



PROJECT TITLE

Vehicle Maintenance Tracking System

STUDENT/TEAM INFORMATION

Team Name if any: Team # on Canvas you have self-signed-up for:	Circuit Clowns Group 5
Team member 1 (Team Lead) (Lastname, Firstname; SDSU email; picture):	Fernandes, Sherwin – sfernandes7008@sdsu.edu 
Team member 2 (Lastname, Firstname; SDSU email; picture):	Nejad, Amir – aghafourinejad@sdsu.edu 

ABSTRACT (15 points)

(Summarize your project (motivation, goals, system design and results). Max 300 words).

When you're driving your own car, you'll know what type of maintenance is needed and when. However, if you (as an individual or company) lend a car out to others, you wouldn't know the state of the car until it's returned. Along with that, there's a ton of sensor data which is limited to scope of the car unless another tool is used. Our project strived to pull that data using the vehicle's OBD2 port and send it to the cloud, allowing users to wirelessly monitor vehicle data, such as speed and dashboard lights, and get real notifications on issues to act early to prevent further damage. We attempted to work on an actual car using an OBD2 shield paired with an ESP32 to pull data, but it ended up producing improper results. Instead, we developed a proof of concept using one board to simulate the vehicle data while the other board receives and sends that data to a cloud server. Although we were able to tackle a few of the main features of the product, it continued to lack features and a few design aspects were never implemented.

INTRODUCTION (15 pts)

Motivation/Background (3 pts)

(Describe the problem you want to solve and why it is important. Max 300 words).

We know that cars contain a lot of sensors, which help the onboard system produce detailed diagnostics when needed. This process would involve having to get/borrow a diagnostics tool like an OBD2 Scanner then go through the process of understanding the information. Having a device always connected to the OBD2 port that could send data to a cloud would make it easier for maintenance. For companies, it'd mean they could rely on dashboards to understand the status of their fleet rather than physically assessing each car. This helps to uphold proper vehicle maintenance and statistics with more convenience.

Project Goals (6 pts)

(Describe the project general goals. Max 200 words).

The goal of our project is to create a simple, real-time dashboard that alerts vehicle owners when there's an issue with their car, like low tire pressure, engine trouble, or low oil levels. This meant having a device connected to the OBD2 port which would use a portable internet connection to constantly send data to a cloud server. The server would handle data processing for visuals and mobile notifications. These implementations would help faster responses to avoid major issues.

Assumptions (3 pts)

(Describe the assumptions (if any) you are making to solve the problem. Max 180 words).

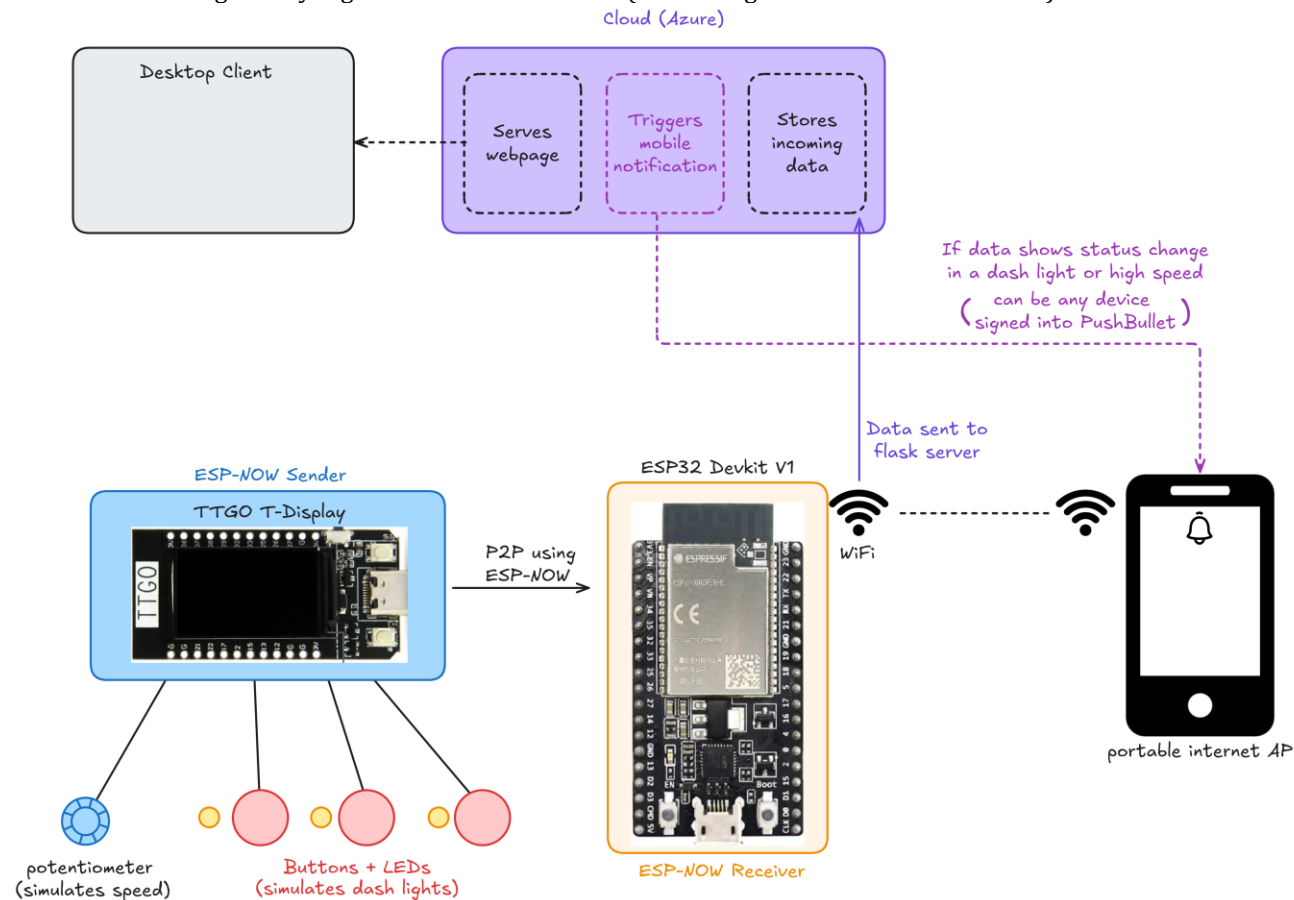
We assumed that the system would be used in a context where a driver or car owner may not always be inside the vehicle, like in fleet or rental car management, but the driver (or car) has a portable internet access point that the TTGO could connect to. The product also requires the user to download PushBullet. Multiple people can sign into the same account to receive notifications. Unfortunately, the account connected for notifications is currently hardcoded.

In terms of environment, we wouldn't have to worry about power constraints since the device could be powered by the vehicle (if the shield was used). Data would be sent to the server whenever the car is on, meaning spaced intervals (every 5 sec. instead of every few ms) wouldn't affect output.

SYSTEM ARCHITECTURE (20 pts)

(Describe the final architecture you have implemented listing sensors, communication protocols (Wi-Fi, BLE, ...), cloud services and user interfaces. Include a block diagram of the system. Max 300 words).

Architecture wise, one ESP32 (TTGO) is used to simulate vehicle data using user-controlled knobs/buttons. This board sends data to another board (Devkit V1) which would then send the data to a flask server. The server would handle storing the data, triggering notifications if issues occur (dash light status changes or over 75 mph), and serving the dashboard page on client request. PushBullet works well as it allows multiple devices to receive the notification as long as they sign into the same account (correlating to account token in code).



FINAL LIST OF HARDWARE COMPONENTS (5 pts)

(Write the final list and quantity of the components you have included in your system)

Component/part	Quantity
ESP32 Board (DevKitV1)	1
TTGO T-Display	1
Push Buttons	3
LED bulbs	3
Potentiometer	1
220 Ω Resistors (for LEDs)	3
Jumper Cables Set	1

PROJECT IMPLEMENTATION (30 PTS)

Tasks/Milestones Completed (15 pts)

(Describe the main tasks that you have completed in this project. Max 250 words).

Task Completed	Team Member
Wire up user inputs (potentiometer + buttons) and handle reading state/values on TTGO	Amir
Sending TTGO data to Devkit V1 using ESP-NOW	Sherwin
Allowing Devkit V1 to handle ESP-NOW receives while using WiFi connection to send data to server	Sherwin
Having flask server store received data	Sherwin
Creating dashboard page and updating visuals as stored data gets updated	Amir
Having mobile notifications trigger as dash lights turn ON/OFF or speed goes over 75	Amir

Challenges/Roadblocks (5 pts)

(Describe the challenges that you have faced and how you solved them if that is the case. Max 300 words).

Initially, we attempted the bold move of using a real car with maintenance lights already on. We bought an OBD2 shield that seemed well documented on how to pull data from the car's CAN bus. However, the shield ended up giving feedback loop data, so no data was pulled from the car. In order to move forward, we used another board to simulate the vehicle data that would act as a proof of concept for our original idea.

For wiring the simulated board, we had the Devkit V1 on the breadboard and connected to user buttons/potentiometer. Unfortunately the ESP32 is a bit too big for the breadboard, only allowing wiring on one side of the board. It almost worked out, but since the potentiometer needed analog input, it would use an ADC channel. The WiFi channel also uses ADC meaning both the potentiometer and ESP-NOW couldn't work together. We got caught on the issue for a while, but the surprisingly simple solution was swapping to have the TTGO taking user inputs since it'd actually allow jumper cables to connect on both sides of the board.

Tasks Not Completed (5 pts)

(Describe the tasks that you originally planned to complete but were not completed. If all tasks were completed, state so. Max 250 words).

Task	Reason
Storing data offline using an SD Card when an internet connection isn't present (assumed vehicle would go to a stationary access point).	This method was scraped since it'd be pointless to only have cloud data updated when the vehicle returns to an internet connection, which would be the same as only knowing the car's status once it's returned.
Ideally, the WiFi SSID/PASS and PushBullet token would be updatable in order to allow any customer to test the product without going into the code.	The PushBullet API would require going onto the site and signing up, requesting a token, then sending it to the server to update it, which we never ended up simplifying/implementing. For WiFi, it would need the Devkit V1 to be setup as an access point in order to setup WiFi connection. Overlapping this with ESP-NOW became an issue we got stuck on and eventually scraping (though we figured it out for swapping between WiFi HTTP request and ESP-NOW).
More inputs to allow for more functionality, such as initial mileage input to allow for notifications on standard maintenance jobs (oil change, tire rotation, etc).	Would require software to know when the last oil change/tire change occurred in order to know when the next one should occur.

WEAK POINTS / FUTURE WORK (15 pts)

(Mention at least two points of your project that have room for improvement. These points can be additions to the existing project setup (components) or improvement of the current implementation. Max 200 words).

As mentioned in the incomplete tasks, we could improve our product by allowing the user to input mileage and allow user to input when they perform maintenance to notify them for the next one (based on how much time passes or mileage value). I'd also love to go back to the original solution and attempt getting the OBD2 shield to properly pull CAN bus data. Since our proof of concept works at a basic level, if the shield works, it'd take a bit more code on the TTGO while maintaining the rest of the architecture. Lastly, it works for me since the values are hard coded, but it's not "other"-user friendly. Having the WiFi credentials and PushBullet token updatable without requiring the user to change the code and compile it again would make the product much easier to use/test.

SOURCE CODE (25 pts)

Please include a link to the source code of your project. A link to a repository (like [GitHub](#)) is preferred.

[CS596---IOT/Final Project at main · emsfur/CS596---IOT](#)