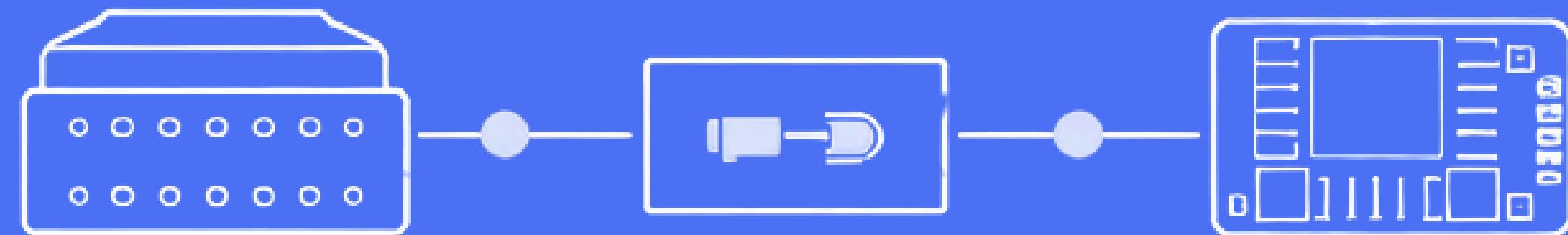


OBD Car Monitor

PROJECT GROUP N. 35

Leonardo Begnoni
Leonardo Magaraggia
Tommaso Lo Magno



OBD

CAN Bus

ESP32

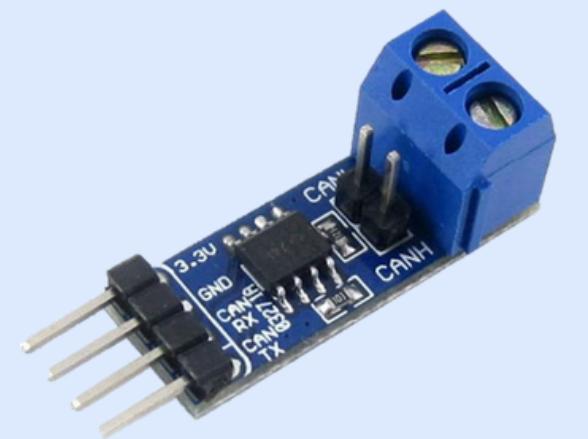


UNIVERSITÀ
DI TRENTO

Components



ESP32D



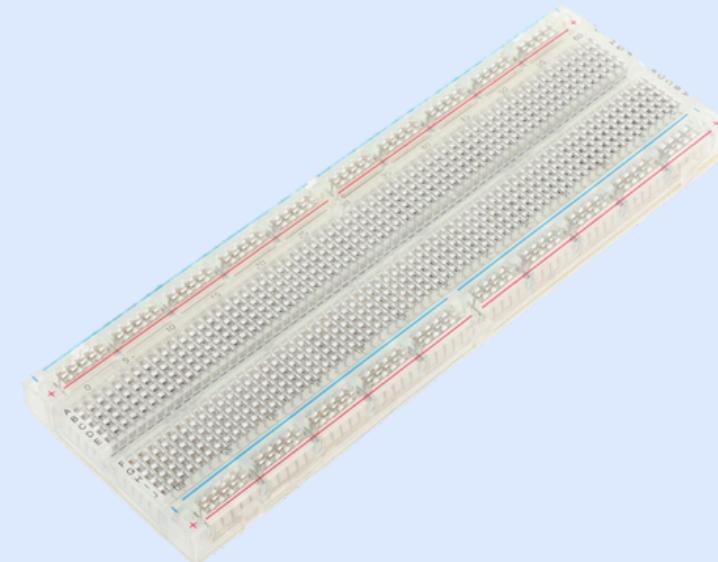
TRANSCEIVER CAN



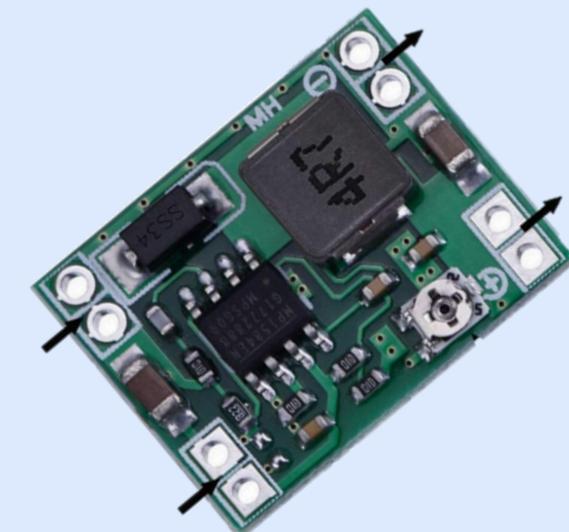
LCD 16X2



OBD CABLE
and WIRING



BREADBOARD



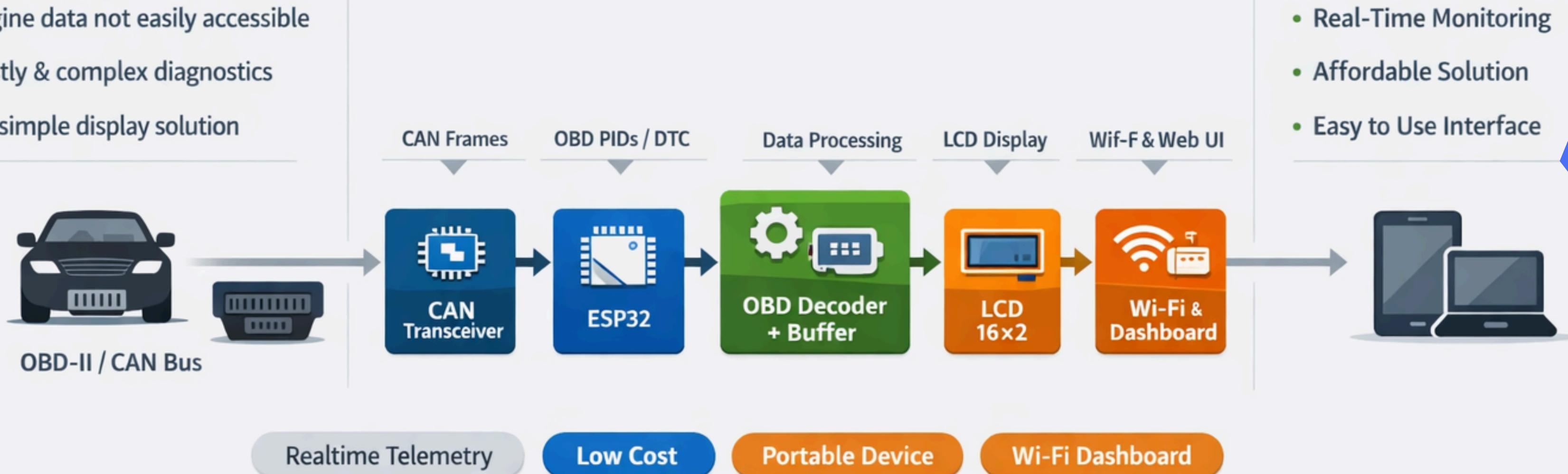
STEPDOWN
12V-5V

! Problem

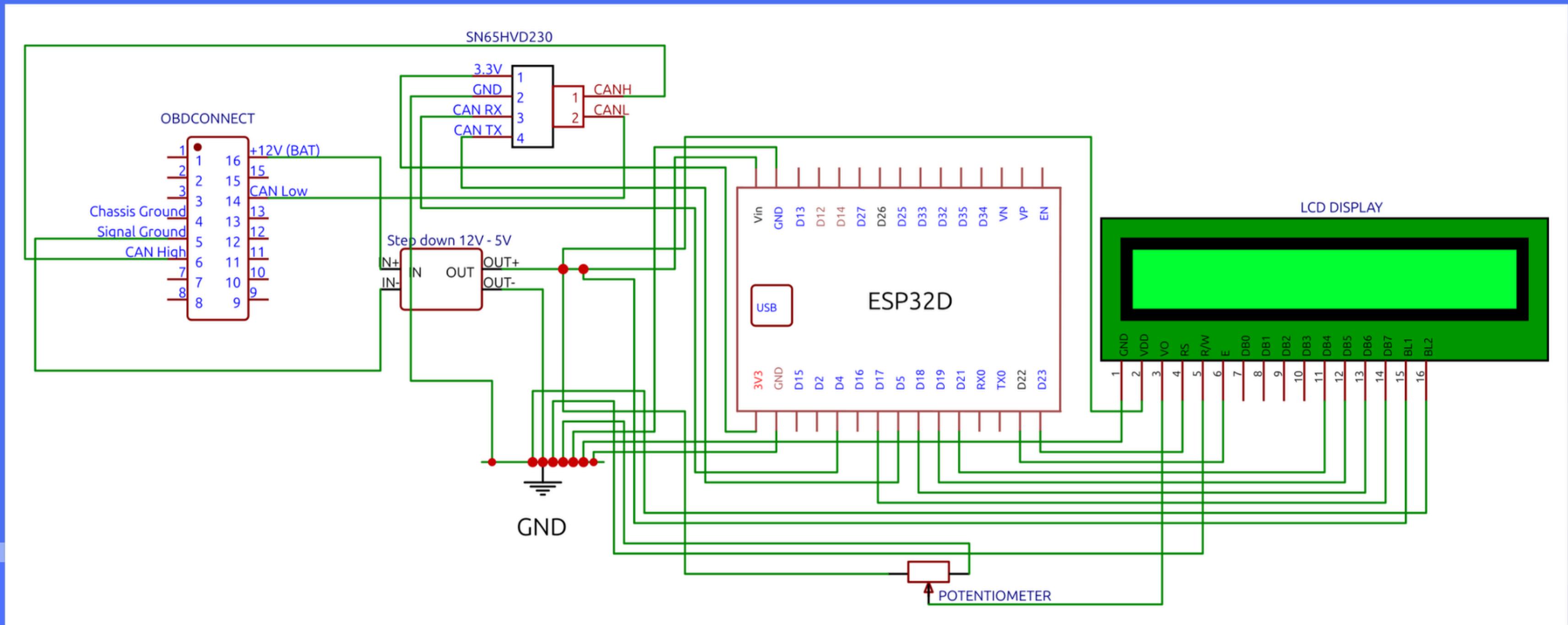
- Engine data not easily accessible
- Costly & complex diagnostics
- No simple display solution

System

✓ Value



SCHEMATICS



```
void can_bus_init(void)
{
...
}
esp_err_t can_bus_send(twai_message_t *msg)
{
...
}
esp_err_t can_bus_receive(twai_message_t *msg, TickType_t timeout)
{
...
}
```

can-bus.c

```
void obd_init(void);

bool obd_read_pid(uint8_t pid, uint8_t out[4]);

void obd_data_init(void);
|
void obd_data_start_polling(void);

obd_full_data_t obd_get_all_data(void);
```

obd.c

MODULES

```
obd_full_data_t d;
static esp_err_t data_handler(httpd_req_t *req)
{...
}

static blob_t get_blob_for_uri(const char *uri, const char **mime_out)
{...
}
static esp_err_t static_handler(httpd_req_t *req)
{...
}

void web_server_start(void)
{...
```

web_server.c

```
void lcd_init(void);

void lcd_clear(void);

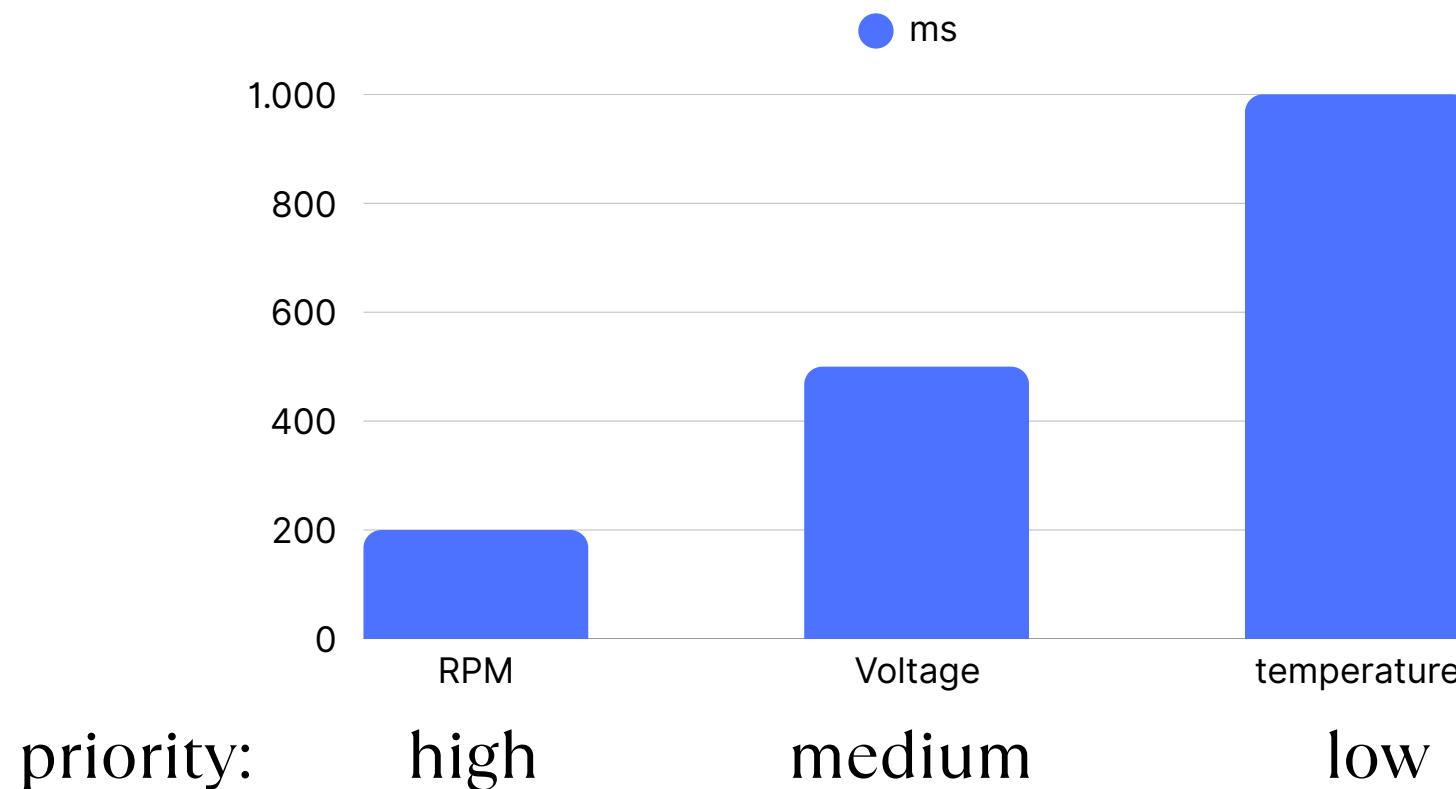
void lcd_gotoxy(uint8_t col, uint8_t row);

void lcd_print(const char *str);
```

lcd.c

Real-Time scheduling

- **Scheduling & periodicity:** polling multiple PIDs with different rates (fast/medium/slow).
- **Policies:** choose the next job using a policy that combines priority and lateness (deadline miss / delay)
- **Data structures:** maintain a job list/array with timing metadata and failure state for each PID.



```
static int pick_next_job(uint32_t tnow)
{
    int best = -1;
    int best_prio = 999;
    int32_t best_lateness = -2147483647;

    for (int i = 0; i < (int)(sizeof(s_jobs) / sizeof(s_jobs[0])); i++)
    {
        pid_job_t *j = &s_jobs[i];

        if (tnow < j->backoff_until)
            continue;
        if (tnow < j->next_due_ms)
            continue;

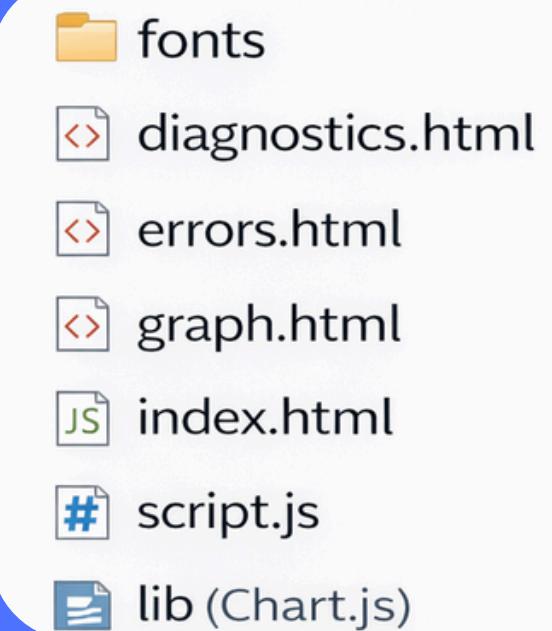
        int prio = (int)j->prio;
        int32_t lateness = (int32_t)(tnow - j->next_due_ms);

        if (prio < best_prio || (prio == best_prio && lateness > best_lateness))
        {
            best = i;
            best_prio = prio;
            best_lateness = lateness;
        }
    }
    return best;
}
```

IoT interface

- **Endpoint HTTP:** expose vehicle sensor as a resource (/data), implemented with esp_http_server (ESP-IDF)
- **JSON schema:** small, stable keys for dashboard use
- **Refresh rate:** client-side polling interval (UI refresh every 100ms)
- **static file hosting:** HTML/CSS/JS embedded in firmware (no external server)

static UI hosting



Testing and issues

Testing approach

- Can > OBD > Scheduler > Web
 - **Debug** using *idf.py monitor + log*
 - *Real testing on vehicles (OBD-II)*
 - Test JSON on browser (*endpoint /data*)
-

Issues Encountered

- JSON value always 0 → unsupported PID/ incorrect timing → filtered supported PIDs + implemented timeout handling
- Firmware too large → application partition too small → extended factory partition to 3MB
- LCD not communicating correctly → fixed init sequence, correct row offset, added microsecond delays

Future improvements

UI update

- **adding DTC reading:** DTC page with status, timestamp, “clear DTC” button.
 - **live notification** and **alerts** (es. over-temp, low-battery)
 - **Trip summary:** start/stop session, estimated distance, avg values.
 - **customizable dashboard:** select widgets
-

Hardware update

- **strong power path:** TVS on 12V, EMI filter...
- **Ignition-aware power management:** reduce drain battery
- **printed case** and **LED status** indicator for alerts

Team contributions:

- **Leonardo Begnoni:** IoT Interface and algorithms
- **Leonardo Magaraggia:** Software architecture and real time scheduling
- **Tommaso Lo Magno:** Hardware and system working scheme