# ARDUINO-BASED TOLL GATE SYSTEM

BY-

SARVAGYA SANJAY

SEM – 04

DEPT. OF ELECTRONICS AND COMMUNICATION

## CONCEPT OF TOLL GATES

- Toll gates, also known as toll plazas or toll booths, are physical checkpoints located on highways, bridges, tunnels, and expressways where vehicles are required to pay a fee, known as a toll, for using the road infrastructure.

National Highways Authority of India (NH-30)
Welcome to Patna - Bakhtiyarpur Tollway Pvt. Ltd.
LANE-2
LANE-3
LANE-4
LANE-5

## TYPES

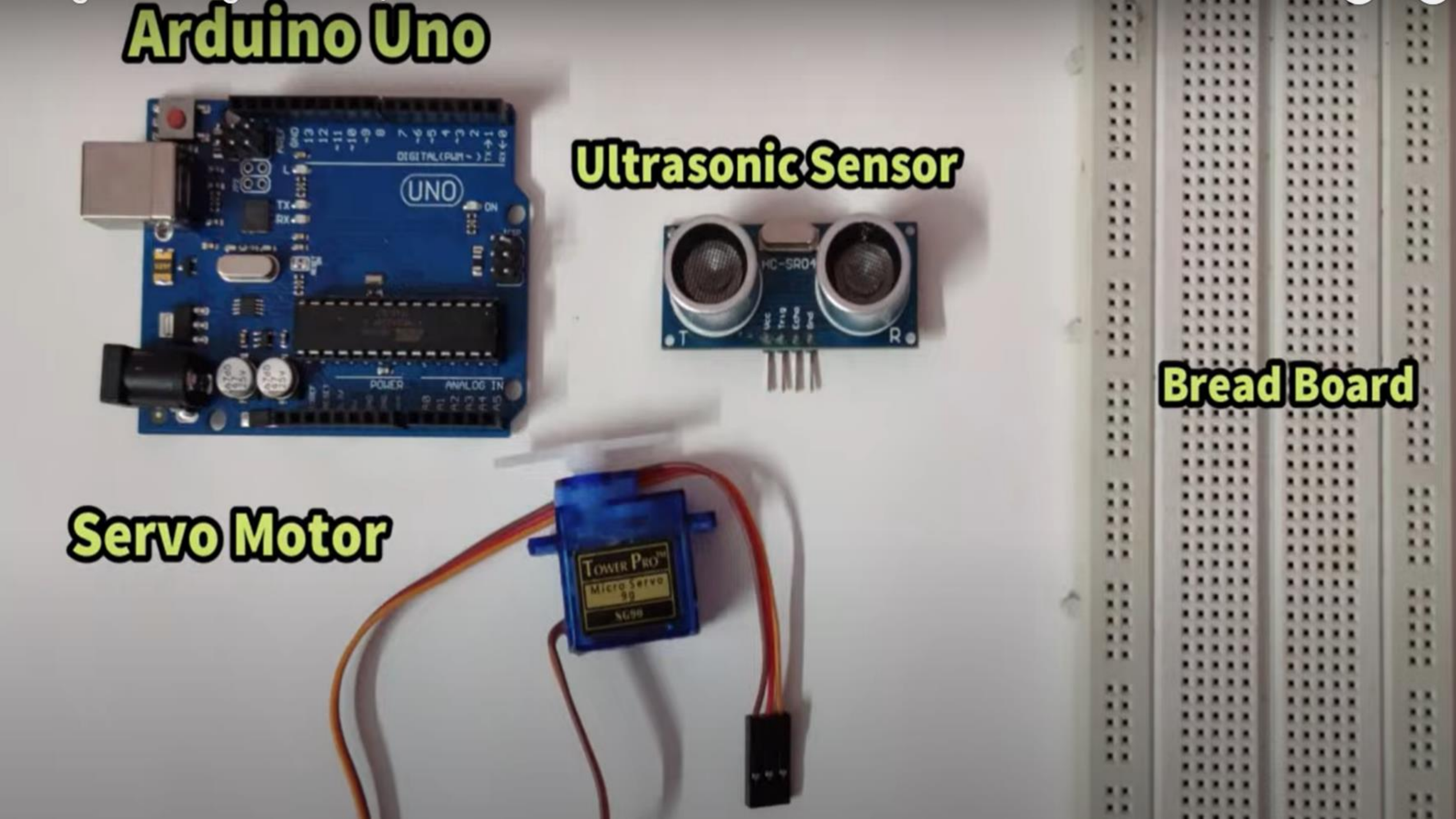
- **Manual toll gates**: Traditional toll booths where toll collectors manually collect tolls from drivers.
- **Automated toll gates**: Modern toll collection systems that use electronic toll collection (ETC) technologies, such as RFID (Radio Frequency Identification) or NFC (Near Field Communication), to automatically collect tolls from vehicles without the need for stopping.
- **Hybrid toll gates**: Toll gates that combine both manual and automated toll collection methods to accommodate different payment preferences and traffic volumes.

## FUNCTIONALITY

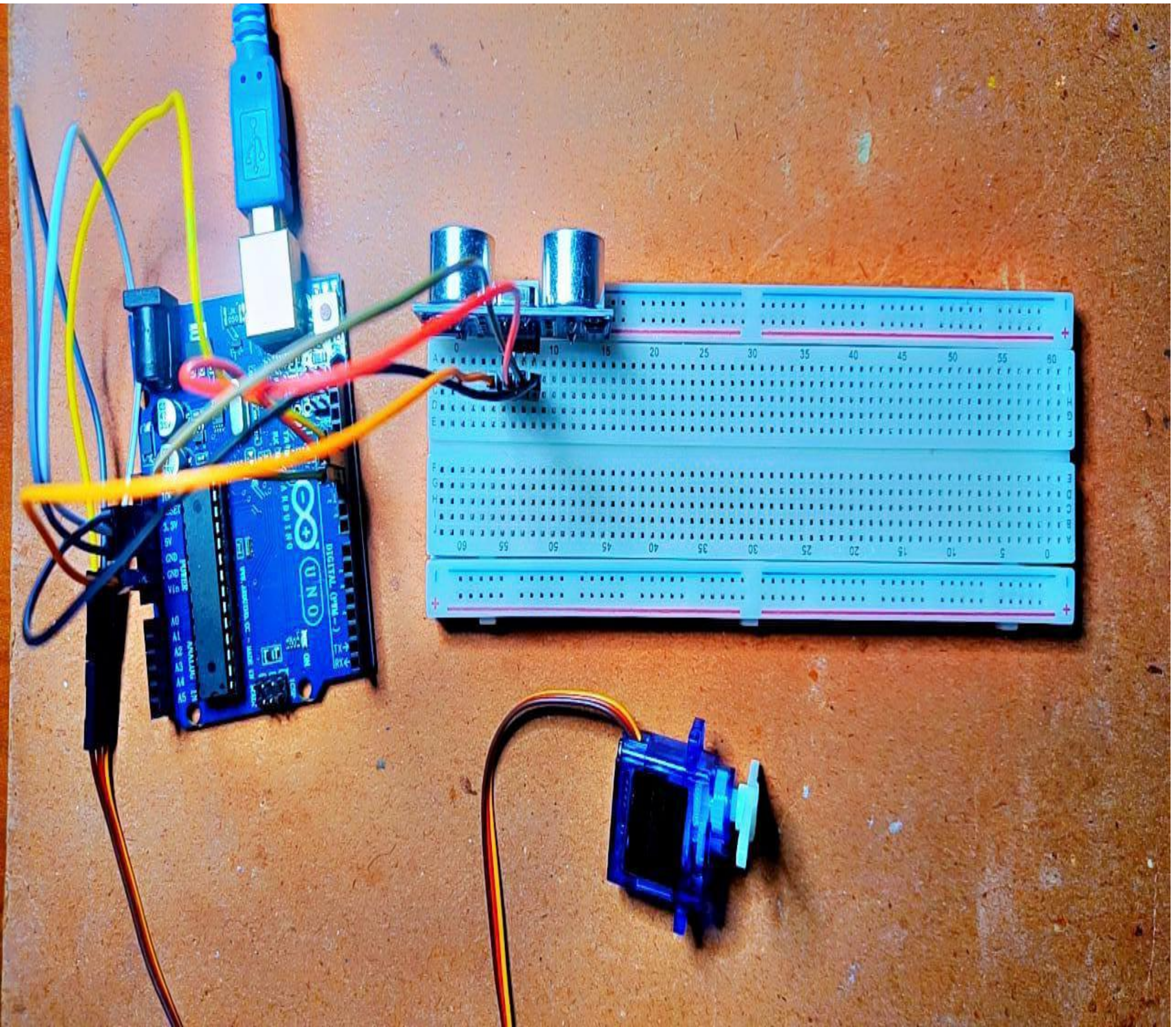
- Vehicle approaches the toll gate
- RFID or NFC tag is scanned for identification
- Toll amount is deducted from the user's account or collected in cash
- Barrier opens to allow passage.
- LCD display provide LCDs status updates

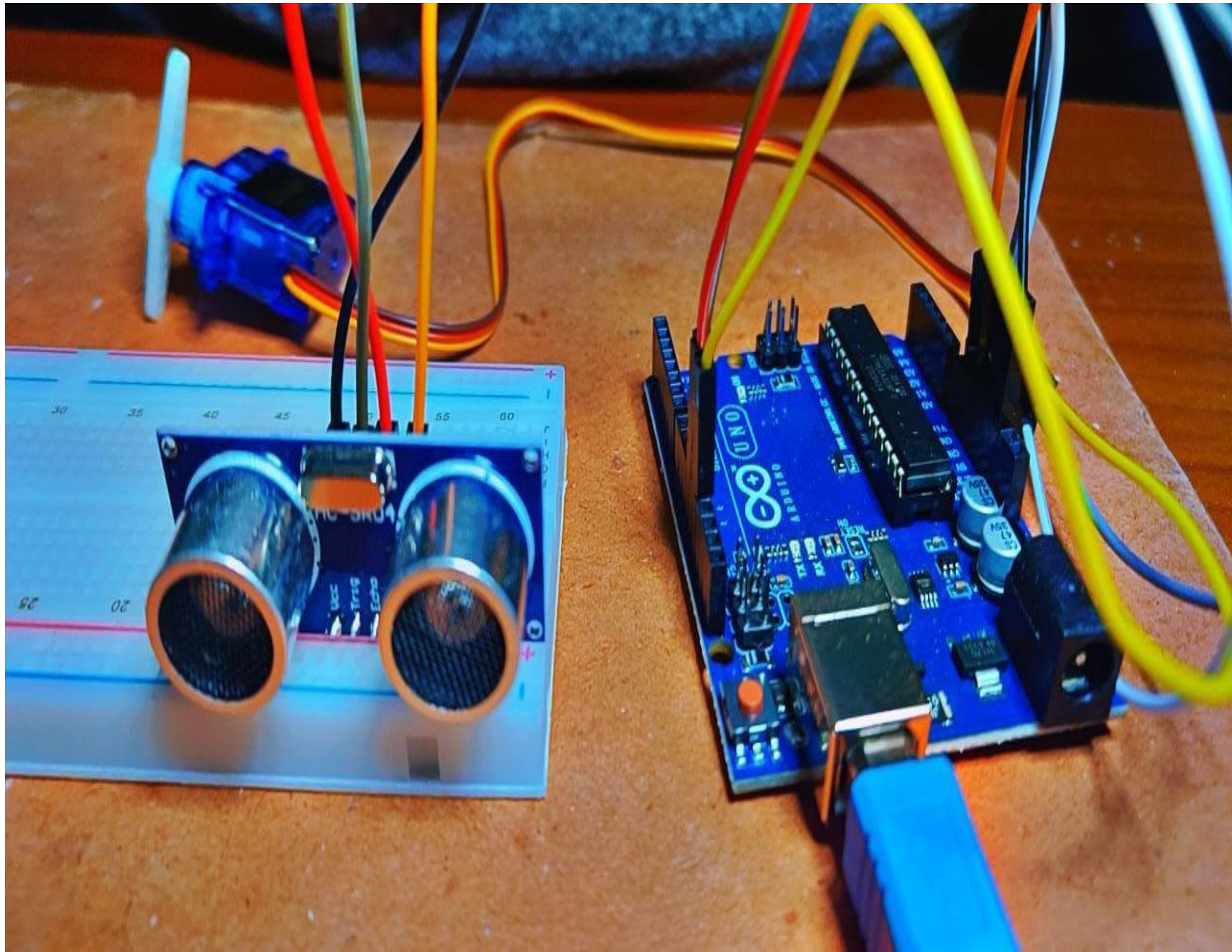
## COMPONENTS used:

- ***Arduino microcontroller***
- ***Servo motors for barrier control***
- ***Ultrasonic sensor***
- ***Bread board***
- ***Jumper cables***

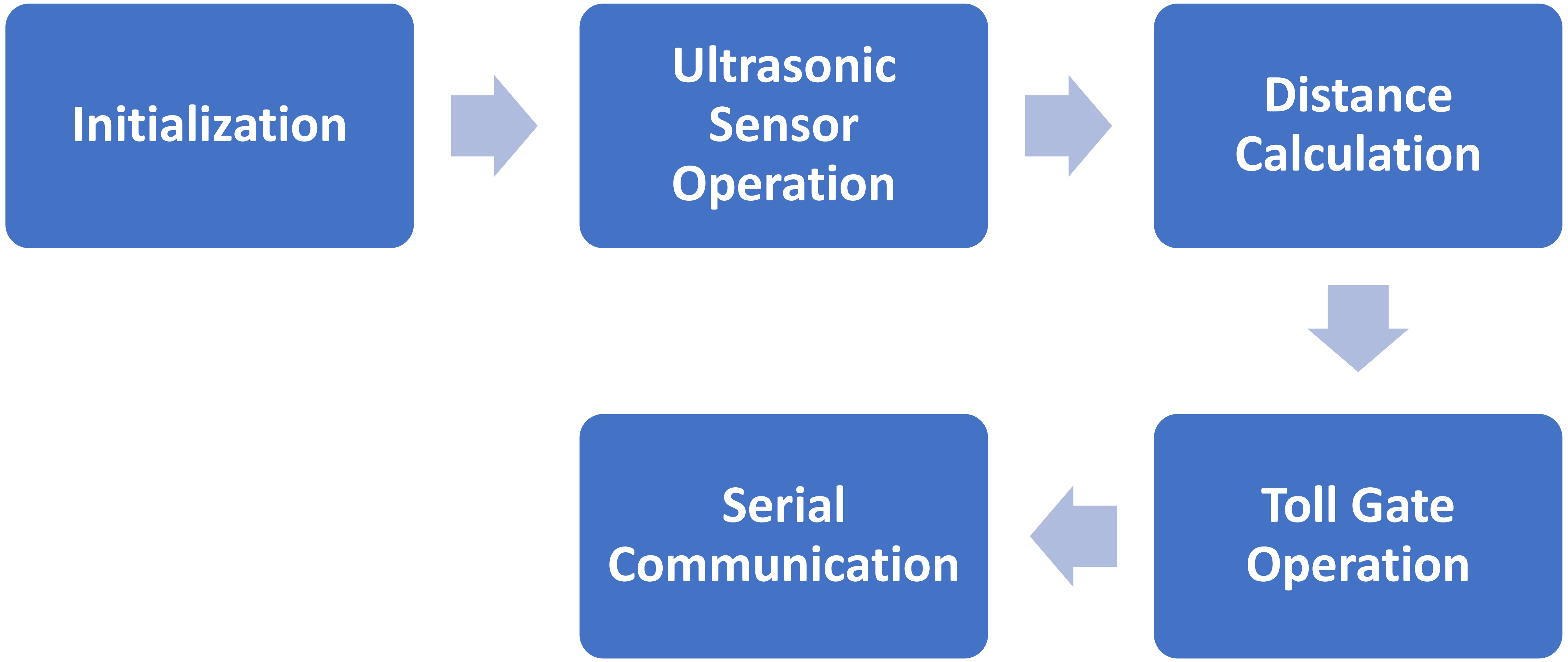

Arduino Uno
Ultrasonic Sensor
Bread Board
Servo Motor
UNO
HC-SR04
TOWER PRO
Micro Servo 9g
SG90

# CONSTRUCTION/CIRCUIT





## FLOW


Initialization
Ultrasonic Sensor Operation
Distance Calculation
Toll Gate Operation
Serial Communication

# Understanding the process

- **Initialization**

The Arduino board initializes the servo motor, ultrasonic sensor, and serial communication.

- **Ultrasonic Sensor Operation**

To measure the distance to an object (a vehicle), the Arduino sends a short pulse to the trigPin, which triggers the ultrasonic sensor to emit an ultrasonic pulse.

The pulse travels from the sensor to the object and back, and the echoPin receives the echo signal.

The Arduino calculates the duration of the pulse using the pulseIn() function, which measures the time it takes for the pulse to travel to the object and back.

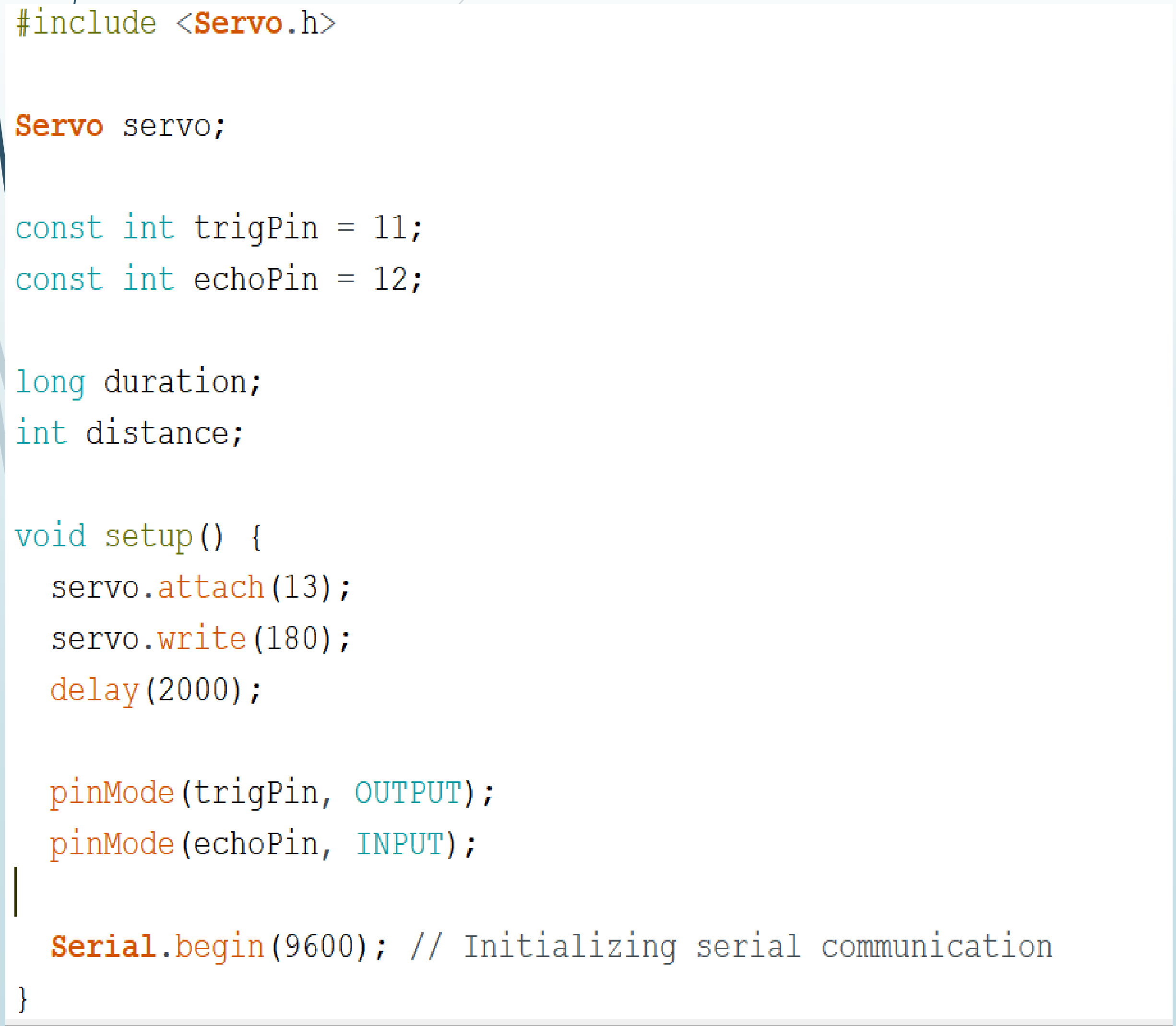
## Distance Calculation

- Using the duration of the pulse, the Arduino calculates the distance to the object (vehicle) from the ultrasonic sensor.
- The distance calculation is based on the speed of sound in air (approximately 0.034 cm/microsecond) and the round-trip time of the ultrasonic pulse.
- The formula `distance = duration * 0.034 / 2;` calculates the distance in centimeters.

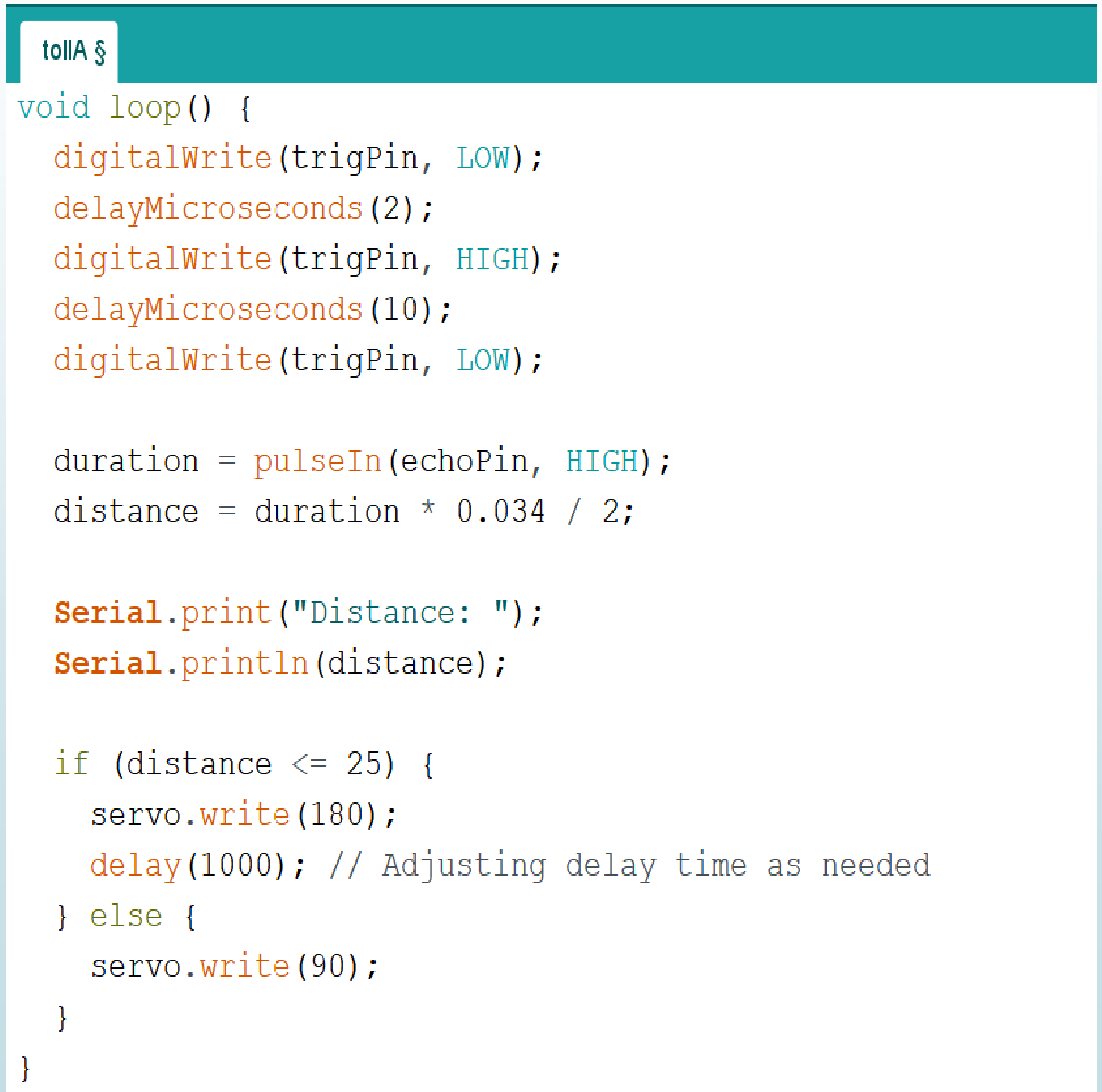
## Toll Gate Operation

- Once the distance to the vehicle is calculated, the Arduino checks if the distance is less than or equal to a predefined threshold (in this case, 25 cm).
- If the distance is within the threshold, indicating that a vehicle is approaching the toll gate, the Arduino activates the servo motor to open the toll gate barrier.
- The servo motor is connected to pin 13 of the Arduino board, and the `servo.write()` function is used to control the position of the servo motor.
- The toll gate remains open for a specified duration (1 second in this case) to allow the vehicle to pass through.
- After the delay, the servo motor returns the toll gate barrier to its closed position.

# ARDUINO CODE

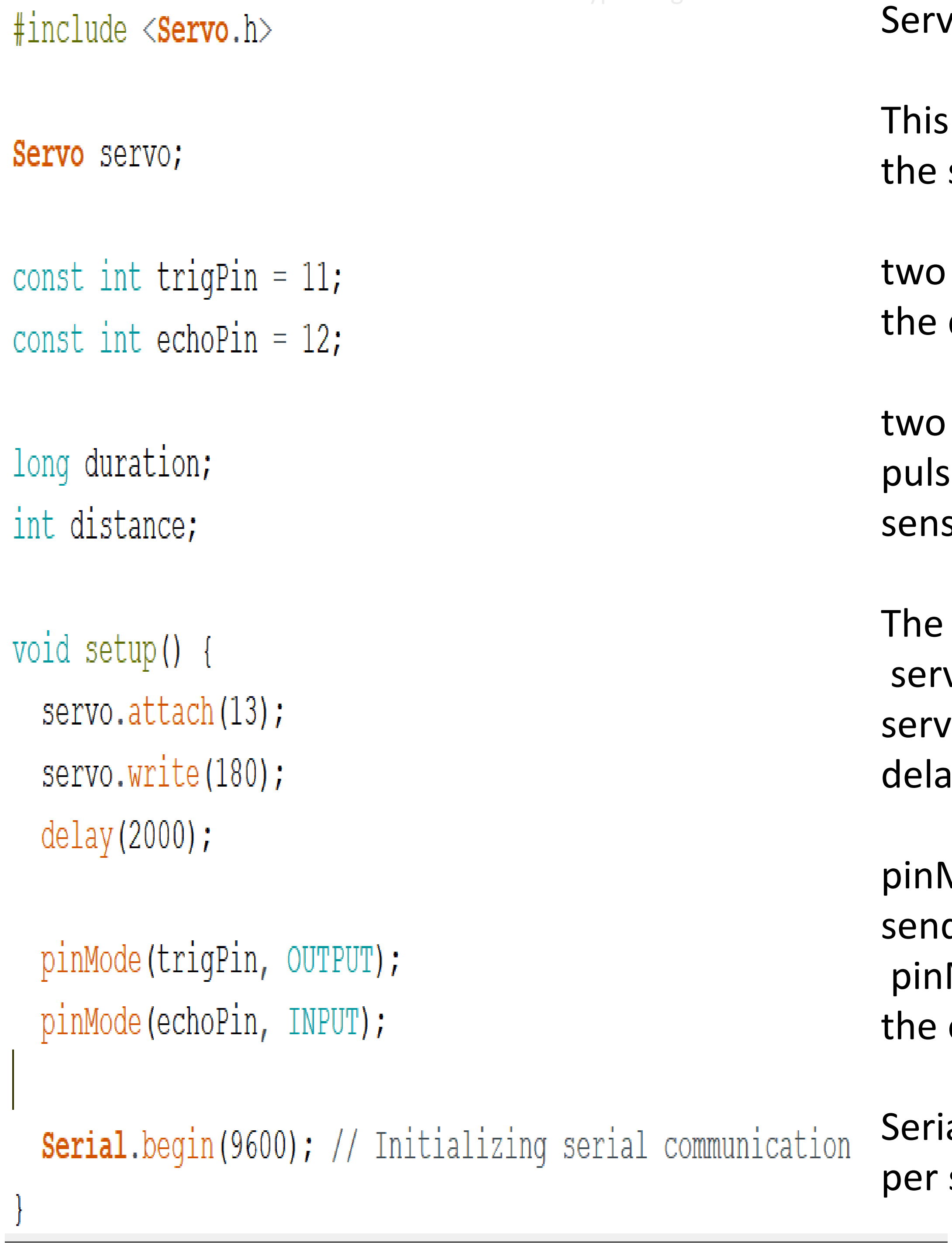


```
#include <Servo.h>

Servo servo;

const int trigPin = 11;
const int echoPin = 12;

long duration;
int distance;

void setup() {
  servo.attach(13);
  servo.write(180);
  delay(2000);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  Serial.begin(9600); // Initializing serial communication
}
```



```
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;

  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance <= 25) {
    servo.write(180);
    delay(1000); // Adjusting delay time as needed
  } else {
    servo.write(90);
  }
}
```

These lines declare two variables: **duration** of type long to store the duration of the ultrasonic pulse, and **distance** of type int to store the calculated distance from the sensor to an object.



```
#include <Servo.h>

Servo servo;

const int trigPin = 11;
const int echoPin = 12;

long duration;
int distance;

void setup() {
  servo.attach(13);
  servo.write(180);
  delay(2000);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  Serial.begin(9600); // Initializing serial communication
}
```

Servo library, which allows you to control servo motors with an Arduino.

This line declares a Servo object named servo, which will be used to control the servo motor connected to the Arduino.

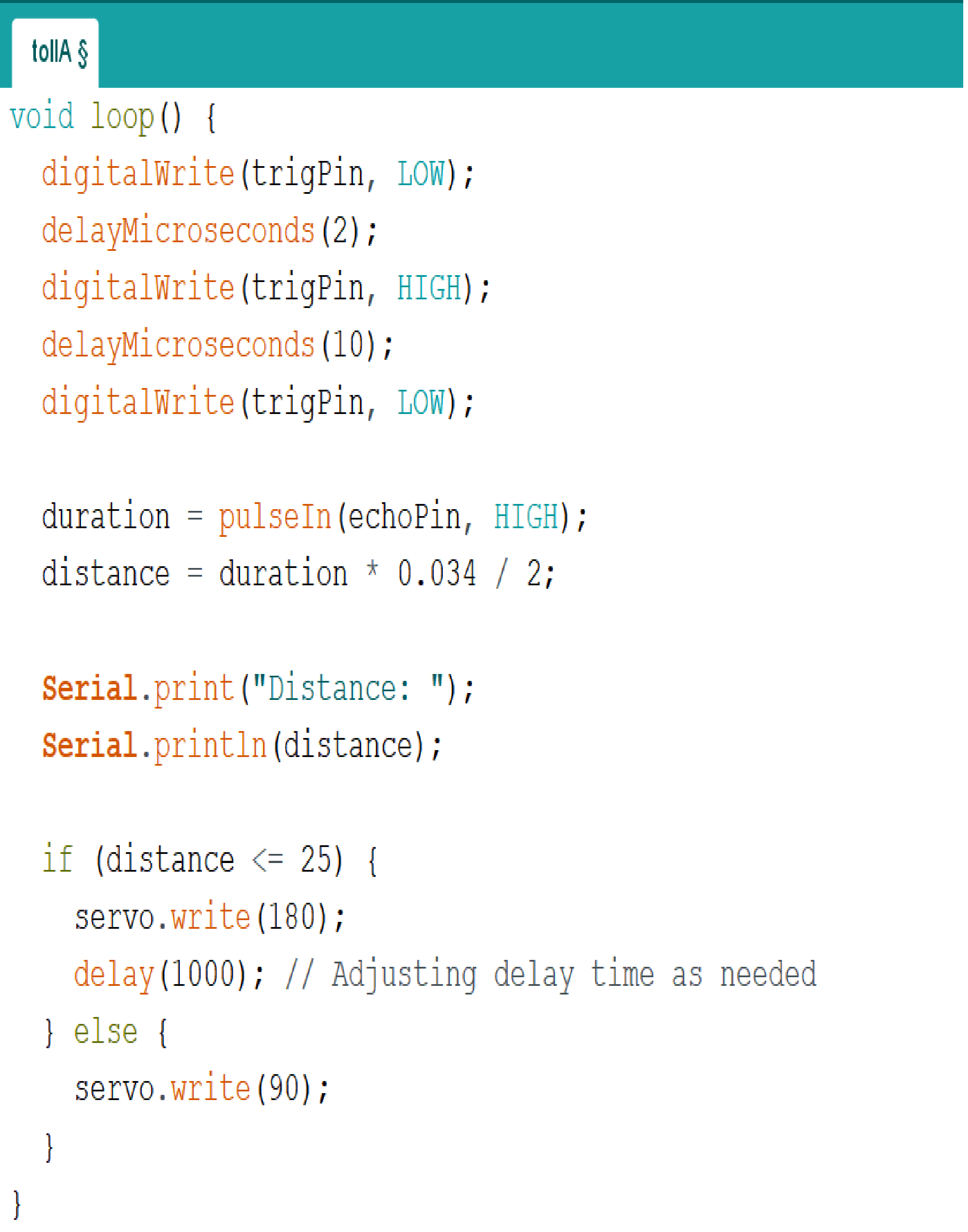
two constant integer variables trigPin and echoPin, which are used to define the digital pins connected to the ultrasonic sensor module.

two variables: duration of type long to store the duration of the ultrasonic pulse, and distance of type int to store the calculated distance from the sensor to an object.

The setup() function is called once when the Arduino starts.
servo.attach(13); attaches the servo motor to pin 13 on the Arduino board.
servo.write(180); positions the servo motor at 180 degrees.
delay(2000); pauses the program for 2000 milliseconds (2 seconds).

pinMode(trigPin, OUTPUT); configures the trigPin as an output pin for sending ultrasonic pulses.
pinMode(echoPin, INPUT); configures the echoPin as an input pin to receive the echo signal from the ultrasonic sensor.

Serial.begin(9600); initializes serial communication at a baud rate of 9600 bits per second.



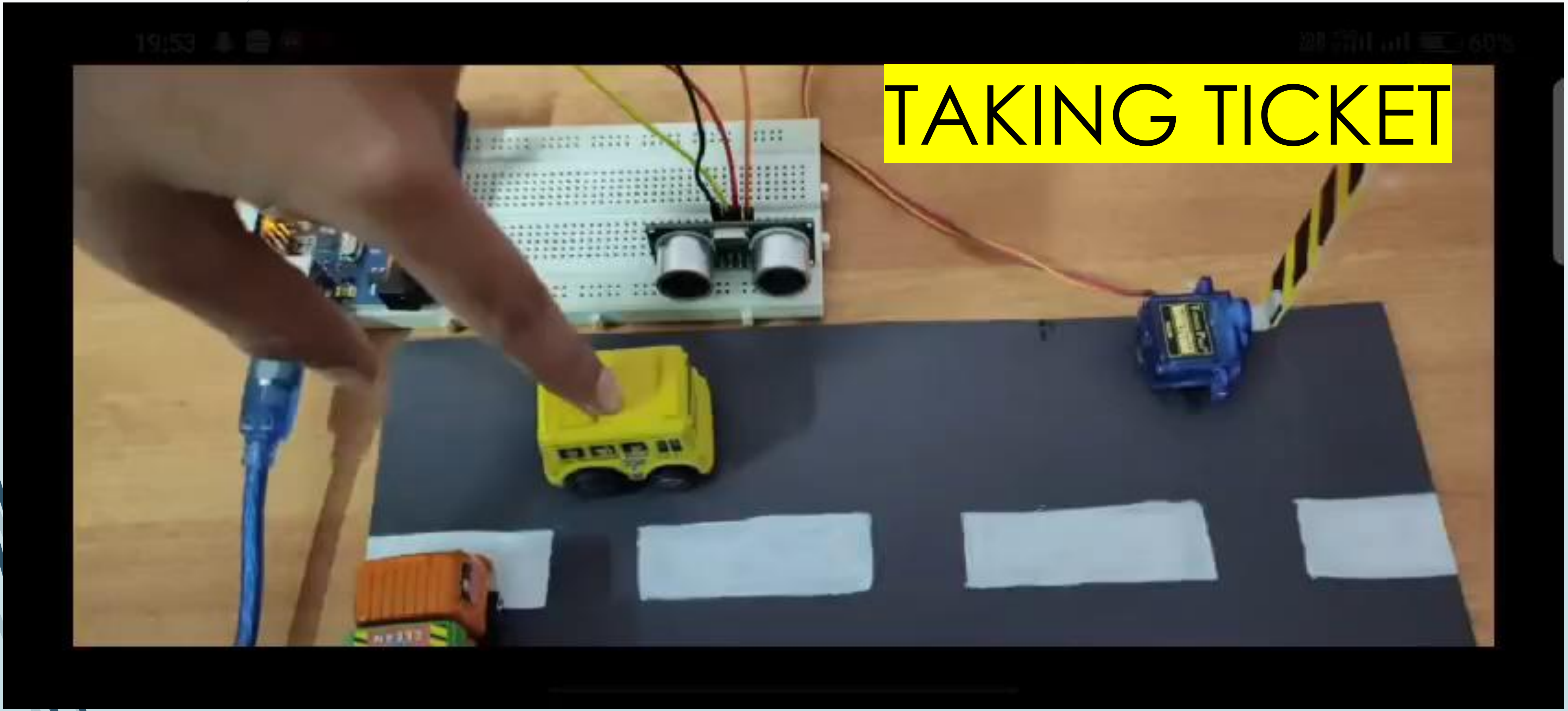
```
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;

  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance <= 25) {
    servo.write(180);
    delay(1000); // Adjusting delay time as needed
  } else {
    servo.write(90);
  }
}
```

The loop() function is continuously executed after the setup() function.
digitalWrite(trigPin, LOW); sets the trigger pin (trigPin) to a low state.
delayMicroseconds(2); pauses for 2 microseconds.
digitalWrite(trigPin, HIGH); sets the trigger pin (trigPin) to a high state.
delayMicroseconds(10); pauses for 10 microseconds.
digitalWrite(trigPin, LOW); sets the trigger pin (trigPin) to a low state.

The duration variable stores the time it takes for the ultrasonic pulse to travel from the sensor to the object and back, measured in microseconds.
distance = duration * 0.034 / 2; calculates the distance based on the duration of the pulse and stores it in the distance variable.

Serial.print("Distance: "); prints the text "Distance: " to the serial monitor.
Serial.println(distance); prints the calculated distance value to the serial monitor.
if distance is less than or equal to 25 units If true, it moves the servo motor to 180 degrees and waits for 1 second. Otherwise, it moves the servo motor to 90 degrees.

## PROCEDURE



THANK YOU