Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică şi Microelectronică
Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator nr. 7.2
La disciplina “Internetul Lucrurilor”
Tema: Comunicare intre dispozitive - Protocol SW Sierial

A efectuat: st. gr. SI-211 Adrian Chihai
A verificat: Valentina Astafi

Chișinău – 2024

1 Definirea problemei

Sa se realizeze o aplicatie ce va implementa comunicatiile intre echipamente dupa cum urmeaza:

1. Protocol logic de comunicare - cererea de date prin interfata serial, in format text respectand un protocol de comunicare care va avea campurile:

Protocol Frame

STX	P nr	SRC	DST	P_ID	CMD	Payload (4 bytes)	SC	ETX

indicator de start pachet

indicator de sfarsit

contorizare pachete

ID emitator

ID receptor

tipul pachetului

<alte campuri optional>

date pachet - Payload

suma de control - suma tuturor valorilor numerice din pachet

cererile venite din interfata seriala vor fi verificate dupa patern, si in caz de pachet valid se va intereta comanda. se va raspunde cu un pachet conform aceluias protocol.
implementare la o comanda obligatorie pentru implementare este cererea de date de la sensorul digital implementat in lab precedent.
sa si implementezi inca o camanda la alegere, pentru diversitate.

2 Descrierea programului

- **setup()**

Inițializează comunicațiile seriale. Configurarea comunicațiilor seriale pentru portul standard și pentru SoftwareSerial cu pinii 10 și 11 pentru RX și TX, respectiv. SoftwareSerial rulează la 4800 bps.

- **loop()**

Coordonează trimiterea solicitărilor și primirea răspunsurilor. Trimite o solicitare periodică la fiecare 2 secunde, citește răspunsul dacă este disponibil, apoi curăță buffer-ul serial pentru a evita date vechi.

- **sendRequest()**

Construiește și trimite o solicitare de date. Trimite un pachet care include un început de text, ID-uri pentru prioritate, sursă și destinație, un comand ID specific, o sarcină ("payload") de exemplu, și un checksum calculat. Termină cu un caracter de sfârșit de text.

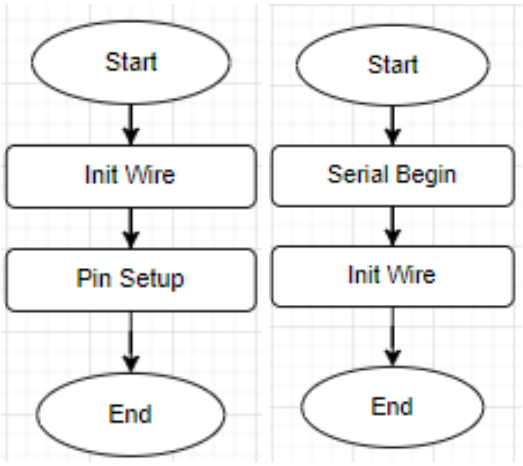
- **handleResponse(String response)**

Procesează răspunsul primit. Extrage și procesează sarcina din răspunsul primit, determină distanța din sarcina de date și afișează aceasta pe portul serial.

- **calculateChecksum(int priority, int source, int dest, int cmd, String payload)**

Calculează un checksum simplu pentru verificarea integrității datelor. Sumă numerică a valorilor de prioritate, sursă, destinație, comandă și convertirea sarcinii la un întreg, oferind un mecanism simplu pentru verificarea integrității datelor transmise.

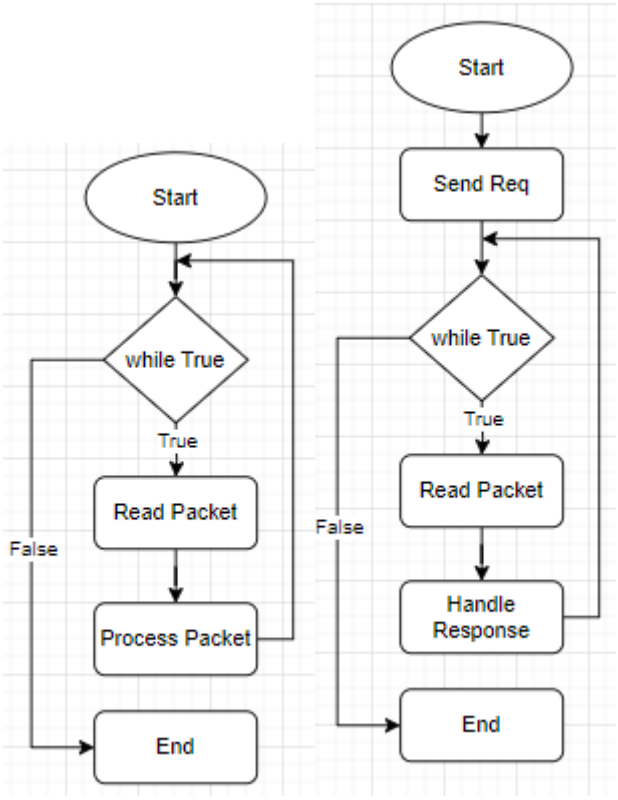
Figura 2.1 reprezintă funcția setup din program. După ce această funcție se termină, se sare la funcția de loop.



a) b)

Fig. 2.1. Funcția *setup*: a pentru MCU1, b pentru MCU2

Figura 2.2. logica funcțiilor loop



a) b)

Fig. 2.2. *loop (a) for MCU1, loop(b) for MCU2* functions

3. Circuitul elaborat

În figura 2.3. este realizat circuitul conform cerințelor

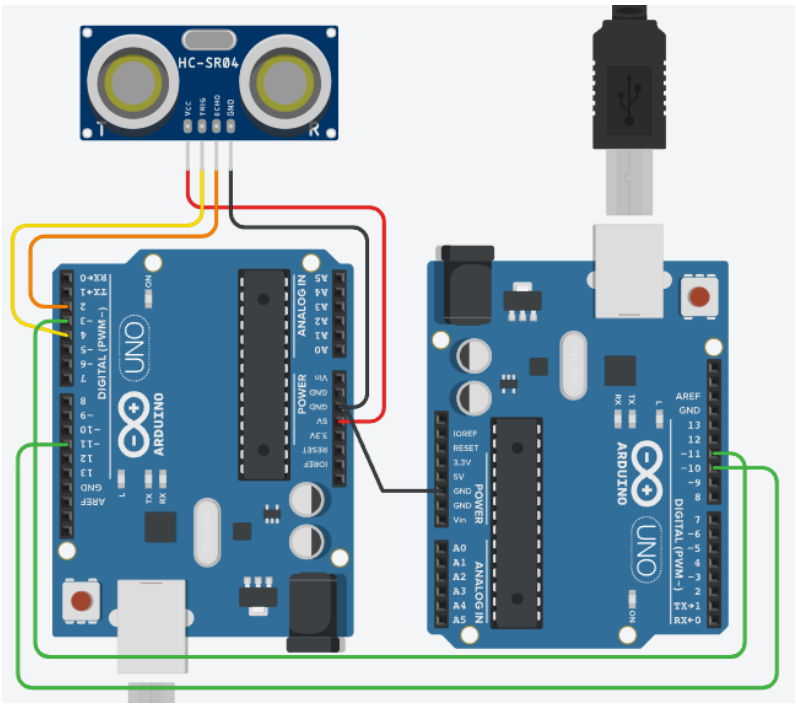


Fig. 2.3. Circuitul asamblat

În figura 2.4. este circuitul pornit și putem observa cum senzorul capturează distanța

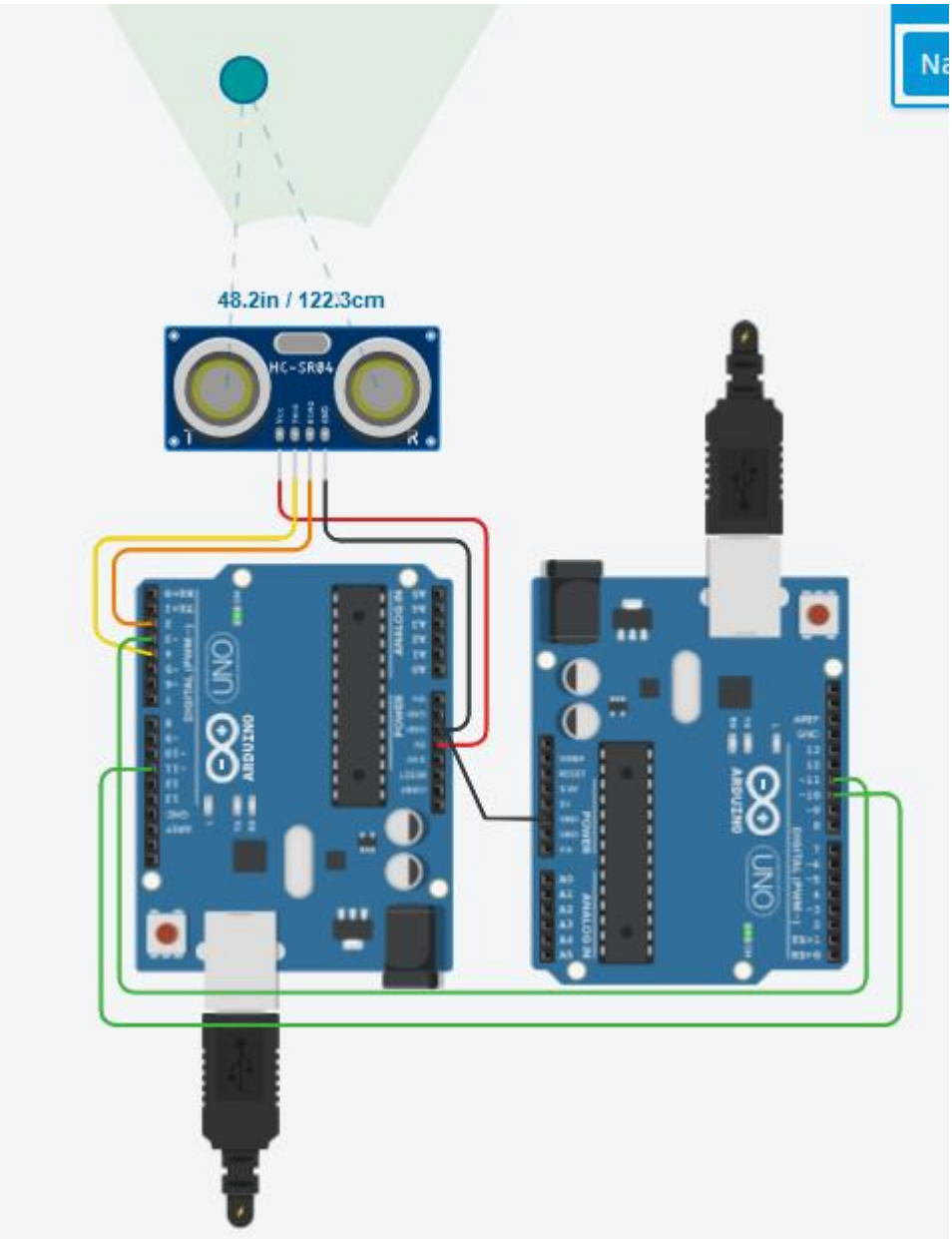


Fig. 2.4. Circuitul pornit

Concluzie

În această lucrare de laborator ne am implementat și utilizat un protocol serial simplu pentru comunicarea între dispozitive. Realizând acest exercițiu practic am înțeles fundamentele comunicării în rețele și necesitatea unui protocol bine structurat pentru a asigura transferul eficient și corect al datelor.

ANEXA

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

const char STX = 0x02; // Start of Text
const char ETX = 0x03; // End of Text
const int PrNr = 1;    // Priority
const int SRC = 1;     // Source ID
const int DST = 2;     // Destination ID
const int CMD = 1;     // Command ID for requesting distance

void setup() {
  Serial.begin(9600);
  mySerial.begin(4800);
}

void loop() {
  sendRequest();
  delay(1000);
  if (mySerial.available()) {
    String response = mySerial.readStringUntil(ETX); // Read the packet data
    if (response.startsWith(String(STX))) {
      handleResponse(response);
    }
  }
  while(mySerial.available()) mySerial.read();
  delay(1000);
}

void sendRequest() {
  mySerial.write(STX);        // Start of packet
  mySerial.print(PrNr);       // Priority
  mySerial.print(SRC);        // Source ID
  mySerial.print(DST);        // Destination ID
  mySerial.print(" ");        // Separator
  mySerial.print(CMD);        // Command ID
  mySerial.print(" ");        // Separator
  mySerial.print("0000");     // Dummy Payload
  mySerial.print(" ");        // Separator
  mySerial.print(calculateChecksum(PrNr, SRC, DST, CMD, "0000")); // Checksum
  mySerial.write(ETX);        // End of packet
  Serial.println("Request sent");
}

void handleResponse(String response) {
  int startIndex = response.indexOf(' ') + 2; // Get start index of the payload
(after STX)
```

```
  int endIndex = response.lastIndexOf(' '); // Get end index of the payload (before
checksum)
  String payload = response.substring(startIndex, endIndex); // Extract payload
  payload.trim(); // Remove any whitespace
  int distance = payload.toInt(); // Convert payload to integer

  Serial.print("Distance received: ");
  Serial.print(distance);
  Serial.println(" cm");
}

int calculateChecksum(int priority, int source, int dest, int cmd, String payload)
{

  // Simple checksum calculation (you can improve this)
  return priority + source + dest + cmd + payload.toInt();
}
```

BIBLIOGRAPHY

1. WOKWI: *Arduino Simulator and Tutorials*. Arduino Examples, ⓒ 2019-2023 CodeMagic LTD [21.02.2024], Link: https://wokwi.com/projects