ABSTRACT

This project presents the design and implementation of an **Automatic Barrier Gate System** utilizing an **Arduino Uno** microcontroller, an **HC-SR04 ultrasonic sensor**, and an **SG90 servo motor**. The system is developed to automate vehicle access control by detecting approaching vehicles and operating a barrier arm without human intervention. The ultrasonic sensor continuously measures the distance to objects in front of it by emitting high-frequency sound waves and calculating the time taken for the echoes to return. When a vehicle is detected within a predefined range of 30 centimeters, the Arduino commands the servo motor to rotate, thereby opening the barrier. After allowing sufficient time for the vehicle to pass, the servo motor returns the barrier to its closed position.

This project aims to provide a cost-effective, reliable, and energy-efficient solution for managing vehicular access in parking lots, gated communities, industrial sites, and other secured areas. The system demonstrates seamless hardware integration and real-time processing, showcasing the practical application of embedded systems in automation. Furthermore, the project lays a foundation for future enhancements, such as wireless connectivity, security integration, and smart control features, to create a more robust and versatile automatic barrier system. Through this project, the understanding of sensor interfacing, motor control, and microcontroller programming is deepened, providing valuable insights into modern microprocessor-based system design.

AUTOMATIC BARRIER GATE SYSTEM

In the modern era of automation and smart infrastructure, controlling and managing vehicular access has become an essential aspect of ensuring security, efficiency, and convenience. Manual gates and barrier systems are often time-consuming, labor-intensive, and prone to human error. To overcome these challenges, this project introduces an **Automatic Barrier Gate System** developed using the **Arduino Uno** microcontroller platform.

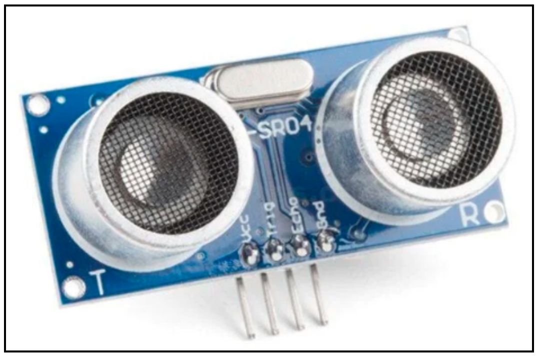
The system utilizes an **ultrasonic sensor (HC-SR04)** to detect the presence of an approaching vehicle and a **servo motor (SG90)** to operate the barrier gate accordingly. When the ultrasonic sensor detects a vehicle within a specified range (30 cm), it sends a signal to the Arduino, which then actuates the servo motor to open the barrier. After a short delay, allowing the vehicle to pass, the barrier automatically returns to its closed position. This entire process is continuous, efficient, and fully automated.

The project showcases the fundamental principles of **sensor interfacing**, **actuator control**, and **real-time decision-making** using microcontroller-based systems. It emphasizes the practical applications of **embedded systems** in solving everyday problems, particularly in areas like traffic management, parking automation, and security access control.

COMPONENTS USED

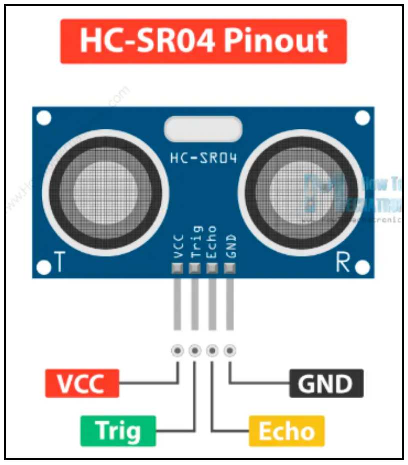
- Arduino Uno (ATmega328P)
- Ultrasonic Sensor (HC-SR04)
- Servo Motor (SG90)
- Jumper Wires
- Breadboard
- USB Cable (Direct AC Power Supply)

ULTRA SONIC SENSOR HC-SR04



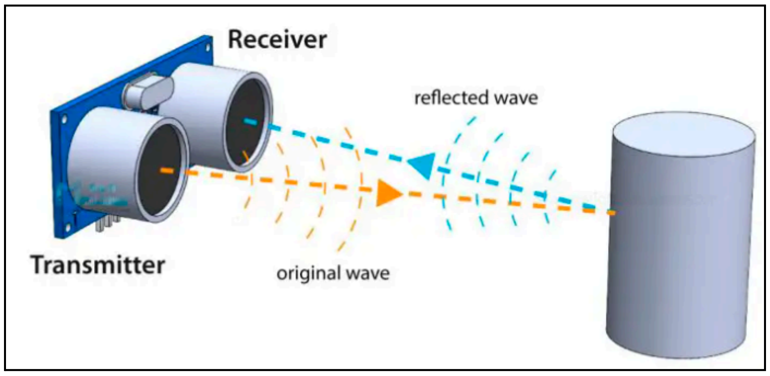
The HC-SR04 is an affordable and easy-to-use distance measuring sensor that has a range from 2cm to 400cm (about an inch to 13 feet). The sensor is composed of two ultrasonic transducers. One is a transmitter, which outputs ultrasonic sound pulses, and the other is a receiver, which listens for reflected waves. It’s basically a SONAR which is used in submarines for detecting underwater objects.

The sensor has 4 pins. ***VCC*** and ***GND*** go to ***5V*** and ***GND*** pins on the Arduino, and the ***Trig*** and ***Echo*** go to any digital Arduino pin. Using the ***Trig*** pin we send the ultrasound wave from the transmitter, and with the ***Echo*** pin we listen for the reflected signal.

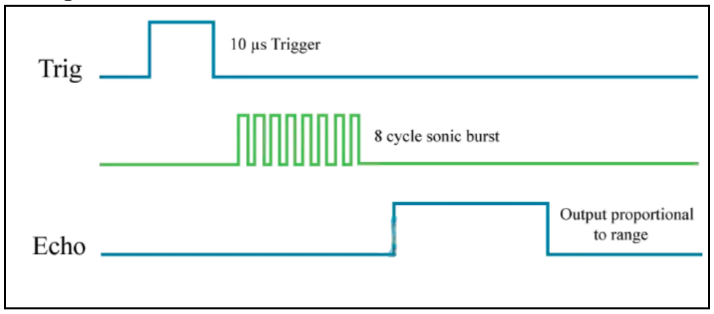


How the HC-SR04 Ultrasonic Distance Sensor Works?

It emits an ultrasound at 40,000 Hz, which travels through the air and if there is an object or obstacle on its path. It will bounce back to the module. Considering the travel time and the speed of sound, you can calculate the distance.



To generate the ultrasound, we need to set the Trig pin to a High State for 10 µs. That will send out an 8-cycle ultrasonic burst, which will travel at the speed of sound. The Echo pin goes high right away after that 8-cycle ultrasonic burst is sent, and it starts listening or waiting for that wave to be reflected from an object. If there is no object or reflected pulse, the Echo pin will time out after 38ms and return to the low state.



If we receive a reflected pulse, the Echo pin will go down sooner than those 38ms. According to the amount of time the Echo pin was HIGH, we can determine the distance the sound wave traveled, thus the distance from the sensor to the object.
For that purpose, we are using the following basic formula for calculating distance:

Distance = Speed x Time

We actually know both the speed and the time values. The time is the amount of time the Echo pin was HIGH, and the speed is the speed of sound, which is 340m/s. There's one additional step we need to do, and that's divide the end result by 2. That's because we are measuring the duration the sound wave needs to travel to the object and bounce back.

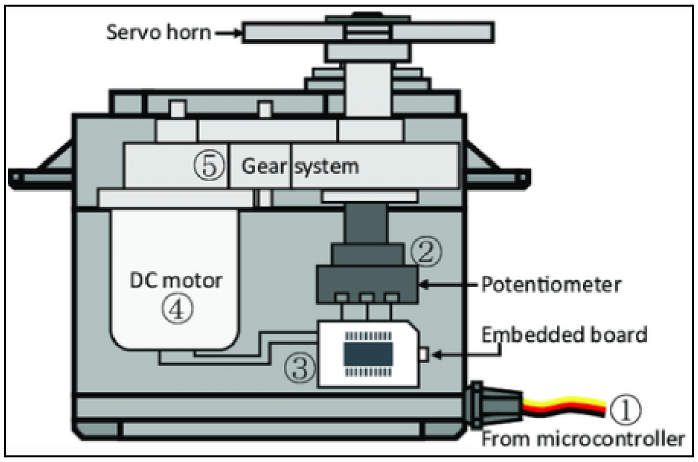
SERVO MOTORS SG90

Servo motors are devices that can rotate to a specific angle or position. They can be used to move robotic arms, steering wheels, camera gimbals, etc. Servo motors have three wires: power, ground and signal. The power wire is usually red and should be connected to the 5V pin on the Arduino board. The ground wire is usually black or brown and should be connected to a ground pin on the board. The signal wire is usually yellow or orange and should be connected to a PWM pin on the board.

Principle

A servo is generally composed of the following parts: case, shaft, gear system, potentiometer, DC motor, and embedded board.
It works like this:

- The microcontroller sends out PWM signals to the servo, and then the embedded board in the servo receives the signals through the signal pin and controls the motor inside to turn.
- As a result, the motor drives the gear system and then drives the shaft after deceleration.
- The shaft and potentiometer of the servo are connected.
- When the shaft rotates, it drives the potentiometer, so the potentiometer outputs a voltage signal to the embedded board.
- Then the board determines the direction and speed of rotation based on the current position, so it can stop exactly at the right position as defined and hold there.



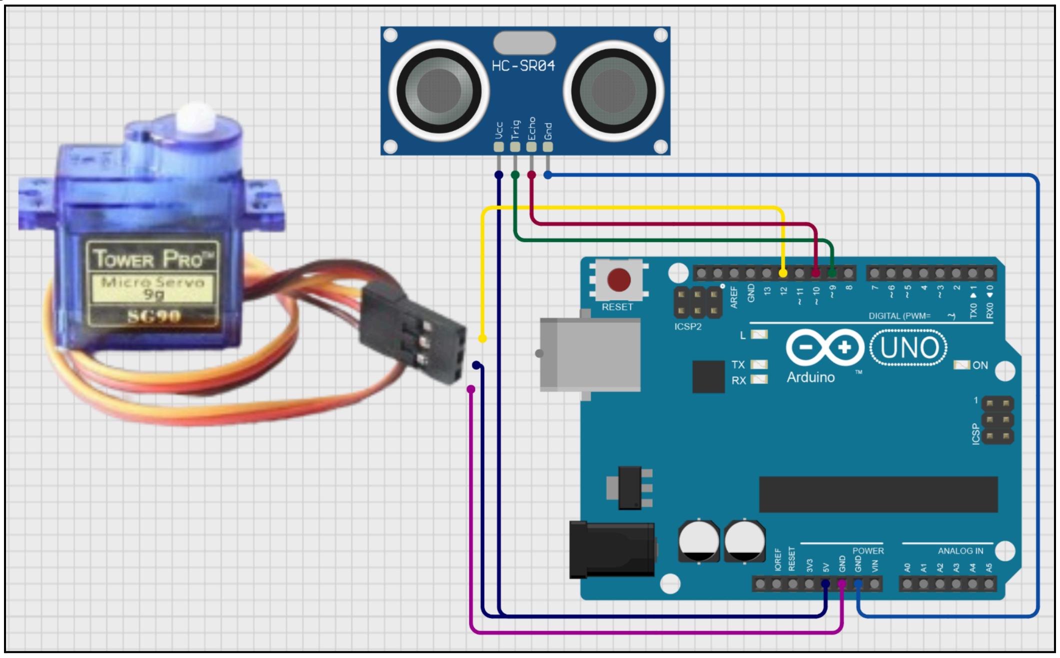
Work Pulse

The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse width Modulation.

- The servo expects to see a pulse every 20 ms. The length of the pulse will determine how far the servo turns.
- For example, a 1.5ms pulse will make the servo turn to the 90 degree position (neutral position).
- When a pulse is sent to a servo that is less than 1.5 ms, the servo rotates to a position and holds its output shaft some number of degrees counterclockwise from the neutral point.
- When the pulse is wider than 1.5 ms the opposite occurs.
- The minimal width and the maximum width of pulse that will command the servo to turn to a valid position are functions of each servo.
- Generally the pulse will be about 0.5 ms ~ 2.5 ms wide.

INTERFACING HC-SR04 & SG90 WITH ARDUINO UNO

Here's our project's schematic:



In this project, the **HC-SR04 Ultrasonic Sensor** and the **SG90 Servo Motor** are interfaced with the **Arduino Uno** to create an automated vehicle detection and barrier control system. The HC-SR04 sensor is responsible for detecting the presence of a vehicle by measuring the distance to the nearest object using ultrasonic sound waves. It operates using two main pins: the **Trigger (Trig)** and the **Echo**. The **Trig pin** (connected to digital pin 9 of the Arduino) is used to send a 10-microsecond pulse that initiates the transmission of ultrasonic waves. When these waves hit an object, they reflect back to the sensor, and the **Echo pin** (connected to digital pin 10) goes HIGH for the duration it takes for the echo to return. This duration is measured by the Arduino and used to calculate the distance to the object using the speed of sound.

The **SG90 servo motor**, which physically controls the barrier arm, is interfaced with the Arduino using digital pin 12. The servo motor requires a PWM (Pulse Width Modulation) signal to rotate to a specific angle. It has three wires: **Signal (Orange)** connected to pin 12, **Power (Red)** connected to the 5V pin, and **Ground (Brown)** connected to the GND pin on the Arduino. The **Servo.h** library is used in the code to control the motor, allowing it to rotate to desired positions by simply passing an angle value. When the sensor detects a vehicle within a predefined threshold distance (e.g., less than 30 cm), the servo motor is activated to open the barrier (by rotating up to 120 degrees) and then return to its original position (0 degrees) after a short delay, thus closing the gate.

This entire setup is powered directly from the Arduino Uno, which is typically connected to a computer or power bank via USB. The schematic diagram illustrates all the electrical connections between the components and the microcontroller, providing a clear overview of how the system is implemented.

WORKING PRINCIPLE

The system functions through continuous distance monitoring and servo-based barrier control. Here's a step-by-step breakdown:

Ultrasonic Sensing

- The **HC-SR04 Ultrasonic Sensor** sends out high-frequency sound pulses via the **Trigger pin**.
- These pulses travel through the air and reflect back upon hitting an object (e.g., a vehicle).
- The **Echo pin** receives the reflected sound waves.
- The **Arduino Uno** calculates the **time duration** of the echo and converts it into **distance** using the formula:

distance = (duration / 2) * speed_of_sound

Vehicle Detection Logic

- The system checks whether the measured distance is **less than 30 cm**.
- If true, it concludes that a vehicle is detected in front of the sensor.

Barrier Gate Operation

- The **SG90 Servo Motor** receives a signal from Arduino to **rotate from 0° to 120°**, lifting the barrier gate.
- A **short delay** is introduced to allow the vehicle to pass through the gate.

Resetting the Barrier

- After the delay, the Arduino commands the servo to **rotate back from 120° to 0°**, lowering the barrier.
- This completes one full cycle of detection and gate operation.

Continuous Monitoring

- The system continuously repeats this process, checking for approaching vehicles and responding automatically.

CODE EXPLANATION

Pin and Object Initialization

```
const int trigPin = 9;
const int echoPin = 10;
Servo myservo;
```

- trigPin and echoPin are constants representing the digital pins connected to the HC-SR04 ultrasonic sensor's Trigger and Echo pins, respectively.
- myservo is an instance of the Servo class used to control the SG90 servo motor.

Setup Function

```
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  myservo.attach(12);
}
```

- The setup() function runs once when the Arduino starts.
- pinMode(trigPin, OUTPUT) sets the Trigger pin as an output so the Arduino can send pulses.
- pinMode(echoPin, INPUT) sets the Echo pin as input to read the returning pulse from the ultrasonic sensor.
- myservo.attach(12) attaches the servo motor control signal wire to pin 12 on the Arduino, enabling servo control.

Loop Function

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

This block sends an ultrasonic pulse:

- The Trigger pin is first set LOW for 2 microseconds to ensure a clean signal start.
- Then it is set HIGH for 10 microseconds, which triggers the ultrasonic sensor to emit a sound pulse.
- Finally, the Trigger pin is set LOW again to end the pulse.

Measuring the Echo Pulse Duration

```
long duration = pulseIn(echoPin, HIGH);
```

- pulseIn() measures the length of time (in microseconds) the Echo pin stays HIGH.
- This duration corresponds to the time taken by the ultrasonic pulse to travel to the object and back.

Calculating Distance

```
float distance = (duration / 2.0) * 0.0343;
```

- The duration is divided by 2 to get the time for the pulse to reach the object (one-way).
- It is multiplied by the speed of sound in cm/µs (0.0343) to calculate the distance to the object in centimeters.

Condition to Detect Object within Range

```
if (distance > 0 && distance < 30) {
```

- The condition checks if the measured distance is greater than zero and less than 30 cm.
- If true, it indicates an object (e.g., a car) is within the sensor's detection range.

Opening the Barrier

```
 for (int pos = 0; pos <= 120; pos++) {
      myservo.write(pos);
      delay(10);
    }
```

- This for loop gradually moves the servo from 0 degrees to 120 degrees.
- myservo.write(pos) sets the servo position.
- delay(10) adds a small delay to slow the rotation for smooth movement.
- This rotation opens the barrier arm.

Pause with Barrier Open

```
 delay(500);
```

- After reaching 120°, the system pauses for 500 milliseconds to keep the barrier open.

Closing the Barrier

```
for (int pos = 120; pos >= 0; pos--) {
      myservo.write(pos);
      delay(10);
    }
```

- This loop moves the servo back from 120 degrees to 0 degrees, closing the barrier.
- Again, a small delay between each degree movement ensures smooth operation.

Pause with Barrier Closed

```
 delay(500);
  }
}
```

- The system pauses for another 500 milliseconds after closing the barrier before restarting the detection loop.

The code repeatedly triggers the ultrasonic sensor to measure the distance. If an object is detected within 30 cm, the servo motor opens the barrier smoothly to 120°. After a short delay, it closes the barrier back to 0°.The process then repeats indefinitely.

FEATURES

- **Automatic Vehicle Detection:** Uses ultrasonic sensing to detect approaching vehicles within a 30 cm range.
- **Smooth Barrier Operation:** Controls a servo motor to open and close the barrier smoothly and precisely.
- **Low Power Consumption:** Utilizes Arduino Uno and simple components, ensuring energy efficiency.
- **Compact and Cost-Effective:** Built with readily available, low-cost components suitable for various access control applications.
- **Real-Time Operation:** Continuously monitors and responds to vehicle presence without manual intervention.
- **Adjustable Detection Range:** Detection distance can be modified by changing the threshold value in the code.
- **Easy Integration:** Can be integrated into existing gate systems with minimal modification.
- **Reliable and Durable:** Uses proven sensor and motor components for consistent operation.

APPLICATIONS

- **Parking Lot Automation:** Automates vehicle entry and exit in parking garages, reducing the need for manual attendants and improving traffic flow.
- **Gated Residential Communities:** Enhance security by controlling vehicle access to private neighborhoods or housing societies.
- **Industrial and Commercial Facilities:** Manages vehicle access in factories, warehouses, and business complexes to ensure authorized entry.
- **Toll Plazas:** Can be adapted to control barriers at toll booths, enabling smooth and automated vehicle passage.
- **Campus Security:** Controls vehicle entry in educational institutions or corporate campuses to restrict unauthorized access.
- **Event Management:** Temporarily deployable for managing vehicle access during events, exhibitions, or festivals.
- **Smart Traffic Systems:** Forms part of intelligent transportation infrastructure to regulate traffic and improve road safety.
- **Restricted Areas:** Useful for controlling access in sensitive or restricted zones such as military bases or government buildings.

FUTURE EXTENSIONS

- **Integration with RFID or NFC:** Combine with RFID/NFC card readers for authorized access control, allowing only registered vehicles to open the barrier.
- **Wireless Communication:** Add wireless modules (e.g., Wi-Fi, Bluetooth, or GSM) for remote monitoring and control of the gate system.
- **Camera Integration:** Use cameras with image processing or license plate recognition to enhance security and automate logging of vehicle entries.
- **Mobile App Control:** Develop a smartphone app for remote operation, status monitoring, and notifications.
- **Multiple Sensor Setup:** Use additional sensors to detect vehicle direction and prevent false triggering or tailgating.
- **Solar Power Supply:** Implement solar panels to power the system for sustainable and off-grid operation.
- **Emergency Override:** Add manual override controls and emergency sensors (like fire alarms) to open the barrier automatically in critical situations.
- **Cloud Data Logging:** Store access logs and system data on the cloud for analysis and auditing.
- **Voice Control:** Incorporate voice commands through smart home assistants for hands-free gate operation.
- **Enhanced Safety Features:** Include obstacle detection to prevent the barrier from closing on vehicles or pedestrians.

REFERENCES

Below are the official datasheets and manuals of the components used in this project:

- **Arduino Uno**
  https://store.arduino.cc/products/arduino-uno-rev3
  Arduino Uno Datasheet (PDF)
- **Ultrasonic Sensor (HC-SR04)**
  HC-SR04 Datasheet (PDF)
- **Servo Motor (SG90)**
  SG90 Servo Motor Datasheet (PDF)
- **pulseIn() Arduino Reference**
  https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/

- **Servo Library Arduino Reference**
  https://www.arduino.cc/reference/en/libraries/servo/

GitHub REPOSITORY

All code, documentation, and files related to this project are hosted on GitHub:

GitHub Repository:
https://github.com/nehanaumankhan/Automatic-Barrier-Gate-System.git

CONCLUSION

The **Automatic Barrier Gate System** designed using Arduino Uno, the HC-SR04 ultrasonic sensor, and the SG90 servo motor demonstrates a practical and efficient solution for automated vehicle access control. By accurately detecting the presence of vehicles and controlling the barrier arm smoothly, this system reduces the need for manual intervention, enhances security, and improves traffic management in various environments such as parking lots, residential areas, and industrial premises.

The project highlights the effective interfacing of sensors and actuators with a microcontroller to build a responsive embedded system. Its simplicity, low cost, and adaptability make it a promising candidate for real-world applications. Moreover, the modular design allows for future enhancements, such as wireless communication, integration with smart access technologies, and improved safety features, to meet evolving user needs.

Overall, this project not only provides a functional prototype but also lays the groundwork for more advanced automatic gate systems, contributing to smarter and safer infrastructure development.