Ministerul Educaţiei, Culturii și Cercetării al Republicii

Moldova

Universitatea Tehnică a Moldovei

Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrare de laborator Nr. 7

**Disciplina:** Internetul lucrurilor

**Tema:** Comunicare

A efectuat: st.gr. TI-204, Agatiev Dumitru

A verificat : asis. univ. Litra Dinu

Chișinău 2023

**Sarcina lucrării**

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

1. Protocol fizic de comunicare - Comunicarea intre DOUA Microcontrollere prin interfata I2C

* MCU1 - implementeaza sensorul digital cu interfata I2C pentru sensorul ultrasonic HCS-04, unde se ececuta colectarea datelor de la interfata sensorului si se retransmite catre interfata I2C la detectarea unei cereri de citire a datelor.
* MCU2 - executa cererea prin interfata I2C catre sesorul digital ultrasonic (MCU+HCS-04)

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

* 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

comanda obligatorie pentru implementare este cererea de date de la sensorul digital implementat in p 1. sa si implementezi inca o camanda la alegere, pentru diversitate.

3. realizarea conexiunii si raportarii datelor achizitionate catre un MQTT server cum ar fi https://thingsboard.io sau echivalent

**Recomandare:**

- Sa utilizeze interfata seriala pentru rapoarte de functionare a automatelor;

- Reutilizati la maxim solutiile prezentate in laboratoarele precedente;

- Revizuiti resursele predate la curs.

1. **Schema**

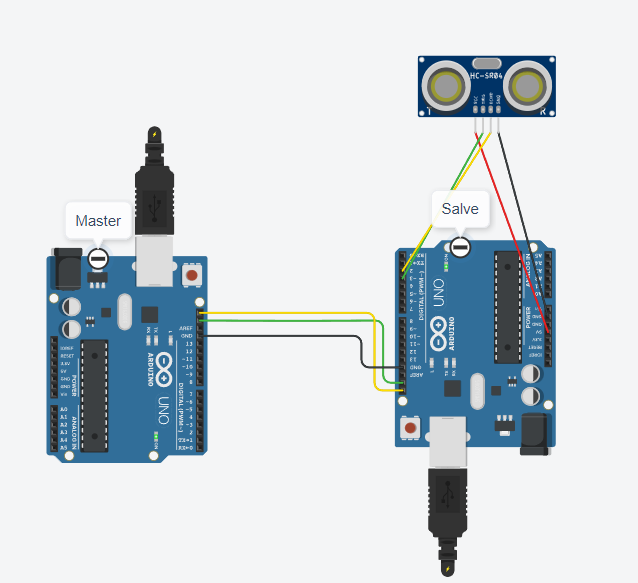


Figura 1.1 Schema circuitului

1. **Codul sursă**

#include <Wire.h>

#define SLAVE\_ADDRESS 0x05 // Adresa I2C pentru slave

const int trigPin = 3;    // Pinul trigger pentru senzorul ultrasonic

const int echoPin = 2;    // Pinul echo pentru senzorul ultrasonic

bool packetAccepted = false; // Flag pentru a verifica dacă pachetul a fost acceptat

// Inițializează pinii pentru senzorul ultrasonic

void UltrasonicInit(int trigPin, int echoPin)

{

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

}

// Citirea datelor de la senzorul ultrasonic

int UltrasonicRead(int trigPin, int echoPin){

  long duration;

  int distance;

  // Trimiterea unui impuls și citirea timpului de întoarcere

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  distance = duration \* 0.034 / 2; // Calculul distanței pe baza timpului de întoarcere

  return distance;

}

// Structura pentru pachetul de date I2C

struct Packet {

  char packetStart; // Caracterul de început pachet

  int packetCounter; // Contor pentru pachete

  int issuerID; // ID-ul emitentului

  int receiverID; // ID-ul receptorului

  int packageType; // Tipul pachetului

  int payload; // Datele pachetului - distanța măsurată

  int checksum; // Suma de control

  char packetEnd; // Caracterul de sfârșit pachet

};

Packet requestPacket; // Pachet pentru cererea primită

// Calculează suma de control pentru un pachet

int calculateChecksum(struct Packet& packet) {

  int sum = packet.packetCounter + packet.issuerID + packet.receiverID + packet.packageType + packet.payload;

  return sum;

}

// Creează un pachet de răspuns pe baza unui pachet de cerere și a citirii senzorului

Packet createResponsePacket(struct Packet& requestPacket, int sensorReading) {

  Packet responsePacket;

  responsePacket.packetStart = '<';

  responsePacket.packetCounter = requestPacket.packetCounter;

  responsePacket.issuerID = requestPacket.receiverID;

  responsePacket.receiverID = requestPacket.issuerID;

  responsePacket.packageType = requestPacket.packageType + 1;

  responsePacket.payload = sensorReading;

  responsePacket.checksum = calculateChecksum(responsePacket);

  responsePacket.packetEnd = '>';

  return responsePacket;

}

// Funcția de recepție pentru I2C, apelată când masterul trimite date la slave

void receiveEvent(int howMany) {

  Wire.readBytes((uint8\_t\*)&requestPacket, sizeof(requestPacket));

  int calculatedChecksum = calculateChecksum(requestPacket);

  // Verifică dacă pachetul este valid prin comparația sumei de control

  if (requestPacket.checksum == calculatedChecksum) {

    packetAccepted = true; // Pachetul este acceptat pentru răspuns

  } else {

    packetAccepted = false; // Pachetul este invalid

  }

}

// Funcția de răspuns pentru I2C, apelată când masterul cere date de la slave

void sendResponse() {

  if(packetAccepted){

    packetAccepted = false; // Resetarea flagului

    int distance = UltrasonicRead(trigPin, echoPin); // Citirea distanței

    Packet responsePacket = createResponsePacket(requestPacket, distance); // Crearea pachetului de răspuns

    Wire.write((uint8\_t\*)&responsePacket, sizeof(responsePacket)); // Trimiterea pachetului de răspuns

  }

}

// Configurația inițială a Arduino

void setup() {

  Serial.begin(9600); // Inițializarea comunicației seriale pentru debug

  UltrasonicInit(trigPin, echoPin); // Inițializarea senzorului ultrasonic

  Wire.begin(SLAVE\_ADDRESS); // Inițializarea comunicării I2C cu adresa slave

  Wire.onReceive(receiveEvent); // Setarea funcției de recepție pentru I2C

  Wire.onRequest(sendResponse); // Setarea funcției de răspuns pentru I2C

  Serial.println("I2C Slave Ready"); // Mesaj de început

}

void loop() {

  // Nici o logică suplimentară în loop deoarece comunicarea este gestionată prin întreruperile I2C

  delay(1000); // Delay pentru a nu supraîncărca bucla

}

#include <Wire.h>

#define SLAVE\_ADDRESS 0x05 // Adresa I2C a dispozitivului slave

// Structura pentru pachetul de date

struct Packet {

  char packetStart; // Caracterul de început pachet

  int packetCounter; // Contor pentru pachete

  int issuerID; // ID-ul emitentului

  int receiverID; // ID-ul receptorului

  int packageType; // Tipul pachetului

  int payload; // Datele pachetului - comanda pentru slave

  int checksum; // Suma de control

  char packetEnd; // Caracterul de sfârșit pachet

};

Packet responsePacket; // Pachet pentru răspunsul primit de la slave

// Calculează suma de control pentru un pachet

int calculateChecksum(struct Packet& packet) {

  int sum = packet.packetCounter + packet.issuerID + packet.receiverID + packet.packageType + packet.payload;

  return sum;

}

// Creează un pachet de cerere

Packet createRequestPacket(int counter, int issuerID, int receiverID, int packageType, int command) {

  Packet requestPacket;

  requestPacket.packetStart = '<';

  requestPacket.packetCounter = counter;

  requestPacket.issuerID = issuerID;

  requestPacket.receiverID = receiverID;

  requestPacket.packageType = packageType;

  requestPacket.payload = command;

  requestPacket.checksum = calculateChecksum(requestPacket);

  requestPacket.packetEnd = '>';

  return requestPacket;

}

// Trimite un pachet de cerere la slave

void sendRequest(struct Packet& requestPacket) {

  Serial.print("Sent ");

  Serial.println(requestPacket.payload);

  Wire.beginTransmission(SLAVE\_ADDRESS); // Începe transmisia I2C

  Wire.write((uint8\_t\*)&requestPacket, sizeof(requestPacket)); // Scrie pachetul în busul I2C

  Wire.endTransmission(); // Termină transmisia I2C

}

void setup() {

  Wire.begin(); // Începe comunicarea I2C ca master

  Serial.begin(9600); // Începe comunicația serială pentru debug

  Serial.println("I2C Master Ready"); // Mesaj de început

}

void loop() {

  // Creează și trimite un pachet de cerere

  Packet requestPacket = createRequestPacket(1, 123, 456, 0, 2);

  sendRequest(requestPacket);

  // Așteaptă și citește răspunsul de la slave

  Wire.requestFrom(SLAVE\_ADDRESS, sizeof(responsePacket));

  while(Wire.available() >= sizeof(responsePacket)){

    Serial.println("Received: ");

    Wire.readBytes((uint8\_t\*)&responsePacket, sizeof(responsePacket)); // Citirea pachetului de răspuns

    Serial.println(responsePacket.payload); // Afișarea informațiilor din pachet

  }

  delay(1000); // Întârziere înainte de următoarea cerere

}

**Rezultate:**

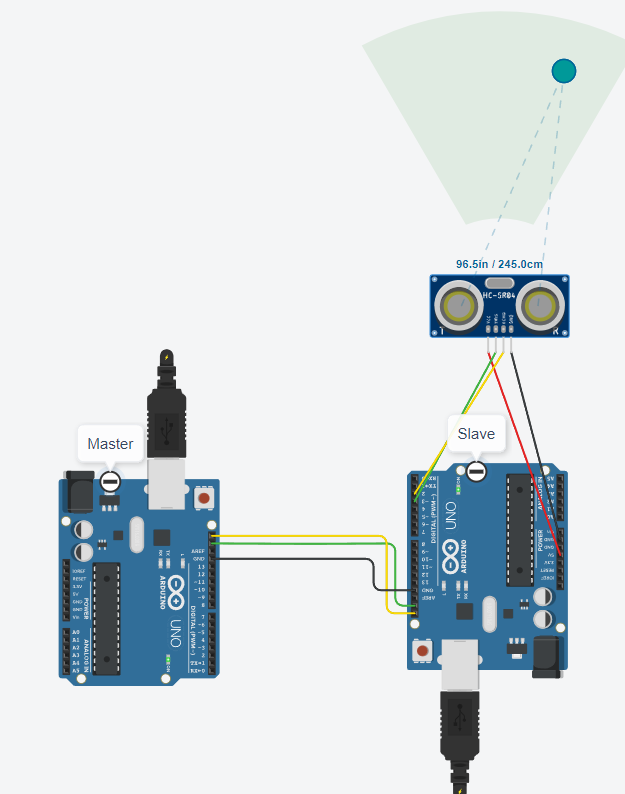


Figura 2 – Simularea aplicației

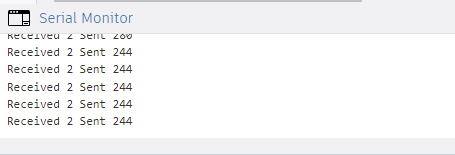


Figura 3 - Rezultatul

**Concluzie :**

În cadrul Laboratorului 7, ne-am propus să implementăm și să înțelegem procesul de comunicare între două microcontrolere folosind interfața I2C și protocolul logic de comunicare prin interfață serială. Această experiență a furnizat oportunitatea ideală pentru a consolida cunoștințele legate de comunicarea între dispozitive în contextul sistemelor încorporate, contribuind la o înțelegere mai profundă a interacțiunii dintre diversele componente ale unui sistem. Pe ansamblu, participarea la acest laborator a subliniat importanța unei comunicări eficiente și sigure între dispozitivele încorporate, oferindu-ne competențele necesare pentru a dezvolta și depana protocoale de comunicare complexe în cadrul sistemelor moderne de automatizare și IoT.