**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**



Звіт до лабораторної роботи №2

з навчальної дисципліни: «Мікроконтролери ч.1»

**ПРОСТЕ КЕРУВАННЯ МІКРОКОНТРОЛЕРОМ ЗА**

**ДОПОМОГОЮ КЛІЄНТСЬКОЇ ПРОГРАМИ НА КОМП’ЮТЕРІ**

Варіант №11

**Виконав:**

ст. гр. ІР-22

Матвійчук А.І.

**Прийняв:**

Ст. викладач кафедри КСА

Лагун І.І.

Львів – 2021

**Порядок виконання роботи:**

1. Згідно варіанту завдання (таблиця 1) зібрати у пакеті симуляції

Proteus схему на основі МК ATmega2560 та написати програму

мовою C++ в Arduino IDE для реалізації вказаного завдання.

2. Розробити клієнтську програму для керування МК.

3. Протестувати роботу клієнтської програми зі схемою в Proteus.

4. Залити програму в Arduino-Kit.

5. Отримані результати представити викладачу.

**Завдання:**

1. Для зв’язку клієнтської програми з Proteus необхідно

встановити пару віртуальних COM-портів.

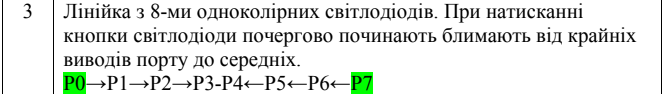
2. Запуск роботи алгоритмів зі світлодіодами виконується

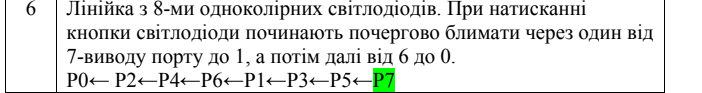
шляхом натиску відповідних кнопок на клієнтській формі.

3. При натисканні кнопок на МК запускаються відповідні

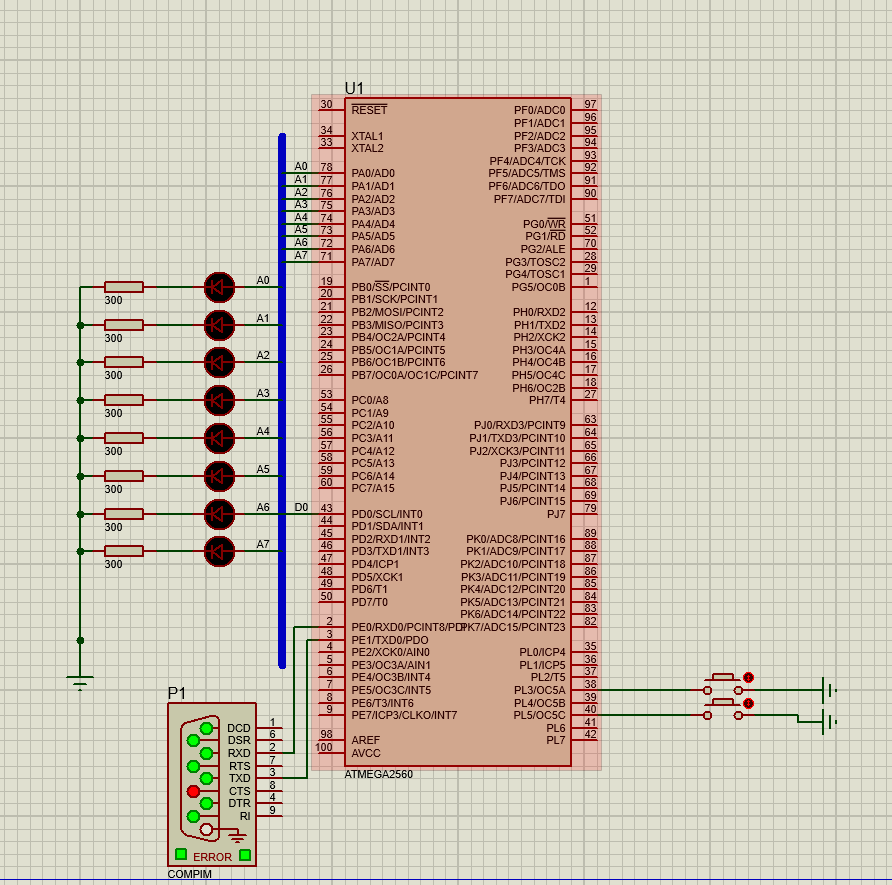
алгоритми блимання іконок на клієнтській формі.

****

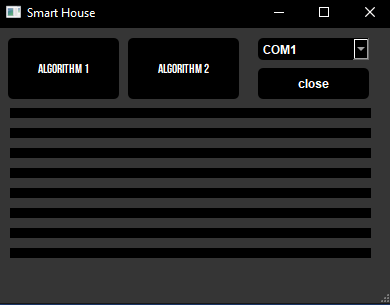
****

****

**Схема у Proteus:**



**Вікно GUI:**



**Відео презентація лабораторної роботи:**

<https://drive.google.com/file/d/1aOxl40-H9tdgdV4-8pGyNeF_sld_2fjT/view>

<https://drive.google.com/file/d/16GRfg6tfbZ3EEECyEJdAADAPCkUwfdG-/view>

<https://drive.google.com/file/d/10XN5R1Rf-TtRgkQpxATkq8cX_cC1HYbi/view>

**Код прошивки:**

const int buttonPinFirst = 46;

const int buttonPinSecond = 44;

const byte animationLed[4] = {

0b10000001,

0b01000010,

0b00100100,

0b00011000

};

void setup() {

DDRA = 0xFF;

PORTA=0;

pinMode(buttonPinFirst,INPUT\_PULLUP);

pinMode(buttonPinSecond,INPUT\_PULLUP);

Serial.begin(9600);

}

void loop() {

if(!digitalRead(buttonPinFirst)){

Serial.print(1);

// showLedsFirst();

}

if(!digitalRead(buttonPinSecond)){

Serial.print(2);

// showLedsSecond();

}

if (Serial.available()) {

char incomingByte = Serial.read();

int number = incomingByte;

if (number == 1) showLedsFirst();

else if(number == 2)showLedsSecond();

}

}

void showLedsFirst(){

for(int i = 0; i<4;i++){

PORTA = animationLed[i];

delay(900);

}

PORTA = 0;

}

void showLedsSecond(){

byte shiftLeds = 0b00000001;

for (int i = 0; i<8; i++){

if(shiftLeds == 0) shiftLeds = 0b00000010;

PORTA = shiftLeds;

shiftLeds = shiftLeds << 2;

delay(900);

}

PORTA = 0;

}

**Код GUI:**

Main.py

import sys  
from PyQt5.QtWidgets import QApplication, QTableWidgetItem  
from PyQt5 import uic,QtGui,QtCore  
from PyQt5.QtGui import \*  
from PyQt5.QtGui import QPalette, QColor  
from PyQt5.QtCore import QThread,QTimer  
from PyQt5 import QtWidgets  
import time  
import os  
import ui\_py  
import serial  
import datetime  
import sqlite3  
import comp  
  
  
from PyQt5 import QtTest  
  
ser = ''  
  
serial\_ports = []  
current\_serial\_port = ""  
  
  
  
ser = serial.Serial("COM29", 9600, bytesize=8, parity='N', stopbits=1, timeout=0)  
  
  
if getattr(sys, 'frozen', False):  
 os.chdir(sys.\_MEIPASS)  
  
class Sensors(QThread):  
  
 def \_\_init\_\_(self):  
 QThread.\_\_init\_\_(self)  
  
  
  
  
 def run(self):  
  
 while True:  
 pass  
  
  
  
  
  
  
  
  
class App(QtWidgets.QMainWindow, ui\_py.Ui\_MainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 self.sensor\_detect = Sensors()  
 self.sensor\_detect.start()  
  
 self.setupUi(self)  
 self.start()  
  
 self.timer = QTimer()  
 self.timer.timeout.connect(self.set\_info)  
 self.timer.start(0.1)  
  
 def set\_info(self):  
 if ser:  
 s = str(ser.readline())  
 if s.find("1") > 0:  
 ex.algorithm\_1(0)  
 elif s.find("2") > 0:  
 ex.algorithm\_2(0)  
  
 def start(self):  
  
 self.setWindowTitle("Smart House")  
 self.show()  
 self.comboBox.currentIndexChanged.connect(lambda: self.set\_combobox\_ser(self.comboBox.currentText()))  
  
 self.pushButton.clicked.connect(lambda: self.update\_serial\_ports())  
 self.pushButton\_2.clicked.connect(lambda: ser.write(str(chr(1)).encode('ascii')))  
 self.pushButton\_3.clicked.connect(lambda: ser.write(str(chr(2)).encode('ascii')))  
  
  
 def algorithm\_1(self,mode=1):  
 if ser:  
 for i in range(6):  
 arr\_led = [0] \* 8  
 if i != 5:  
 arr\_led[i] = 1  
 arr\_led[7 - i] = 1  
 self.show\_led(arr\_led)  
 QtTest.QTest.qWait(800 + 100\*mode)  
  
  
  
  
  
 def algorithm\_2(self,mode=1):  
 if ser:  
 for i in range(7,0,-2):  
 arr\_led = [0] \* 8  
 arr\_led[i]=1  
 self.show\_led(arr\_led)  
 QtTest.QTest.qWait(800 + 100\*mode)  
  
 QtTest.QTest.qWait(800 + 100\*mode)  
  
 for i in range(6,-1,-2):  
 arr\_led = [0] \* 8  
 arr\_led[i]=1  
 self.show\_led(arr\_led)  
 QtTest.QTest.qWait(800 + 100\*mode)  
  
 self.show\_led([0] \* 8)  
  
  
  
 def show\_led(self,arr):  
 for i in range(8):  
 self.set\_led(i,arr[i])  
  
  
 def set\_led(self,number,mode):  
 color = "black"  
 if mode ==1:  
 color = "red"  
  
 if number is 7:  
 self.line.setStyleSheet("background-color:"+ color +";")  
 elif number is 6:  
 self.line\_2.setStyleSheet("background-color:"+ color +";")  
 elif number is 5:  
 self.line\_3.setStyleSheet("background-color:" + color + ";")  
 elif number is 4:  
 self.line\_4.setStyleSheet("background-color:" + color + ";")  
 elif number is 3:  
 self.line\_5.setStyleSheet("background-color:" + color + ";")  
 elif number is 2:  
 self.line\_6.setStyleSheet("background-color:" + color + ";")  
 elif number is 1:  
 self.line\_7.setStyleSheet("background-color:" + color + ";")  
 elif number is 0:  
 self.line\_8.setStyleSheet("background-color:" + color + ";")  
  
  
  
  
 def set\_combobox\_ser(self,newText):  
  
 global current\_serial\_port  
 global ser  
 current\_serial\_port = newText  
  
 try:  
 pass  
 # ser = serial.Serial(current\_serial\_port, 9600)  
 except serial.SerialException:  
 print  
 "failed to write to port %s" % current\_serial\_port  
 sys.exit()  
  
 def update\_serial\_ports(self):  
 global serial\_ports  
 global current\_serial\_port  
 serial\_ports = comp.serial\_ports()  
 self.comboBox.clear()  
 for port in serial\_ports:  
 self.comboBox.addItem(port)  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
  
  
 app = QApplication(sys.argv)  
  
 app.setStyle("Fusion")  
  
 # Now use a palette to switch to dark colors:  
  
 palette = QPalette()  
 palette.setColor(QPalette.Window, QColor(53, 53, 53))  
  
 palette.setColor(QPalette.Base, QColor(25, 25, 25))  
 palette.setColor(QPalette.AlternateBase, QColor(53, 53, 53))  
 palette.setColor(QPalette.ToolTipBase, QColor(0, 0, 0))  
 palette.setColor(QPalette.ToolTipText, QColor(255, 255, 255))  
 palette.setColor(QPalette.Text, QColor(255, 255, 255))  
 palette.setColor(QPalette.Button, QColor(53, 53, 53))  
 palette.setColor(QPalette.ButtonText, QColor(255, 255, 255))  
 palette.setColor(QPalette.BrightText, QColor(255, 0, 0))  
 palette.setColor(QPalette.Link, QColor(42, 130, 218))  
 palette.setColor(QPalette.Highlight, QColor(100, 0, 0))  
 palette.setColor(QPalette.HighlightedText, QColor(0, 0, 0))  
 app.setPalette(palette)  
  
  
 ex = App()  
 ex.update\_serial\_ports()  
 app.exec\_()

ui.py

# -\*- coding: utf-8 -\*-  
  
# Form implementation generated from reading ui file 'ui.ui'  
#  
# Created by: PyQt5 UI code generator 5.13.2  
#  
# WARNING! All changes made in this file will be lost!  
  
  
from PyQt5 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(395, 275)  
 font = QtGui.QFont()  
 font.setPointSize(9)  
 MainWindow.setFont(font)  
 self.centralwidget = QtWidgets.QWidget(MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.comboBox = QtWidgets.QComboBox(self.centralwidget)  
 self.comboBox.setGeometry(QtCore.QRect(260, 10, 111, 22))  
 font = QtGui.QFont()  
 font.setFamily("Adderley")  
 font.setPointSize(9)  
 font.setBold(True)  
 font.setWeight(75)  
 self.comboBox.setFont(font)  
 self.comboBox.setStyleSheet("#comboBox{\n"  
"background:black;\n"  
"color:white;\n"  
"border-radius:5px;\n"  
"}")  
 self.comboBox.setObjectName("comboBox")  
 self.pushButton = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton.setGeometry(QtCore.QRect(260, 40, 111, 31))  
 font = QtGui.QFont()  
 font.setFamily("Adderley")  
 font.setPointSize(9)  
 font.setBold(True)  
 font.setWeight(75)  
 self.pushButton.setFont(font)  
 self.pushButton.setStyleSheet("#pushButton{\n"  
"background:black;\n"  
"color:white;\n"  
"border-radius:5px;\n"  
"}")  
 self.pushButton.setObjectName("pushButton")  
 self.pushButton\_2 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_2.setGeometry(QtCore.QRect(10, 10, 111, 61))  
 font = QtGui.QFont()  
 font.setFamily("Adderley")  
 font.setPointSize(10)  
 font.setBold(True)  
 font.setWeight(75)  
 self.pushButton\_2.setFont(font)  
 self.pushButton\_2.setStyleSheet("#pushButton\_2{\n"  
"background:black;\n"  
"color:white;\n"  
"border-radius:5px;\n"  
"}")  
 self.pushButton\_2.setObjectName("pushButton\_2")  
 self.pushButton\_3 = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton\_3.setGeometry(QtCore.QRect(130, 10, 111, 61))  
 font = QtGui.QFont()  
 font.setFamily("Adderley")  
 font.setPointSize(10)  
 font.setBold(True)  
 font.setWeight(75)  
 self.pushButton\_3.setFont(font)  
 self.pushButton\_3.setStyleSheet("#pushButton\_3{\n"  
"background:black;\n"  
"color:white;\n"  
"border-radius:5px;\n"  
"}")  
 self.pushButton\_3.setObjectName("pushButton\_3")  
 self.line = QtWidgets.QFrame(self.centralwidget)  
 self.line.setGeometry(QtCore.QRect(12, 80, 361, 10))  
 self.line.setAutoFillBackground(False)  
 self.line.setStyleSheet("#line{\n"  
"border:0;\n"  
"background:black;\n"  
"}")  
 self.line.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line.setObjectName("line")  
 self.line\_2 = QtWidgets.QFrame(self.centralwidget)  
 self.line\_2.setGeometry(QtCore.QRect(12, 100, 361, 10))  
 self.line\_2.setAutoFillBackground(False)  
 self.line\_2.setStyleSheet("#line\_2{\n"  
"border:0;\n"  
"background:black;\n"  
"}")  
 self.line\_2.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line\_2.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line\_2.setObjectName("line\_2")  
 self.line\_3 = QtWidgets.QFrame(self.centralwidget)  
 self.line\_3.setGeometry(QtCore.QRect(12, 120, 361, 10))  
 self.line\_3.setAutoFillBackground(False)  
 self.line\_3.setStyleSheet("#line\_3{\n"  
"border:0;\n"  
"background:black;\n"  
"}")  
 self.line\_3.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line\_3.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line\_3.setObjectName("line\_3")  
 self.line\_4 = QtWidgets.QFrame(self.centralwidget)  
 self.line\_4.setGeometry(QtCore.QRect(12, 140, 361, 10))  
 self.line\_4.setAutoFillBackground(False)  
 self.line\_4.setStyleSheet("#line\_4{\n"  
"border:0;\n"  
"background:black;\n"  
"}")  
 self.line\_4.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line\_4.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line\_4.setObjectName("line\_4")  
 self.line\_5 = QtWidgets.QFrame(self.centralwidget)  
 self.line\_5.setGeometry(QtCore.QRect(12, 160, 361, 10))  
 self.line\_5.setAutoFillBackground(False)  
 self.line\_5.setStyleSheet("#line\_5{\n"  
"border:0;\n"  
"background:black;\n"  
"}")  
 self.line\_5.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line\_5.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line\_5.setObjectName("line\_5")  
 self.line\_6 = QtWidgets.QFrame(self.centralwidget)  
 self.line\_6.setGeometry(QtCore.QRect(12, 180, 361, 10))  
 self.line\_6.setAutoFillBackground(False)  
 self.line\_6.setStyleSheet("#line\_6{\n"  
"border:0;\n"  
"background:black;\n"  
"}")  
 self.line\_6.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line\_6.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line\_6.setObjectName("line\_6")  
 self.line\_7 = QtWidgets.QFrame(self.centralwidget)  
 self.line\_7.setGeometry(QtCore.QRect(12, 200, 361, 10))  
 self.line\_7.setAutoFillBackground(False)  
 self.line\_7.setStyleSheet("#line\_7{\n"  
"border:0;\n"  
"background:black;\n"  
"}")  
 self.line\_7.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line\_7.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line\_7.setObjectName("line\_7")  
 self.line\_8 = QtWidgets.QFrame(self.centralwidget)  
 self.line\_8.setGeometry(QtCore.QRect(12, 220, 361, 10))  
 self.line\_8.setAutoFillBackground(False)  
 self.line\_8.setStyleSheet("#line\_8{\n"  
"border:0;\n"  
"background:black;\n"  
"}")  
 self.line\_8.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line\_8.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line\_8.setObjectName("line\_8")  
 MainWindow.setCentralWidget(self.centralwidget)  
 self.menubar = QtWidgets.QMenuBar(MainWindow)  
 self.menubar.setGeometry(QtCore.QRect(0, 0, 395, 21))  
 self.menubar.setObjectName("menubar")  
 MainWindow.setMenuBar(self.menubar)  
 self.statusbar = QtWidgets.QStatusBar(MainWindow)  
 self.statusbar.setObjectName("statusbar")  
 MainWindow.setStatusBar(self.statusbar)  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))  
 self.pushButton.setText(\_translate("MainWindow", "close"))  
 self.pushButton\_2.setText(\_translate("MainWindow", "ALGORITHM 1"))  
 self.pushButton\_3.setText(\_translate("MainWindow", "ALGORITHM 2"))

**Висновок:**

В даній лабораторній роботі я навчився розробиляти клієнтську програму для керування МК. Також навчився як з’єднанувати мікроконтролер з персональним комп’ютером через comport