**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

**Інститут комп’ютерних технологій, автоматики та метрології**

**Кафедра "Комп'ютеризовані системи автоматики"**



**ЗВІТ**

про виконання **лабораторної роботи №5**

з курсу **«Мікроконтролери, частина 1»**

**«Розроблення завадостійкого протоколу обміну**

**даних в мережі мікроконтролерів на базі RS-485»**

**Виконав:**

студент групи ІР-22

Яцків О. О.

Варіант 38

**Прийняла:**

к.т.н., старший викладач КСА

Лагун І. І.

Львів — 2023

**Порядок виконання**

1. Згідно варіанту завдання (таблиця 1) зібрати у пакеті симуляції Proteus схему та написати програми для реалізації вказаного завдання.

2. Розробити клієнтську програму для керування МК-системою.

3. Протестувати роботу клієнтської програми зі схемою в Proteus.

4. Залити програми в Arduino-Kits.

5. Отримані результати представити викладачу

**Завдання (варіант 38)**

1. Спроектувати принципову схему системи обміну даними між мікроконтролерною мережею (один ATmega2560 та два ATmega328P) на основі інтерфейсу RS-485 та персональним комп’ютером;

2. Розробити протокол обміну між мікроконтролерами «master» та «slave» мережі на основі RS-485, а також між клієнтською програмою Windows та МК «master» (RS-232); У протоколі мають бути наявні контрольні суми CRC (таблиця 2).

3. Для МК «master» та «slave» написати програмний код, що реалізовуватиме поставлену задачу:

- 1й Slave надсилає «Прізвище Ім’я По-батькові» студента;

- 2й Slave надсилає «Дату народження» студента;

Продублювати 5 раз ці повідомлення з внесеними в них спотвореннями згідно варіанту завдання.

4. Розробити клієнтську програму (С++, C#, JavaScript) та написати для неї програмний код, що реалізовуватиме зв’язок між комп’ютером та МК «master» і відображатиме та аналізуватиме отримані дані з МК «slave» на наявність помилок передачі.

**RS-485:** швидкість передачі 4800 Бод

**RS-232:** швидкість передачі 4800 Бод

**Aдреси МК:** Slave1 = 87, Slave2 = 77

**CRC:** CRC-8 (таблична реалізація) для кожних 6 байтів інформації

**Перелік спотворень:**

1) без спотворень

2) 2й розряд найстаршого байту даних Slave1

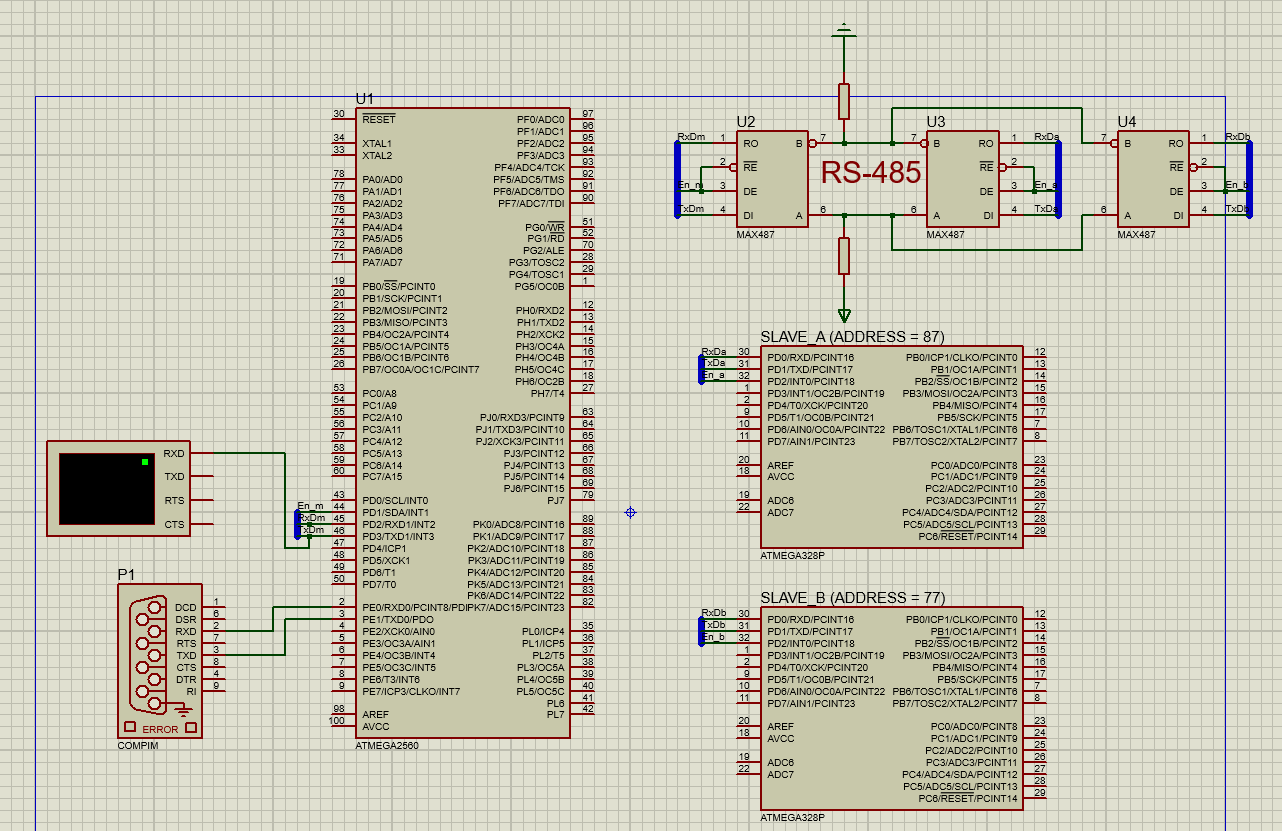
3) 5й розряд наймолодшого байту даних Slave2

4) 3й та 4й розряд 3го байту даних Slave2

5) 1, 2, 6 розряди 8го байту даних Slave1

**Виконання роботи**

У пакеті симуляції Proteus зібрав необхідну схему на основі мікроконтролерів ATMega2560 та ATMega328P:



Програма мовою C++ в Arduino IDE на мікроконтролері master:

#include <Arduino.h>

bool isAddress = true;

bool isCommand = false;

byte command;

void setWriteModeRS485() {

  byte port = PORTD;

  PORTD |= 1 << PD1; // режим відправки

  delay(1);

}

// переривання при завершенні передачі

ISR(USART1\_TX\_vect)

{

  PORTD &= ~(1 << PD1); // режим прийому

}

void setup() {

  delay(1000);

  // En\_m - на вихід + низький рівень

  DDRD |= 1 << PD1;

  PORTD &= ~(1 << PD1);

  // initialize UART0 (RS-232)

  Serial.begin(4800);

  // initialize RS-485

  Serial1.begin(4800, SERIAL\_8N1);

  Serial1.write("");

  UCSR1B |= (1 << UCSZ12) | (1 << TXCIE1);

}

void loop() {

  if (Serial.available()) {

    byte inByte = Serial.read();

    if (isAddress) {

      setWriteModeRS485();

      UCSR1B |= 1 << TXB81;

      Serial1.write(inByte);

      isAddress = false;

      isCommand = true;

    } else if (isCommand) {

      command = inByte;

      isCommand = false;

      setWriteModeRS485();

      // після запису даних у буфер передачі необхідно

      // надати дозвіл на переривання для події спорожнення буфера UDR.

      UCSR1B &= ~(1 << TXB81);

      Serial1.write(inByte);

      if (command == 0xB1) isAddress = true; // Команда читання 1 байту

    } else { // data byte

      isAddress = true;

      setWriteModeRS485();

      UCSR1B &= ~(1 << TXB81);

      Serial1.write(inByte);

    }

  }

  if (Serial1.available()) { // if we receive something from slaves, we also process it.

    byte inByte1 = Serial1.read();

    Serial.write(inByte1);

  }

}

Програма мовою C++ в Arduino IDE на мікроконтролері slave A:

#include <Arduino.h>

#define SLAVE\_ADDRESS 0x57

byte address;

int v = 0;

void writeData();

char crc8(const char \*data,int length);

void setWriteModeRS485() {

  PORTD |= 1 << PD2;

  delay(1);

}

// переривання по завершенню передачі

ISR(USART\_TX\_vect) {

  PORTD &= ~(1 << PD2);

}

void setup() {

  delay(1000);

  DDRD = 0b00000111;

  PORTD = 0b11111000;

  Serial.begin(4800, SERIAL\_8N1);

  UCSR0B |= (1 << UCSZ02) | (1 << TXCIE0);

  UCSR0A |= (1 << MPCM0);

  delay(1);

}

void loop() {

  if (Serial.available()) {

    byte inByte = Serial.read();

    if (SLAVE\_ADDRESS == inByte) {

      UCSR0A &= ~(1 << MPCM0);

      setWriteModeRS485();

      writeData();

      delay(200);

    }

  }

}

void writeData()

{

  byte byteMessage[24];

  String message = "Yatskiv Oleh Olehovych\0";

  message.getBytes(byteMessage, 24);

  byte encryptedByteMessage[26];

  for (int i = 0; i < 24; i++) {

    encryptedByteMessage[i] = byteMessage[i];

  }

  for(int k = 0; k < 5; k++) {

    if (v == k) {

      for (unsigned int i = 0; i < sizeof(encryptedByteMessage); i++) {

        if (k == 1 && i == 0) {

          encryptedByteMessage[i] ^= (1 << 2);

        } else if (k == 4 && i == 8) {

          encryptedByteMessage[i] ^= ((1 << 1) | (1 << 2) | (1 << 6));

        }

      }

    }

  }

  byte byteMessageModified[24];

  for (int i = 0; i < 24; i++) {

    byteMessageModified[i] = encryptedByteMessage[i];

  }

  unsigned short checkSumCRC = crc8(byteMessageModified, sizeof(byteMessageModified));

  byte firstByteOfCheckSum = (checkSumCRC >> 8) & 0xFF;

  byte secondByteOfCheckSum = checkSumCRC & 0xFF;

  encryptedByteMessage[24] = firstByteOfCheckSum;

  encryptedByteMessage[25] = secondByteOfCheckSum;

  for (unsigned int i = 0; i < sizeof(encryptedByteMessage); i++) {

    Serial.write(encryptedByteMessage[i]);

  }

  v++;

  if (v == 5) {

    v = 0;

  }

}

// імплементація CRC8

char crc8(const char \*data, int length)

{

   char crc = 0x00;

   char extract;

   char sum;

   for(int i=0;i<length;i++)

   {

      extract = \*data;

      for (char tempI = 8; tempI; tempI--)

      {

         sum = (crc ^ extract) & 0x01;

         crc >>= 1;

         if (sum)

            crc ^= 0x8C;

         extract >>= 1;

      }

      data++;

   }

   return crc;

}

Програма мовою C++ в Arduino IDE на мікроконтролері slave B:

#include <Arduino.h>

#define SLAVE\_ADDRESS 0x4C

byte address;

int v = 0;

void writeData();

char crc8(const char \*data, int length);

void setWriteModeRS485() {

  PORTD |= 1 << PD2;

  delay(1);

}

// переривання по завершенню передачі

ISR(USART\_TX\_vect) {

  PORTD &= ~(1 << PD2);

}

void setup() {

  delay(1000);

  DDRD = 0b00000111;

  PORTD = 0b11111000;

  Serial.begin(4800, SERIAL\_8N1);

  UCSR0B |= (1 << UCSZ02) | (1 << TXCIE0);

  UCSR0A |= (1 << MPCM0);

  delay(1);

}

void loop() {

  if (Serial.available()) {

    byte inByte = Serial.read();

    if (SLAVE\_ADDRESS == inByte) {

      UCSR0A &= ~(1 << MPCM0);

      setWriteModeRS485();

      writeData();

      delay(200);

    }

  }

}

void writeData()

{

  byte byteMessage[12];

  String message = "11.09.2003\0";

  message.getBytes(byteMessage, 12);

  byte encryptedByteMessage[14];

  for (int i = 0; i < 12; i++) {

    encryptedByteMessage[i] = byteMessage[i];

  }

  for(int k = 0; k < 5; k++) {

    if (v == k) {

      for (unsigned int i = 0; i < sizeof(encryptedByteMessage); i++) {

        if (k == 2 && i == sizeof(encryptedByteMessage) - 5) {

          encryptedByteMessage[i] ^= (1 << 5);

        } else if (k == 3 && i == 3) {

          encryptedByteMessage[i] ^= ((1 << 3) | (1 << 4));

        }

      }

    }

  }

  byte byteMessageModified[12];

  for (int i = 0; i < 12; i++) {

    byteMessageModified[i] = encryptedByteMessage[i];

  }

  unsigned short checkSumCRC = crc8(byteMessageModified, sizeof(byteMessageModified));

  byte firstByteOfCheckSum = (checkSumCRC >> 8) & 0xFF;

  byte secondByteOfCheckSum = checkSumCRC & 0xFF;

  encryptedByteMessage[12] = firstByteOfCheckSum;

  encryptedByteMessage[13] = secondByteOfCheckSum;

  for (unsigned int i = 0; i < sizeof(encryptedByteMessage); i++) {

    Serial.write(encryptedByteMessage[i]);

  }

  v++;

  if (v == 5) {

    v = 0;

  }

}

// імплементація CRC8

char crc8(const char \*data, int length)

{

   char crc = 0x00;

   char extract;

   char sum;

   for(int i=0;i<length;i++)

   {

      extract = \*data;

      for (char tempI = 8; tempI; tempI--)

      {

         sum = (crc ^ extract) & 0x01;

         crc >>= 1;

         if (sum)

            crc ^= 0x8C;

         extract >>= 1;

      }

      data++;

   }

   return crc;

}

Клієнтська програма мовою Python з використанням бібліотеки PyQt:

from PyQt5 import QtWidgets

from PyQt5 import QtGui

from PyQt5 import QtCore

from PyQt5.QtWidgets import QApplication, QMainWindow

import sys

import serial

import serial.tools.list\_ports as ports\_list

import crc8

class Application(QMainWindow):

    def \_\_init\_\_(self) -> None:

        super(Application, self).\_\_init\_\_()

        width = 400

        height = 300

        self.setGeometry(int((1920 - width) / 2), int((1080 - height) / 2), width, height)

        self.setWindowTitle("Lab 5")

        self.setStyleSheet("background-color: #000420; font-size: 22px;")

        self.setWindowIcon(QtGui.QIcon("mcicon.png"))

        self.initUI()

    def initUI(self):

        label\_top = QtWidgets.QLabel(self)

        label\_top.setGeometry(QtCore.QRect(0, 0, self.width(), 50))

        label\_top.setStyleSheet("background-color: #000412;")

        label\_text = QtWidgets.QLabel(label\_top)

        label\_text.setText("COM порт")

        label\_text.move(10, int((label\_top.height() - label\_text.height()) / 2))

        label\_text.setStyleSheet("color: #FFFFFF; background: none;")

        self.dropdown = QtWidgets.QComboBox(label\_top)

        for port in ports\_list.comports():

            self.dropdown.addItem(port.name)

        dropdown\_width = self.dropdown.width()

        for i in range(self.dropdown.count()):

            if (self.dropdown.itemText(i).\_\_len\_\_() > dropdown\_width / 22):

                dropdown\_width = self.dropdown.itemText(i).\_\_len\_\_() \* 22

        self.dropdown.setGeometry(label\_text.width() + 20, int((label\_top.height() - self.dropdown.height()) / 2), dropdown\_width, self.dropdown.height())

        self.dropdown.setStyleSheet("background-color: #000412; color: #FFFFFF; border: 1px solid #FFFFFF; border-radius: 6px;")

        self.dropdown.setCursor(QtGui.QCursor(QtCore.Qt.CursorShape.PointingHandCursor))

        self.open\_button = QtWidgets.QPushButton(label\_top)

        self.open\_button.setText("Відкрити")

        self.open\_button.setGeometry(label\_text.width() + self.dropdown.width() + 35, int((label\_top.height() - 28) / 2), self.open\_button.width(), 28)

        self.open\_button.setStyleSheet("color: #FFFFFF; font-size: 18px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

        self.open\_button.setCursor(QtGui.QCursor(QtCore.Qt.CursorShape.PointingHandCursor))

        self.open\_button.installEventFilter(self)

        self.slave1\_button = QtWidgets.QPushButton(self)

        self.slave1\_button.setGeometry(35, label\_top.height() + 30, self.slave1\_button.width() + 45, 40)

        self.slave1\_button.setText("Slave 1")

        self.slave1\_button.setStyleSheet("color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

        self.slave1\_button.setCursor(QtGui.QCursor(QtCore.Qt.CursorShape.PointingHandCursor))

        self.slave1\_button.installEventFilter(self)

        self.slave2\_button = QtWidgets.QPushButton(self)

        self.slave2\_button.setGeometry(self.width() - self.slave1\_button.width() - 35, label\_top.height() + 30, self.slave2\_button.width() + 45, 40)

        self.slave2\_button.setText("Slave 2")

        self.slave2\_button.setStyleSheet("color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

        self.slave2\_button.setCursor(QtGui.QCursor(QtCore.Qt.CursorShape.PointingHandCursor))

        self.slave2\_button.installEventFilter(self)

        self.slave1\_text = QtWidgets.QLabel(self)

        self.slave1\_text.setText("Slave 1: ")

        self.slave1\_text.setStyleSheet("color: #ffffff; font-size: 20px;")

        self.slave1\_text.setGeometry(10, label\_top.height() + self.slave1\_button.height() + 40, self.slave1\_text.width(), self.slave1\_text.height())

        self.slave1\_error = QtWidgets.QLabel(self)

        self.slave1\_error.setText("")

        self.slave1\_error.setStyleSheet("color: #e80202; font-size: 18px;")

        self.slave1\_error.setGeometry(10, label\_top.height() + self.slave1\_button.height() + self.slave1\_text.height() + 40, self.slave1\_error.width(), self.slave1\_error.height())

        self.slave2\_text = QtWidgets.QLabel(self)

        self.slave2\_text.setText("Slave 2: ")

        self.slave2\_text.setStyleSheet("color: #ffffff; font-size: 20px;")

        self.slave2\_text.setGeometry(10, label\_top.height() + self.slave1\_button.height() + self.slave1\_text.height() + self.slave1\_error.height() + 40, self.slave2\_text.width(), self.slave2\_text.height())

        self.slave2\_error = QtWidgets.QLabel(self)

        self.slave2\_error.setText("")

        self.slave2\_error.setStyleSheet("color: #e80202; font-size: 18px;")

        self.slave2\_error.setGeometry(10, label\_top.height() + self.slave1\_button.height() + self.slave1\_text.height() + self.slave1\_error.height() + self.slave2\_text.height() + 40, self.slave2\_error.width(), self.slave2\_error.height())

        self.exception\_text = QtWidgets.QLabel(self)

        self.exception\_text.setText("")

        self.exception\_text.setStyleSheet("color: #e80202; font-size: 18px;")

        self.exception\_text.setGeometry(int((self.width() - self.exception\_text.width()) / 2), label\_top.height() + self.slave1\_button.height() + 30 + 70 + 40 + 30, self.exception\_text.width(), self.exception\_text.height())

    def eventFilter(self, obj, event):

        if (event.type() == QtCore.QEvent.Type.HoverEnter and obj is self.open\_button):

            self.open\_button.setStyleSheet("background-color: #000435; color: #FFFFFF; font-size: 18px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

            return True

        if (event.type() == QtCore.QEvent.Type.MouseButtonPress and obj is self.open\_button):

            self.exception\_text.setText("")

            self.open\_button.setStyleSheet("background-color: #000460; color: #FFFFFF; font-size: 18px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

            try:

                s = serial.Serial(port = self.dropdown.currentText(), baudrate = 4800, bytesize = 8, timeout = 3)

                if self.open\_button.text() == "Відкрити":

                    self.open\_button.setText("Закрити")

                else:

                    self.open\_button.setText("Відкрити")

            except serial.serialutil.SerialException:

                self.exception\_text.setText("Serial port access denied!")

                self.exception\_text.adjustSize()

                self.exception\_text.setGeometry(int((self.width() - self.exception\_text.width()) / 2), self.exception\_text.geometry().top(), self.exception\_text.width(), self.exception\_text.height())

            QtCore.QTimer.singleShot(200, lambda: self.on\_open\_click())

            return True

        if (event.type() == QtCore.QEvent.Type.HoverLeave and obj is self.open\_button):

            self.open\_button.setStyleSheet("color: #FFFFFF; font-size: 18px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

            return True

        if (event.type() == QtCore.QEvent.Type.HoverEnter and obj is self.slave1\_button):

            self.slave1\_button.setStyleSheet("background-color: #000435; color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

            return True

        if (event.type() == QtCore.QEvent.Type.MouseButtonPress and obj is self.slave1\_button):

            self.exception\_text.setText("")

            self.slave1\_button.setStyleSheet("background-color: #000460; color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

            try:

                s = serial.Serial(port = self.dropdown.currentText(), baudrate = 4800, bytesize = 8, timeout = 3)

                s.write(bytes([0x57]))

                res = s.read\_until(b"\0")

                self.slave1\_text.setText("Slave 1: " + res.decode("utf-8"))

                self.slave1\_text.adjustSize()

                if self.crc8(res) != self.crc8("Yatskiv Oleh Olehovych\0".encode("utf-8")):

                    self.slave1\_error.setText("Error: corrupted data!")

                    self.slave1\_error.adjustSize()

                else:

                    self.slave1\_error.setText("")

                    self.slave1\_error.adjustSize()

            except serial.serialutil.SerialException:

                self.exception\_text.setText("Serial port access denied!")

                self.exception\_text.adjustSize()

                self.exception\_text.setGeometry(int((self.width() - self.exception\_text.width()) / 2), self.exception\_text.geometry().top(), self.exception\_text.width(), self.exception\_text.height())

            QtCore.QTimer.singleShot(200, lambda: self.on\_timerslave1\_click())

            return True

        if (event.type() == QtCore.QEvent.Type.HoverLeave and obj is self.slave1\_button):

            self.slave1\_button.setStyleSheet("color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

            return True

        if (event.type() == QtCore.QEvent.Type.HoverEnter and obj is self.slave2\_button):

            self.slave2\_button.setStyleSheet("background-color: #000435; color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

            return True

        if (event.type() == QtCore.QEvent.Type.MouseButtonPress and obj is self.slave2\_button):

            self.exception\_text.setText("")

            self.slave2\_button.setStyleSheet("background-color: #000460; color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

            try:

                s = serial.Serial(port = self.dropdown.currentText(), baudrate = 4800, bytesize = 8, timeout = 3)

                s.write(bytes([0x4C]))

                res = s.read\_until(b"\0")

                self.slave2\_text.setText("Slave 2: " + res.decode('utf-8'))

                self.slave2\_text.adjustSize()

                if self.crc8(res) != self.crc8("11.09.2003\0".encode("utf-8")):

                    self.slave2\_error.setText("Error: corrupted data!")

                    self.slave2\_error.adjustSize()

                else:

                    self.slave2\_error.setText("")

                    self.slave2\_error.adjustSize()

            except serial.serialutil.SerialException:

                self.exception\_text.setText("Serial port access denied!")

                self.exception\_text.adjustSize()

                self.exception\_text.setGeometry(int((self.width() - self.exception\_text.width()) / 2), self.exception\_text.geometry().top(), self.exception\_text.width(), self.exception\_text.height())

            QtCore.QTimer.singleShot(200, lambda: self.on\_timerslave2\_click())

            return True

        if (event.type() == QtCore.QEvent.Type.HoverLeave and obj is self.slave2\_button):

            self.slave2\_button.setStyleSheet("color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

            return True

        return False

    def on\_open\_click(self):

        self.open\_button.setStyleSheet("background-color: #000435; color: #FFFFFF; font-size: 18px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

    def on\_timerslave1\_click(self):

        self.slave1\_button.setStyleSheet("background-color: #000435; color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

    def on\_timerslave2\_click(self):

        self.slave2\_button.setStyleSheet("background-color: #000435; color: #FFFFFF; font-size: 25px; border: 1px solid #FFFFFF; padding-bottom: 4px; border-radius: 10px;")

    def crc8(self, data):

        crc = 0x00

        for byte in data:

            extract = byte

            for \_ in range(8):

                sum = (crc ^ extract) & 0x01

                crc >>= 1

                if sum:

                    crc ^= 0x8C

                extract >>= 1

        return crc

def window() -> None:

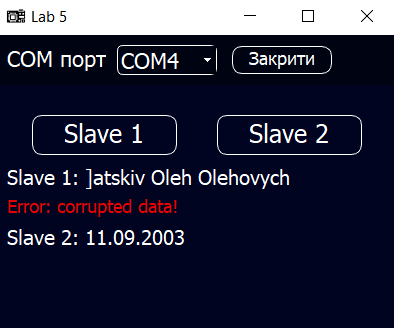
    app = QApplication(sys.argv)

    win = Application()

    win.show()

    sys.exit(app.exec\_())

window()



Демонстрація виконання завдання в клієнтській програмі

**Висновок:**

На цій лабораторній роботі я навчився працювати з інтерфейсом RS-485, організовувати комунікацію між мікроконтролерами, а також навчився імплементовувати та використувати CRC для перевірки цілісності даних.