Ministerul Educației și Cercetării
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Raport

Curs: Internetul lucrurilor

Tema: Comunicare cu periferii - SW

A elaborat: st.gr.SI-211 Chirița Stanislav
A verificat: Asist. Univ.Astafi Valentina

Chișinău 2024

Scopul lucrării

Realizarea unei aplicații care implementează comunicarea între două dispozitive utilizând protocolul SoftwareSerial pentru schimbul de date și un protocol logic personalizat pentru gestionarea interacțiunilor.

Mersul lucrării

1. Protocol logic de comunicare

Protocolul logic implementat include următoarele câmpuri obligatorii:

- Indicator de start al pachetului: <
- Indicator de sfârșit al pachetului: >
- Contorizare pachete: Incrementat la fiecare transmisie.
- ID-ul emițătorului: Identifică dispozitivul care trimite pachetul.
- ID-ul receptorului: Identifică dispozitivul care primește pachetul.
- Tipul pachetului: Cod numeric pentru tipul de comandă.
- Datele pachetului (payload): Valoarea transmisă (ex. datele primite de la un senzor).
- Sumă de control: Calculată ca suma tuturor valorilor numerice din pachet pentru verificarea integrității.

2. Funcționalități implementate

1. Cererea de date de la senzor:
    - Dispozitivul emițător colectează date de la un senzor digital și le transmite conform protocolului SoftwareSerial.
    - Dispozitivul receptor validează pachetul și afișează datele primite.
2. Comandă suplimentară:
    - Dispozitivul receptor poate trimite o comandă pentru activarea unui LED conectat la emițător.
    - LED-ul este pornit pentru 1 secundă, apoi oprit.

3. Implementarea codului

Cod pentru Emițător:

- Implementarea include colectarea datelor de la senzor, calculul checksum-ului și transmiterea pachetului conform protocolului logic.

Cod pentru Receptor:

- Implementarea verifică validitatea pachetului prin analiza indicatorilor de start/sfârșit și checksum.
- Procesarea pachetului include afișarea datelor primite sau activarea LED-ului, conform comenzii primite.

4. Conexiuni hardware

- SoftwareSerial:
    - RX (Receptor) → TX (Emițător).
    - TX (Emițător) → RX (Receptor).
    - GND comun între dispozitive.
- Senzor:
    - Pin de date conectat la un pin digital al emițătorului.
    - Alimentare: VCC și GND.
- LED:
    - Anod → Pin digital al emițătorului printr-un rezistor de 220Ω.
    - Catod → GND.

5. Rezultate obținute

- Sistemul transmite cu succes datele de la senzor folosind protocolul SoftwareSerial.
- Dispozitivul receptor validează pachetele și afișează informațiile primite.
- Comanda suplimentară (activarea LED-ului) este procesată corect, demonstrând flexibilitatea sistemului.
- Verificările pentru pachete invalide funcționează conform așteptărilor.

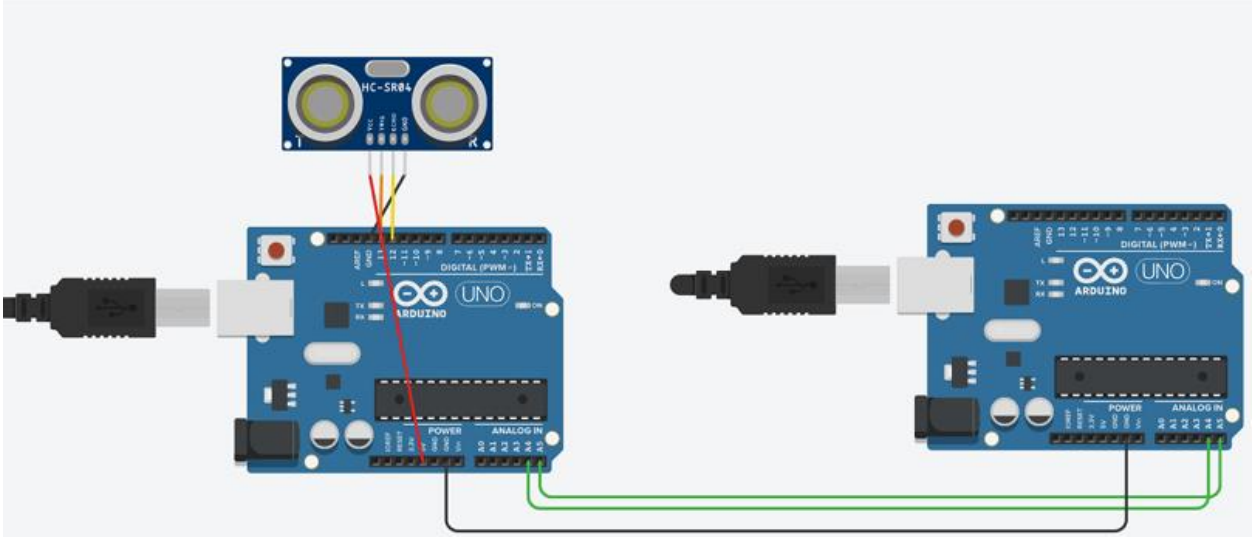

Figura 1 – Circuitul

Lucrarea propusă demonstrează implementarea cu succes a unui sistem de comunicație bidirecțională între două dispozitive utilizând protocolul SoftwareSerial și un protocol logic personalizat. Sistemul proiectat asigură transferul sigur și eficient al datelor, validând integritatea acestora prin checksum și alți indicatori.

Funcționalitățile testate, precum transmiterea datelor de la un senzor și comanda pentru activarea unui LED, evidențiază flexibilitatea și fiabilitatea sistemului, fiind în conformitate cu cerințele proiectului. Conexiunile hardware au fost realizate corect, asigurând interoperabilitatea dispozitivelor.

Rezultatele obținute confirmă aplicabilitatea protocolului logic personalizat într-un mediu practic, oferind o bază solidă pentru viitoare extinderi, cum ar fi integrarea mai multor dispozitive sau adăugarea unor tipuri suplimentare de comenzi. Sistemul demonstrează o abordare practică și eficientă în utilizarea resurselor hardware și software ale dispozitivelor embedded.

```
// Cod pentru MCU1 (Emitator)
#include <Wire.h>
#include <SoftwareSerial.h>

const int trigPin = 13;
const int echoPin = 12;
const int ledPin = 9;
const int emitterID = 1;
const int receiverID = 2;
const int packetSize = 15;
SoftwareSerial Serial1(10, 11); // RX, TX pentru interfata seriala suplimentara

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
```

```
    Serial.begin(9600); // Interfata seriala principala
    Serial1.begin(9600); // Interfata seriala secundara

    Wire.begin(emitterID); // Initializare I2C ca emitator
    Serial.println("MCU1 initializat.");
}

void loop() {
    long duration, distance;

    // Generare impuls ultrasonic
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.034 / 2; // Distanta in cm

    // Trimiterea datelor via I2C
    Wire.beginTransmission(receiverID);
    Wire.write("<"); // Indicator start
    Wire.write(emitterID);
    Wire.write(receiverID);
    Wire.write(0x10); // Comanda pentru senzor
    Wire.write((byte*)&distance, sizeof(distance));
    Wire.write(calculateChecksum(distance));
    Wire.write(">"); // Indicator sfarsit
    Wire.endTransmission();

    delay(500); // Pauza intre transmisii
}

byte calculateChecksum(long data) {
    byte checksum = emitterID + receiverID + 0x10;
    for (int i = 0; i < sizeof(data); i++) {
        checksum += ((byte*)&data)[i];
    }
    return checksum;
}

// Cod pentru MCU2 (Receptor)
#include <Wire.h>

const int receiverID = 2;
const int packetSize = 15;
```

```
void setup() {
   Wire.begin(receiverID); // Initializare I2C ca receptor
   Wire.onReceive(receiveEvent); // Functie apelata cand se primesc date
   Serial.begin(9600);
   Serial.println("MCU2 initializat.");
}

void loop() {
   delay(100); // Asteptare evenimente
}

void receiveEvent(int howMany) {
   char buffer[packetSize];
   int index = 0;

   while (Wire.available()) {
      buffer[index++] = Wire.read();
   }

   if (validatePacket(buffer, index)) {
      processPacket(buffer, index);
   } else {
      Serial.println("Pachet invalid.");
   }
}

bool validatePacket(char* packet, int size) {
   if (packet[0] != '<' || packet[size - 1] != '>') {
      return false;
   }
   byte checksum = 0;
   for (int i = 1; i < size - 2; i++) {
      checksum += packet[i];
   }
   return checksum == packet[size - 2];
}

void processPacket(char* packet, int size) {
   byte cmd = packet[3];

   if (cmd == 0x10) { // Comanda pentru senzor
      long distance;
      memcpy(&distance, &packet[4], sizeof(distance));
      Serial.print("Distanta primita: ");
      Serial.print(distance);
      Serial.println(" cm");
   } else if (cmd == 0x20) { // Comanda suplimentara: control LED
      digitalWrite(ledPin, HIGH);
      delay(1000);
```

```
        digitalWrite(ledPin, LOW);
        Serial.println("LED activat.");
    }
}
```