IntelliGate

Automatic Car Gate System with Vehicle Identification (AI-Powered)

By: Nuriye Selcuk & Rukiye Selcuk

🔷 Overview

This project demonstrates how to build an **automatic car gate system** that uses **artificial intelligence** to detect vehicles and trigger a gate mechanism. The AI model, trained via Teachable Machine, distinguishes between vehicles and non-vehicle objects. A gate opens only when a **car** is detected and closes after the car has passed through; all with real-time feedback using **RGB LEDs**, a **servo motor**, and **ultrasonic sensors**.

🔷 How It Works

- An AI model (trained on a webcam using Teachable Machine) detects objects in real-time.
- If a **car** is detected, a **Blue LED lights up** and a **servo motor opens the gate**.
- If something else (like a hand or background) is detected, a **red LED lights up**, and the gate remains closed.
- Two ultrasonic sensors track whether the car is at the entrance and when it passes through.
- The system resets after the car leaves, ready for the next vehicle.

🔷 Hardware Components

Component	Quantity
Arduino Uno	1
HC-SR04 Ultrasonic Sensors	2
Servo Motor (Gate control)	1

- RGB LED 1
- Jumper wires (Male–Male) 18–20
- Breadboard (Large) 1
- USB cable (for Arduino) 1
- Webcam (for AI detection) 1
- Servo Motor Arm (3D printed) 1

🔷 Circuit and Layout Setup

To build the system correctly, follow the wiring using a Tinkercad model or this step-by-step guide.

⚠️ Power Note:

- **Ground** = Black wire
- **Power (Vcc)** = Red wire

✅ Step-by-Step Hardware Assembly:

Step 1: Power Rails

- Connect Arduino **GND** to the **- rail** of the breadboard.
- Connect Arduino **5V** to the **+ rail**.

Step 2: Ultrasonic Sensors

- Place **Sensor 1** on the left side of the breadboard.
    - Trig = pin 10
    - Echo = pin 9
- Place **Sensor 2** on the right side.
    - Trig = pin 12

    - Echo = pin 11

Step 3: Servo Motor

- Attach the servo in the middle of the breadboard using tape or a 3D printed holder.
- Connect:
    - Brown wire → GND
    - Red wire → 5V
    - Orange wire → Pin 3 (signal)

Step 4: LEDs

- Connect the **Blue LED** to pin 6 (via resistor).
- Connect the **Red LED** to pin 7 (via resistor).

Step 5: Status LED (optional)

- Connect another LED to pin 4 to show gate activity.

Step 6: Copy/Paste Arduino code

- Copy code:

```
#include <Servo.h>

// Pin Definitions
String inputString = "";
bool aiInputLocked = false;

const int trigPin1 = 10;
const int echoPin1 = 9;

const int RedPin = 7;
const int GreenPin = 6;

const int trigPin2 = 12;
const int echoPin2 = 11;
```

```
const int servoPin = 3;
const int ledPin = 4;

Servo gateServo;

const int threshold = 11; // Trigger distance in cm
bool gateIsOpen = false;
bool sensorsEnabled = false;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(RedPin, OUTPUT);
  pinMode(GreenPin, OUTPUT);
  gateServo.attach(servoPin);
  gateServo.write(90); // Start closed
  digitalWrite(ledPin, LOW);
}

void loop() {
  if (!aiInputLocked && Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n') {
      inputString.trim();
      Serial.println("Received: " + inputString);

      if (inputString == "Car") {
        analogWrite(GreenPin, 255);
        analogWrite(RedPin, 0);
        sensorsEnabled = true;
        aiInputLocked = true;
      } else if (inputString == "Hand" || inputString == "Background")
{
```

```
      analogWrite(RedPin, 255);
      analogWrite(GreenPin, 0);
      sensorsEnabled = false;
    } else {
      analogWrite(RedPin, 0);
      analogWrite(GreenPin, 0);
      sensorsEnabled = false;
    }

    inputString = "";
  }
}

if (sensorsEnabled) {
  long distance1 = getDistance(trigPin1, echoPin1);
  long distance2 = getDistance(trigPin2, echoPin2);

  Serial.print("Sensor1: ");
  Serial.print(distance1);
  Serial.print(" cm\tSensor2: ");
  Serial.println(distance2);

  if (!gateIsOpen && distance1 > 0 && distance1 <= threshold) {
    gateServo.write(0);
    digitalWrite(ledPin, HIGH);
    gateIsOpen = true;
    delay(1000);
  }

  if (gateIsOpen && distance2 > threshold) {
    delay(1000);
    distance2 = getDistance(trigPin2, echoPin2);
    if (distance2 > threshold) {
      gateServo.write(90);
      digitalWrite(ledPin, LOW);
      gateIsOpen = false;
      sensorsEnabled = false;
      aiInputLocked = false;
```

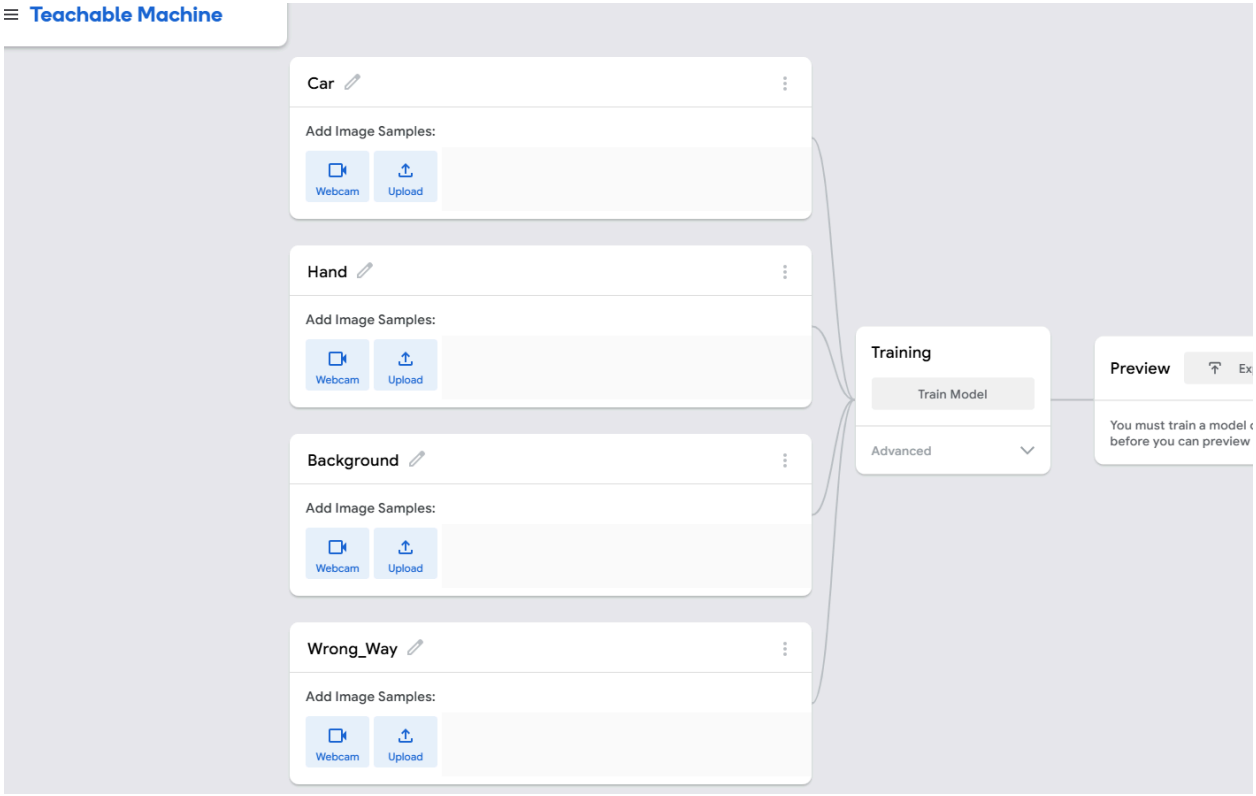
```
        analogWrite(GreenPin, 0);
      }
    }
  }

  delay(200);
}

long getDistance(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH, 30000);
  if (duration == 0) return 999;

  return duration * 0.034 / 2;
}
```

- Attach the usb to arduino, then transfer it into the arduino by clicking this:

🔷 AI Model (Teachable Machine)

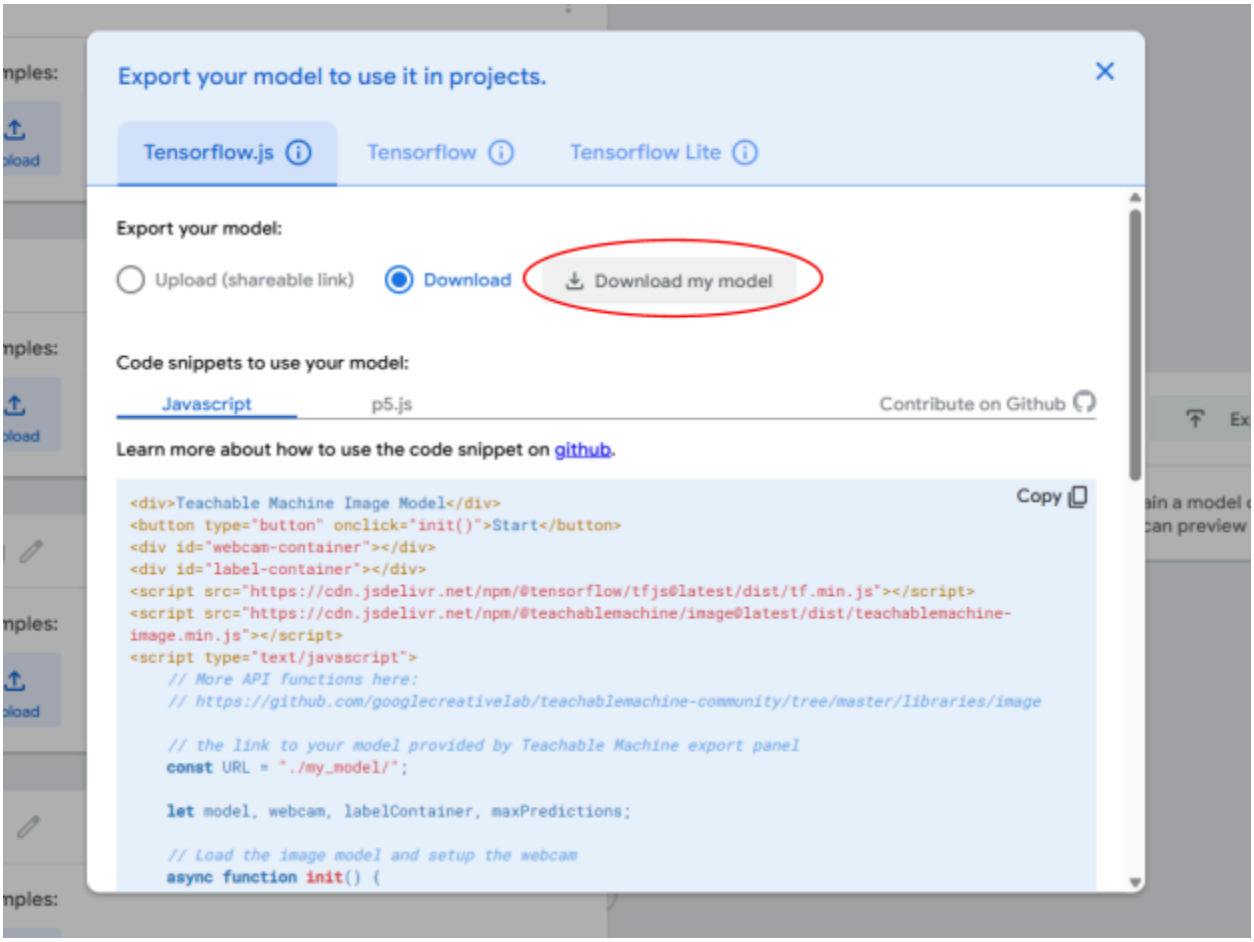
1. **Classes Trained**: Car, Hand, Background, Wrong_Way
   a. Open Teachable Machines ; Select ‘Get Started’

b. Then select ‘Image Model’
c. Lastly, select ‘Standard Image model’
d. Once that is set, attach the webcam through usb.
e. After, Create 4 Classes like so;



2. **Collection Method**: Captured images using a **webcam** from various angles and lighting.
    a. Start taking images of the vehicles, Hands, and background in different forms to truly give the AI many options to train with.
    b. Make at least 900 - 1200 images per class, then start training your model by clicking ‘Train Model’

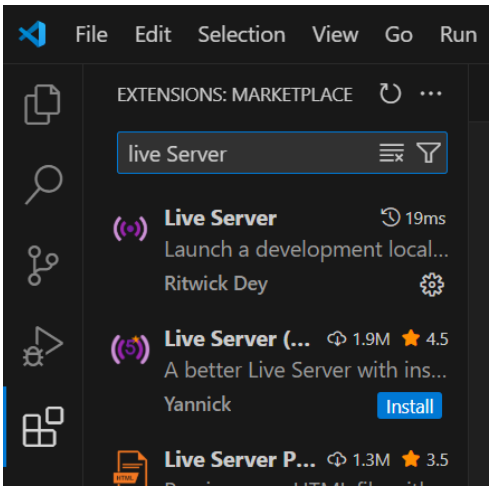
3. **Export Format**: Exported model as **TensorFlow.js:**
    a. Name file something that you can find easily



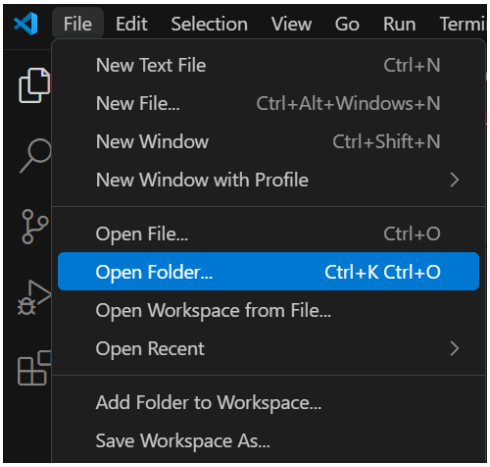
4. **Platform Used**: Visual Studio Code with live webcam input
    a. Download Visual Studio on your device

🔷 Visual Studio Code (HTML)

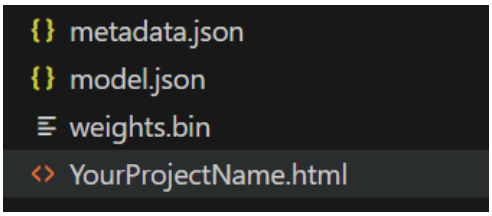
a. Go to Extensions and Install “Live Server”



a. Open folder where you put your AI model



1. Make a new file and name your file _____.html

a. 

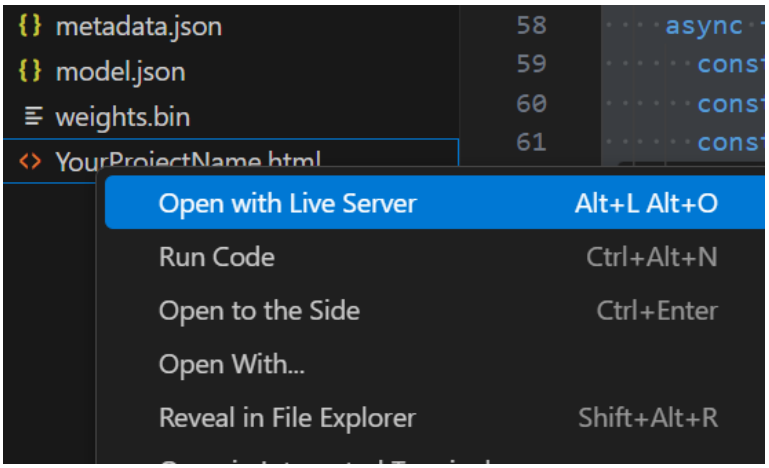
2. Write your code
    a. Copy/Paste

```
b. <!DOCTYPE html>
c. <html>
d. <head>
e.   <title>Thumbs Detection</title>
f.   <script
   src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
```

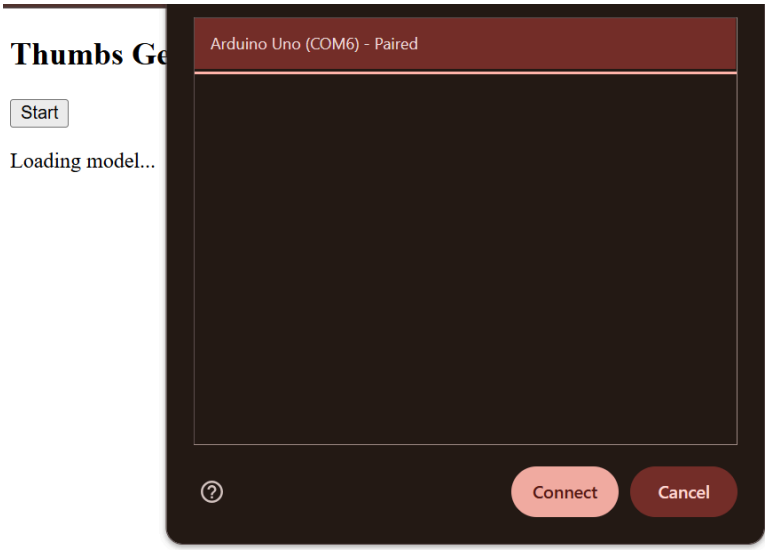
```
  <script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@latest/dist/teachablemachine-image.min.js"></script>
</head>
<body>
  <h2>Thumbs Gesture Detection (Offline + Arduino)</h2>
  <button onclick="init()">Start</button>
  <p id="status">Waiting...</p>

  <!-- ✅ Your custom JS goes here -->
  <script>
    let model, webcam, writer;
    let lastClass = "";
    let lastSendTime = 0;
    const sendInterval = 400; // milliseconds between predictions

    async function init() {
      try {
        document.getElementById("status").innerText = "Loading model...";

        const modelURL = "model.json";
        const metadataURL = "metadata.json";
        model = await tmImage.load(modelURL, metadataURL);

        webcam = new tmImage.Webcam(200, 200, true);
        await webcam.setup();
        await webcam.play();
        document.body.appendChild(webcam.canvas);

        const port = await navigator.serial.requestPort();
        await port.open({ baudRate: 9600 });
        const encoder = new TextEncoderStream();
        encoder.readable.pipeTo(port.writable);
        writer = encoder.writable.getWriter();
```

```
mm.
nn.         document.getElementById("status").innerText = "Camera
   + Serial Ready!";
oo.         window.requestAnimationFrame(loop);
pp.       } catch (err) {
qq.         console.error("Setup failed:", err);
rr.         document.getElementById("status").innerText = "Error:
   " + err;
ss.       }
tt.     }
uu.
vv.     async function loop() {
ww.          webcam.update();
xx.       const now = Date.now();
yy.
zz.       if (now - lastSendTime > sendInterval) {
aaa.           await predict();
bbb.           lastSendTime = now;
ccc.         }
ddd.
eee.         window.requestAnimationFrame(loop);
fff.     }
ggg.
hhh.       async function predict() {
iii.     const prediction = await model.predict(webcam.canvas);
jjj.
kkk.       let maxIndex = 0;
lll.     for (let i = 1; i < prediction.length; i++) {
mmm.         if (prediction[i].probability >
   prediction[maxIndex].probability) {
nnn.           maxIndex = i;
ooo.           }
ppp.       }
qqq.       // If you have more Classes add them here. [...,
   "Class4", "Class5", ...], also add DisplayName#...
rrr.     const classNames = [ “Car”, “Hand”, “Background”, “Wrong_Way”];
```

```
sss.        const displayNames = [ “Car”, “Hand”, “Background”,
   “Wrong_Way”];
ttt.
uuu.        const selectedClass = classNames[maxIndex];
vvv.
www.        // Only send to Arduino if changed
xxx.        if (selectedClass !== lastClass && writer) {
yyy.            await writer.write(selectedClass + "\n");
zzz.            lastClass = selectedClass;
aaaa.       }
bbbb.
cccc.       // Show a friendly name on screen
dddd.       document.getElementById("status").innerText =
   displayNames[maxIndex]; // You can change this text
eeee.       }
ffff.   </script>
gggg.  </body>
hhhh.  </html>
iiii.
```

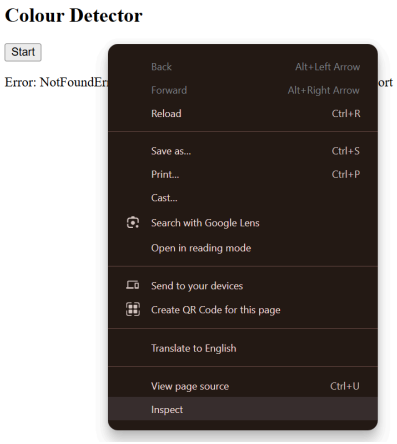
3. Open with Live Server
   a. 

4. Press Start and connect your arduino (Use chrome as your browser)

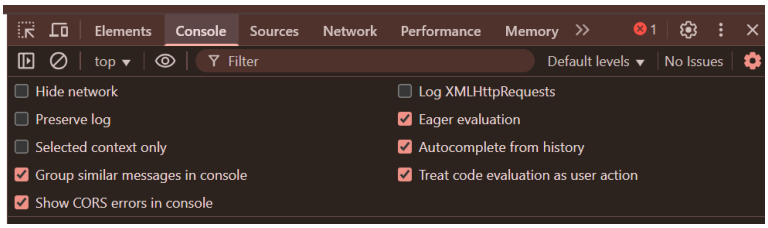
a. 

Connecting External Camera

1. Right Click on a chrome page

a. 

2. Open console

a. 

3. Write code below

```
navigator.mediaDevices.enumerateDevices().then(devices => {
  devices.forEach(device => {
    if (device.kind === "videoinput") {
      console.log("Camera:", device.label || "Unnamed", device.deviceId);
    }
  });
```

});

4. Copy the id
   a.
```
Camera: Live Streamer CAM 313 (07ca:313a)
0e8061431ede9dca7f7e6065fa07fe03e9d723110fc0444e2b726f01a614bb52
```
5. Change your code
   a.
```
await webcam.setup();
```
   b. To
   c.
```
const deviceId = "0e8061431ede9dca7f7e6065fa07fe03e9d723110fc0444e2b726f01a614bb52";
```
   d.
```
await webcam.setup({ facingMode: "user", deviceId: deviceId });
```

5. The AI communicates with Arduino through **Serial.print()** from JavaScript running in the browser.

🔷 Software

- **Arduino Code**: Written in C++ using the Arduino IDE.
- **AI Script**: JavaScript in the browser (Visual Studio Code + TensorFlow.js).
- **Communication**: AI predictions sent via **Serial USB** to Arduino Uno.

🔷 Photos and Visuals

- ✅ Working hardware prototype (breadboard + components)
- ✅ Tinkercad circuit diagram
- ✅ Teachable Machine interface screenshots
- ✅ AI detection samples (Car vs. Hand detection)

✅ Optional: 3D printed part designs for ultrasonic covers and servo arm

Google Drive:

🔷 Reflection

What Worked Well:

- The AI model could successfully detect cars with decent accuracy.
- The ultrasonic sensors performed well in detecting object presence and passage.

- The gate opening/closing logic was reliable and responsive.

Challenges:

- Calibrating the distance sensors for small-scale models.
- Serial communication syncing between the browser AI and Arduino.
- Keeping servo movement consistent during rapid object detection.

Future Improvements:

- Add **license plate detection** for advanced access control.
- Integrate with **Bluetooth/Wi-Fi** for remote control.
- Create a full mobile/web app dashboard.
- Add a **buzzer or speaker** for audio alerts.
- Use **solar power** or **battery packs** for standalone deployment.