LAB 5 - Mobile App DC Motor Control with Grafana Dashboard

1. Overview

In this lab, students will design a **mobile application** (using **MIT App Inventor**) to remotely control a **DC motor** via an **ESP32 (MicroPython)** web server.
The ESP32 will expose HTTP endpoints (/forward, /backward, /stop, /speed), while the mobile app will send commands over Wi-Fi to control direction and speed.
All control actions and speed updates will be recorded to an **IoT dashboard (InfluxDB + Grafana)** for real-time monitoring and analysis.

---

2. Learning Outcomes (CLO Alignment)

After completing this lab, students will be able to:

- Design and implement a full IoT actuation system integrating hardware (ESP32 + L298N), mobile app, and cloud dashboard.
- Develop a custom mobile interface using MIT App Inventor to send REST commands.
- Use MicroPython to create a lightweight web API for motor control.
- Configure InfluxDB and Grafana to log and visualize actuator data (speed, direction, timestamp).
- Evaluate system performance (response latency, data accuracy) and propose improvements.

---

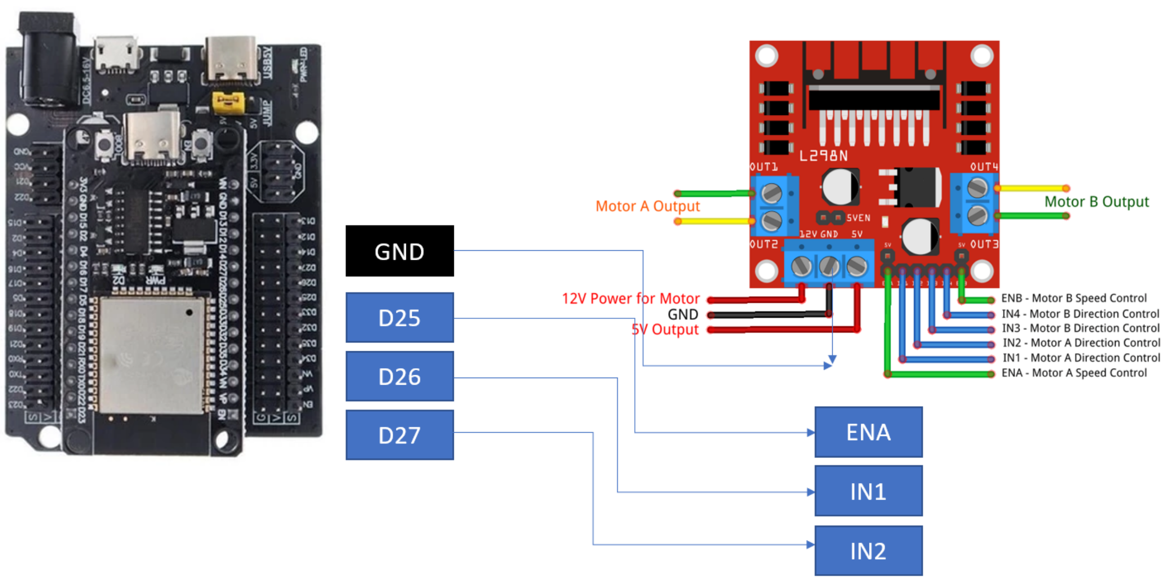
3. Equipment

- ESP32 Dev Board (MicroPython flashed)
- L298N motor driver
- DC motor + power supply (7–12 V)
- Jumper wires + breadboard
- Laptop with Thonny IDE
- Android phone with MIT App Inventor installed
- Wi-Fi access point
- Grafana Cloud account or local InfluxDB server

---

4. Wiring Diagram

ESP32 Pin	L298N Pin	Function
25	ENA	PWM (speed)
26	IN1	Motor direction 1
27	IN2	Motor direction 2
GND	GND	Common ground



---

5. Tasks & Checkpoints

Task 1 – ESP32 Web Server (15 pts)

- Implement endpoints /forward, /backward, /stop, and /speed?value=.
- Print every command and speed value to the serial monitor.
- Evidence: serial output screenshot showing commands.

Task 2 – Mobile App Design (20 pts)

- Build an MIT App Inventor interface with:
  - Buttons: Forward, Backward, Stop
  - Slider: Speed (0–100 %)
  - Label for status.
- Buttons send requests to http://<ESP32_IP>/forward?speed=NN etc.
- Evidence: app screenshot and demonstration video.

Task 3 – Data Logging to InfluxDB (25 pts)

- Add HTTP POSTs in ESP32 code to send logs like:
- {"timestamp": "<ISO_time>", "action": "forward", "speed": 70}

  to InfluxDB or Node-RED endpoint.

- Verify the data appears in InfluxDB bucket/table.
- Evidence: terminal log + InfluxDB data preview.

Task 4 – Grafana Dashboard (20 pts)

- Configure Grafana to visualize:
  - Motor speed vs time
  - Last command direction
  - Table of events with timestamps
- Evidence: screenshot of Grafana dashboard updating in real time.

Task 5 – Reliability & Analysis (10 pts)

- Add Wi-Fi auto-reconnect logic.
- Handle bad HTTP requests gracefully (print error and continue).
- Discuss response delay and accuracy issues.

Task 6 – Documentation (10 pts)

Submit a **README.md** including:

- Wiring diagram/photo
- App layout and URLs used
- Grafana dashboard setup steps
- Screenshot of working system
- Reflection on latency and data logging.

---

6. Submission & Academic Integrity

Submit to a private GitHub repo (add instructor as collaborator).
Include:

- main.py (ESP32 code)
- .aia file (exported MIT App Inventor project)
- README.md with wiring and setup
- Screenshots of app, serial logs, and Grafana dashboard
- Short demo video (1–2 min) showing app control and dashboard updates