CAN Display using Arduino nano &mcp2515

# Aim:

The objective is to interface a microcontroller (Arduino Nano) with a CAN controller (MCP2515) to receive battery state-of-charge data and compute the remaining range (DTE) of an electric vehicle. The calculated DTE is then displayed in real-time on a 4-digit 7-segment display (Catalex/TM1637), allowing users to monitor how far the vehicle can travel before the battery depletes.

# Principle:

This project is based on the principles of **Controller Area Network (CAN) communication**, **battery energy estimation**, and **digital display control**.

# Components:

Arduino Nano

4-digit 7-segment display (Catalex/TM1637Display)

Mcp2515 CAN Module

Battery

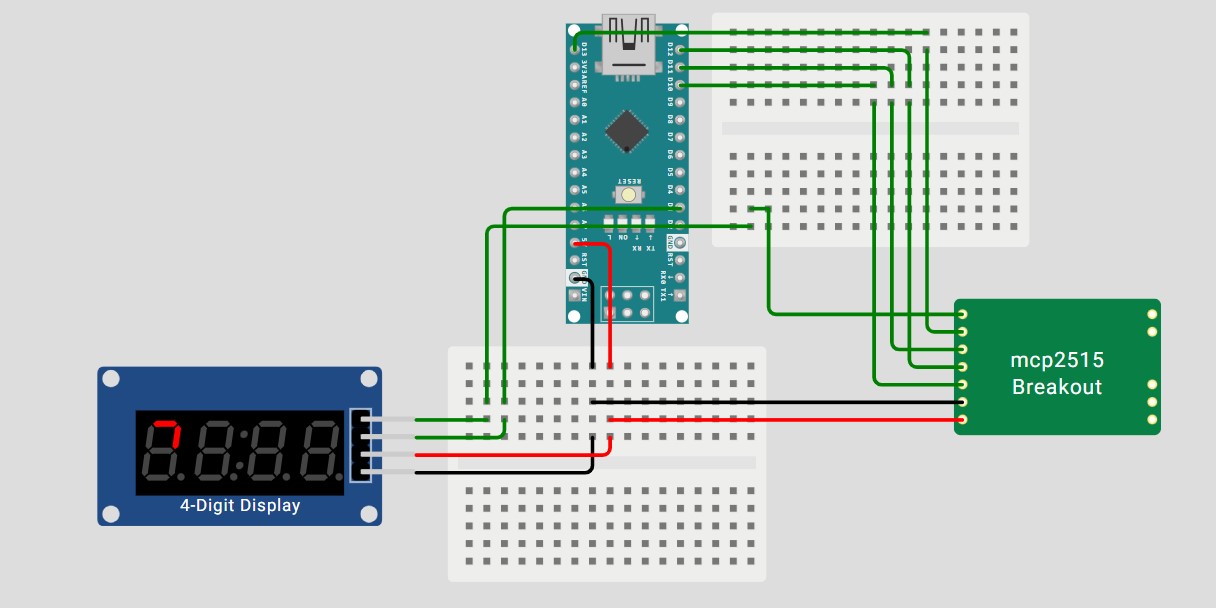
Arduino IDE

# Libraries to install:

TM1637TinyDisplay

Mcp\_can

# Circuit Diagram:



# Hardware Connections:

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | MODULE USED | ARDUINO NANO | MODULE PINS |
| 1 | MCP2515 CAN Module | 5V | VCC |
| 2 | GND | GND |
| 3 | D10 | CS |
| **4** | **D11** | **MOSI** |
| 5 | D12 | MISO |
| 6 | D13 | SCL |
| 7 | D2 | INT |

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | MODULE USED | ARDUINO NANO | MODULE PINS |
| 1 | Catalex/TM1637 | 5V | VCC |
| 2 | GND | GND |
| 3 | D3 | DIO |
| 4 | D2 | CLK |

# CODE:

#include <TM1637TinyDisplay.h>

#include <SPI.h>

#include <Wire.h>

#include <mcp\_can.h>

#include <TM1637TinyDisplay.h>

const int SPI\_CS\_PIN = 10;

MCP\_CAN CAN(SPI\_CS\_PIN);

#define CLK 2

#define DIO 3

TM1637TinyDisplay display(CLK, DIO);

unsigned long lastCanMsgTime = 0;

const unsigned long timeout = 2000; // 2 seconds timeout

void setup() {

Serial.begin(115200);

if (CAN.begin(MCP\_ANY, CAN\_500KBPS, MCP\_8MHZ) == CAN\_OK) {

Serial.println("CAN Bus Initialized");

} else {

Serial.println("CAN Bus Initialization Failed");

while (1);

}

CAN.setMode(MCP\_NORMAL);

display.setBrightness(7);

display.showNumber(0); // Show 00 at start

}

void loop() {

// Check for CAN message

if (CAN\_MSGAVAIL == CAN.checkReceive()) {

unsigned long id;

byte len;

byte data[8];

CAN.readMsgBuf(&id, &len, data);

int idShort = id & 0xFFF;

if (idShort != 0x008) return;

// Update last received time

lastCanMsgTime = millis();

// Parse DTE

unsigned long remainingAhRaw =

((unsigned long)data[1] << 24) |

((unsigned long)data[2] << 16) |

((unsigned long)data[3] << 8) |

((unsigned long)data[4]);

float remainingAh = remainingAhRaw \* 0.001;

float remainingPower = (remainingAh \* 51.2) - 230.4;

float DTE = remainingPower / 18;

Serial.print("DTE: ");

Serial.println(DTE);

// Display DTE (integer part only)

int dteInt = (int)DTE;

display.showNumber(dteInt);

}

// Timeout check: if no CAN message in `timeout` period

if (millis() - lastCanMsgTime > timeout) {

display.showNumber(0); // Show 00 when battery/CAN is inactive

}

}

# Formulas:

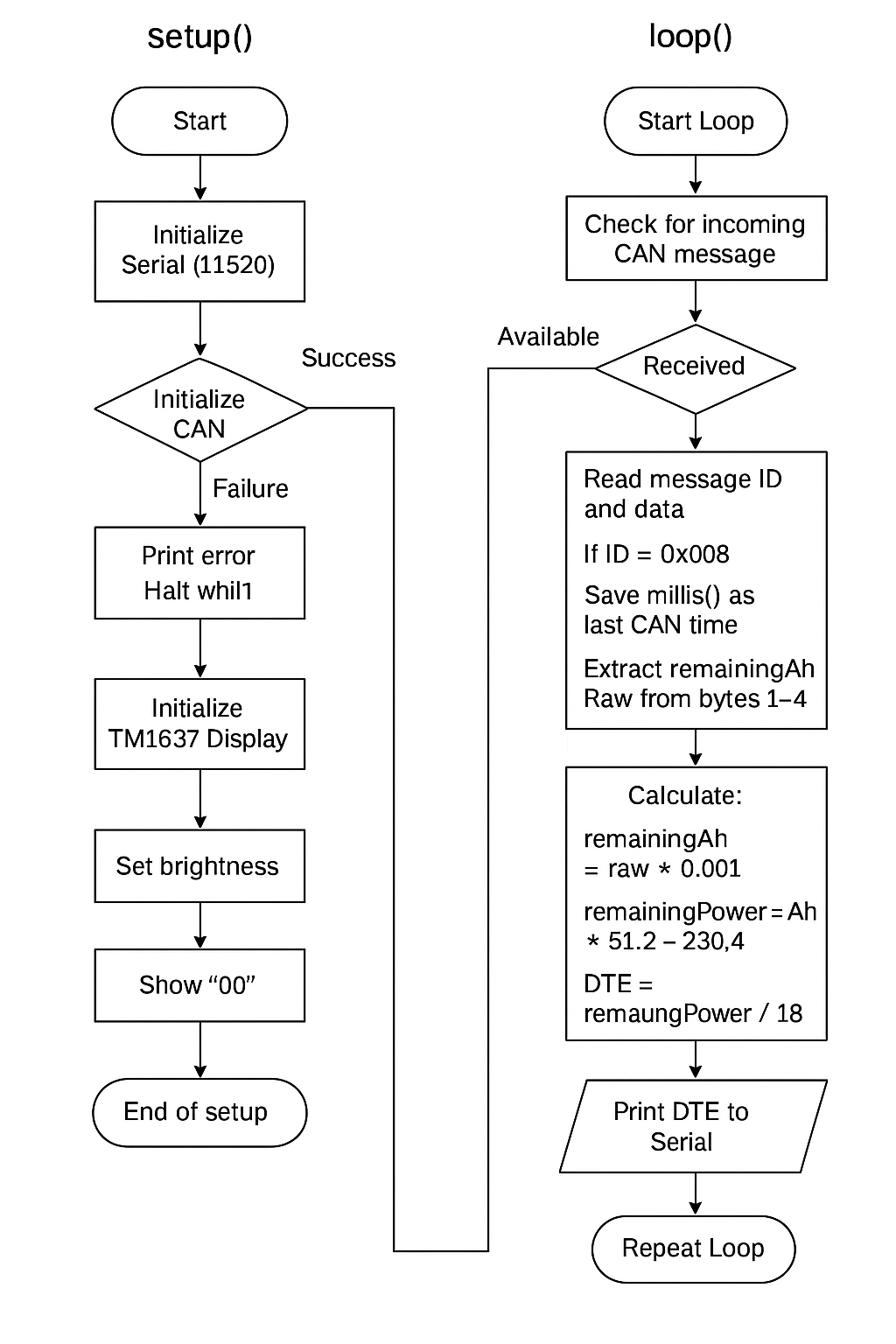
RemainingAh=(data[1]×2^24+data[2]×2^16+data[3]×2^8+data[4])×0.001.

RemainingPower (Wh)=(RemainingAh×51.2)−230.4.

DTE (km)=RemainingPower (Wh)/18​ .

**🔧 Benefits**

1. Reduces need for a full CAN logger setup
2. Compact and low-cost display module
3. Easy to integrate into existing CAN systems
4. Reusable and extensible for other CAN messages (e.g., motor control, temperatures)



**⚠️ Challenges Faced in the Project**

During the development of the CAN-based DTE (Distance To Empty) display system using Arduino Nano, MCP2515, and a TM1637 display, several challenges were encountered:

**🔌 1. CAN Communication Initialization**

* **Issue:** Difficulty in establishing stable communication with the MCP2515 module.
* **Reason:** Incorrect SPI settings or wrong crystal frequency (8 MHz vs. 16 MHz).
* **Solution:** Verified MCP2515 frequency and selected correct parameters: MCP\_8MHZ in code.

**📟 2. Display Not Updating**

* **Issue:** TM1637 display was stuck showing 88:88 or garbage values.
* **Reason:** Wrong library used (TM1637TinyDisplay has limitations vs. TM1637Display).
* **Solution:** Matched display pins in code and used compatible library functions for numeric display.

**📁 3. Library Compatibility Issues**

* **Issue:** Errors like showNumberDecEx not recognized.
* **Reason:** That function exists in TM1637Display.h, not TM1637TinyDisplay.h.
* **Solution:** Adapted code to display integer-only DTE values or scaled decimal manually.

**⚙️ 4. No CAN Data Received**

* **Issue:** Display always showed 00 even though code was correct.
* **Reason:** Incorrect message ID filter or no actual CAN traffic.
* **Solution:** Ensured matching message ID (0x008), verified with serial monitor, and confirmed CAN sender functionality.

**🚀 How This Project Can Be Improved in the Future:**

To enhance the **accuracy, functionality, and real-world usability** of the CAN-based Distance To Empty (DTE) display project, here are key improvements and upgrades you can explore:

**1. 🔋 Battery Data Accuracy Enhancement**

* **Current Method:** DTE is calculated based on basic Ah and voltage.
* **Improvement:**
  + Add **real-time current sensing** (with INA219 or ACS712).
  + Factor in **temperature and battery health** from CAN data or sensors.

**2. 📲 Mobile App Integration**

* **Improvement:** Use **Bluetooth (HC-05 or ESP32)** to send DTE data to a smartphone.
* Display live DTE, battery status, and CAN errors in a **user-friendly Android/iOS app**.

**3. 🧠 Intelligent DTE Estimation**

* **Improvement:** Use **machine learning models** or adaptive algorithms that:
  + Learn from user driving patterns.
  + Adjust DTE in real-time based on speed, terrain (with GPS), and load.

**4. 📦 Compact Custom PCB**

* **Improvement:** Instead of using modules and jumper wires:
  + Design a **custom PCB** combining Arduino, MCP2515, and TM1637.
  + Increases reliability, reduces size, and is production-ready.

**5. 🧾 Datalogging with SD Card**

* **Improvement:** Add an **SD card module** to log:
  + SOC, DTE, current, voltage, temperature, errors.
  + Helpful for diagnostics, warranty tracking, and performance analysis.

**6. 🗺️ GPS & Terrain-Based Adjustments**

* **Improvement:** Integrate a **GPS module (like NEO-6M)** to:
  + Track real-time location.
  + Adjust DTE based on distance to next charging point or uphill/downhill.

**7. ⚠️ Critical Alerts**

* **Improvement:** Add **LED indicators or a buzzer**:
  + If DTE < 10 km.
  + For battery errors or temperature warnings.