REVERSE CAR PARKING SENSOR

eeo8

Reverse/Rear Car Parking Sensor Interface

An IoT-Based Smart Parking system using ESP32 and Ultrasonic Sensor with Mobile App System

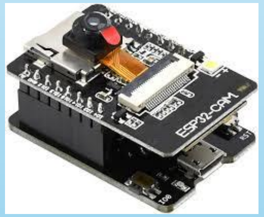
Presented by Enrico Obeng eeo8

Introduction

This project consist in a parking sensor system that combines the ESP32-CAM and an ultrasonic sensor to provide drivers with real-time visual and auditory feedback. This smart parking solution increases safety while reversing and helps drivers park efficiently

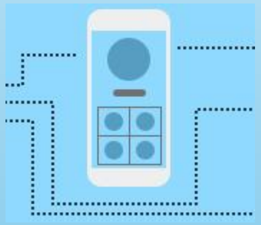
System Components

Hardware: ESP32-CAM



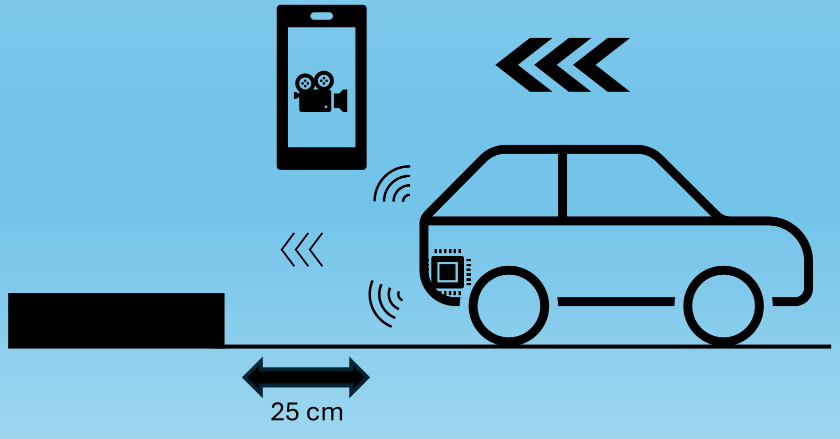
Ultrasonic Sensor and cables

Software: Arduino IDE, Mobile App/Web Interface



Outcome

The goal of this project is to develop a reverse car parking sensor that uses an ESP32-CAM for visual feedback, an ultrasonic sensor to measure distance to obstacles, and a mobile app interface that provides real-time video streaming and distance data. The system also generates sound feedback to alert the driver as the car gets closer to the obstacle.



System Architecture / Core Features

Visual Feedback: The ESP32-CAM captures the rear-view video
Distance Measurement: The ultrasonic sensor measures the distance to the closest obstacle
WiFi: Transmits data to mobile app
Mobile App Interface: The app displays both the video feed and the measured distance, providing real-time feedback to the driver. The system also emits a sound that increases as the distance decreases

Process Diagram

Start: Power on system
WiFi Connection: connects to WiFi
Camera Streaming: Live video is captured
Distance Measurement: The ultrasonic sensor detects the distance to the obstacle
Feedback System: If the distance is less than 20 cm, the buzzer activates
App Interface: The app displays the video stream, the distance data and the sound

Complication

The ultrasonic sensor might detect unintended objects in the way.
WiFi obstruction from other wireless signals or network congestion in the area could cause reducing the system's reliability. Battery saving options.

Scenario

- If your car doesn’t come equipped with reverse parking assist, this innovative device is the perfect solution to help you park safely and with confidence.

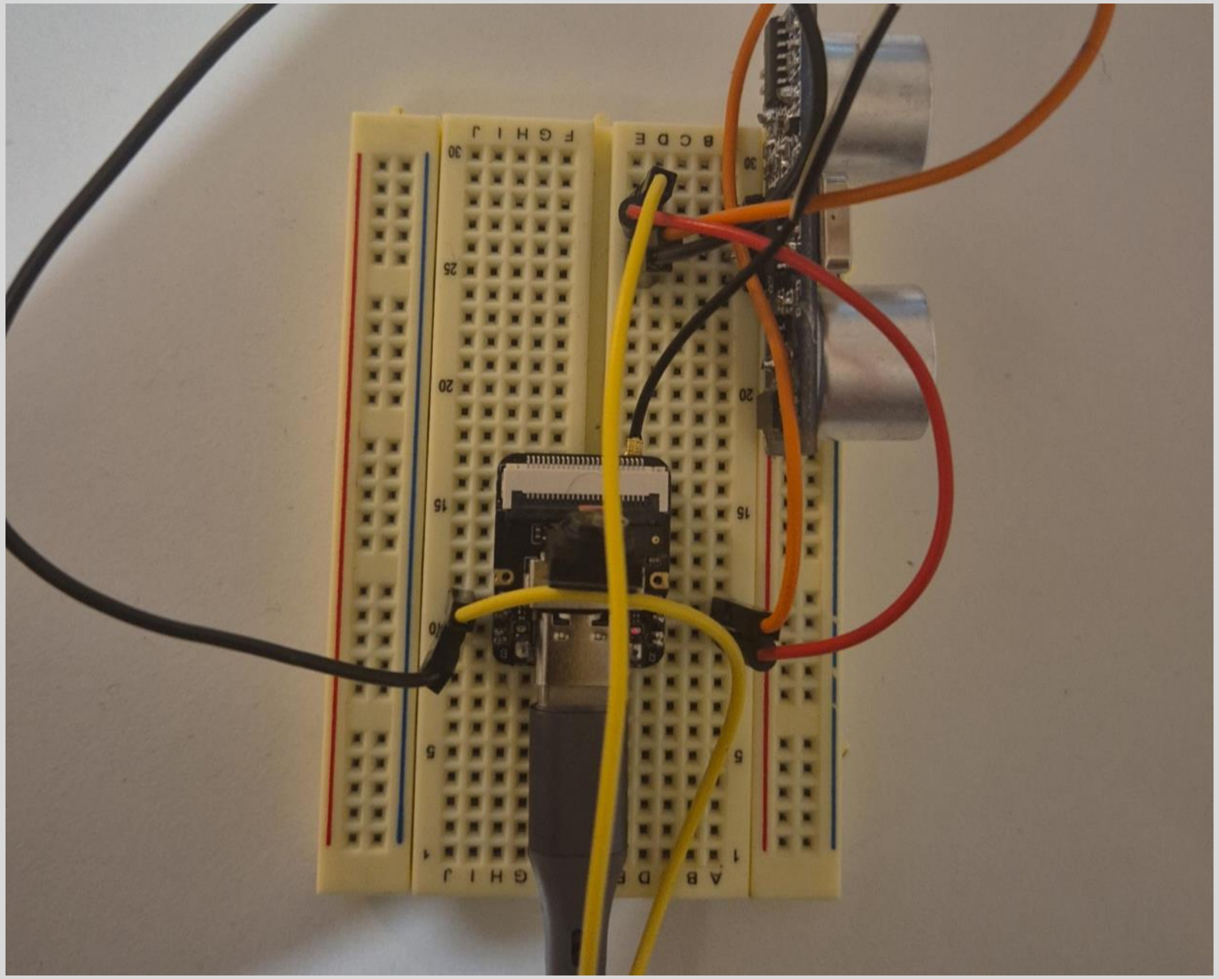
Original idea

- My idea is to install the device at the rear of the car, facing potential obstacles and hazards. When it’s time to park, the driver can access a mobile app that provides a simultaneous live video feed and real-time distance measurements. Additionally, the system includes an audible buzzer that increases in intensity as the vehicle gets closer to an object, offering an intuitive parking assistance solution.

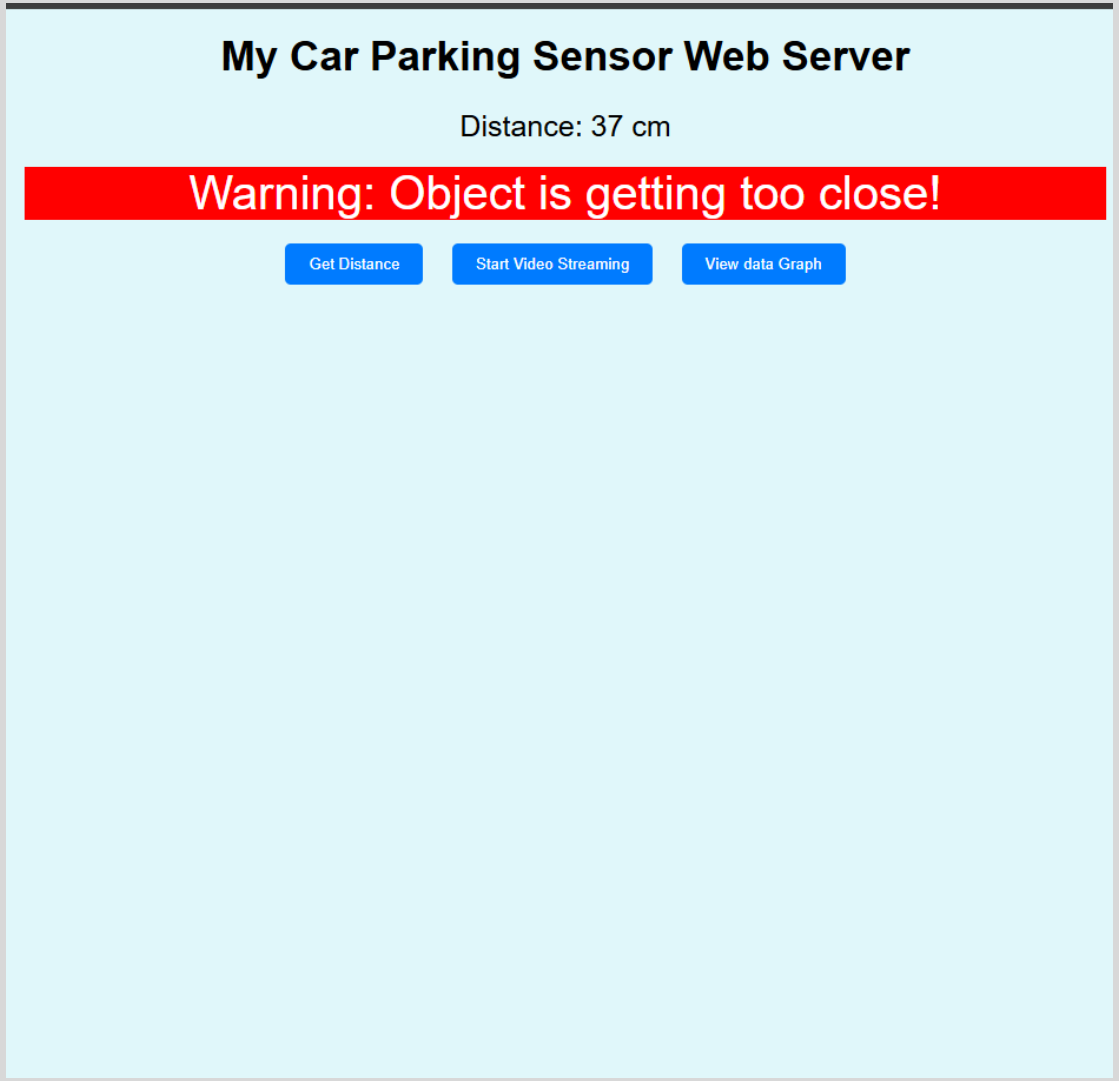
Final implementation

- I opted not to include a buzzer, as it would operate outside the car and might not effectively assist the driver. Instead, I chose to use a web server for its responsiveness and user-friendly interface, making it ideal for delivering real-time data.
- Given the ESP32’s limitation to handle one connection at a time, I structured the system into three dedicated webpages: one for real-time distance data, another for the live video feed, and a third displaying a graphical representation of the distance data.
- The main page features a clean and simple design with three buttons providing access to these functions. It also displays real-time distance data, with a built-in safety feature: if you come within 50 cm of an obstacle, a warning message appears. Should you move closer and cross the 25 cm threshold, the web server automatically redirects to the live video feed, ensuring the driver has the visual information needed to park safely and confidently.

Hardware setup



Software

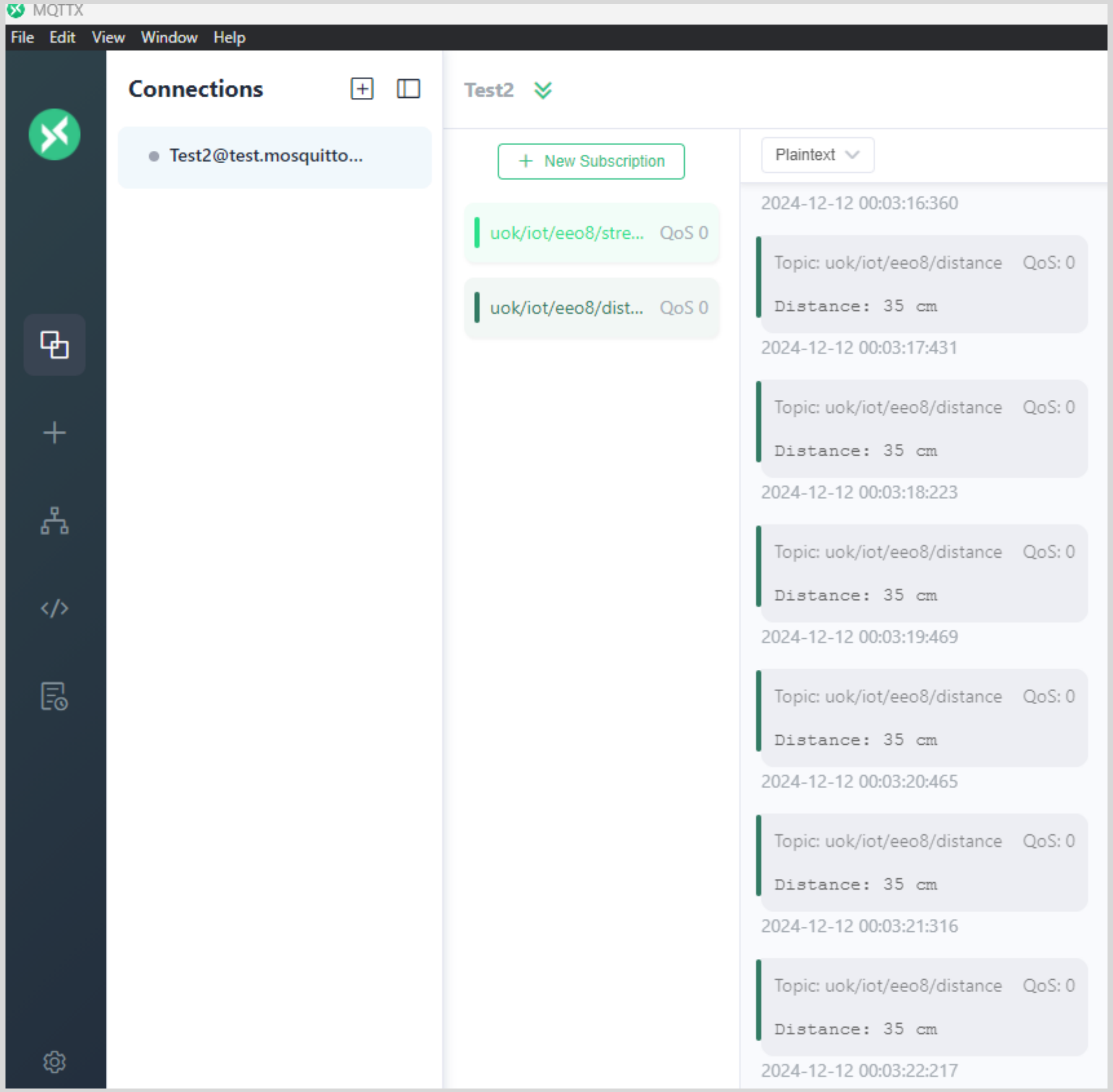


My Car Parking Sensor Web Server

Distance: 37 cm

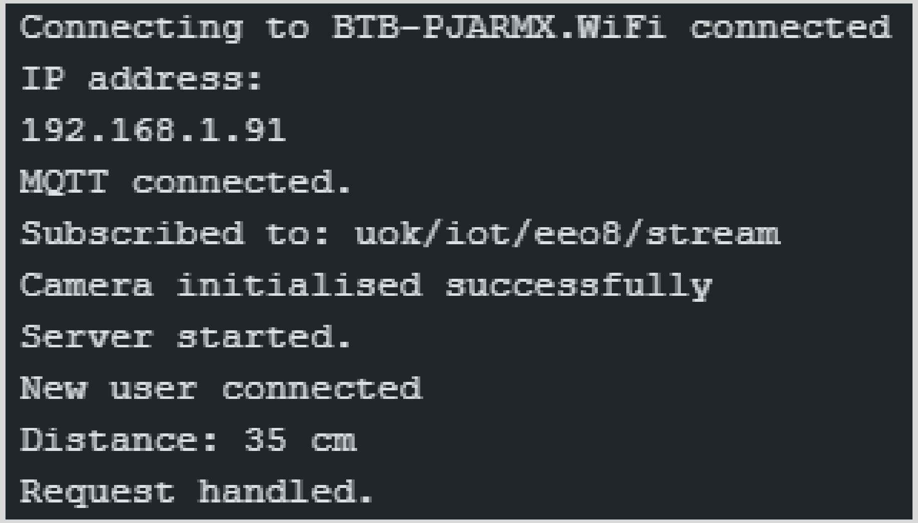
Warning: Object is getting too close!

MQTT communication


MQTTX
File Edit View Window Help
Connections
Test2@test.mosquitto...
Test2
New Subscription
uok/iot/eeo8/stre... QoS 0
uok/iot/eeo8/dist... QoS 0
Plaintext

```
2024-12-12 00:03:16:360
Topic: uok/iot/eeo8/distance QoS: 0
Distance: 35 cm
2024-12-12 00:03:17:431
Topic: uok/iot/eeo8/distance QoS: 0
Distance: 35 cm
2024-12-12 00:03:18:223
Topic: uok/iot/eeo8/distance QoS: 0
Distance: 35 cm
2024-12-12 00:03:19:469
Topic: uok/iot/eeo8/distance QoS: 0
Distance: 35 cm
2024-12-12 00:03:20:465
Topic: uok/iot/eeo8/distance QoS: 0
Distance: 35 cm
2024-12-12 00:03:21:316
Topic: uok/iot/eeo8/distance QoS: 0
Distance: 35 cm
2024-12-12 00:03:22:217
```

Serial monitor

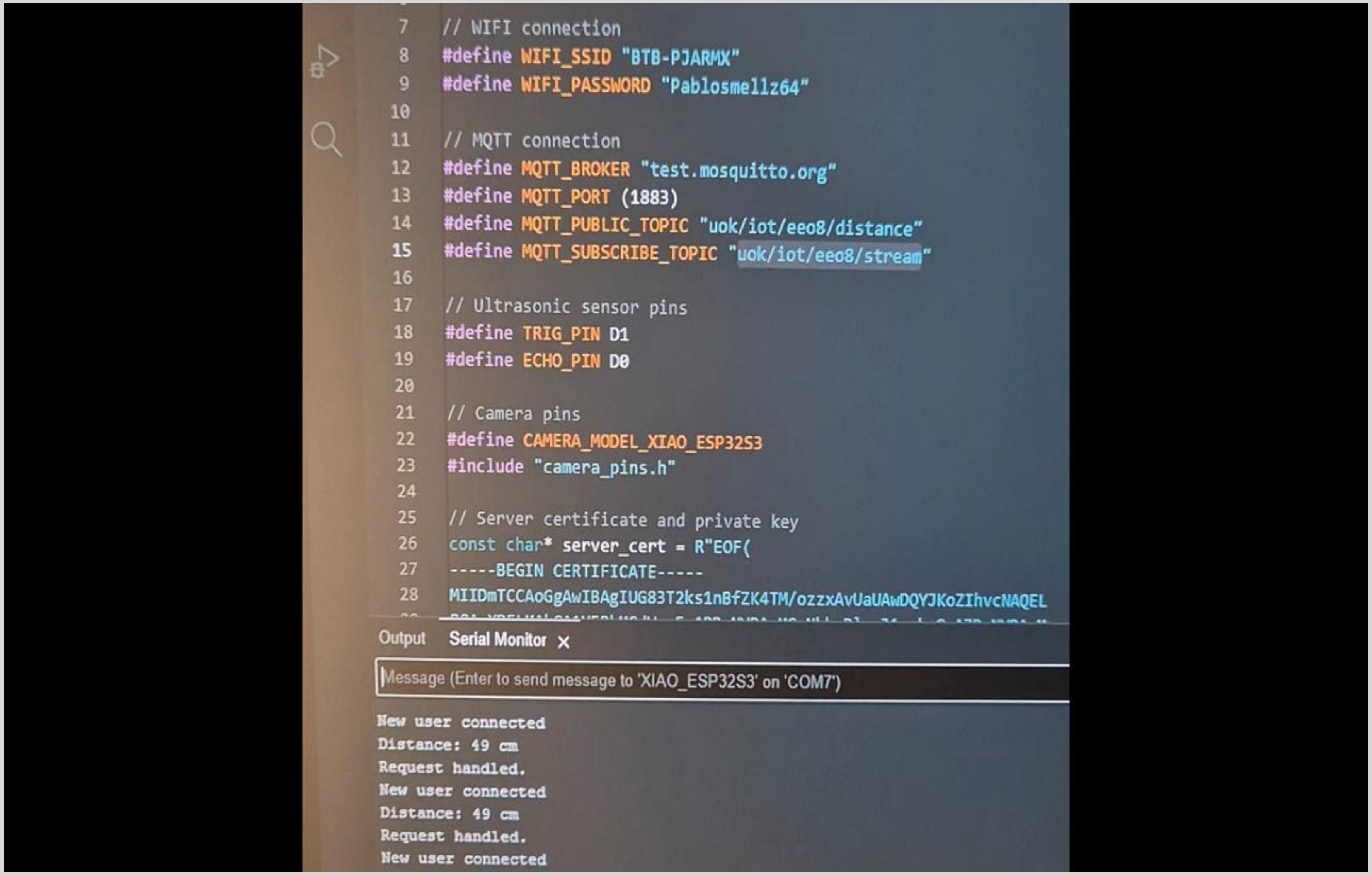


```
Connecting to BTB-PJARMX.WiFi connected
IP address:
192.168.1.91
MQTT connected.
Subscribed to: uok/iot/eeo8/stream
Camera initialised successfully
Server started.
New user connected
Distance: 35 cm
Request handled.
```

My project

- Libraries: Enables Wi-Fi communication; Handles MQTT communication; Manages the camera module; Supports secure connections (e.g., HTTPS).
- Credentials for connecting the ESP32 to a Wi-Fi network.
- MQTT configuration: Broker (server for communication); Port 1883; Topics (publishing distance data)
- Ultrasonic sensor pins (D1, D0)
- WiFiServer on port 80; WiFiClient: Handles users' connections; PubSubClient: Manages MQTT communication using wifiClient
- The distance is calculated using an ultrasonic sensor: TRIG_PIN emits a pulse; ECHO_PIN measures the duration of the returned pulse; Then the formula converts duration of the pulse to distance
- Camera initialisation: Initialises the camera with specific settings; prints success or failure messages
- Setup function: Connects to Wi-Fi and MQTT broker; Subscribes to the streaming topic; Starts the camera and web server.
- Loop function: Keeps MQTT connection active; Interacts with client requests from the web server; Publishes live distance data to MQTT.
- Web server: Main Page presents an HTML page with live distance updates and three buttons for fetching distance(/distance provides the real time distance value); streaming video(/stream shows the live video using the camera); and viewing a graph. (/graph displays an interactive graph showing live distance data)

Video demonstration



```
// WIFI connection
#define WIFI_SSID "BTB-PJARMX"
#define WIFI_PASSWORD "Pablosmellz64"

// MQTT connection
#define MQTT_BROKER "test.mosquitto.org"
#define MQTT_PORT (1883)
#define MQTT_PUBLIC_TOPIC "uok/iot/eeo8/distance"
#define MQTT_SUBSCRIBE_TOPIC "uok/iot/eeo8/stream"

// Ultrasonic sensor pins
#define TRIG_PIN D1
#define ECHO_PIN D0

// Camera pins
#define CAMERA_MODEL_XIAO_ESP32S3
#include "camera_pins.h"

// Server certificate and private key
const char* server_cert = R"EOF(
-----BEGIN CERTIFICATE-----
MIIDmTCCAoGgAwIBAgIUG83T2ks1nBfZK4TM/ozzxAvUaUAwDQYJKoZIhvcNAQEL
```

```
New user connected
Distance: 49 cm
Request handled.
New user connected
Distance: 49 cm
Request handled.
New user connected
```

https://youtu.be
/z7WquGnLcaE

Technical challenges

Why I didn’t add duty cycling

- I explored implementing duty cycling to optimise power consumption by leveraging the ESP32's light sleep mode. The idea was to keep critical functionalities, like connectivity and the ultrasonic sensor, operational while putting non-essential activities to sleep.
- The ultrasonic sensor would periodically check every 30 seconds for objects within a 100 cm range. If an object was detected, the device would exit light sleep mode and activate fully; otherwise, it would remain in low-power mode until the next check.
- After successfully uploading the changes, I encountered an issue where the web server failed to update properly. Despite troubleshooting, including making adjustments and resetting the ESP32, I encountered persistent errors like 'No data received on Port COM 7.' Eventually, the ESP32 resumed normal operation, but the risk of further instability led me to reconsider implementing duty cycling.
- Ultimately, I decided to prioritise reliability over additional power savings, especially since my production plan involves powering the device directly through the car's 12V system.

Why I didn’t add more internet security protocols

- I took steps to enhance security by using OpenSSL to generate a customised certificate and security key, configuring the web server to run on port 443 for HTTPS. Despite successfully implementing the certificate and key, the server failed to recognise them when I attempted to run it.
- For production, I plan to acquire an officially validated certificate, ensuring the server is fully secure and adheres to industry standards. This will provide users with a reliable and encrypted connection, enhancing both security

Social & Environmental Impact

- Key Positive Impacts
  - Enhanced Safety: The system helps prevent collisions with objects or hazards while parking
  - The real-time warning messages and automated redirection to the video feed provide immediate feedback
  - Energy Efficiency: The inclusion of features like duty cycling (even if not implemented fully) reflects an effort to minimise power consumption.
  - User-Friendly Interaction: The simple web-based interface eliminates the need for app installations or complex configurations. Anyone in the household can easily operate the system.
- Value Offered
  - This system provides a cost-effective, reliable, and user-friendly parking. By leveraging real-time data and live visuals, it delivers peace of mind and convenience, especially for individuals without advanced parking assist features in their vehicles.

Systems Thinking

- Picked the Ultrasonic Sensor for its affordability and reliability in measuring distance. It provides the accuracy required for detecting obstacles within the targeted range.
- Picked the ESP32 microcontroller for connectivity and processing reasons. Its built-in Wi-Fi and Bluetooth capabilities, which are essential for IoT systems. and compatibility with Arduino.
- Picked the camera version for live streaming.
- Picked MQTT Protocol for the efficient communication over the network. This was ideal for publishing sensor data.
- User Interaction, Visualisation and Alerts (HTML/CSS/JavaScript) these technologies were used for designing the web interface. They provided the flexibility to create an intuitive, visually appealing, and interactive interface. Chart.js was integrated to generate dynamic graphs, offering a clear visualisation of distance data.

Execution

- Key functionalities retained
  - Both the initial concept and the implemented system rely on ultrasonic sensors to provide accurate distance data, ensuring obstacle detection remains a central feature.
  - A camera module integrated into the ESP32 delivers real-time video streaming, consistent with the original idea of enhancing situational awareness.
- Refinements for Practicality:
  - While the original idea included a buzzer for auditory feedback, it was removed due to its impracticality outside the car.
  - A web server was chosen over a mobile app for its ease of use, responsiveness, and platform independence.
  - The implementation divides functionality into three dedicated webpages for real-time data, live video, and a graphical display of distance data. This structure improves clarity and ensures the system remains responsive despite the ESP32's connection limitations.
- Robustness of the System:
  - Automatic alerts and redirection enhance the system's reliability by reducing the chances of human error.
  - Although power-saving strategies were explored, duty cycling was not implemented due to technical issues. However, reliance on the car's 12V power supply mitigates this limitation.
  - While HTTPS implementation faced challenges, the plan is to integrate official certificate validation in production.

Creativity & Innovation

- Unlike most commercial systems that rely on pre-installed car software or mobile apps, this solution uses a web server for accessibility. This approach eliminates the need for additional app downloads.
- The separation of functionality into three dedicated webpages (real-time distance data, live video feed, and graphical data) is a unique design choice. This decision ensures clarity, prioritising essential information while allowing users to navigate to specific views. The popup warnings and smooth navigation through the system elevates the driver's parking experience.
- The system's warning when and object falls within 50 cm and the dynamic redirection to the live video feed when objects come within 25 cm is an intelligent feature. This automatic switch focuses the driver's attention on the most relevant data at critical moments, providing an enhanced sense of control and situational awareness.