**Bike Dashboard that shows Real-Time Metrics with a Custom GUI on 5 inch Display Using Raspberry Pi (for Both EV and IC Vehicles)**

**Abstract**

This project is about creating a bike dashboard that shows real-time information for both Electric Vehicles (EVs) and Internal Combustion Engine Vehicles (ICVs). It uses a Raspberry Pi connected to an LCD screen to display important data while riding. For EVs, the dashboard shows speed, battery voltage, current, and state of charge (SoC). For ICVs, it shows speed, engine RPM, battery voltage, and engine temperature. The system collects data from the vehicle’s sensors using serial or GPIO connections. The Raspberry Pi reads this data and displays it live on the screen. This setup is flexible, easy to customize, and can be improved further by adding features like data logging.

**Introduction**

To build the dashboard’s graphical interface, I used various tools for both design and development. I first created the visual layout and components of the GUI using Figma and Photoshop, which helped in planning how the screen would look and where each metric would be placed. Once the design was ready, I recreated it using Python’s Tkinter library to make it functional on the Raspberry Pi. I also added basic animations and smooth transitions, especially for speed updates, to make the dashboard more visually responsive and easier to read while riding.

**Objectives**

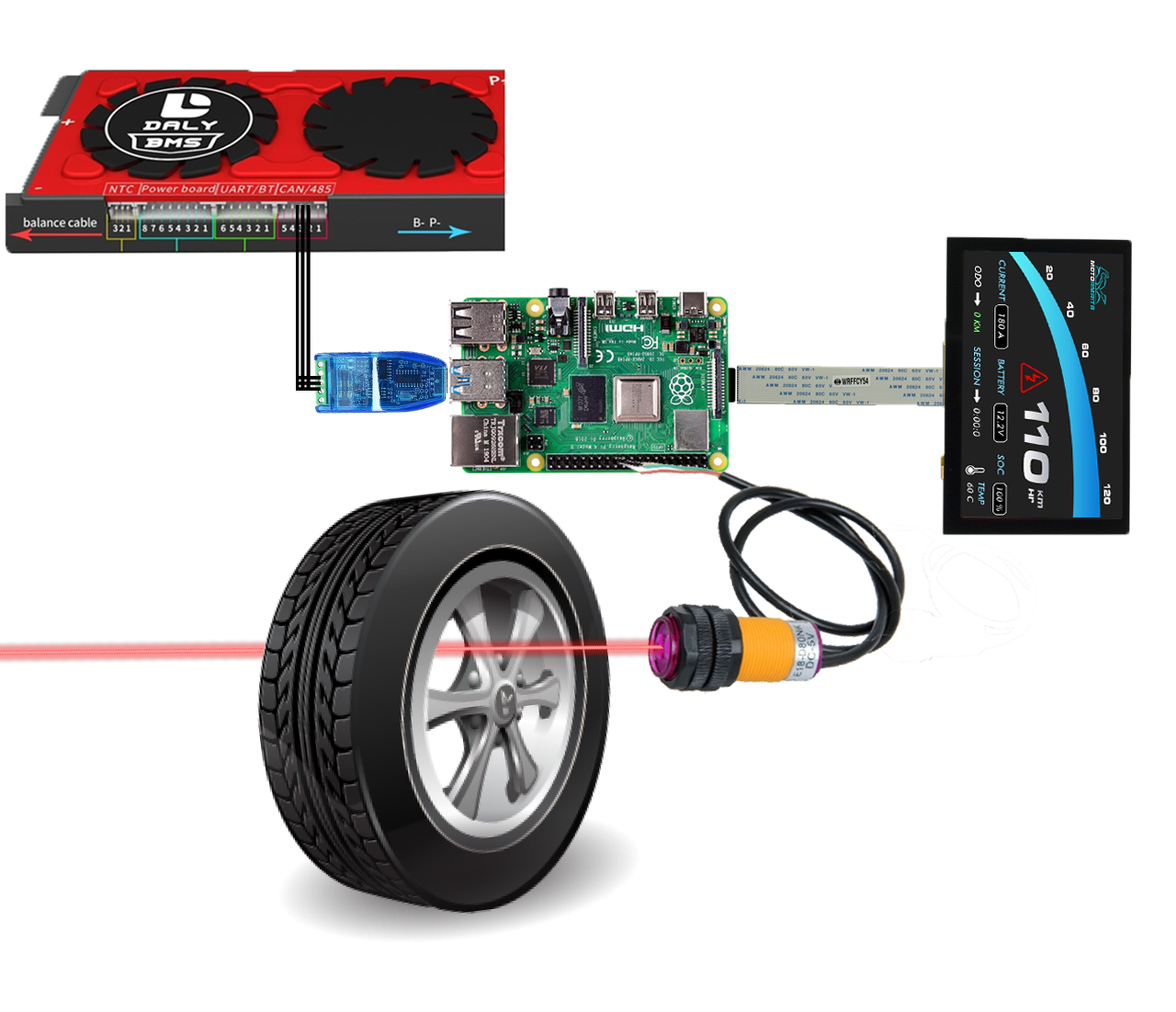
* Develop a real-time dashboard system that supports both EV and ICV bikes.
* Interface vehicle-specific sensors with the Raspberry Pi to obtain real-time data.
* Design a GUI that adapts to the vehicle type, displaying appropriate metrics.
* Provide clear, responsive visualization of live data on an LCD screen.
* Design a data acquisition system for a bike to collect and store sensor data for analysis.

**Components Used**

* Raspberry Pi 4 Model B
* IR Sensor (For both ICV and EV)
* OBD – II to Bluetooth
* ELM327 Bluetooth Module
* USB to RS485 Convertor
* 12V to 3.3V Convertor Module
* 16 Pin OBD2 Male to Female Convertor
* 5 Inch Display from CrystalFontz
* Wiring, connectors, and power supply
* **Software tools:** Python, GUI library Tkinter, Photoshop, Figma, Tkdesigner Tool

**System Design**

**Hardware (EV)**



DALY BMS

.

5’ INCH DISPLAY

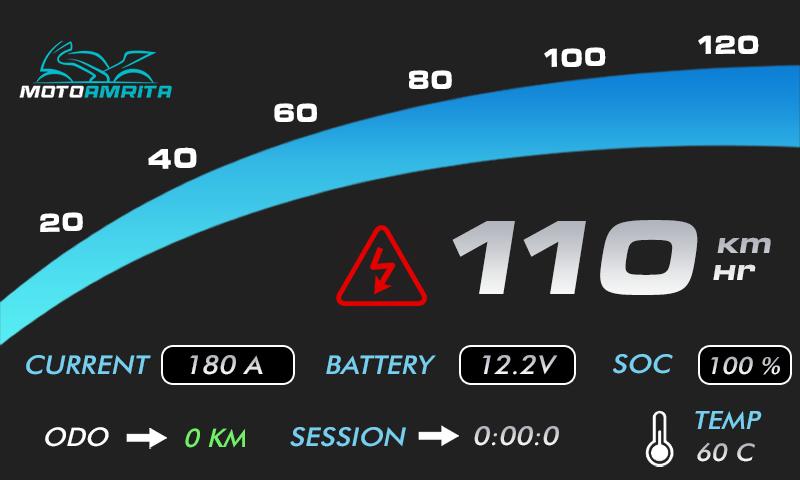
RASPBERRY PI 4

USB TO RS485

Wheel

IR Sensor

DSI CABLE

**Software (EV)**

The GUI design was created using Photoshop and Figma, and then converted to Tkinter code using a tool called TkDesigner.

**Implementation (EV)**

For the Electric Vehicle (EV) setup in our project, we use a Daly Battery Management System (BMS) to monitor the battery's health and performance. This BMS supports different communication methods such as CAN, RS485, UART, and Bluetooth. In our case, we chose RS485 communication, as it is reliable for long-distance and stable data transmission in automotive applications.

To connect the Daly BMS to our system, we used an RS485 to USB converter. This converter allows us to link the BMS with a laptop or a Raspberry Pi 4 through a USB port. Once connected, the BMS continuously sends real-time battery data through the RS485 interface.

On the Raspberry Pi 4, I wrote a Python script using the open-source daly-bms library. This library helps to communicate with the BMS and fetch key battery parameters such as:

* Battery voltage
* Current
* State of Charge (SoC)
* Cell voltages
* Battery temperature
* Range of vehicle

The script reads this data at regular intervals and passes it to a custom-designed Graphical User Interface (GUI) that I built using Python's Tkinter GUI library. The GUI displays the live data in a simple and easy-to-read layout, showing the most important metrics for the rider or engineer during testing.  
The speed of the bike is calculated from the wheel RPM using an IR sensor, which detects the number of interruptions (or cuts) made by the wheel’s rim. The speed is then determined using the appropriate formula based on these readings.

This dashboard makes it easier to monitor battery health and performance in real-time during vehicle operation or testing. The system can also be expanded to log data for further analysis.