TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

Laboratory work nr. 7.2

Communication between devices - SW Serial Protocol

Elaborated: **Cristian Brinza**
st. gr. FAF-212

Verified: **Moraru Dumitru**
lect. univ

Chișinău, 2024

THE TASK OF THE LABORATORY WORK:

To create an MCU-based application that will take a signal from a signal source, and display the physical parameter at a terminal (LCD and/or Serial).

Each student will select a sensor either analog or digital (not binary) from the attached PDF or: http://www.37sensors.com/.

- **To acquire the signal from the sensor;**
- **To display the data on the LCD and / or Serial display.**
- **To condition the signal involving digital filters and other methods**

PROGRESS OF THE WORK

1 Main functions/methods used to execute the task

Explication about this chapterIn this chapter I will explain the functionality of diferent parts of the executed task:

1. In the sketch.ino File (Transmitter):

- **setup() Function**: This function initializes the serial communication and configures the ultrasonic sensor pins. It is foundational in setting up the serial communication to ensure the master device can send data effectively. Configuring the trigger and echo pins of the ultrasonic sensor prepares it for distance measurement. The setup phase is crucial as it establishes the operational baseline for the sensor and communication components, ensuring they are correctly initialized and ready for data acquisition and transmission.
- **loop() Function**: Manages the core operational cycle, including reading sensor data and sending packets over serial communication. This function embodies the continuous execution logic of the system, beginning with the acquisition of raw sensor data from the ultrasonic sensor. It processes these readings to create a data packet and sends this packet via serial communication, providing real-time data transmission. This loop ensures the system consistently monitors the environment and transmits the latest information.

2. In the sketch.ino File (Receiver):

- **setup() Function**: This function initializes the serial communication for receiving data packets from the transmitter.
- **loop() Function**: Manages the core operational cycle of receiving and validating data packets. It continuously checks for available serial data, reads the data packet, and validates it. If the packet is valid, it extracts and prints the distance value. This loop ensures the system consistently receives and processes incoming data packets.

3. Packet Transmission and Validation:

- **Packet Transmission (Transmitter)**: Constructs a data packet with distance readings and a checksum for validation. It sends this packet over the serial communication. The packet includes start and end markers, a packet number, sender and receiver IDs, packet type, distance, and a checksum to ensure data integrity.
- **Packet Validation (Receiver)**: Parses and validates the received data packet. It checks the start and end markers, extracts the checksum, and calculates the checksum from the packet data to verify its integrity. If the packet is valid, it extracts and prints the distance value.

Block Diagram

- **Main Program (sketch.ino)**: Serves as the heart of the system, initializing the Arduino pins for serial communication and the ultrasonic sensor. It orchestrates the operational flow, including acquiring sensor data, processing this data, and transmitting it via serial communication.
- **Setup Function (Transmitter)**: Responsible for initializing the serial communication and configuring the sensor pins for input.
- **Loop Function (Transmitter)**: Represents the continuous execution cycle of the Main Program. It handles reading sensor data, creating a data packet, and sending it via serial communication.
- **Setup Function (Receiver)**: Responsible for initializing the serial communication for receiving data packets.
- **Loop Function (Receiver)**: Represents the continuous execution cycle of the Main Program. It handles receiving data packets, validating them, and extracting the distance value.
- **Packet Transmission and Validation**: Manages the construction, transmission, reception, and validation of data packets.

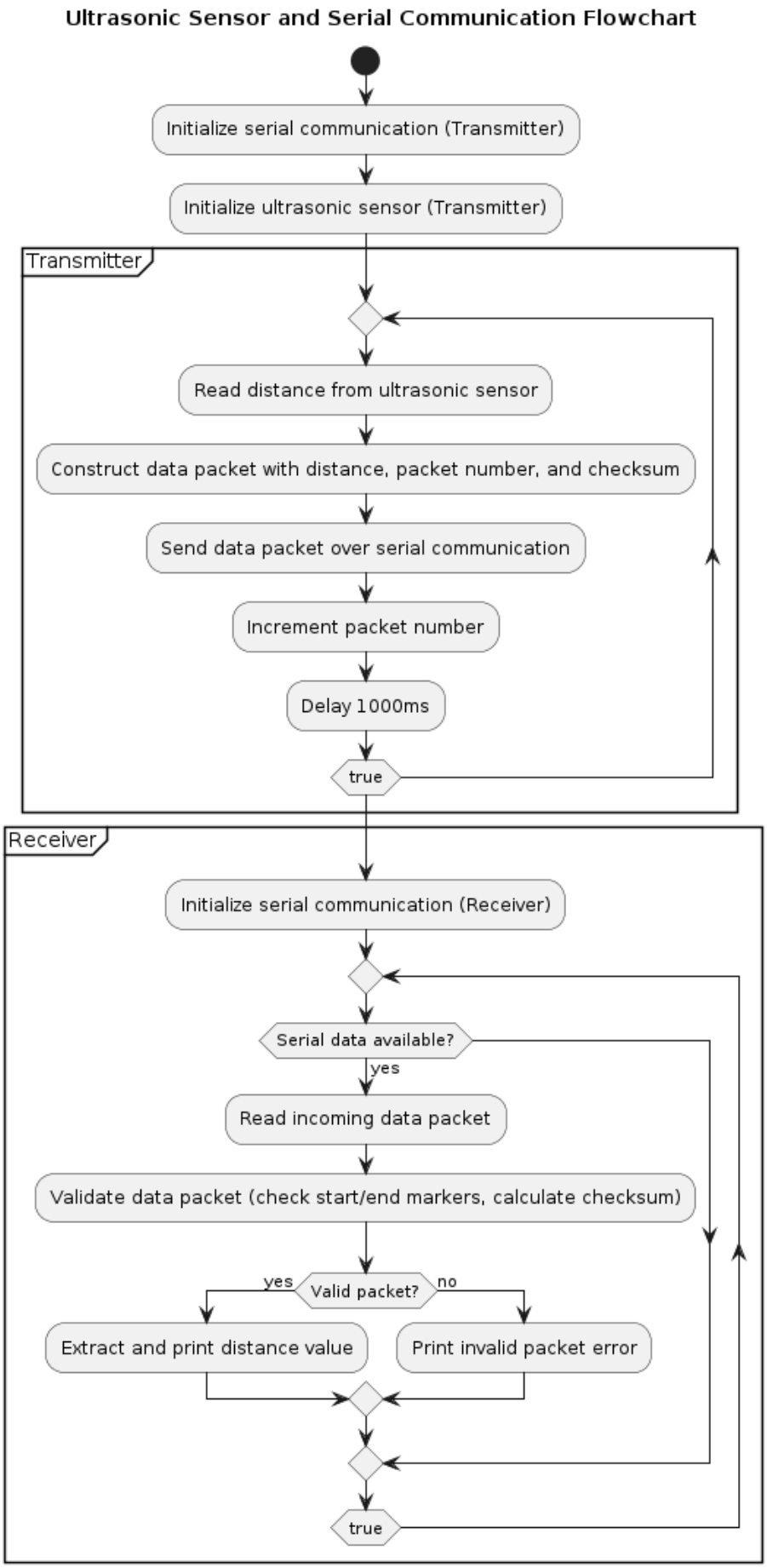

Figure 1 Program schema.

Simulated or real assembled electrical schematic diagram :

The Figure 2 depicted the phisical board schema.

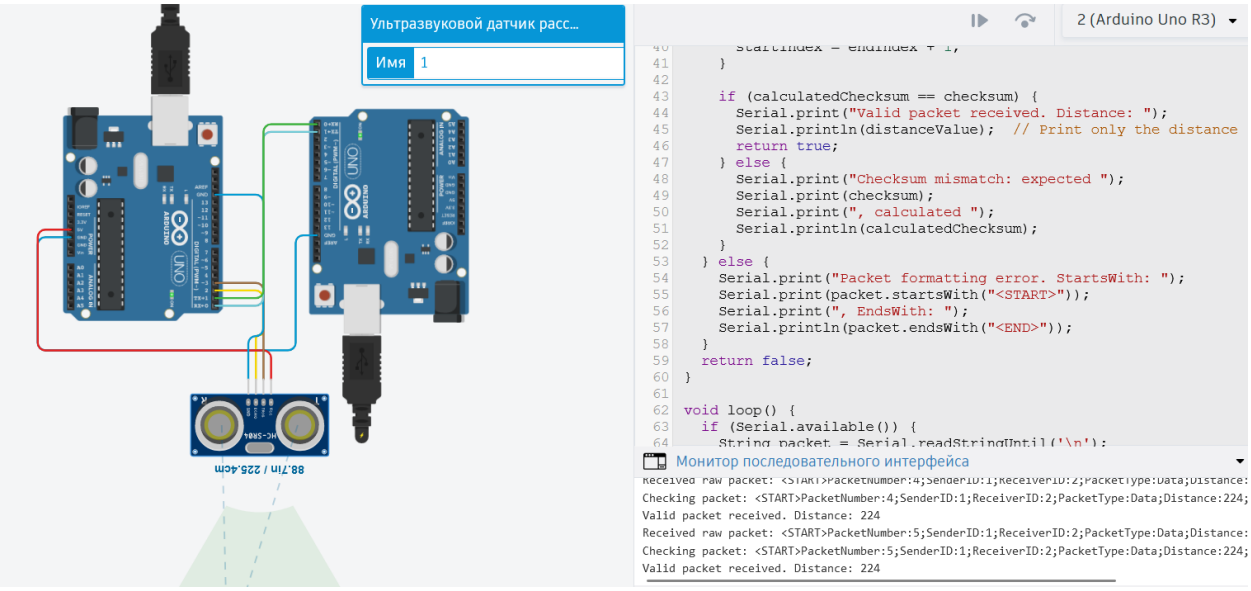

Figure 2 Phisical board schema.

CONCLUSION

In completing this lab, we have navigated through the detailed intricacies of working with Arduino, particularly focusing on distance measurement and communication using an ultrasonic sensor and serial communication. This project has exemplified the harmonious integration of hardware and software components, illustrating our capability to collect raw sensor data, implement communication protocols, and visually communicate system statuses through the Serial Monitor.

We initiated the project by setting up crucial components: preparing the serial communication for data transmission and configuring the Arduino to interface with the ultrasonic sensor. The development of the sketch.ino file for both the transmitter and receiver formed the backbone of our endeavor, incorporating both setup() and loop() functions that are essential for the system's operation. The setup() function was pivotal in ensuring that all components were correctly initialized and operational, whereas the loop() function played a continuous role in reading distance data, creating data packets, sending them, and validating received packets.

This lab not only underscored the importance of precise and responsive data handling in embedded systems but also highlighted the critical role of communication protocols in real-time monitoring. By implementing a straightforward yet effective communication logic, we addressed the challenges associated with maintaining accurate data transmission. Additionally, the practice of interpreting sensor output to enact communication decisions reinforced the necessity of integrating accurate electronics principles and understanding the behavior of sensors and communication interfaces.

The lab served as more than just a practical exercise; it was a robust demonstration of the foundational principles of embedded system design, encompassing sensor interfacing, communication protocol implementation, and user interaction. The skills and insights gained from this project provide a solid base for future explorations, whether they involve further applications in sensor data management, Internet of Things (IoT) implementations, or broader ventures within the expansive fields of electronics and embedded systems. Through this project, we have strengthened our understanding and are better prepared for the technological challenges and opportunities that lie ahead.

BIBLIOGRAPHY

1 EDUCBA: *Introduction to Embedded Systems.* Cursuri electronice online ©2020 [quote 20.03.2024] Available: https://www.educba.com/what-is-embedded-systems/

2 ARDUINO: *Arduino UNO.* The official website of Arduino modules, ©2020 [quote 20.03.2024]. Available: https://www.arduino.cc/.

3 TUTORIALSPOINT: *Embedded Systems Tutorial*. Comprehensive guide for beginners and professionals to learn Embedded System basics to advanced concepts, including microcontrollers, processors, and real-time operating systems. ©2023 [quote 20.03.2024] Available: https://www.tutorialspoint.com/embedded_systems/index.htm

4 GURU99: *Embedded Systems Tutorial: What is, History & Characteristics*. A detailed introduction to embedded systems, covering microcontrollers, microprocessors, and the architecture of embedded systems, as well as their applications and advantages. ©2023 [quote 20.03.2024] Available: https://www.guru99.com/embedded-systems-tutorial.html

5 JAVATPOINT: *Embedded Systems Tutorial.* Offers basic and advanced concepts of Embedded System, designed for beginners and professionals. ©2023 [quote 20.03.2024] Available: https://www.javatpoint.com/embedded-systems-tutorial

6 DEEPBLUE*: Embedded Systems Tutorials Introduction* | Embedded Systems Online Course. ©2023 [quote 20.03.2024] Available: https://deepbluembedded.com/embedded-systems-tutorials/

APPENDIX 1

Code of main.ino:

```
void setup() {
  Serial.begin(9600);
}
bool parsePacket(String packet) {
Serial.print("Checking packet: ");
Serial.println(packet);
if (packet.startsWith("<START>") && packet.endsWith("<END>")) {
packet.replace("<START>", "");
packet.replace("<END>", "");
int checksumIndex = packet.lastIndexOf("Checksum:");
if (checksumIndex == -1) {
Serial.println("Checksum not found.");
return false;
}
String checksumStr = packet.substring(checksumIndex + 9);
int checksum = checksumStr.toInt();
int calculatedChecksum = 0;
int startIndex = 0;
int distanceValue = 0;  // Variable to store the distance
while (startIndex < checksumIndex) {
int endIndex = packet.indexOf(';', startIndex);
if (endIndex == -1) endIndex = checksumIndex;
String field = packet.substring(startIndex, endIndex);
int colonIndex = field.indexOf(':');
if (colonIndex != -1) {
String key = field.substring(0, colonIndex);
String valueStr = field.substring(colonIndex + 1);
int value = valueStr.toInt();
calculatedChecksum += value;
if (key == "Distance") {  // Check if the key is "Distance"
distanceValue = value;  // Assign the distance value}}
startIndex = endIndex + 1;}
if (calculatedChecksum == checksum) {
Serial.print("Valid packet received. Distance: ");
Serial.println(distanceValue);  // Print only the distance
return true;
} else {
```

```
Serial.print("Checksum mismatch: expected ");
Serial.print(checksum);
Serial.print(", calculated ");
Serial.println(calculatedChecksum);
}} else {
Serial.print("Packet formatting error. StartsWith: ");
Serial.print(packet.startsWith("<START>"));
Serial.print(", EndsWith: ");
Serial.println(packet.endsWith("<END>"));}
return false;}

void loop() {
if (Serial.available()) {
String packet = Serial.readStringUntil('\n');
packet.trim(); // Trim whitespace and newlines
Serial.print("Received raw packet: "); // Debugging line to see what exactly is
received
Serial.println(packet);

if (!parsePacket(packet)) {
Serial.println("Invalid packet detected.");}}}
```