UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Calculatoare Încorporate

PROIECT RSI

Student:
Teodor-Constantin IURAȘCU
Alexandru-Robert BALAN
Marina-Dumitrița HRIȚCU

Iași, 2026

Cuprins

Capitolul 1. Introducere

1.1. Descriere

Acest proiect implementează un sistem de control distribuit utilizând protocolul de comunicație industrială **CAN (Controller Area Network)**. Obiectivul principal este transferul de date de la nodurile de intrare (senzori) către nodurile de execuție (actuatori), sub supravegherea unui nod dedicat de monitorizare. Întregul sistem este construit pe platforma **NXP FRDM-MCXN947**.

1.1.1. Arhitectura Sistemului și Distribuția Echipelor

Sistemul este compus din 5 noduri, fiecare fiind gestionat de o echipă specifică:

1. **Potențiometru (Input):** achiziția analogică (ADC) și transmiterea poziției către rețea pentru controlul unghiular sau de intensitate.
2. **Senzor de Temperatura/Umiditatem (Input):** monitorizarea condițiilor de mediu și partajarea acestor parametri pe magistrală.
3. **Monitor:** interceptarea pachetelor. Acesta nu modifică datele, ci analizează traficul pentru validarea comunicării între celelalte noduri.
4. **Servomotor (Output):** Interpretează datele primite (de la potențiometru si senzor) și generează semnalul PWM necesar poziționării mecanice.
5. **LED RGB (Output):** Oferă feedback vizual pe baza datelor de la senzori.

1.1.2. Schema Logică a Transmisiei

Figura 1.1 ilustrează modul în care datele circulă de la cele două surse de intrare către restul componentelor sistemului.

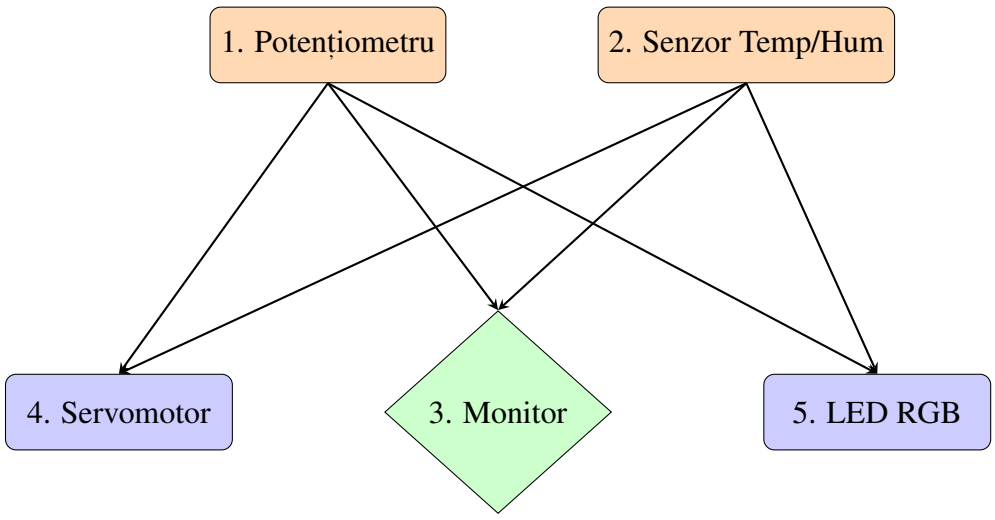

Figura 1.1. Schema de distribuție

1.2. Placa de dezvoltare

Placa de dezvoltare **FRDM-MCXN947** este o platformă compactă și scalabilă, concepută pentru evaluarea microcontrolerelor din seria **MCX N94x**. Aceasta este ideală pentru aplicații de tip Edge Computing, automatizări industriale și Internet of Things (IoT).

1.3. Arhitectură Microcontroler (MCU)

Nucleul plăcii este reprezentat de MCU-ul **MCX N947**, care dispune de următoarele caracteristici:

- **Dual-Core:** 2x Arm® Cortex®-M33 rulând la frecvențe de până la 150 MHz.
- **Accelerator AI/ML:** Include unitatea de procesare neurală **eIQ® Neutron NPU** pentru sarcini de inteligență artificială la nivel hardware.
- **Securitate:** Subsistem de securitate **EdgeLock® Secure Enclave** (Core Profile).
- **Memorie:** Până la 2 MB de memorie Flash și 512 KB de memorie SRAM cu paritate/ECC.

1.4. Caracteristici Hardware și Periferice

Placa FRDM-MCXN947 integrează o gamă largă de componente pentru prototipare rapidă:

- **Conectivitate Ethernet:** Port RJ45 integrat pentru aplicații de rețea.
- **Interfețe USB:** Port USB Type-C High-Speed pentru date și alimentare.
- **Expansiune:** Compatibilitate cu shield-uri **Arduino Uno R3** și conectori **MikroBus™** pentru senzori și actuatori.
- **Debug Integrat:** On-board **MCU-Link** debugger bazat pe CMSIS-DAP, eliminând necesitatea unui programator extern.
- **Interfață Audio:** Suport pentru ieșiri audio de tip MQS (Medium Quality Sound).

1.5. Ecosistem de Dezvoltare

Suportul software este asigurat prin **MCUXpresso Developer Experience**, oferind:

1. **IDE:** MCUXpresso IDE, VS Code (cu extensii NXP) sau IAR/Keil.
2. **SDK:** Drivere periferice optimizate, stive de protocoale și exemple de cod (RTC, PWM, Ethernet).
3. **Configurare:** Instrumente grafice pentru rutarea pinilor și configurarea ceasurilor (Pins, Clocks, Peripherals Tool).

1.6. Implementare Tehnică

Pentru realizarea comunicării, s-au utilizat următoarele resurse hardware ale microcontrolerului MCX N947:

- **FlexCAN:** Configurarea baud-rate-ului și a filtrelor de acceptare (Hardware Acceptance Filters).
- **GPIO & PWM:** Pentru controlul direct al servomotorului și al LED-ului.
- **ADC:** Pentru conversia semnalului analogic de la potențiometru.

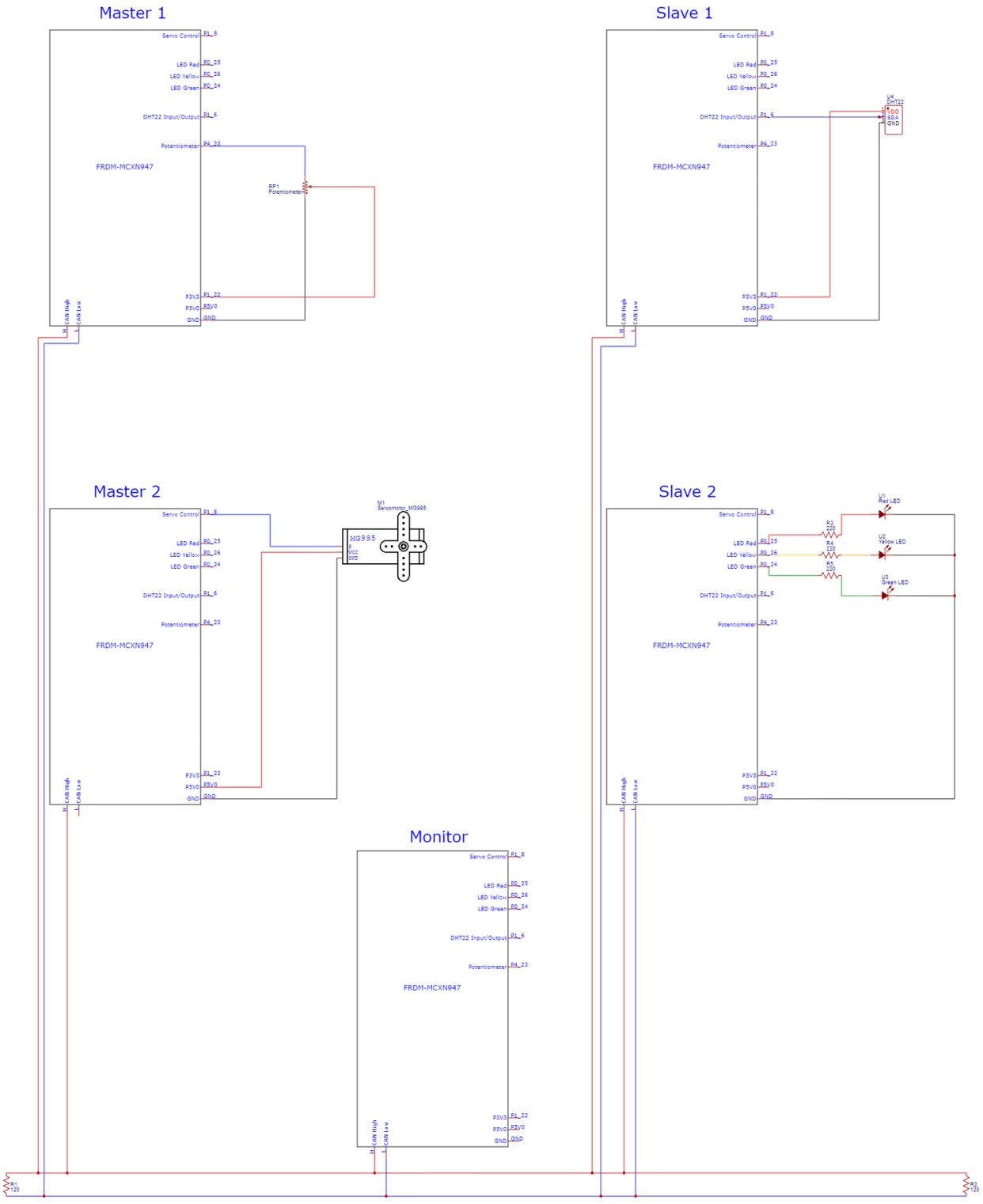

Figura 1.2. Schemă de conectare generală

Capitolul 2. Implementarea Controlului pentru Servomotor

2.1. Descriere Generală

În cadrul acestui proiect, pentru nodul 4 s-a utilizat ca output **servomotorul MG955**.

Acest modul utilizează placa de dezvoltare **FRDM-MCXN947** pentru a controla un servomotor prin intermediul protocolului **CAN FD**. Rolul principal al sistemului este de a recepționa date de poziționare și viteză de pe magistrala CAN și de a le converti în semnale PWM (Pulse Width Modulation) de 50Hz.

Sistemul implementează o mișcare de tip *asincron*, permițând servomotorului să execute tranziții line între unghiuri, evitând mișcările bruște.

2.2. Descriere Hardware și Conectică

Servomotorul MG995 este un dispozitiv de mare putere cu angrenaje metalice, necesitând o alimentare stabilă de 5V.

- **Tensiune de operare:** 4.8V până la 7.2V (alimentat extern pentru a proteja regulatorul plăcii).
- **Tip semnal:** PWM Analogic (frecvență de 50 Hz).
- **Interval de mișcare:** 180 grade.

Conexiunile principale sunt detaliate în Figura 2.1:

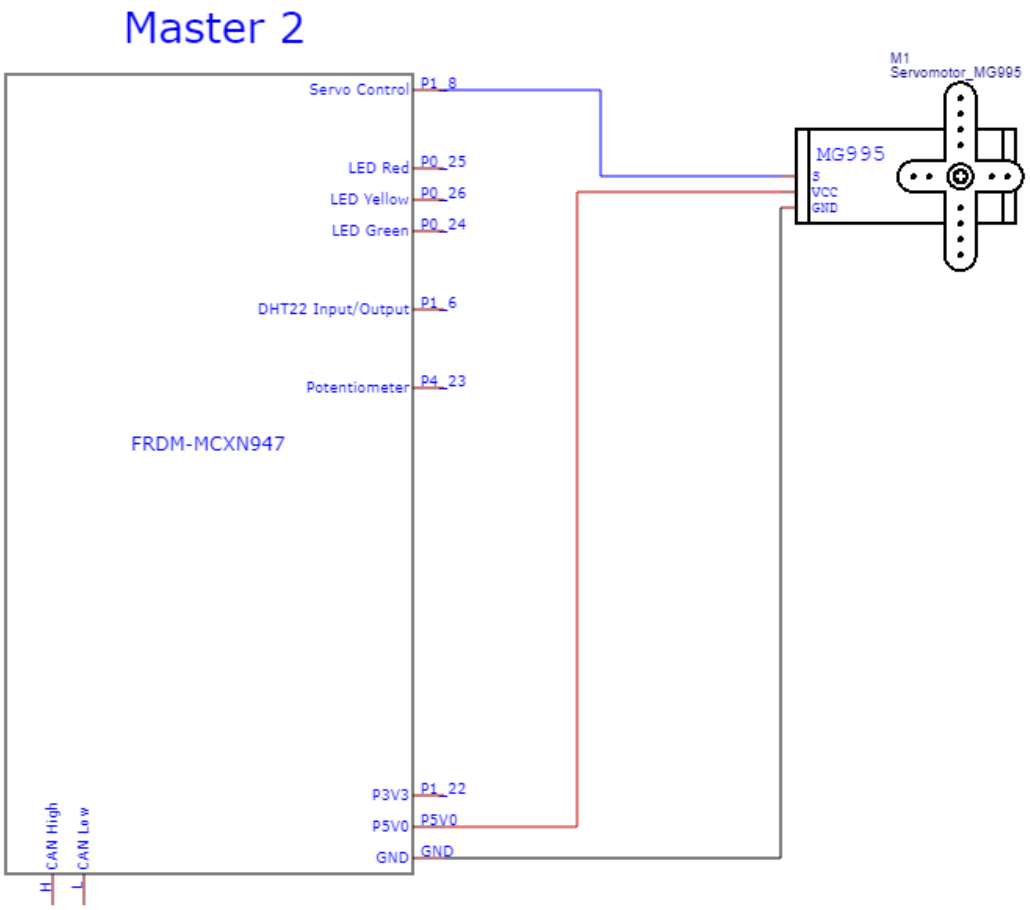

Figura 2.1

Detalierea conexiunilor din schemă:

- **VCC:** Conectat la pinul **P5V0** pentru alimentare.
- **GND:** Conectat la borna comună de masă a sistemului.
- **Semnal:** Conectat la pinul de control definit prin SCTimer (**P0_29**).
- **Magistrala CAN:** Pinii **CAN High** și **CAN Low** asigură recepția pachetelor de date FD.

2.3. Configurarea Bitilor și Parametrii PWM

Parametru	Valoare	Explicație
SERVO_PERIOD_US	20000	Perioada semnalului (50Hz)
MIN_PULSE_US	500	Puls pentru 0°
MAX_PULSE_US	2500	Puls pentru 180°
SCTimer Clock	Bus Clock	Frecvența de referință pentru PWM

2.4. Analiză Software

2.4.1. Logica de mișcare

Nucleul sistemului este reprezentat de rutina de întrerupere `SCT0_IRQHandler`. Aceasta rulează la fiecare 20ms și gestionează incrementarea:

```
if (g_current_angle != g_target_angle) {
    g_tick_counter++;
    if (g_tick_counter >= g_speed_delay_ticks) {
        double step = (g_target_angle > g_current_angle) ? 1 : -1;
        g_current_angle += step;
        // Update PWM Hardware
        uint32_t pulse_us = 500U + ((uint32_t)g_current_angle * 2000U /
   180U);
        SCTIMER_UpdatePwmDutycycle(BOARD_SCTIMER, BOARD_SCT_OUT, (pulse_us
    * 100U) / 20000U, sctimer_event_number);
        g_tick_counter = 0;
    }
}
```

2.4.2. Calculul parametrilor și maparea semnalului

Pentru a converti unghiul dorit (α), exprimat în grade, în lățimea pulsului corespunzătoare (t_{pulse}) exprimată în microsecunde, se utilizează o funcția:

$$t_{pulse} = MIN\_PULSE\_US + \left(\alpha \cdot \frac{MAX\_PULSE\_US - MIN\_PULSE\_US}{180} \right) \tag{2.1}$$

Factorul de umplere (Duty Cycle) necesar pentru funcția `SCTIMER_UpdatePwmDutycycle` este ulterior calculat ca raport între t_{pulse} și perioada totală a semnalului:

$$DutyCycle(\%) = \frac{t_{pulse} \cdot 100}{SERVO\_PERIOD\_US} \tag{2.2}$$

2.4.3. Configurarea SCTimer

- **Modul unified (32-bit):** Prin activarea flag-ului `enableCounterUnify`, cele două timere interne de 16 biți ale SCT0 sunt concatenate pentru a forma un singur numărător de 32 de biți. Această configurație permite o rezoluție mult mai fină a semnalului PWM.
- **Sursa de ceas:** Perifericul este sincronizat cu `Bus Clock`, asigurând o bază de timp stabilă, independentă de variațiile de sarcină ale nucleului procesorului.

2.5. Mecanismul de Recepție și Decodificare CAN FD

Sistemul utilizează perifericul **FlexCAN** configurat pentru suport **CAN FD** (Flexible Data-rate), permițând o rată de transfer de 2 Mbps pentru zona de date. Recepția mesajelor este implementată printr-o metodă de interogare (polling) a indicatorilor de stare ai bufferelor de mesaje (Message Buffers).

2.5.1. Procesul de recepție la nivel hardware

Fiecare mesaj CAN este recepționat în bufferul `RX_MESSAGE_BUFFER_NUM` (configurat ca MB9). Procesul urmează pașii următori:

1. **Verificarea Flag-ului:** monitorizarea flag-ului de stare se face prin `FLEXCAN_GetMbStatusFlags`. Acest flag este activat automat de hardware atunci când un mesaj valid, care trece de filtrul de mască (`0x7FF`), este stocat în buffer.
2. **Citirea Datelor:** Odată confirmată prezența unui mesaj, se utilizează funcția `FLEXCAN_ReadFDRxMb` pentru a transfera conținutul din regiștrii perifericului într-o structură de tip `flexcan_fd_frame_t`.
3. **Resetarea Stării:** flag-ul este șters manual pentru a permite perifericului să semnalizeze următoarea recepție.

2.5.2. Protocolul de comunicare și extragerea datelor

Mesajul recepționat conține informații compactate în cuvântul de date `dataWord[0]`. Pentru a minimiza traficul pe bus, s-a utilizat un protocol personalizat unde ID-ul expeditorului și valoarea propriu-zisă sunt transmise în același cadru de date prin tehnici de *bit-shifting*.

Algoritmul de decodificare este următorul:

```
uint32_t canData = rxFrame.dataWord[0];
int data_read = (canData >> 24) & 255;  // Extragerea valorii utile (8
    biti superiori)
int sender_id = (canData >> 16) & 3;    // Identificarea sursei (2 biti)
```

2.5.3. Interpretarea surselor de date

În funcție de valoarea extrasă pentru `sender_id`, sistemul ia decizii diferite, demonstrând flexibilitatea nodului de output:

- **Sender ID 1 (Potențiometru):** Valoarea `data_read` este interpretată ca o poziție absolută. Aceasta este mapată direct către variabila `angle`, provocând o mișcare a servomotorului către noua destinație.
- **Sender ID 2 (Senzor):** Valoarea este utilizată pentru a calcula dinamica mișcării. Formula `100 * (100-data_read) / 60` transformă input-ul senzorului într-o întârziere temporală (`speed_delay_ticks`), ajustând astfel viteza cu care servomotorul parcurge distanța până la unghiul țintă.

Concluzii

Proiectul a demonstrat succesul implementării unui sistem de control distribuit utilizând protocolul **CAN FD** pe platforma **NXP FRDM-MCXN947**. Integrarea nodului de execuție pentru servomotorul **MG995** a evidențiat capacitatea microcontrolerului de a gestiona sarcini de timp real prin utilizarea perifericului **SCTimer** în mod *unified* pe 32 de biți, asigurând o precizie ridicată a semnalului PWM și o mișcare asincronă fluidă.

Eficiența sistemului este susținută de mecanismul robust de recepție prin interogare (polling) și de algoritmul de decodificare optimizat, care permite procesarea diferențiată a datelor în funcție de sursa acestora (potențiometru sau senzor). Rezultatul final este un sistem scalabil, capabil să asigure o interacțiune precisă între nodurile de intrare și actuatori, respectând cerințele de siguranță hardware și integritate a datelor.

Bibliografie

[1] https://docs.nxp.com/bundle/UM12018/page/topics/Evk_overview.html, NXP Semiconductors, "FRDM-MCXN947 Development Board User Guide"

[2] NXP Semiconductors, MCX N Series Microcontrollers Reference Manual, rev. 1, 2024, oferind detalii despre perifericele LPADC, FlexCAN si SCTimer

[3] International Organization for Standardization (ISO), ISO 11898-1:2015 Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling, utilizat pentru standardizarea protocolului CAN-FD.

[4] https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf, "MG995 High Speed Digital Servo Datasheet"