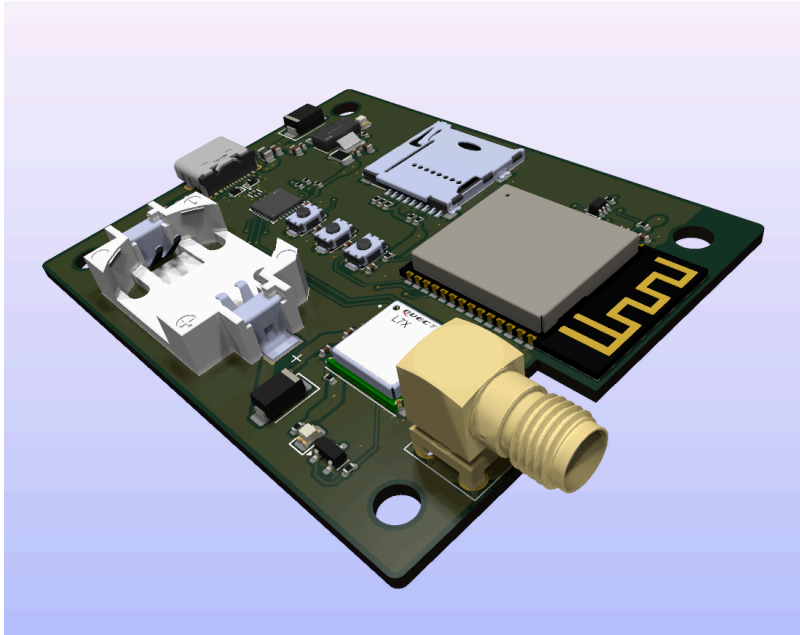


Jake Odgers

Mileage Logger Project



Scope

To log and display mileage for business purposes. With the ability for switching from personal journey to business journey.

Technical Design Skills

- RF waveguide design with a GCPW (Grounded Coplanar Waveguide)
- Differential pair routing for USB C datalines
- EMC and transient considerations
 - TVS (Transient Voltage Suppression)
 - Common mode chock
 - 4 layer board design with ground pours and power plane

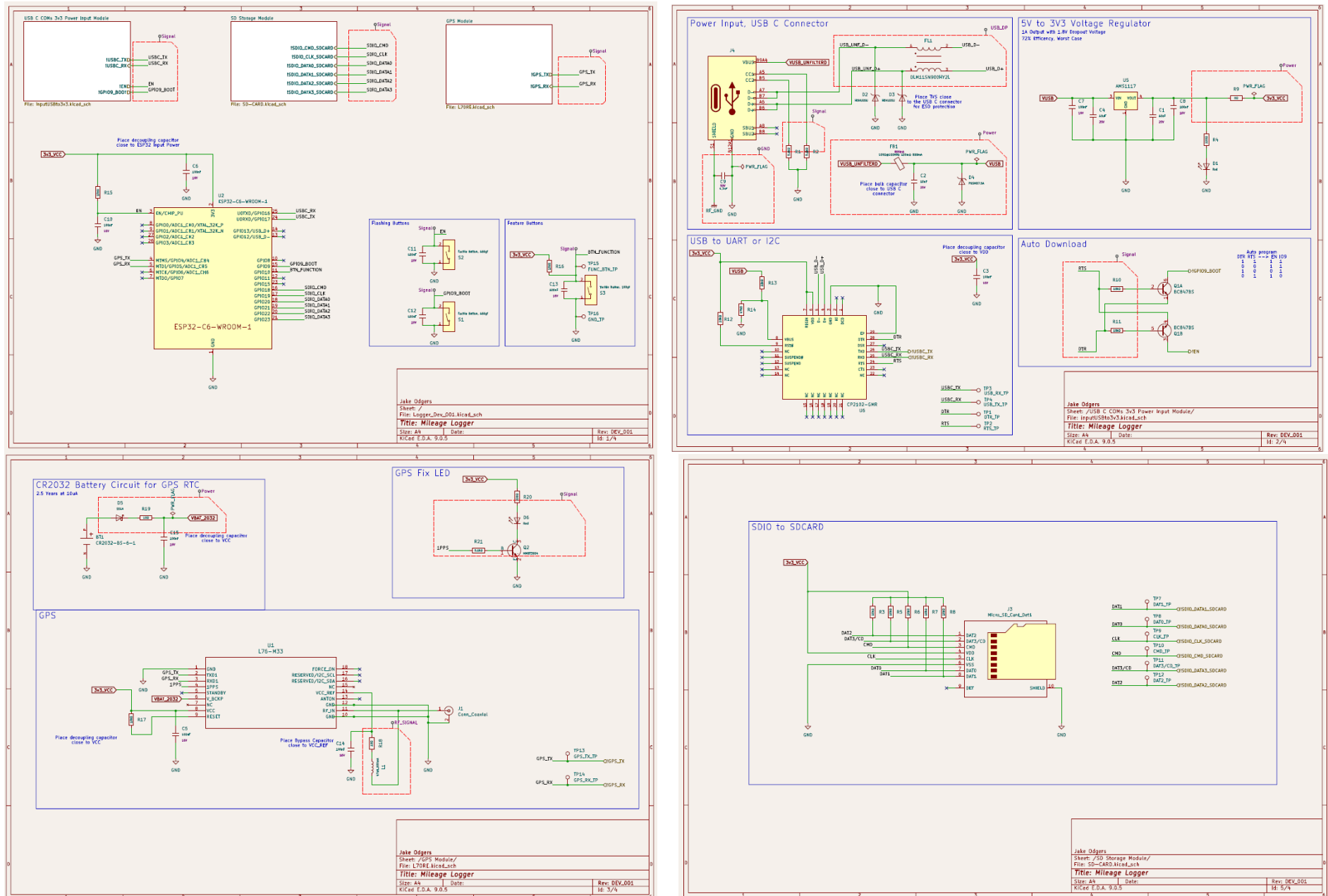
Display

Option 1 - Display through a GUI on a computer made through Python Tkinter.

Jake Odgers

Mileage Logger Project

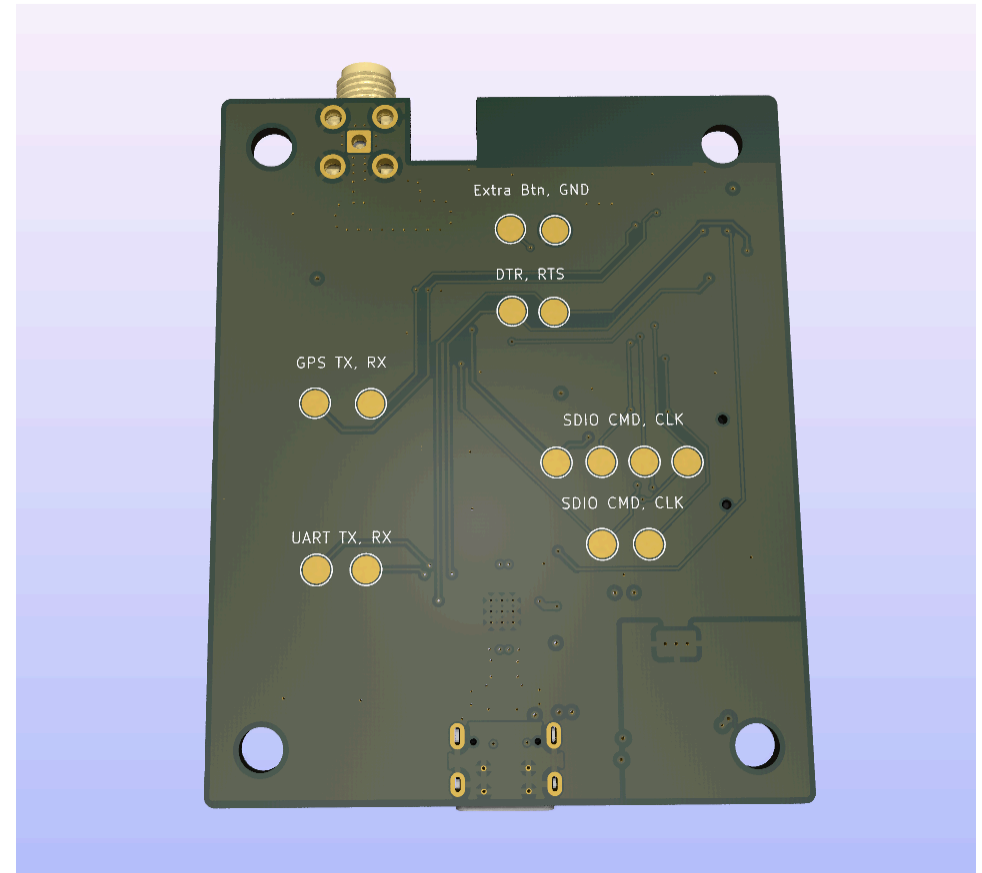
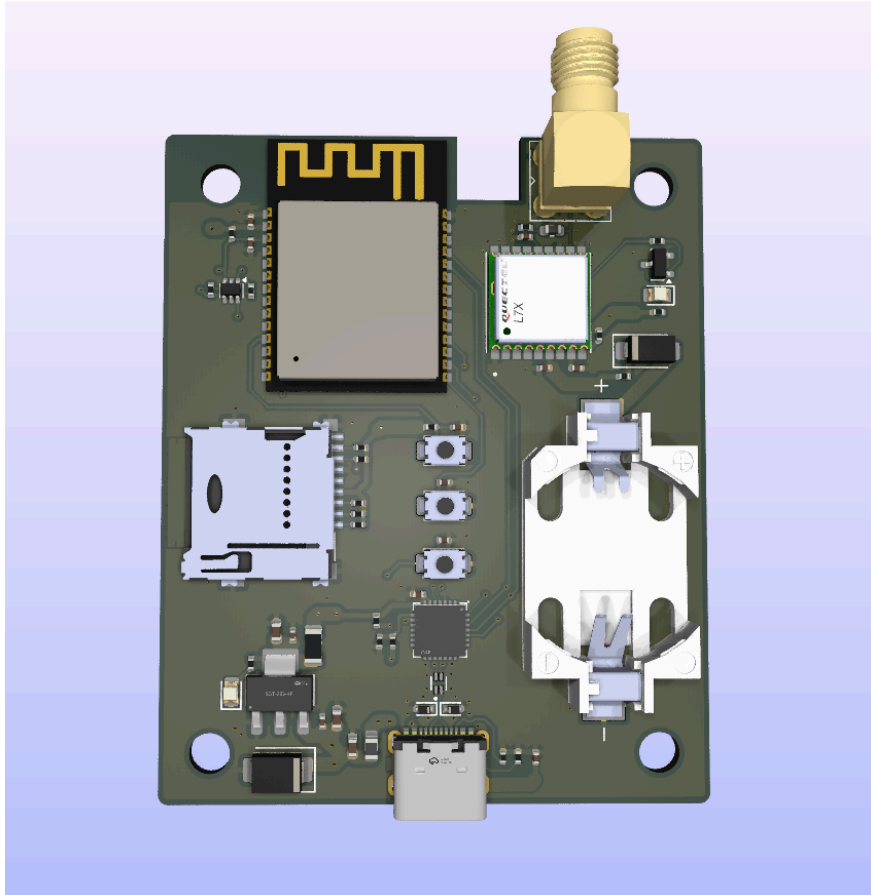
Schematic



Jake Odgers

Mileage Logger Project

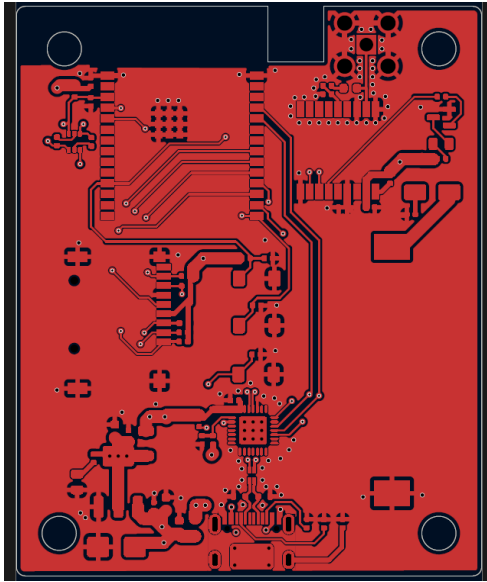
PCB Layout



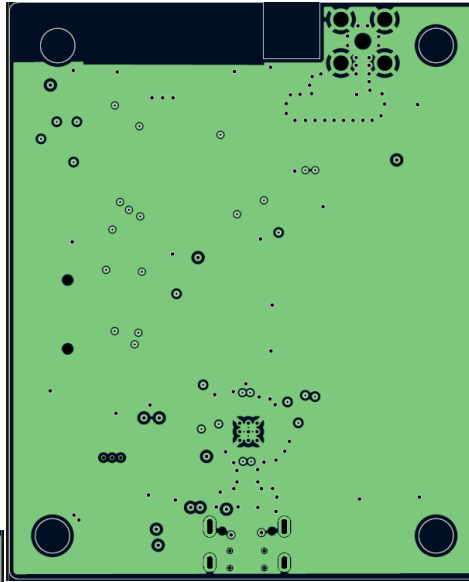
Jake Odgers

Mileage Logger Project

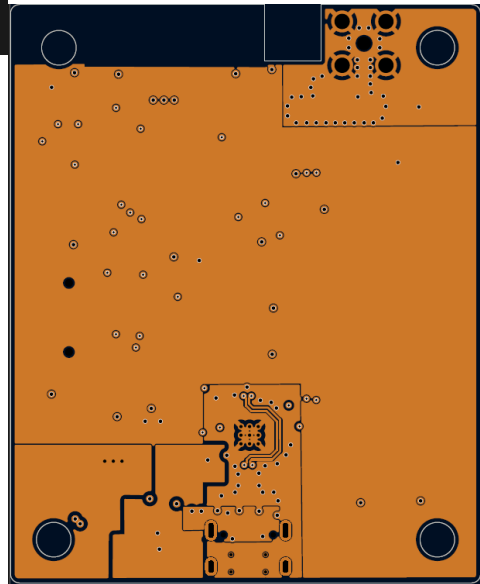
PCB Layer View



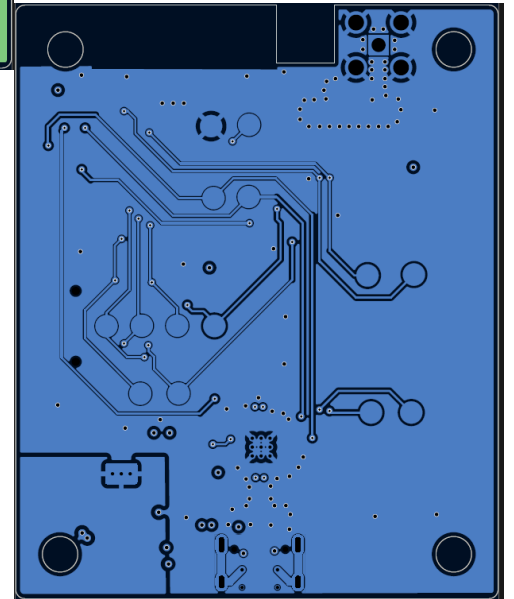
Layer 1 - Signal
(with GND pour)



Layer 2 - GND



Layer 3 - Power
(with GND pour
around differential
pair)



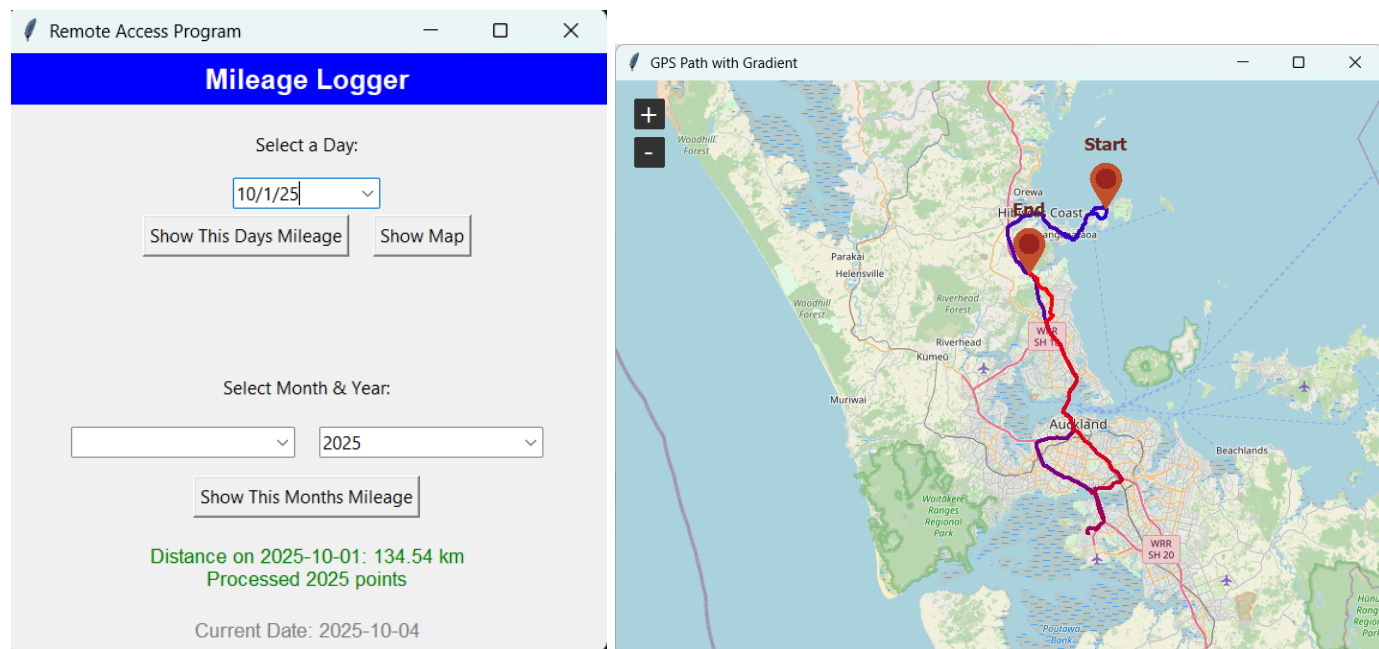
Layer 4 - Signal
(with GND pour)

Jake Odgers

Mileage Logger Project

GUI

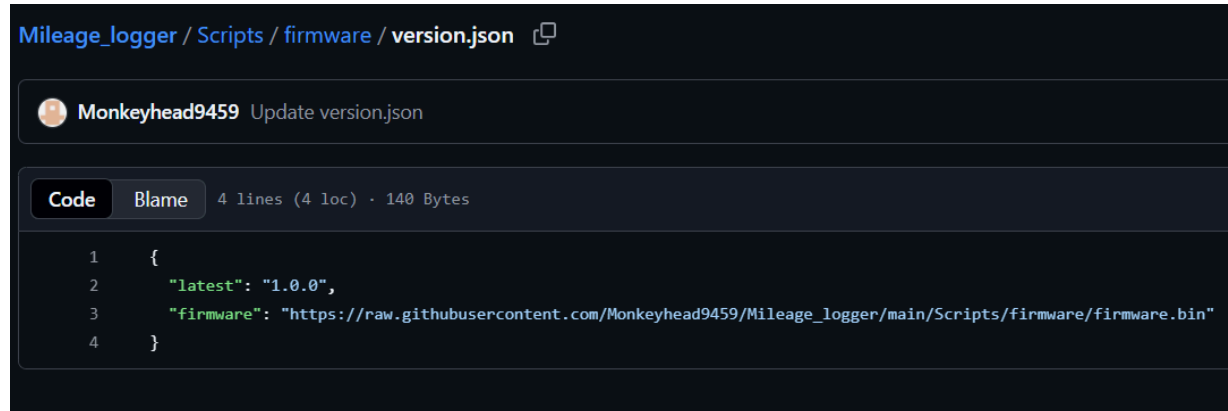
Option 1 - Computer Display



Mileage Logger Project

Back End

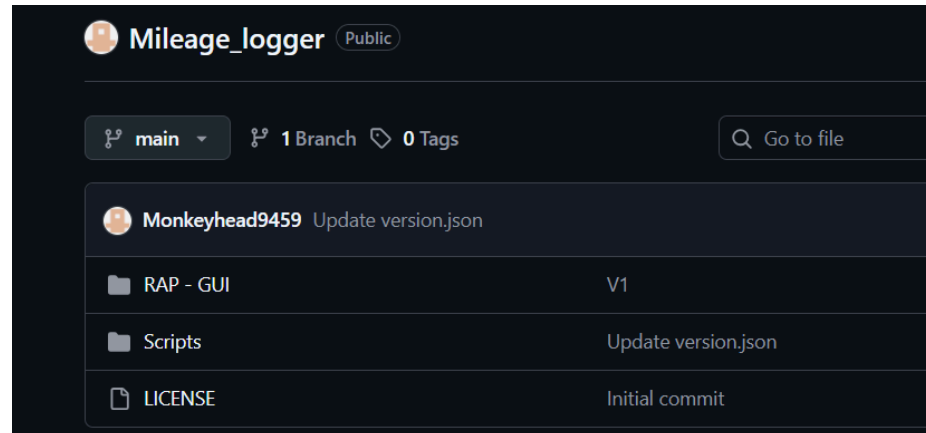
ESP32 REMOTE FIRMWARE UPGRADE THROUGH GITHUB



A screenshot of a GitHub file view for the file `version.json` located at `Mileage_logger / Scripts / firmware /`. The file was updated by user `Monkeyhead9459`. The code is displayed in a dark-themed editor with line numbers 1 through 4. It shows a JSON object with two properties: `latest` and `firmware`. The `latest` property is set to `"1.0.0"`, and the `firmware` property is set to a raw.githubusercontent.com URL pointing to `firmware.bin`.

```
1  {  
2    "latest": "1.0.0",  
3    "firmware": "https://raw.githubusercontent.com/Monkeyhead9459/Mileage_logger/main/Scripts/firmware/firmware.bin"  
4  }
```

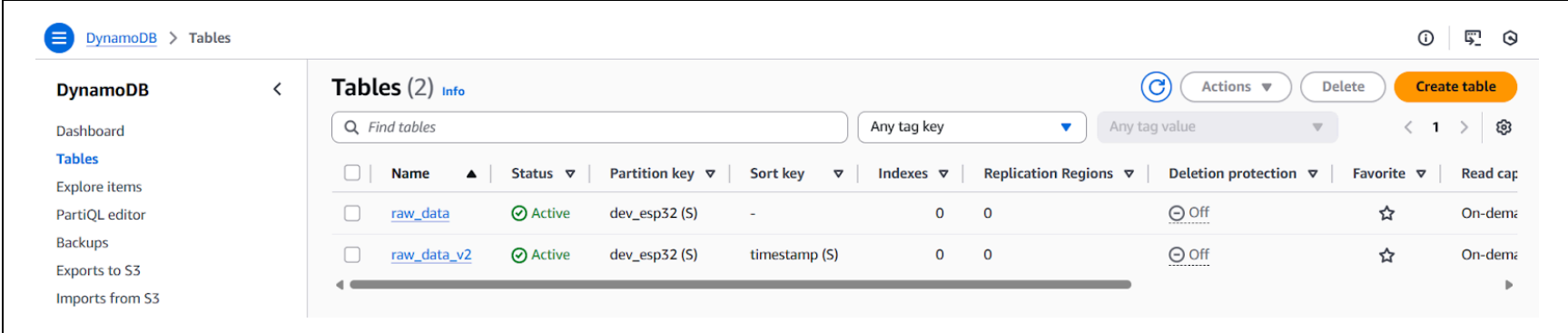
GITHUB VERSION CONTROL



Mileage Logger Project

Back End

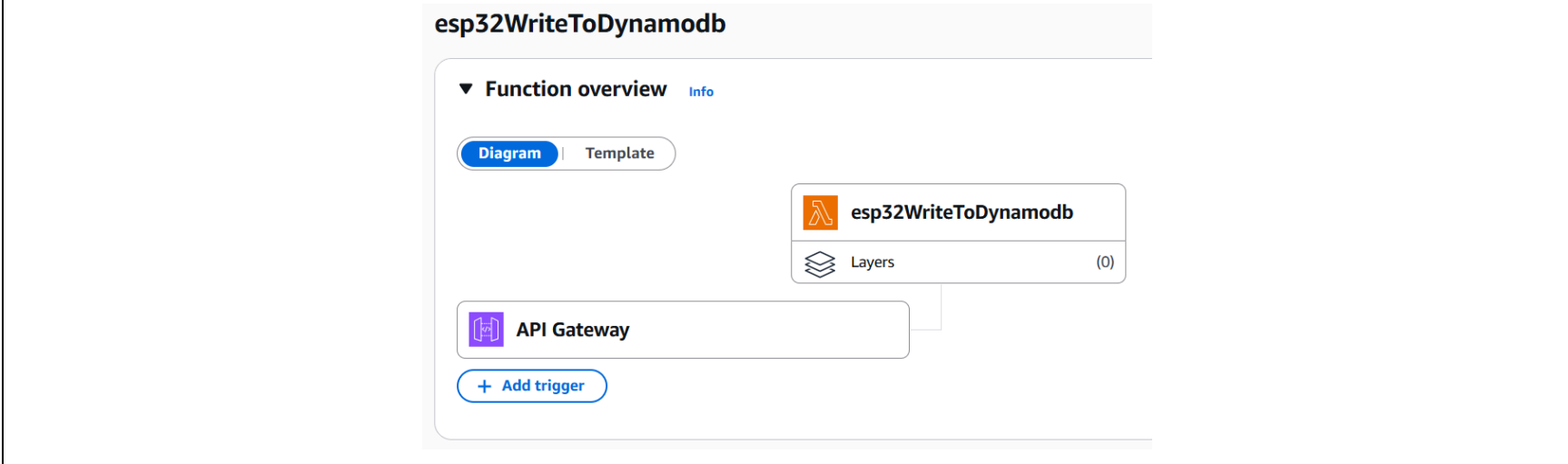
AWS DYNAMODB TABLE FOR CLOUD STORAGE



The screenshot shows the AWS DynamoDB console interface. On the left is a navigation menu with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, and Imports from S3. The main area is titled 'Tables (2)' and contains a table with the following data:

	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read cap
<input type="checkbox"/>	raw_data	Active	dev_esp32 (S)	-	0	0	Off	☆	On-demand
<input type="checkbox"/>	raw_data_v2	Active	dev_esp32 (S)	timestamp (S)	0	0	Off	☆	On-demand

API GATEWAY TO LAMDA FUNCTION FOR ESP32 DATA STORAGE



The screenshot shows the AWS Lambda console for the function 'esp32WriteToDynamodb'. It includes a 'Function overview' section with tabs for 'Diagram' and 'Template'. The 'Diagram' tab shows a visual representation of the function's architecture, including an 'API Gateway' icon and a button to '+ Add trigger'. The function details show the name 'esp32WriteToDynamodb' and 'Layers (0)'.

Mileage Logger Project

Improvements:

- Incorporate Google Roads API for more accurate mileage logging by snapping to roads
- Make a portable option for battery connection.
 - SMPS (Increase efficiency)
 - Smaller profile
 - ESP32-C3-MINI-1U
 - Offboard wifi and bluetooth antenna
 - Local/Smaller GPS antenna
- Option 2 for display is making an app through Flutter to connect to the ESP32 through bluetooth and store and display results as well as change journey status from business to personal.