**Title**

Automated Toll Collection System with Ultrasonic Sensors

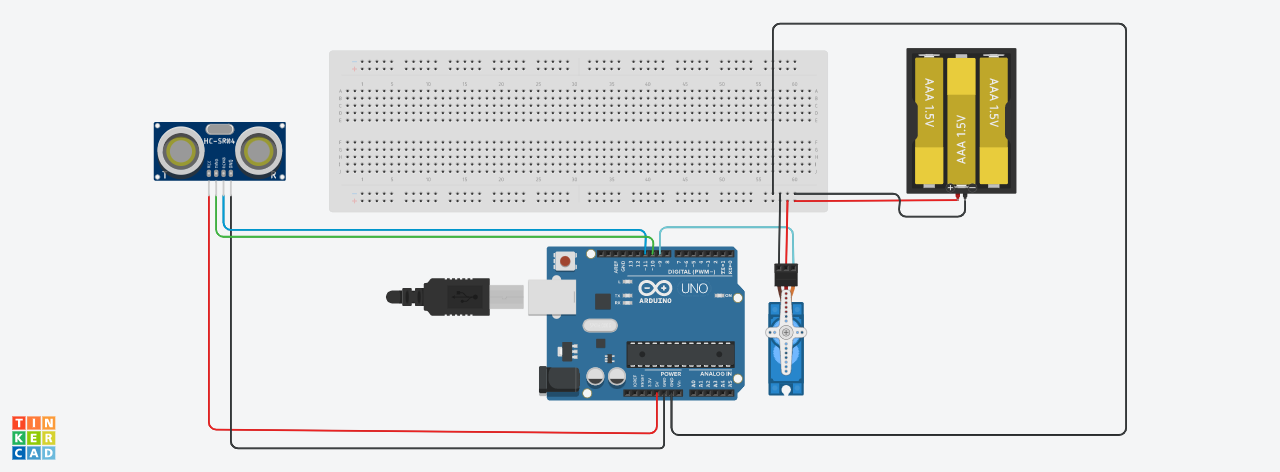
**Objective**

* + Develop an automated toll collection system that eliminates the need for manual intervention at toll booths. This system will utilize ultrasonic sensors to identify vehicles and an Arduino microcontroller for system control. The goal is to create an automated toll collection system that eliminates manual intervention at toll booths. It will employ ultrasonic sensors for vehicle identification and an Arduino microcontroller for system control. By doing so, the system enhances efficiency and accuracy while reducing the reliance on human involvement.
  + The project aims to implement a precise distance measurement technique using ultrasonic sensors to detect vehicles at the toll booth accurately. By leveraging this technology, the system can effectively distinguish between occupied and empty lanes. The ultrasonic sensors emit waves and analyze the reflected signals to determine the distance between the vehicle and the toll booth. This enables accurate detection of vehicle presence, allowing for efficient lane management. The implementation of this technique enhances the overall effectiveness and reliability of the toll system, optimizing traffic flow and improving the user experience at the toll booth.
  + Incorporate a servo motor system to simulate the opening and closing of the toll gate when a vehicle is successfully detected in front of the toll booth. This feature ensures regulated vehicle access and enhances the security of the toll booth. The project incorporates a servo motor system as a vital component to replicate the opening and closing of the toll gate in response to the successful detection of a vehicle in front of the toll booth. This feature serves multiple purposes, including ensuring regulated vehicle access and enhancing the overall security of the toll booth. When a vehicle is correctly identified and detected by the ultrasonic sensors, the servo motor system is activated. It triggers the mechanical movement of the toll gate, allowing the vehicle to pass through the toll booth. By simulating the opening and closing of the toll gate, the system controls the entry and exit of vehicles, maintaining a regulated flow of traffic and preventing unauthorized access. By integrating the servo motor system into the automated toll collection system, the project ensures a smooth and regulated process for vehicles passing through the toll booth. It contributes to the efficient management of traffic flow and provides a heightened level of security, allowing authorized vehicles to enter while deterring unauthorized access or potential risks.
  + Evaluate the efficiency and reliability of the toll system by assessing factors such as reaction time, accuracy of vehicle detection, and overall system stability. This evaluation will be conducted under various traffic scenarios and user loads to ensure the system performs effectively in different conditions.

**Apparatus**

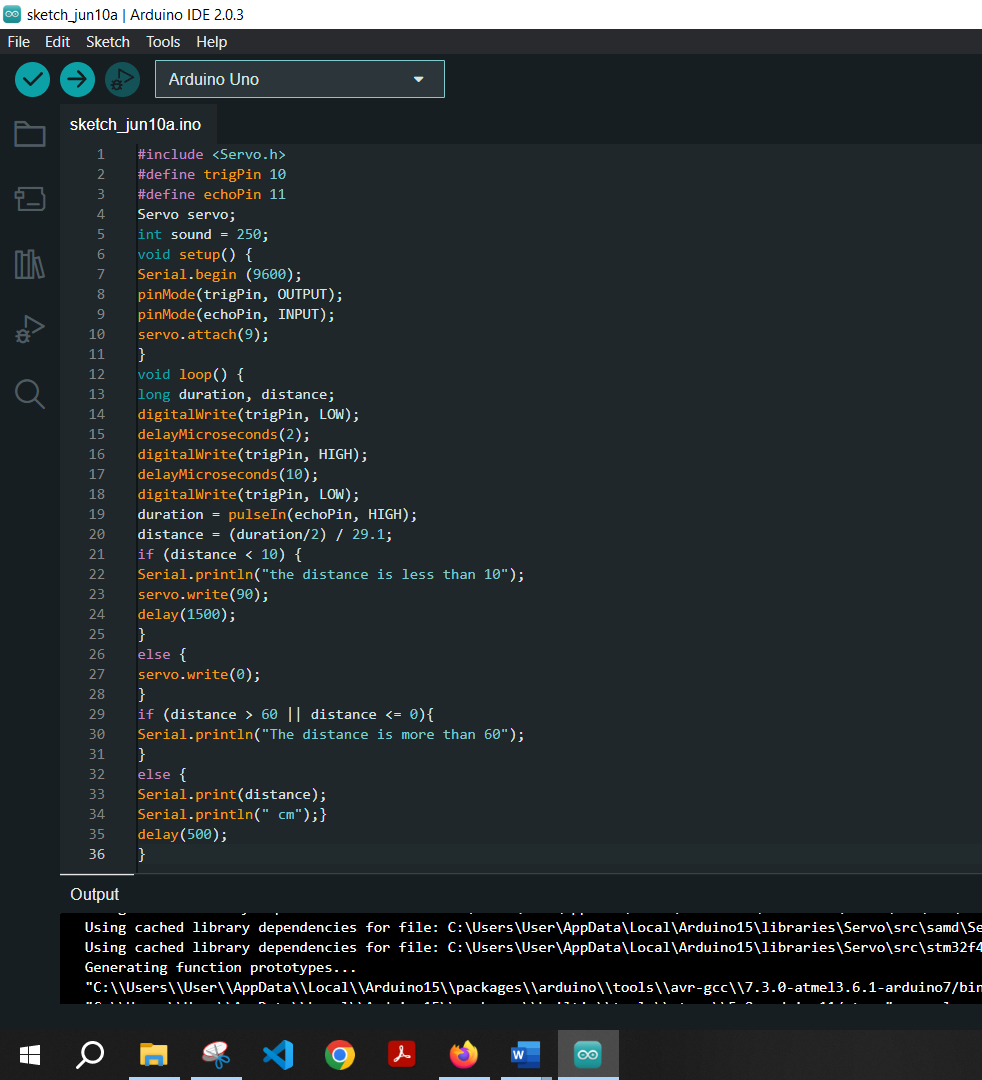
|  |  |  |
| --- | --- | --- |
| Serial No | Component | Quantity |
| 1. | Arduino Uno R3 | 1 |
| 2. | Positional Micro Servo | 1 |
| 3. | Ultrasonic Distance Sensor | 1 |
| 4. | 3.7v Battery | 1 |
| 5. | Breadboard | 1 |
| 6. | Jumper Wire | As required |
| 7. | Laptop | 1 |

**Working Diagram**



**Code**

**Arduino IDE Implementation**

****

#include <Servo.h>

#define trigPin 10

#define echoPin 11

Servo servo;

int sound = 250;

void setup() {

Serial.begin (9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

servo.attach(9);

}

void loop() {

long duration, distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) / 29.1;

if (distance < 10) {

Serial.println("the distance is less than 10");

servo.write(90);

delay(1500);

}

else {

servo.write(0);

}

if (distance > 60 || distance <= 0){

Serial.println("The distance is more than 60");

}

else {

Serial.print(distance);

Serial.println(" cm");

}

delay(500);

}

Code explaination

1. #include <Servo.h> This line includes the Servo library, which provides functions for controlling servo motors.
2. #define trigPin 10 This line defines a constant called trigPin and assigns it the value 10. It is used to specify the pin number connected to the trigger pin of the ultrasonic sensor.
3. #define echoPin 11 This line defines another constant called echoPin and assigns it the value 11. It is used to specify the pin number connected to the echo pin of the ultrasonic sensor.
4. Servo servo; This line declares a Servo object called servo. It will be used to control the servo motor.
5. int sound = 250; This line declares an integer variable called sound and assigns it the value 250. The purpose of this variable is not clear from the provided code snippet.
6. void setup() { ... } This is the setup function which runs once when the Arduino is powered on or reset. Here's what it does:
   * Serial.begin(9600); initializes the serial communication with a baud rate of 9600. This allows the Arduino to communicate with a computer via a serial connection, and you can use the Serial Monitor to debug or monitor the output.
   * pinMode(trigPin, OUTPUT); sets the trigPin (pin 10) as an output. This pin is responsible for sending the trigger signal to the ultrasonic sensor.
   * pinMode(echoPin, INPUT); sets the echoPin (pin 11) as an input. This pin is used to receive the echo signal from the ultrasonic sensor.
   * servo.attach(9); attaches the servo object servo to pin 9 of the Arduino. This pin will be used to control the servo motor.

That's the breakdown of the code up to the setup function. The rest of the code should include the loop function where the main logic of the project is implemented.

 void loop() { ... } This is the loop() function, which is executed repeatedly after the setup() function. Here's what it does:

 long duration, distance; These lines declare two variables of type long called duration and distance. They will be used to store the measured duration and calculated distance values.

 digitalWrite(trigPin, LOW); This line sets the trigPin (pin 10) to a LOW state. This ensures that the trigger pin of the ultrasonic sensor is initially low.

 delayMicroseconds(2); This line introduces a delay of 2 microseconds.

 digitalWrite(trigPin, HIGH); This line sets the trigPin to a HIGH state. This triggers the ultrasonic sensor to send an ultrasonic pulse.

 delayMicroseconds(10); This line introduces a delay of 10 microseconds.

 digitalWrite(trigPin, LOW); This line sets the trigPin back to a LOW state.

 duration = pulseIn(echoPin, HIGH); This line measures the duration it takes for the ultrasonic pulse to travel to the object and return. It uses the pulseIn() function, which waits for the echo pin (pin 11) to go HIGH and then returns the duration in microseconds.

 distance = (duration/2) / 29.1; This line calculates the distance based on the measured duration. The formula used here assumes that the speed of sound is approximately 343 meters per second (or 34300 centimeters per second). By dividing the duration by 2 and then dividing by 29.1, we obtain the distance in centimeters.

 if (distance < 10) { ... } This condition checks if the calculated distance is less than 10 centimeters.

 Serial.println("the distance is less than 10"); If the distance is less than 10 centimeters, this line prints a message to the Serial Monitor.

 servo.write(90); This line sets the servo motor's position to 90 degrees. It is an example action that can be performed based on the condition. You can modify this value to control the servo motor according to your requirements.

 delay(1500); This line introduces a delay of 1.5 seconds (1500 milliseconds).

 else { ... } If the distance is not less than 10 centimeters, the code execution continues here.

 servo.write(0); This line sets the servo motor's position to 0 degrees. It is an example action that can be performed based on the condition. You can modify this value to control the servo motor according to your requirements.

 if (distance > 60 || distance <= 0) { ... } This condition checks if the calculated distance is greater than 60 centimeters or if it's less than or equal to 0 (which can happen if the ultrasonic sensor doesn't detect any object). This condition checks if the calculated distance is greater than 60 centimeters or if it's less than or equal to 0. It handles situations where the distance measurement is out of range or invalid.

1. Serial.println("The distance is more than 60"); If the distance is greater than 60 centimeters or less than or equal to 0, this line prints a message to the Serial Monitor indicating that the distance is beyond the expected range or invalid.
2. else { ... } If the distance is within the valid range (greater than 0 and less than or equal to 60), the code execution continues here.
3. Serial.print(distance); This line prints the measured distance (stored in the distance variable) to the Serial Monitor without adding a newline character at the end.
4. Serial.println(" cm"); This line prints the string " cm" to the Serial Monitor, which adds a newline character. It completes the line by printing the distance value followed by the unit of measurement (cm).
5. delay(500); This line introduces a delay of 500 milliseconds (half a second) before starting the next iteration of the loop() function. This delay can help control the rate at which the measurements and actions are performed. You can adjust this value based on your project's requirements.

That's the breakdown of the code snippet. It demonstrates how to control the servo motor based on the measured distance and provides feedback through the Serial Monitor regarding the distance measured.

**Discussion**

The objective of the Automated Toll Collection System with Ultrasonic Sensors project was to address the drawbacks of manual toll collection processes by introducing an automated system. This project utilized ultrasonic sensors, an Arduino microcontroller, and other components to streamline the toll collection process and enhance user experience.

The incorporation of ultrasonic sensors for vehicle detection played a crucial role in the project. These sensors provided a reliable and accurate method of identifying vehicles at the toll booth. By measuring the distance between the car and the toll booth, the technology effectively determined whether a lane was occupied or empty. This innovation significantly reduced human errors and improved the efficiency of toll collection.

Additionally, a servo motor system was implemented to simulate the opening and closing of the toll gate, ensuring an extra layer of security and control. When a car was accurately detected, the toll gate opened, allowing the vehicle to pass through. This technology not only facilitated regulated vehicle entry but also enhanced the safety of the toll booth.

The evaluation phase focused on assessing the efficiency and reliability of the toll system. Various traffic scenarios and user loads were considered to examine factors such as response time, vehicle detection accuracy, and system stability. Through extensive testing and analysis of performance data, the project successfully demonstrated its efficiency and effectiveness. Several factors were examined during the evaluation, including response time, vehicle detection accuracy, and system stability. Response time refers to the speed at which the system reacts to a vehicle approaching the toll booth, calculates the toll, and opens the gate. Vehicle detection accuracy was assessed to ensure that the ultrasonic sensors correctly identified and classified vehicles passing through the toll booth, which is crucial for accurate toll calculation. System stability focused on determining the robustness and consistency of the toll system under different situations, ensuring that it operated reliably over an extended period.

To gather meaningful insights, extensive testing was conducted, and performance data was collected and analyzed. The project team likely performed numerous test runs, recorded data on the system's behavior, and analyzed the results to understand how well the toll system performed. The analysis of performance data allowed them to quantify the efficiency and effectiveness of the system in terms of accurate toll calculation, timely response, and overall reliability.

By successfully demonstrating its efficiency and effectiveness through the evaluation phase, the project team provided evidence that the automated toll system with ultrasonic sensors was capable of performing its intended functions reliably. This data-driven approach adds credibility to the project's outcomes and reinforces the system's viability for real-world implementation.

In conclusion, the Automated Toll Collection System with Ultrasonic Sensors project showcased the potential for automation and innovation in the field of toll collection. By leveraging ultrasonic sensors and an Arduino microcontroller, the project effectively addressed the challenges associated with manual toll collection, resulting in improved efficiency, enhanced user experience, and heightened security. The success of this project creates a foundation for future enhancements, scalability, and acceptance of automated toll systems in real-world applications.