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

Laboratory work nr. 7

User interaction - Automates Finite – Semafor(Trafic light)

A elaborat: **Brinza Cristian**
st. gr. FAF-212

A verificat: **Moraru Dumitru**
lect. univ

Chișinău 2024

THE TASK OF THE LABORATORY WORK:

To create an MCU-based application that will implement communications between equipment as follows:

- Physical communication protocol - Communication between TWO Microcontrollers through the I²C interface
- MCU1 - implements the digital sensor with the I²C interface for the HCS-04 ultrasonic sensor, where data collection is executed from the sensor interface and retransmitted to the I2C or SPI interface when a data reading request is detected.
- MCU2 - executes the request through the I²C interface to the ultrasonic digital sensor (MCU+HCS-04) and displays the data on the serial interface
- as the serial interface will be used, the implementation for the serial interface will be a textual one

Reuse as much as possible the solutions presented in the previous labs

review the resources taught in the course

- use the serial interface for automata operation reports
- reuse as much as possible the solutions presented in the previous labs
- review the resources taught in the course clocking

Marking:

- 5 - simple device activation application
- +1 - for modular implementation of the project
- +1 - MCU1 sends packet data through the serial interface,
- +1 - MCU2 decodes packets coming from the serial interface
- +1 - for demonstrating evidence of physical implementation

NOTE: maximum time limit possible only when presenting physical performance!!

Penalties:

- 1 - penalty for NOT using STDIO
- 1 - penalty for each week late from the deadline
- 1 - penalty for non-compliance with the report format

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

In the Main Sketch File:

- *setup():*This function initializes the Arduino's serial communication and sets up the I2C communication protocol via the Wire library. It prepares the system for communication with the slave device by setting up the necessary peripherals and variables before entering the main program loop..
- *loop():* Acting as the core of the Arduino sketch, this function continuously executes commands to communicate with the I2C slave device. It requests data from the slave, processes the received information, and outputs it to the serial monitor.

In Device Control and Status Reporting:

- *initiateTransmission():*Initiates the transmission to the I2C slave device, sending a request for data. This function is crucial for establishing communication with the slave device and ensuring data exchange.
- *requestData():*Requests 2 bytes of data from the I2C slave device, expected to represent 'distance'. It combines the two received bytes into a single integer and returns the distance value.
- *displayDistance(int distance):* Displays the received distance value on the serial monitor. This function updates the user with the current measurement, ensuring real-time feedback.

State Management:

- *Serial Monitor:* Utilized for displaying the received data from the I2C slave device. The Serial.begin(), Serial.print(), and Serial.println() functions are used extensively to provide real-time feedback about the system's status, including the current distance measurement.

Explanation of Chapters

This section has outlined the implementation and functionality of the main sections of our code. Through the setup and loop functions, the sketch manages to establish a responsive system capable of interpreting I2C commands for data retrieval from a slave device. The integration of direct control methods, along with real-time feedback on the serial monitor, exemplifies a practical approach to interactive device management in a laboratory setting.

Device control functions such as initiating transmission and requesting data highlight the application's ability to interact with other devices through digital communication protocols, a core aspect of automation and control systems. Meanwhile, the serial monitor feedback ensures that the user remains informed about the system's current status, fostering an intuitive interaction between the user and the system.

2 Block Diagram

The diagram is consisting of the following main blocks:

- **Main Program (sketch.ino):** The central point that initializes and controls other components and manages the program flow. It orchestrates the communication between the Arduino and the I2C slave device.
- **I2C Communication:** Represents the logic for interfacing with the I2C slave device, including data requests and transmission handling.
- **Serial Monitor:** Used for debugging and displaying the received data from the I2C slave device. Provides real-time feedback about the distance measurements.
- **Arduino Board:** The physical layer where the LEDs are connected, with GPIO pins used to control the traffic lights.

The Figure 1 depicted the UML program flow.

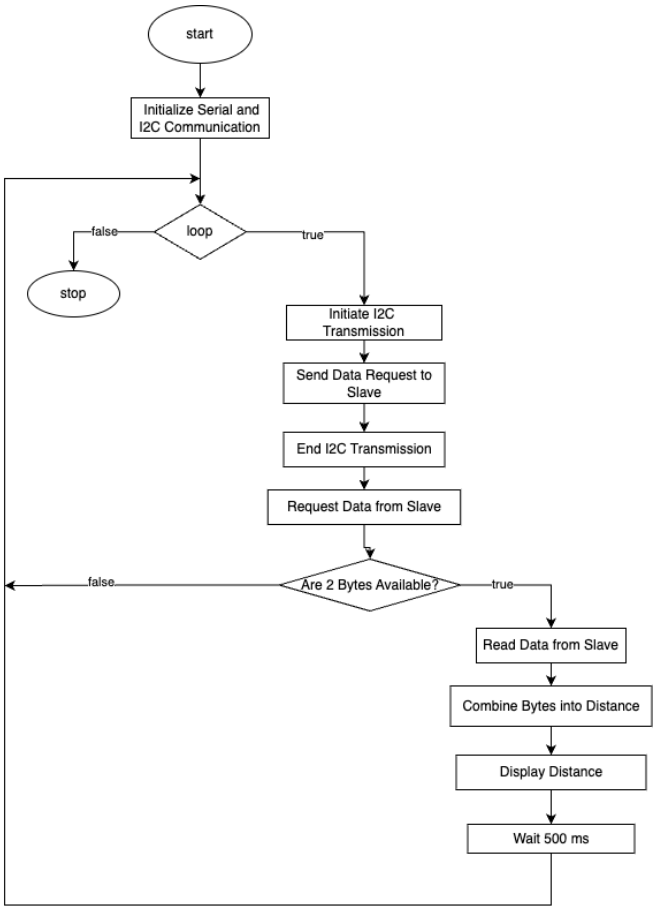

Figure 1 Program schema.

3 Simulated or real assembled electrical schematic diagram :

The Figure 2 depicted the phisical board schema.

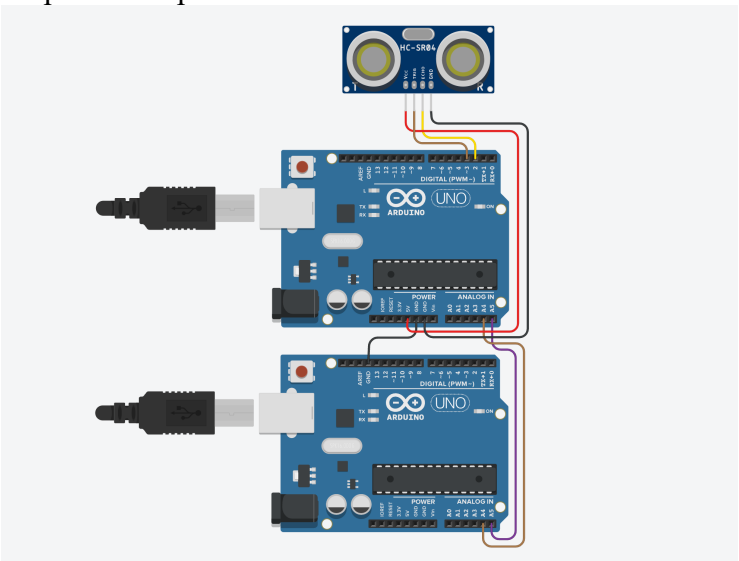

Figure 2 Phisical board schema.

4 Screenshots of the simulation execution:

The Figure 3 depicted a sceenshot of the TINKERCAD simulation :

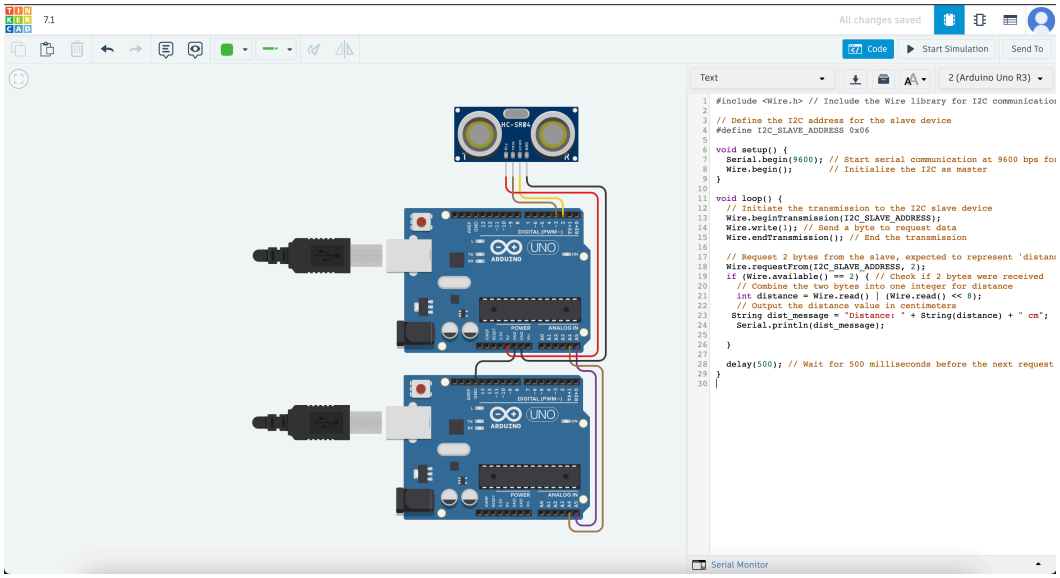

Figure 3 Screenshot of the simulation.

CONCLUSION

This project embarked on the innovative task of creating a dynamic control system, utilizing the versatile Arduino platform to communicate with an I2C slave device for distance measurement. The journey from conceptualization to realization has been both challenging and rewarding, offering deep insights into the integration of hardware and software to achieve precise control and feedback mechanisms.

Throughout the development process, we adhered to best coding practices, emphasizing readability and maintainability. The use of clear variable names and structured functions facilitated a seamless debugging experience and ensured that our system was both efficient and scalable. A notable highlight of our project was the successful implementation of I2C communication to request and process data from a slave device, demonstrating the power of interactive systems in real-world applications.

The Arduino ecosystem, with its extensive libraries and community support, played a crucial role in the swift development and troubleshooting of our system. This project underscored the importance of understanding both the limitations and capabilities of each component, from the intricacies of I2C communication to the nuances of serial output.

In conclusion, this project was not just a technical achievement but a comprehensive learning venture that covered the spectrum of embedded system design, from software logic to hardware manipulation. The skills honed and the knowledge gained throughout this project provide a robust foundation for future endeavors in automation and control systems. The successful implementation of this control system stands as a testament to our ability to bridge the gap between theoretical concepts and practical applications, sparking a continued interest in exploring the vast possibilities within the realm of embedded systems.

BIBLIOGRAPHY

1 EDUCBA: *Introduction to Embedded Systems.* Cursuri electronice online ©2020 [quote 10.02.2024] Available: https://www.educba.com/what-is-embedded-systems/

2 ARDUINO: *Arduino UNO.* The official website of Arduino modules, ©2020 [quote 10.02.2024]. 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 10.02.2024] 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 10.02.2024] 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 10.02.2024] Available: https://www.javatpoint.com/embedded-systems-tutorial

6 DEEPBLUE*: Embedded Systems Tutorials Introduction* | Embedded Systems Online Course. ©2023 [quote 10.02.2024] Available: https://deepbluembedded.com/embedded-systems-tutorials/

APPENDIX 1

Code of main.ino:

```
#include <Wire.h> // Include the Wire library for I2C communication

// Define the I2C address for the slave device
#define I2C_SLAVE_ADDRESS 0x06

void setup() {
  Serial.begin(9600); // Start serial communication at 9600 bps for debugging
  Wire.begin();       // Initialize the I2C as master
}

void loop() {
  // Initiate the transmission to the I2C slave device
  Wire.beginTransmission(I2C_SLAVE_ADDRESS);
  Wire.write(1); // Send a byte to request data
  Wire.endTransmission(); // End the transmission

  // Request 2 bytes from the slave, expected to represent 'distance'
  Wire.requestFrom(I2C_SLAVE_ADDRESS, 2);
  if (Wire.available() == 2) { // Check if 2 bytes were received
    // Combine the two bytes into one integer for distance
    int distance = Wire.read() | (Wire.read() << 8);
    // Output the distance value in centimeters
   String dist_message = "Distance: " + String(distance) + " cm";
    Serial.println(dist_message);

  }

  delay(500); // Wait for 500 milliseconds before the next request
}
```