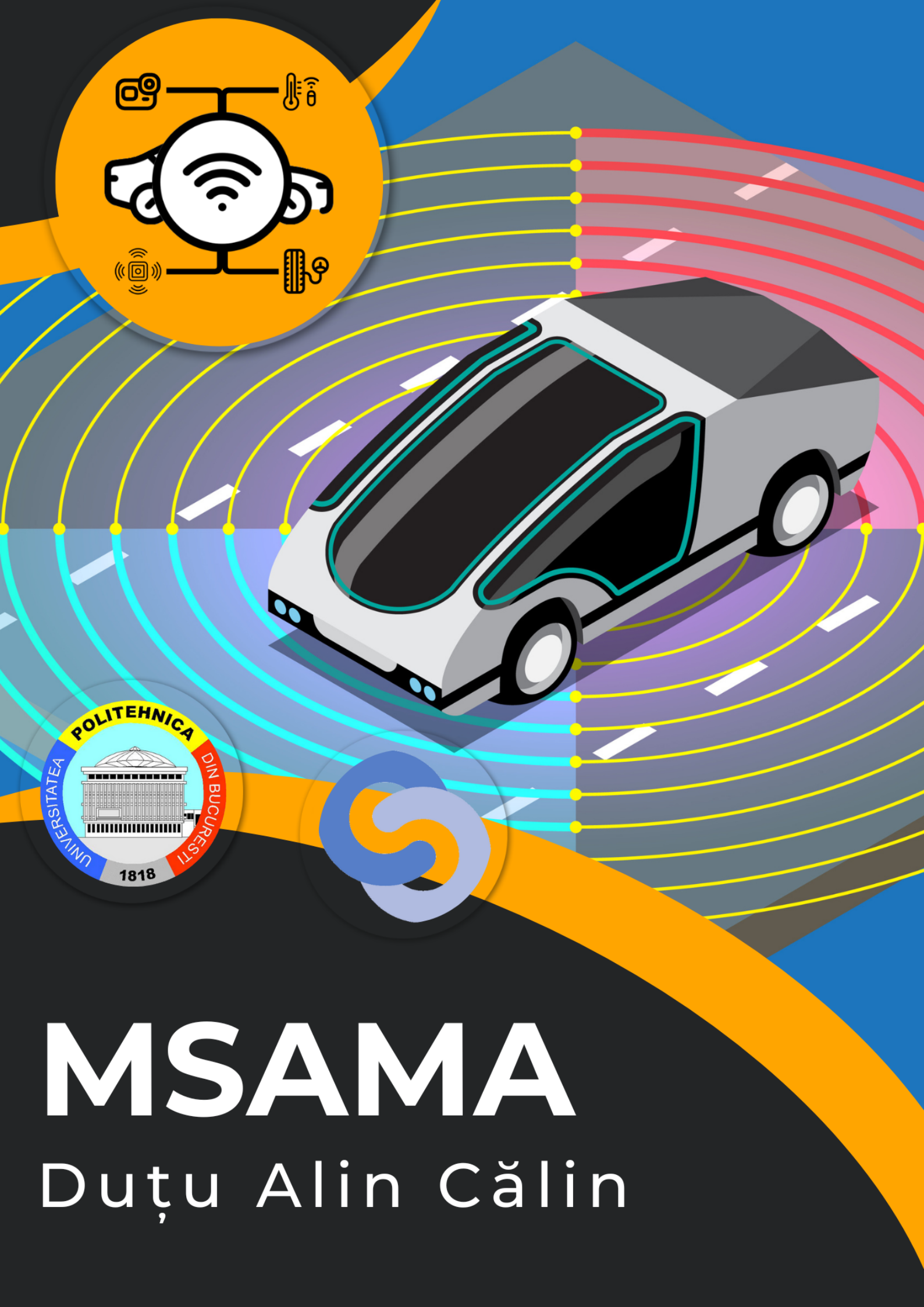

MSAMA

Duțu Alin Călin

CONTENTS

1 INTRODUCTION

1.1 Context

In 2022 it became mandatory for car companies to include an essential kit with assistance and safety sensors in their new models, but some models still lack essential sensors. However, according to DRPCIV, at the end of 2023, more than 81% of the cars in Romania had been used for more than 10 years [1], so it is clear that most of them don't have the essential set of sensors.

1.2 Objectives

This project aims to implement a system for automotive that is capable to monitor multiple aspect of a car and provide assistance to the drivers with older vehicle models to equip essential sensors for ensuring a safe and comfortable driving experience. The following objectives are taken into consideration:

- Implement a modular system to suit any vehicle
- Develop reliable modules that require minimal maintenance
- Create an UI application which provides important data to the user

2 PROPOSED SOLUTION

2.1 Solution presentation

The project aims to implement a modular system for automotive monitoring and assistance that should be easy to use and maintain while providing reliability through a long lifespan. Knowing the project's purpose and the complexity level, the following architecture has been considered:

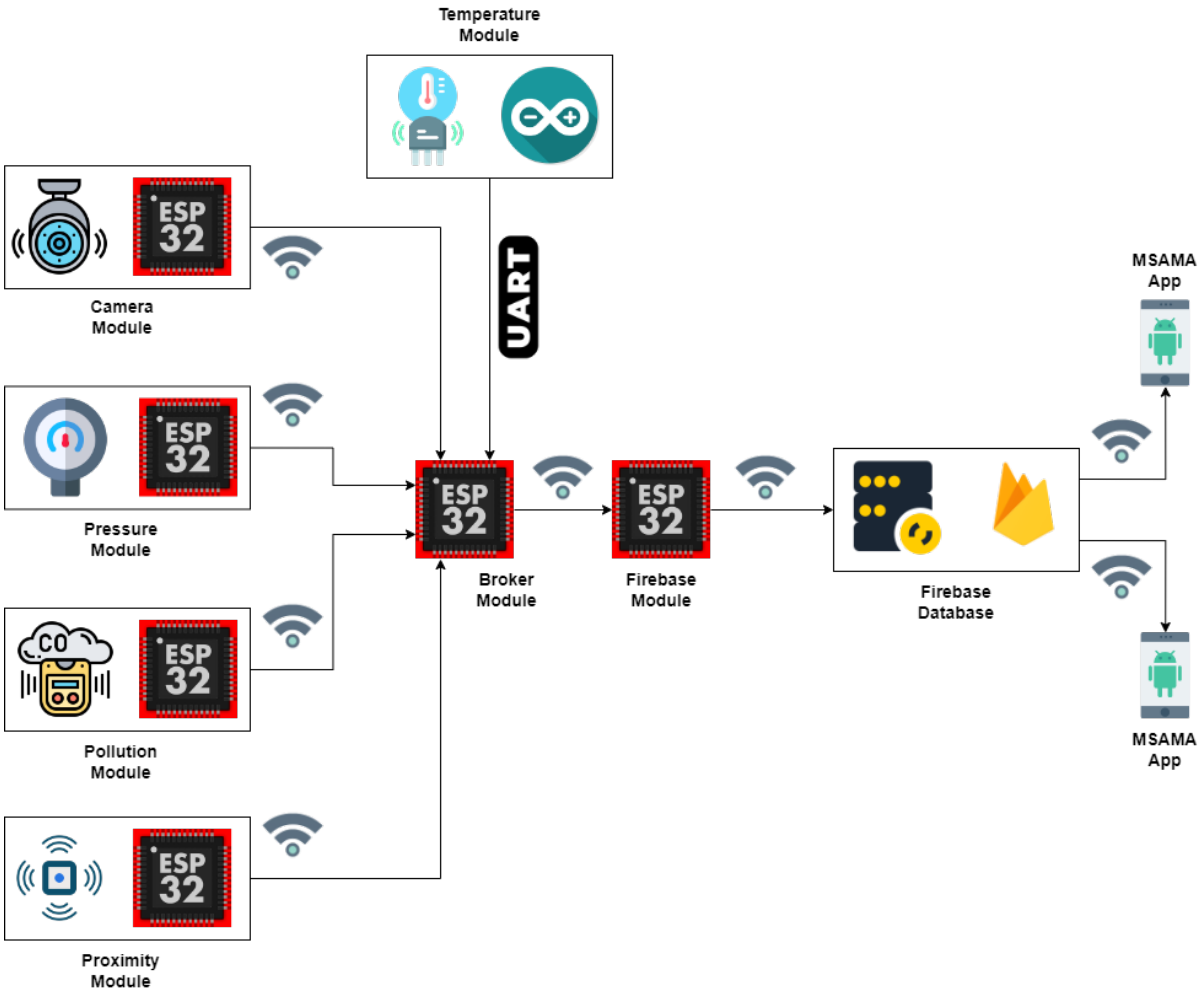

Figure 1: Architecture

There are multiple components identified in the architecture:

- Temperature Module - Sends temperature readings
- Camera Module - Sends camera images
- Pressure Module - Sends pressure readings
- Pollution Module - Sends CO readings
- Proximity Module - Sends distance readings
- Broker Module - Module that gathers data from the sensor modules and prepares them to be sent to the database
- Firebase Module - Module that sends data to the database
- Firebase Database - Database used for storing data
- MSAMA app - Android app that gathers database data and presents it to the user

The main requirement for this setup is to ensure internet connectivity since most communication between modules uses the Wi-Fi protocol to make the setup cableless, offering modularity and ease in the installation procedure.

The database is an instance of Firebase[2] that stores the sensor data. The database can be administered, at the moment, using the Firebase web app, a great tool for administrators to manage connections and useful database settings.

3 IMPLEMENTATION DETAILS

3.1 Hardware

The architecture schematic (fig. 1) offers a brief envision of the idea. However, there are more details to be discussed in this section starting with the required hardware for the project:

- 4 x ESP-WROOM-32 microcontrollers
- 1 x Arduino Uno microcontroller
- 1 x NodeMCU32S microcontroller
- 2 x HC-SR04 sensor[3]
- 1 x DHT11 sensor[4]
- 1 x Wi-Fi ESP32-CAM[5]
- 1 x BMP280 sensor[6]
- 1 x MQ7 sensor[7]
- 5 x MB102 boards, 3.3V - 5V, 700mA
- 6 x 9V battery cable adapters

3.1.1 Camera module

The camera module is used mainly to capture and provide the video output, process it, and manage Wi-Fi connections. The camera module has 2 layers of hardware:

- The camera component (ESP32-CA) - The top hardware layer, it is used mainly to capture and provide the video output to the motherboard
- The motherboard component (ESP32-CAM-MB) - The bottom hardware layer's purpose is to process the captured video and manage Wi-Fi connections to transmit the video to a live server.

This module can be powered by any external source that provides an output of at least 5V.

A relevant picture of the module's hardware and the pinout of the top and bottom layers are presented in the following figures:

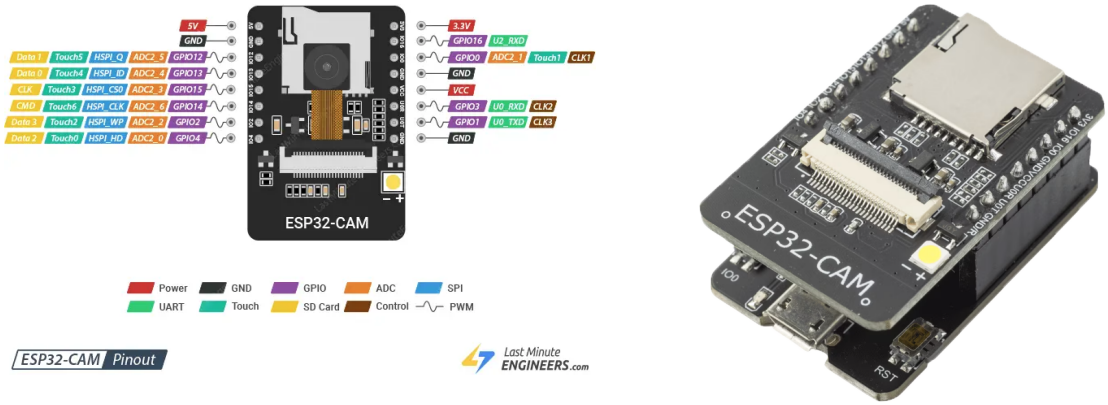

Figure 2: Camera pinout and the assembled module

3.1.2 Pollution module

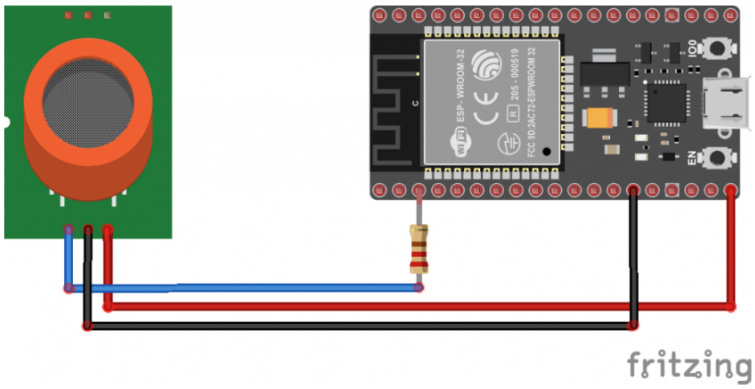

Figure 3: Pollution module

The pollution module consists of an MQ7 component that measures the quantity of CO coming from the exhaust and sends it to the ESP32 component, which transmits the data to the broker through Wi-Fi. It can be powered by any external source that provides an output of at least 5V.

3.1.3 Pressure module

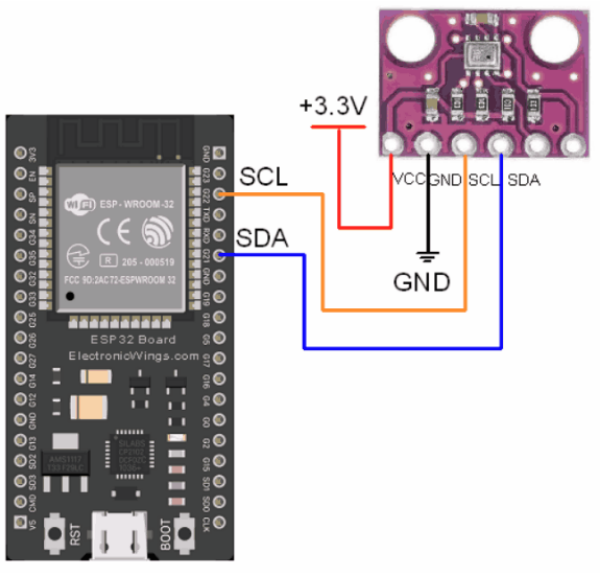

Figure 4: Pressure module

The pressure module consists of a BMP280 component that measures a tire's pressure, temperature, and altitude and sends the measurements to the ESP32 component, which transmits the data to the broker through Wi-Fi. It can be powered by any external source that provides an output of at least 3.3V.

3.1.4 Proximity module

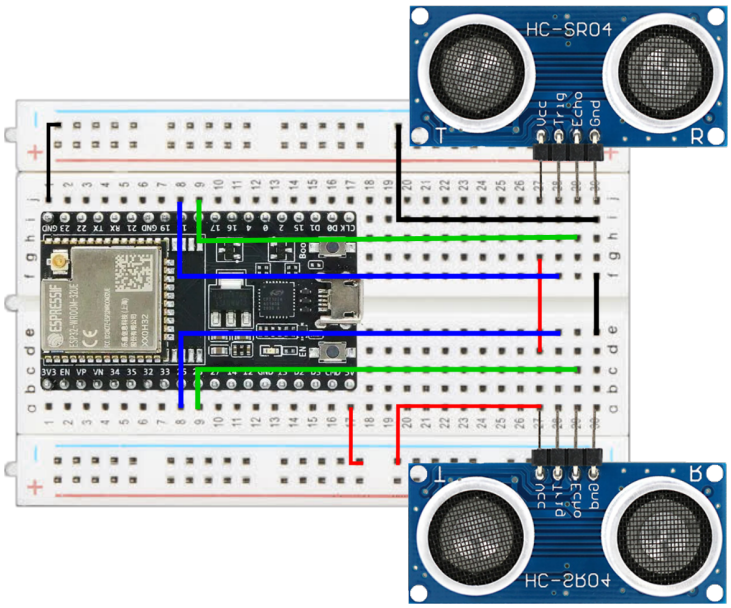

Figure 5: Pressure module

The proximity module consists of 2 HC-SR04 components that should be positioned on the front and the back of the car. They measure the distance between the sensor and the closest object. The measurements are sent to the ESP32 component, which further transmits them to the broker through Wi-Fi. At the moment the components are connected to the same ESP32, however, it is planned to use separate microcontrollers. The module requires a power supply that offers at least 5V.

3.1.5 Temperature Module

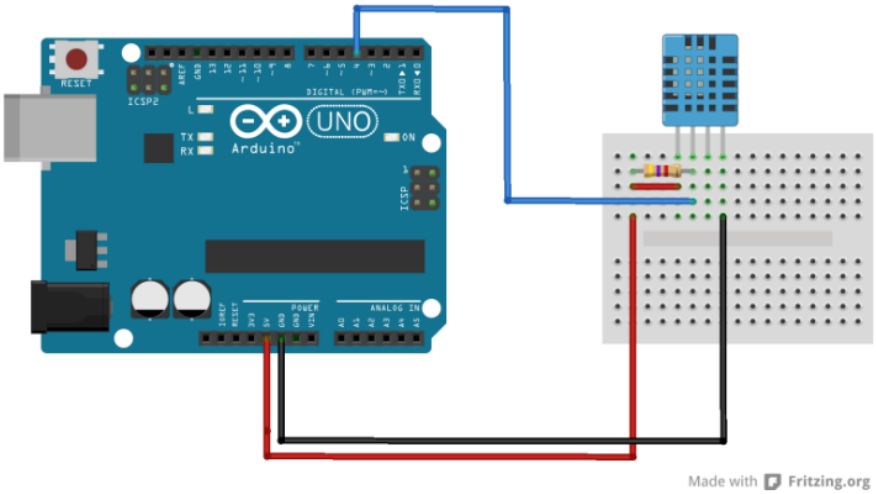

Figure 6: Temperature module

The temperature module consists of a DHT11 component that measures the inside temperature and sends the values to the Arduino Uno component, which transmits the data to the broker through UART. The module requires a power supply that offers at least 5V.

3.1.6 Broker Module

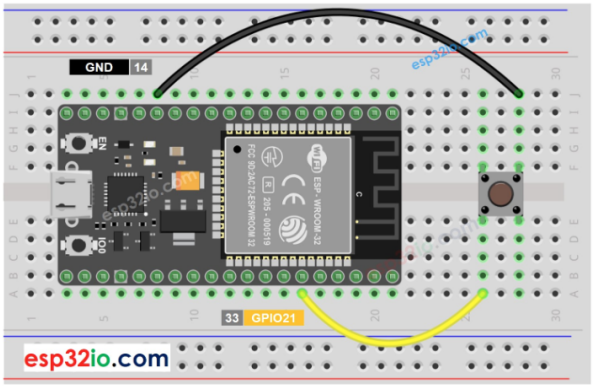

Figure 7: Broker module

The broker module consists of an ESP32 that will mainly collect all the data received from the modules enumerated above, process it, and send it to the Firebase module through Wi-Fi. Additionally, the module has a button that will trigger the $gearmode$ which will be thoroughly discussed in the Software section. The module requires a power supply that offers 3.3V.

3.1.7 Firebase module

The Firebase module has a simple ESP32 that will send the data processed by the broker and send it to the Firebase database. Since there is only one ESP in this module a source with a 3.3V output is sufficient to power the module.

3.2 Software

3.2.1 MQTT protocol

MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport which is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. MQTT today is used in various industries, such as automotive, manufacturing, telecommunications, oil and gas, etc. [8]. For the project, the MQTT is used to easily transfer data from multiple sensors to a single broker that will process all the data provided by the sensors with some functionality additions. The MQTT communication layer has been implemented using the PicoMQTT library[9] and used in the following manner:

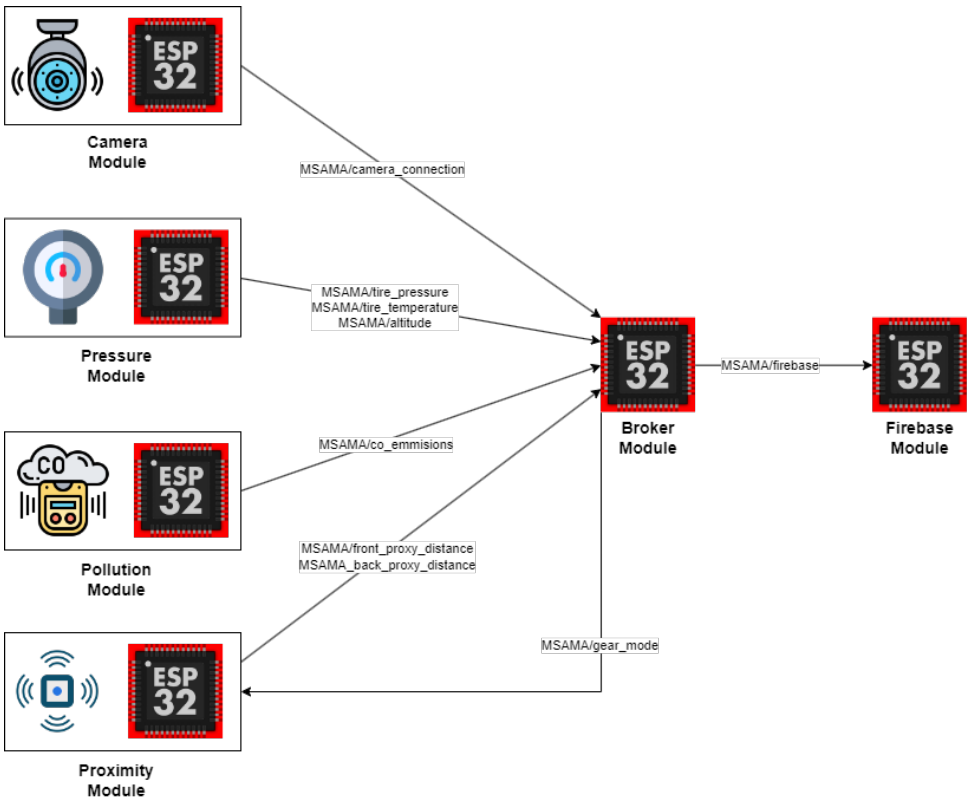

Figure 8: MSAMA MQTT network

3.2.2 Sensor modules

At initialization, every module besides the Temperature module will connect to the provided Wi-Fi followed by the connecting to the MQTT broker, and will retry until it connects. If the connection with the MQTT broker is lost, the sensors at the beginning of a new loop cycle will retry until the connection with the broker is set.

Every sensor will communicate with its microcontroller through either SPI or digital reading. Every client will asynchronously get the values from the sensors and send them to the broker through Wi-Fi if the microcontroller is an ESP 32 or UART if it is an Arduino UNO.

Additionally, the proximity sensor will wait for the $gear\ mode$ to change to parking mode to send proximity data. The $gear\ mode$ shall be interpreted as 1 when the car is parking and 0 otherwise. The $gear\ mode$ is controlled by the broker.

3.2.3 The Broker

The broker's main functionality is to aggregate data from sensors. It will receive data through MQTT protocol topics, aggregate them, and send them to the Firebase module through a specific MQTT topic for Firebase.

At initialization, the broker will connect to the Wi-Fi, initialize data variables to empty strings, and wait for new connections and data to be received. If no data has been received or the Firebase Module is not connected, the broker won't send any data to Firebase.

Every time a sensor connects, it will show in Serial that it is connected. If a client disconnects, there will be a timeout sequence of around 70 seconds which is enough time for sensors to reconnect. At the end of a connection, the broker will announce the loss of connection in the Serial interface and set the corresponding variable to an empty string.

The broker is also in control of the gear mode feature which is controlled at the press of a button. If the $gear\ mode$ is set to 0, everything will work as usual. However, if the button is pressed, the $gear\ mode$ will change, the broker will send it to the Proximity Module through the $gear\ mode$ topic and then, it will start receiving data from the proximity sensors as well. Due to the importance of this information, it must be processed quickly, so the broker will send more data to Firebase in a significantly shorter amount of time.

Switching back to normal mode will turn off the proximity sensors and the data sending speed will get back to normal.

3.2.4 Firebase database

Since Firebase is a horizontally scalable database which is a must-have for an IOT project, along with great documentation, easy setup, and seamless integration, it perfectly suits the project.

The Firebase module at initialization connects to the Internet via Wi-Fi and tries to establish contact with the Firebase database and it will retry until a connection is established. Next, the module connects to the MQTT broker, and by doing so, the broker has immediate access to the database.

The received data is stored in a database object and immediately available to the user. The database object apart from the sensor measurements, has a timestamp with the time of creation used for identification.

3.2.5 Android application

This app was created with the MIT App Inventor tool along with other plugins that completed the implementation such as FastBase[10], for communication with Firebase, and CustomWebView[11] to make it possible to show the dashcam output. The MSAMA app has 3 interactive screens:

- Car status screen
- Dashcam screen
- Parking mode screen

Car status screen

This screen is used to show the data saved in the Firebase Database. The app will fetch the database once every 2 seconds and will update the data accordingly. If an empty string is received for a specific value, then its text field won't show only the unit of measurement only but rather make it disappear from the screen.

Additionally, if there is a camera link that is associated with the camera being on, the camera button will be visible, but if not, it will be invisible and therefore inaccessible.

Dashcam screen

The dashcam screen is used to show live what is the camera capturing. The user can go back to the first screen by pressing the back button from the phone UI.

Parking mode screen

The parking mode screen is automatically activated when the $gear\ mode$ is set to 1. The user cannot go back through the phone UI, as a safety feature. It will automatically go back to the first screen when $gear\ mode$ is set to 0.

On this screen, the app will show visual and auditive feedback based on the proxy values fetched in the database.

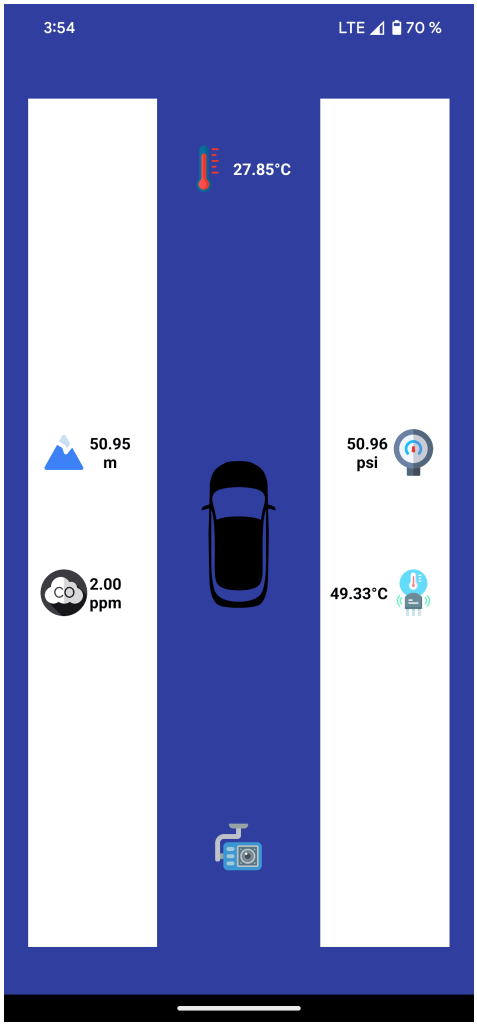

Figure 9: Car status screen

3.3 Testing

For testing the hardware reliability and software functionalities, the following methods have been used:

- Smoke tests - Testing that hardware connections are working and data is successfully sent to the user in the Android App.
- Stress tests - Testing the Broker's computational power to the limit by pairing and operating all modules at once
- Fuzz testing - Testing the modules on different environments to check for any inconsistencies in the measured input

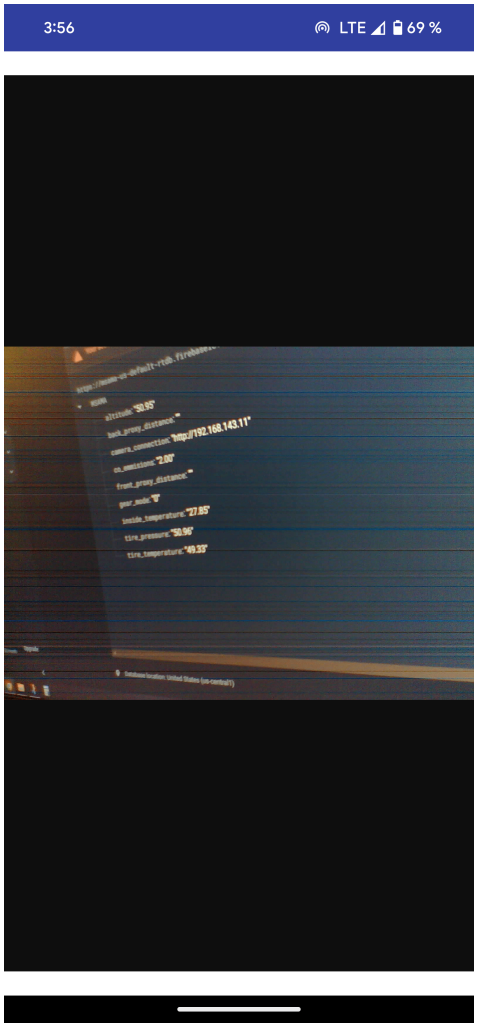

Figure 10: Dashcam screen

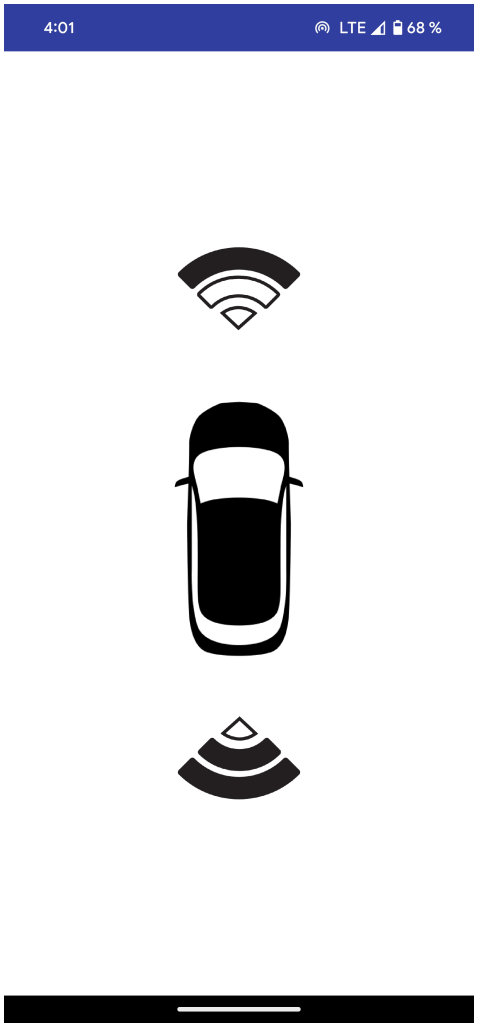

Figure 11: Parking mode screen

4 CONCLUSIONS

In conclusion, this program aims to provide valuable insights into the process of creating a system for automotive that is capable of monitoring multiple aspects of a car and providing assistance to the drivers. Although this is only a project concept, the research and development opened new and unique opportunities. The software, combined with the hardware and the right idea could offer a great product that might be useful for a big category of users.

BIBLIOGRAPHY

[1] DRPCIV Romania. Parc auto la data de 31.12.2023 - DRPCIV. https://dgpci.mai.gov.ro/news-details/statistica/65bb3a421f34e39b8431ba42. Latest access: 27.09.2024.

[2] Google. Firebase Database. https://firebase.google.com/. Latest access: 27.09.2024.

[3] Random Nerd Tutorials. Tutorial HC-SR04. https://randomnerdtutorials.com/esp32-hc-sr04-ultrasonic-arduino/. Latest access: 27.09.2024.

[4] Random Nerd Tutorials. Tutorial DHT11. https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-web-server-arduino-ide/. Latest access: 27.09.2024.

[5] Random Nerd Tutorials. Tutorial ESP32 Cam. https://randomnerdtutorials.com/upload-code-esp32-cam-mb-usb/. Latest access: 27.09.2024.

[6] Electronic Wings. Tutorial BMP280. https://www.electronicwings.com/esp32/bmp280-barometer-sensor-interfacing-with-esp32. Latest access: 27.09.2024.

[7] fjebaker. MQ7. https://github.com/fjebaker/MQ7. Latest access: 27.09.2024.

[8] Mqtt protocol. https://mqtt.org/. Latest access: 27.09.2024.

[9] mlesniew. PicoMQTT. https://github.com/mlesniew/PicoMQTT. Latest access: 27.09.2024.

[10] Mr Koder. FastBase. https://community.appinventor.mit.edu/t/fastbase-extension-to-retrieve-data-from-firebase/97538. Latest access: 27.09.2024.

[11] vknow360. CustomWebView plugin. Latest access: 27.09.2024.

[12] mobizt. Firebase ESP Client. https://github.com/mobizt/Firebase-ESP-Client/tree/main. Latest access: 27.09.2024.