Sistem de Monitorizare RPM via CAN-Bus (OBD-II) cu Shift Light

Introducere

Proiectul reprezintă un sistem hardware-software de diagnoză auto activă, conceput pentru a extrage și interpreta date în timp real de pe magistrala CAN (Controller Area Network) a unui autovehicul.

- **Ce face:** Se conectează la portul OBD-II al mașinii (în cazul de față un Ford Fiesta 2007), interoghează unitatea de control a motorului (ECU) pentru valoarea turației (RPM), procesează pachetele de date și le afișează pe un ecran LCD, controlând simultan un sistem de avertizare vizuală tip “Shift Light” format din 3 LED-uri.
- **Scopul lui:** Realizarea unui instrument “standalone” pentru monitorizarea performanțelor motorului și asistarea șoferului pentru schimbarea optimă a treptelor de viteză.
- **Ideea de la care am pornit:** Dorința de a decoda și vizualiza traficul intern al mașinii (o rețea industrială complexă, extrem de zgomotoasă și aglomerată) fără a folosi testere comerciale (gen ELM327), ci comunicând nativ prin microcontroler.
- **Utilitate:** Demonstrează conceptele critice de rețele auto, gestiunea corectă a întreruperilor hardware pentru prevenirea pierderii de pachete și necesitatea izolării și stabilizării alimentării auto (12V → 5V) prin convertoare în comutație.
- **Elemente de noutate (5%):** Spre deosebire de o simplă interogare OBD-II clasică prin care se solicită un parametru (PID 0x0C), sistemul meu acționează și ca un CAN-Sniffer activ. Acesta interceptează direct pachetele de tip broadcast aruncate de ECU pe rețeaua mașinii (ex: ID 0x201 specific vehiculelor Ford), obținând astfel turația motorului instantaneu, cu latență zero, fără a mai încărca magistrala cu cereri proprii.

Ipoteză, Planificare și Măsurători

Ipoteză științifică: Credem că efectuarea citirii asincrone în cascadă a bufferului CAN (în interiorul rutinei de loop) combinată cu decodarea mesajelor broadcast native ale vehiculului (ID 0x201) va îmbunătăți considerabil performanța afișajului RPM deoarece va preveni fenomenul de “Interrupt Overrun” și va elimina latența cauzată de timpii de așteptare ai interogărilor standard OBD-II.

Metrici și Ținte de Performanță (Măsurători):

- **Stabilitatea tensiunii:** Menținerea tensiunii de ieșire la 5.0V (+/- 0.1V) folosind sursa Step-Down MP1584EN, indiferent de fluctuațiile alternatorului mașinii (care urcă până la 14.4V).
- **Latența comunicației:** Actualizarea datelor pe ecranul LCD și a sistemului Shift-Light în mai puțin de 150ms de la variația fizică a turației motorului.
- **Rata de succes a pachetelor:** Procesarea traficului fără pierderi (0% pachete ignorate) și fără

blocarea magistralei I2C sau SPI la turații mari.

Planificarea activităților (Grafic Gantt):

- **Săptămâna 1:** Cercetare protocoale (ISO 15765-4), desenare scheme bloc/electrice și achiziție componente.
- **Săptămâna 2:** Interconectare magistrală I2C (LCD) și SPI (MCP2515) pe breadboard. Scanare adrese I2C.
- **Săptămâna 3:** Implementarea logicii de polling, setarea întreruperilor hardware (INT0) și decodarea pachetelor OBD-II în C++.
- **Săptămâna 4:** Integrarea sursei Step-Down, calibrarea tensiunilor și testarea practică pe autovehicul (Ford Fiesta). Implementarea "CAN Sniffing-ului" pentru ID 0x201.
- **Săptămâna 5:** Testare finală de stabilitate termică, wire-management, urcare cod pe repository-ul Git și redactarea documentației Wiki.

Descriere generală

Arhitectura sistemului este modulară, fiecare protocol (CAN, SPI, I2C, UART) având un rol bine definit:

- **Sursa de alimentare auto (OBD-II Pin 16):** Furnizează aproximativ 12-14.4V (bateria auto/alternator).
- **Modul Step-Down DC-DC (MP1584EN):** Preia voltajul instabil al mașinii și îl coboară eficient, prin comutație (fără disipare termică excesivă), la un nivel logic stabil de 5.0V necesar microcontrolerului și ecranului.
- **Transceiver & Controller CAN (MCP2515 + TJA1050):** Partea de transceiver (TJA1050) transformă semnalele diferențiale (CAN_H și CAN_L) în biți RX/TX. Partea de controller (MCP2515) stochează pachetele în buffere și anunță microcontrolerul printr-un pin de întrerupere hardware (INT).
- **Unitatea de procesare (ATmega328P Xplained Mini):** Comunică prin SPI cu modulul CAN pentru a trimite cereri (PID-uri) și a citi buffere. Rulează un timer non-blocant, decodează octeții conform standardului ISO 15765-4 și calculează turația.
- **Afișajul (LCD 1602 + PCF8574):** Primește comenzi prin protocolul I2C (SDA/SCL) de la microcontroler pentru a afișa textul aferent.
- **Modul Shift Light:** Un grup de 3 LED-uri (Verde, Galben, Roșu) acționate pe pinii GPIO ai ATmega328P.

Hardware Design

Listă de piese:

- Placă de dezvoltare ATmega328P Xplained Mini
- Modul CAN-Bus (Controller MCP2515 + Transceiver TJA1050) (Cristal 8MHz)
- Sursă Coborâtoare (Step-Down Buck Converter) MP1584EN
- Ecran LCD 1602 cu modul I2C integrat
- Modul semafor (sau 3x LED-uri cu rezistențe de 330Ω)
- Cablu / Mufă tată OBD-II pigtail

- Breadboard și fire conexiune Dupont

Mapare Pini (Conexiuni logice):

- **SPI (MCP2515):** CS → PB2 (Pin 10), MOSI → PB3 (Pin 11), MISO → PB4 (Pin 12), SCK → PB5 (Pin 13)
- **Interrupt:** INT → PD2 (Pin 2) - Configurat pe declanșare FALLING.
- **I2C (LCD):** SDA → PC4, SCL → PC5
- **GPIO (LED-uri):** Verde → PD3 (Pin 3), Galben → PD4 (Pin 4), Roșu → PD5 (Pin 5)
- **Alimentare:** OBD-II Pin 16 & Pin 4 → IN (+/-) Step-Down → OUT (+/-) ajustat pe multimetru la 5.0V → Breadboard Power Rails.

Software Design

Descrierea codului aplicației (firmware):

- **Mediu de dezvoltare:** Proiectul a fost dezvoltat în Visual Studio Code folosind extensia PlatformIO, framework-ul Arduino și toolchain-ul avr-gcc.
- **Librării 3rd-party:**
  - `mcp_can` (de Cory J. Fowler) - Manipularea registrelor interne ale MCP2515 via SPI.
  - `LiquidCrystal_I2C` - Controlul afișajului.
- **Profilarea codului:** Evaluarea performanței codului s-a realizat urmărind evitarea blocajelor procesorului. Datorită utilizării logicii asincrone cu funcția `millis()`, procesorul (CPU) are un Idle Time ridicat, intervenind doar pentru a drena bufferul SPI când pinul de întrerupere pică pe LOW. Nu există funcții `delay()` în bucla principală care să monopolizeze microcontrolerul.
- **Algoritmi implementați:**
  - **Cod non-blocant:** S-a utilizat Timer0 (`millis()`) pentru a interoga CAN-ul la intervale precise de 100ms.
  - * if (millis() - lastRequestTime >= 100) {
    // Cerem RPM (PID 0x0C) la fiecare 100ms
    unsigned char st[8] = {0x02, 0x01, 0x0C, 0, 0, 0, 0, 0};
    CAN.sendMsgBuf(0x7DF, 0, 8, st);
    lastRequestTime = millis();
    }

  - **Interrupt Overrun Handling:** Rutina de tratare folosește un sistem în cascadă `while (CAN_MSGAVAIL == CAN.checkReceive())` care citește simultan toate mesajele adunate pentru a debloca microcontrolerul în cazul unui trafic auto intens.
  - * while (CAN_MSGAVAIL == CAN.checkReceive()) {
    long unsigned id;
    unsigned char len = 0;
    unsigned char buf[8];

    CAN.readMsgBuf(&id, &len, buf); // Golim bufferul repetat
    processCanMessage(id, buf);
    }

  - **Decodare dual**ă: Software-ul identifică răspunsurile standard la interogări OBD-II (ID 0x7E8) și

funcționează simultan ca un sniffer pentru a prinde direct cadrele transmise broadcast de mașină (ID 0x201).

Rezultate Obţinute

Sistemul a fost testat fizic pe un autoturism Ford Fiesta. A reușit negocierea cu ECU la viteza de 500kbps în modul MCP_NORMAL. Datele au fost citite și afișate fără latențe vizibile. S-a observat sensibilitatea ecranelor LCD la fluctuații de tensiune, problema afișării neclare fiind corectată prin calibrarea precisă a sursei Step-Down cu potențiometru la pragul de 5.0V sub sarcină. Shift-light-ul răspunde instantaneu la turațiile motorului.

Concluzii

Proiectul a dovedit cu succes că accesarea rețelelor CAN din vehicule este posibilă folosind echipamente hardware low-cost, atât timp cât protocolul și zgomotul de pe magistrală sunt corect filtrate software prin buffere și întreruperi. S-a demonstrat importanța stabilizării tensiunii în domeniul automotive și necesitatea scrierii de cod non-blocant pentru sisteme de tip “real-time”.

Download

- Repository GitHub (Istoric Commit-uri)
- Arhivă Cod Sursă (.zip)

Jurnal

- **Etapa 1:** Configurare mediu PlatformIO și depanare port USB Mac.
- **Etapa 2:** Interconectare I2C (Ecran LCD) și depanare scanner de adrese.
- **Etapa 3:** Interconectare SPI, setare module și testare cod de bază OBD-II cu LED-uri pe GPIO.
- **Etapa 4:** Testarea pe vehicul real. Rezolvare eroare de blocare a programului datorită întreruperilor suprapuse (citire buffer continuă).
- **Etapa 5:** Integrare sursă alimentare standalone (MP1584EN) de la 12V la 5V, rezolvare probleme

de contrast pe ecran.

Bibliografie/Resurse

Resurse Software:

- Documentație PlatformIO: https://docs.platformio.org/
- Librăria mcp_can (Cory J Fowler): https://github.com/coryjfowler/MCP_CAN_lib
- Wikipedia: OBD-II PIDs

Resurse Hardware:

- Datasheet ATmega328P (Microchip)
- Datasheet MCP2515 Stand-Alone CAN Controller
- Datasheet MP1584EN Step-Down Converter

Export to PDF