**TECHNICAL UNIVERSITY OF MOLDOVA**

**FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS**

**DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS**

**Laboratory work #7**

**Communication.**

**Performed the work:**

**Basso Nicolae, FAF-191**

**Checked by:**

**Moraru Dumitru**

**2022**

**1. The task of the laboratory work**

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 dae de la sensorul digital implementat in p 1. sa si implementezi inca o camanda la alegere, pentru diversitate.

Recomandare:

asa cum se va utiliza interfata seriala, se implementarea pentru interfata seriala va fi un textuala

pentru citirea valorilor numerice se recomanda utilizarea functiei Serial.parseint()

Reutilizati la maxim solutiile prezentate in laboratoarele precedente

revizuiti resursele predate la curs

Pontaj:

nota 5 - simpla aplicatie de comunicare

+1.0 - pentru implementare modulara a proiectului

+1.0 - MCU2 trimite date impachetare prin interfata seriala

+1.0 - MCU2 decodifica pachete venite de pe interfata seriala

+1.0 - MCU1 detecteaza cererile de la MCU2 prin interfata I2C

+1.0 - MCU2 receptioneaza corect atele de la MCU1 prin interfata I2C

+1.0 - pentru demonstrarea probelor de implementare fizica

penalitati

-1 - penalizare pentru fiecare saptamana intarziere de la deadline

-1 - penalizare pentru nerespectare format raport

**2. The progress of the work**

**2.1 Description of the main functions used to perform the tasks**

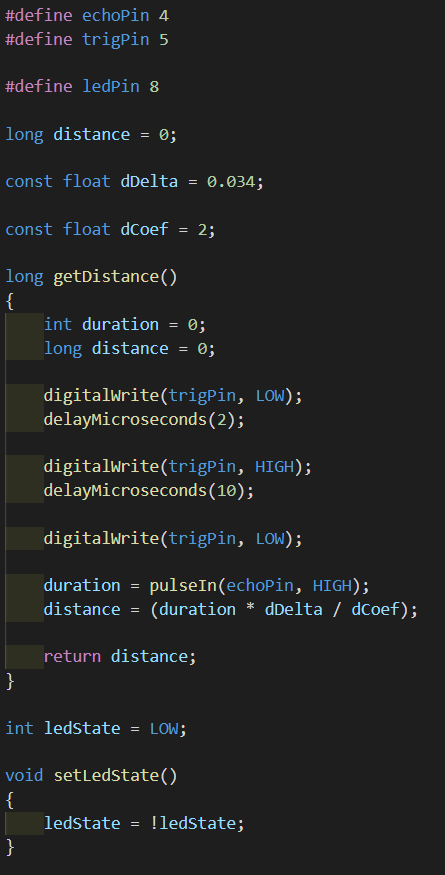
The problem was to create two solutions for two Arduino microconrtollers: Master and Slave:

1. The Master, which receives commands „light” or „dist” from serial controller, sends protocolled messages to the Slave, waits the response from the Slave and then prints the result in serial controller:



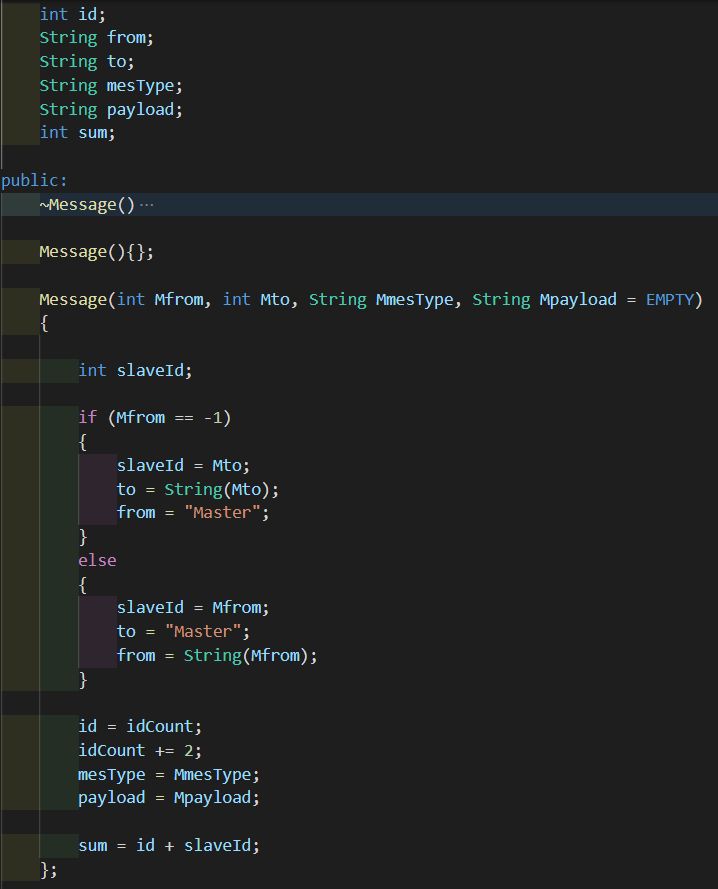
**Figure 1** The main Master code for the task.

1. The Slave, which waits for messages from Master, processes it, and then sends response to the Master:



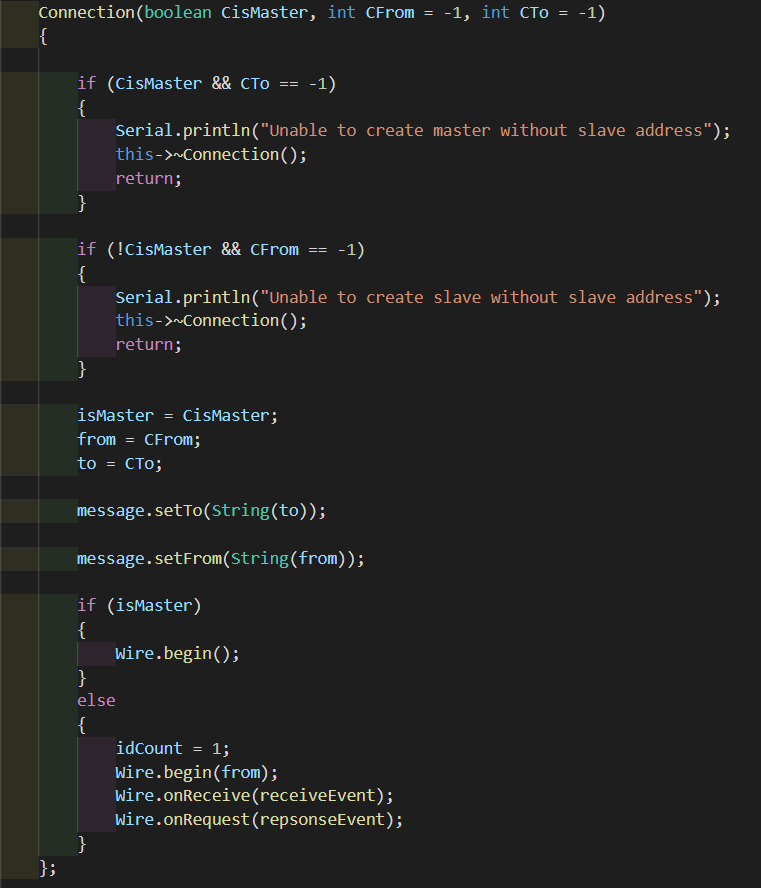
**Figure 2** The general main Slave code for the task.

1. The Message protocol solution, which is represented as class with fields to create standardized messages sending to the other microcontroller:



**Figure 3** The general Message code for the task.

1. The Connection protocol solution, which is represented as class with fields to create standardized action for sending to the other microcontroller and receiving from it messages (represented as receiveEvent(), responseEvent() functions in Slave and requestFrom() in Master) (all the code can be checked in the source zip archive):



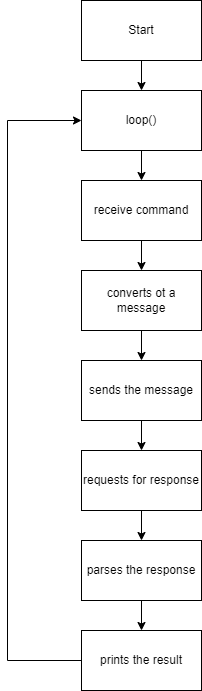
**Figure 4** The Connection constructor code for the task.

1. The message itself consists of several strings that are consequently being sent to the other side:

* MessageStart
* MessageId
* From which device is being sent (address, if it is a Slave, “Master” if it is a Master)
* To which device is being sent (the same thing as in “From”)
* Type of the message (currently exists four types of the messages: “LIGHT” is being sent by Master to Slave to toggle the LED light; “GET” is being sent by Master to Slave to request data from sensor; “LIGHT\_OK” is being sent by Slave to Master when the LED light was toggled; “DATA” is being sent by Slave to Master, where in Payload is saved distance in cm from the sensor.)
* Payload of the message (usually is “empty”, but if the message type is “DATA”, there have to be number, which represents the distance data from sensor).
* Sum (checksum of some fields of the message and represented as integer)
* MessageEnd

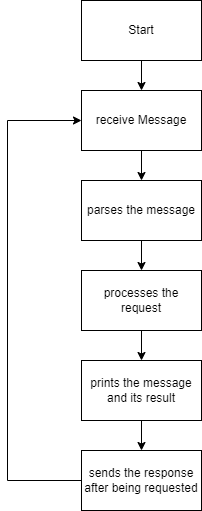
**2.2 The programs’ diagram block**

During this part of laboratory work, all the code is being executed consequently, all the main code is located in several main function and classes that process data. Of course, these functions contain operations and other functions invocations. Afrer all the data was received, it prints it.



**Figure 5** Flowchart diagram for the main functions of Master.

The same thing is about slave, but it firstly waits for the message from master, then processes it, and after responses:

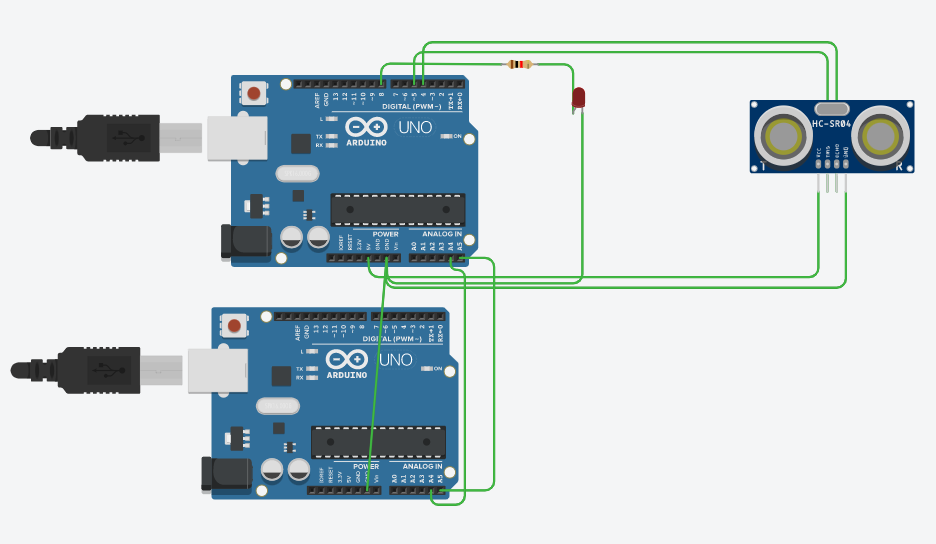


**Figure 6** Flowchart diagram for the main functions of Slave.

**2.3 Simulated electrical schematic**

1) Schematic of the solution for the sequential type of software:

MCU-based applications that communicate between eachother sending and receiving messages to process data and print it or to send response to the other side.



**Figure 7** Simulated schematic for the solution.

**Conclusions**

During this laboratory work, we implemented an MCU-based application that controls two microcontrollers with commands received from Serial terminal and some data printed using Serial terminal. Both solutions are using I2C protocol for the intercommunication, which major part is implemented as “Wire.h” for the Arduino Platform. The first solution is a Master, which receives user’s commands from Serial terminal, wraps it into a message that can be acknowledged by both devices, sends it to the other side (Slave) and waits for the response from it. The other side at this moment is receiving the message from the Master, parses it to get useful data and processes this data, after processing, the Slave sends response to the Master using the same standard (protocol, if you wish) and prints that data was processed. After that, Master receives the message from the Slave and prints the result in the Serial terminal.

**Bibliography**

1. ARDUINO: *Arduino Language Reference*. Arduino official site, ©2022 [quote 03.05.2022]. Access link: <https://www.arduino.cc/reference/en/>
2. ARDUINO: *Master Reader/Slave Sender*. Arduino official site, ©2022 [quote 03.05.2022]. Access link: <https://www.arduino.cc/en/Tutorial/LibraryExamples/MasterReader>
3. ARDUINO: *Wire Arduino Reference*. Arduino official site, ©2022 [quote 03.05.2022]. Access link: <https://www.arduino.cc/reference/en/language/functions/communication/wire/>

  Что вершит судьбу человечества в этом мире? Некое незримое существо или закон, подобно Длани Господней парящей над миром? По крайне мере истинно то, что человек не властен даже над своей волей.