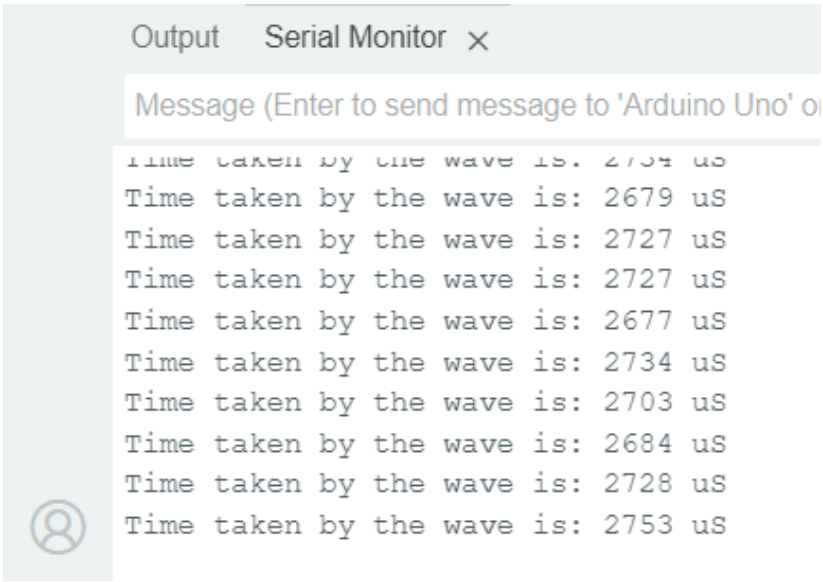
LAB TASKS

Q1: Develop the program to read the data from ultrasonic sensor and display it on serial monitor. Capture it using snipping tool and add this image in your lab report.

```
#define trigPin 4 // defining trigger global variable with pin value
#define echoPin 5 // defining echo global variable with pin value

int duration; // defining variable to store time value

void setup() {
  pinMode(trigPin, OUTPUT); // Assigning output port for trigger pin
  pinMode(echoPin, INPUT); // Assigning input port for echo pin
  Serial.begin(9600); // Calling serial monitor for data display
}

void loop() {
  digitalWrite(trigPin, LOW); // Clearing already storred code in Arduino from previous run (if
any)
  delayMicroseconds(2); // Defining duration for this clear code to run

  digitalWrite(trigPin, HIGH); // Assigning trigger pin with max voltage
  delayMicroseconds(10); // Assigning duration for the above trigger input code

  digitalWrite(trigPin, LOW); // Assigning trigger pin with zero voltage to stop code

  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time
in microseconds and stores in duration

  Serial.print("Time taken by the wave is: "); // Printing data to serial monitor
  Serial.print(duration); // Printing duration value to serial monitor
  Serial.println(" uS"); // Printing rest of the text to serial monitor
}
```



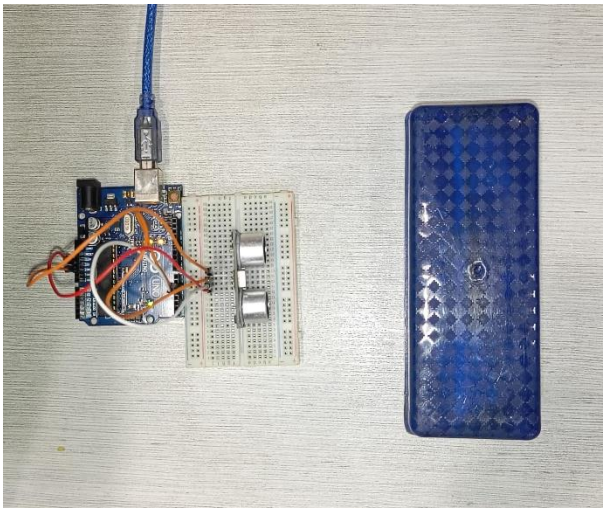


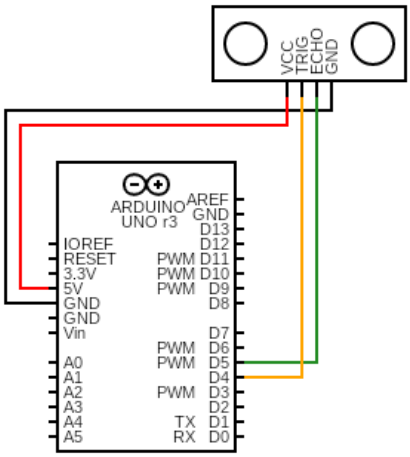


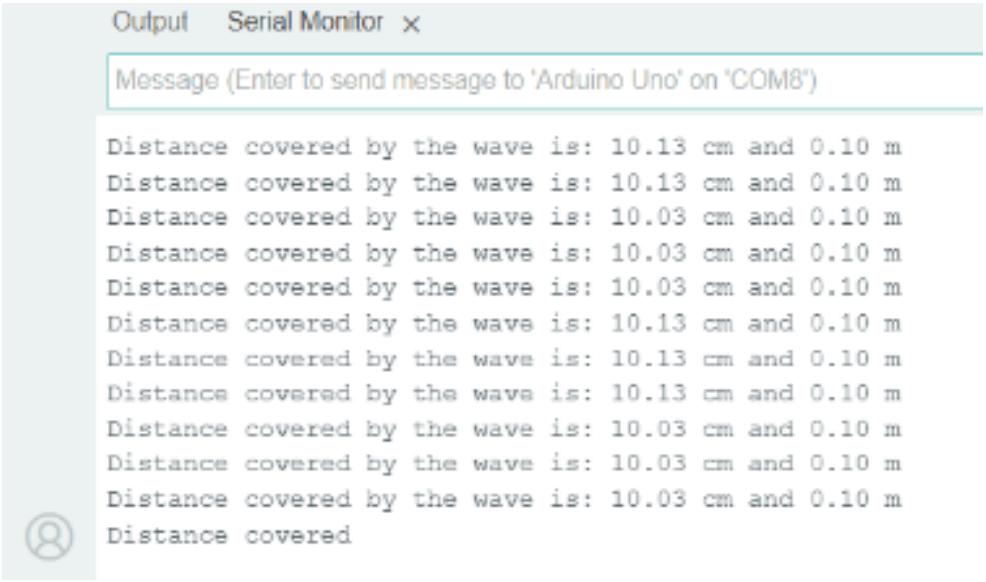
Q2: Develop a digital measuring device which displays data in cm and m based on the time you calculated in question 1. Use basic velocity formula with consistent units. Capture it using snipping tool and add this image in your lab report.

```
#define trigPin 4 // Defining trigger global variable with pin value
#define echoPin 5 // Defining echo global variable with pin value

float distance; // Defining variable to store cm distance value
float mdistance; // Defining variable to store m distance value
int duration; // Defining variable to store time value

void setup() {
  pinMode(trigPin, OUTPUT); // Assigning output port for trigger pin
  pinMode(echoPin, INPUT); // Assigning input port for echo pin
  Serial.begin(9600); // Calling serial monitor for data display
}

void loop() {
  digitalWrite(trigPin, LOW); // Clearing already storred code in Arduino from previous run (if
any)
  delayMicroseconds(2); // Defining duration for this clear code to run

  digitalWrite(trigPin, HIGH); // Assigning trigger pin with max voltage
  delayMicroseconds(10); // Assigning duration for the above trigger input code

  digitalWrite(trigPin, LOW); // Assigning trigger pin with zero voltage to stop code

  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time
in microseconds and stores in duration

  distance = duration * 0.034 / 2; // Using speed distance formulae to convert time into
distance using speed of sound
  mdistance = distance / 100; // Converting cm distance to m distance

  Serial.print("Distance covered by the wave is: "); // Printing data to serial monitor
```

```
  Serial.print(distance); // Printing cm distance value to serial monitor
  Serial.print(" cm and "); // Printing rest of the text to serial monitor
  Serial.print(mdistance); // Printing m distance value to serial monitor
  Serial.println(" m"); // Printing rest of the text to serial monitor
}
```





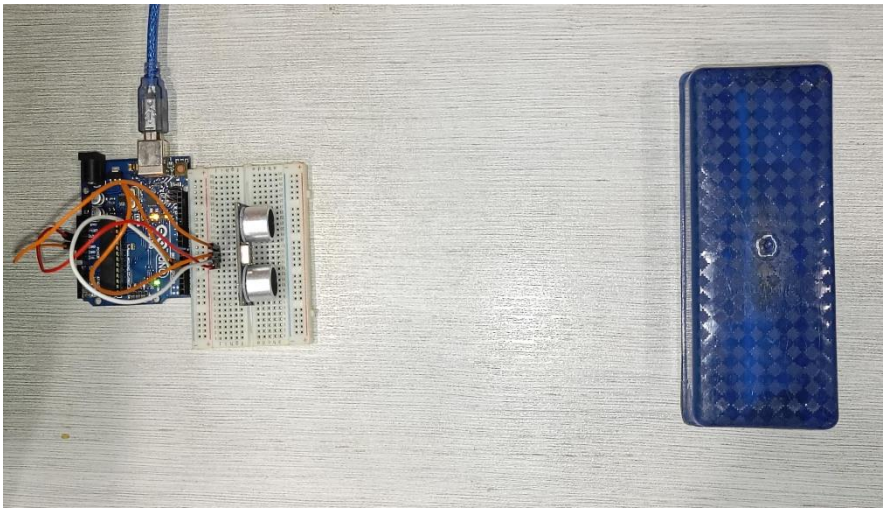


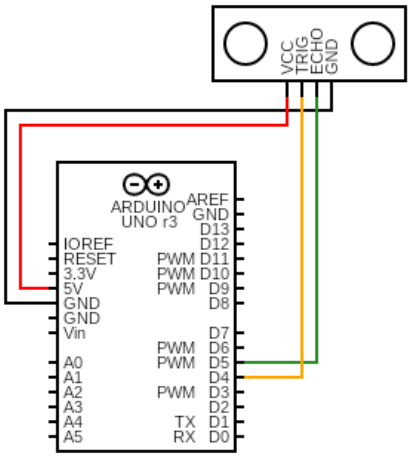
Q3: Develop a vehicle reverse car parking device that will assist the driver while reversing the car. (Capture the results using snipping tool and add in your lab report.) The device must consist of an ultrasonic sensor which would sense the data between rear bumper and an obstacle. Write the program such that

- **if the distance is between 30 and 45 cm, it prints “SLOW!!”,**
- **if the distance is between 15 and 30 cm, it prints “WATCH OUT!!”,**
- **if the distance is between 0 and 15 cm, it prints “STOP!!”.**

```
#define trigPin 4 // Defining trigger global variable with pin value
#define echoPin 5 // Defining echo global variable with pin value

float distance; // Defining variable to store cm distance value
float mdistance; // Defining variable to store m distance value
int duration; // Defining variable to store time value
```

```
// Defining function to store multiple print statement and print all when called
float distanceFunction(float x, float y) {
  Serial.print("Distance covered by the wave is: "); // Printing data to serial monitor
  Serial.print(x); // Printing cm distance value to serial monitor
  Serial.print(" cm and "); // Printing rest of the text to serial monitor
  Serial.print(y); // Printing m distance value to serial monitor
  Serial.println(" m"); // Printing rest of the text to serial monitor
}

void setup() {
  pinMode(trigPin, OUTPUT); // Assigning output port for trigger pin
  pinMode(echoPin, INPUT); // Assigning input port for echo pin
  Serial.begin(9600); // Calling serial monitor for data display
}

void loop() {
  digitalWrite(trigPin, LOW); // Clearing already storred code in Arduino from previous run (if
any)
  delayMicroseconds(2); // Defining duration for this clear code to run

  digitalWrite(trigPin, HIGH); // Assigning trigger pin with max voltage
  delayMicroseconds(10); // Assigning duration for the above trigger input code

  digitalWrite(trigPin, LOW); // Assigning trigger pin with zero voltage to stop code

  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time
in microseconds and stores in duration

  distance = duration * 0.034 / 2; // Using speed distance formulae to convert time into
distance using speed of sound
  mdistance = distance / 100; // Converting cm distance to m distance

  if ((distance > 30) && (distance < 45)) { // Checking cm distance to be above 30 and below 45
and execute below code when true
    Serial.println("SLOW!!"); // Printing text to serial monitor when above condition is true
  }
  if ((distance > 15) && (distance < 30)) { // Checking cm distance to be above 15 and below 30
and execute below code when true
    Serial.println("WATCH OUT!!"); // Printing text to serial monitor when above condition is
true
  }
  if ((distance > 0) && (distance < 15)) { // Checking cm distance to be above 0 and below 15
and execute below code when true
    Serial.println("STOP!!"); // Printing text to serial monitor when above condition is true
  }

  distanceFunction(distance, mdistance); // Calling function to call same print code and print
}
```





POST LAB TASKS

Q1: Write down the model number of your sensor and list the pins with their usage.

The HC-SR04 Ultrasonic Sensor has four pins: VCC, Trig (Trigger), Echo, and GND.

- VCC: Connects to the 5V pin on the Arduino for power.
- Trig (Trigger): Sends an ultrasonic signal to the object.
- Echo: Receives the ultrasonic signal reflected from the object.
- GND: Connects to the ground (GND) pin on the Arduino.

Q2: What is the speed and frequency of sound waves emitted by Sonar sensor?

The HC-SR04 Ultrasonic Sensor emits ultrasonic waves at a frequency of around 40 kHz, and the speed of these sound waves in air at room temperature is approximately 343 meters per second.

Q3: Why time is divided by 2 in the distance formula?

Time is divided by 2 in the distance formula because the sensor measures the total time taken by the ultrasonic waves to travel to the object and back. Dividing this total time by 2 gives the time taken for the wave to travel to the object only.

Q4: What will be the return value of pulseIn() if no wave is detected?

If no wave is detected by the pulseIn() function, it typically returns a value of 0. This indicates that the echo pulse wasn't received within the specified time duration.

Q5: Suggest any process that can made using Arduino and Ultrasonic Sensor which can be used in any industry for automation

The ultrasonic sensor offers flexible capabilities in experimental setups. For instance, it can be effectively utilized to calculate intervals, as demonstrated in experiments involving moving pendulums. By positioning the sensor strategically, it can detect the intervals of the pendulum's motion. As the pendulum swings past the sensor, creating a change in distance, the sensor registers this change. However, it's essential to note that when calculating the total count, it needs to be divided by 2 due to the pendulum swinging past the sensor every half cycle. Furthermore, integrating a time counter allows measurement of the duration of the pendulum's swinging motion until it comes to rest, providing valuable data for further analysis and experimentation.