ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

10. Parent application - virtual serial port

Lead Partner: Warsaw University of Technology

Authors: Daniel Krol

University of Applied Sciences in Tarnow

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Declaration

This report has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

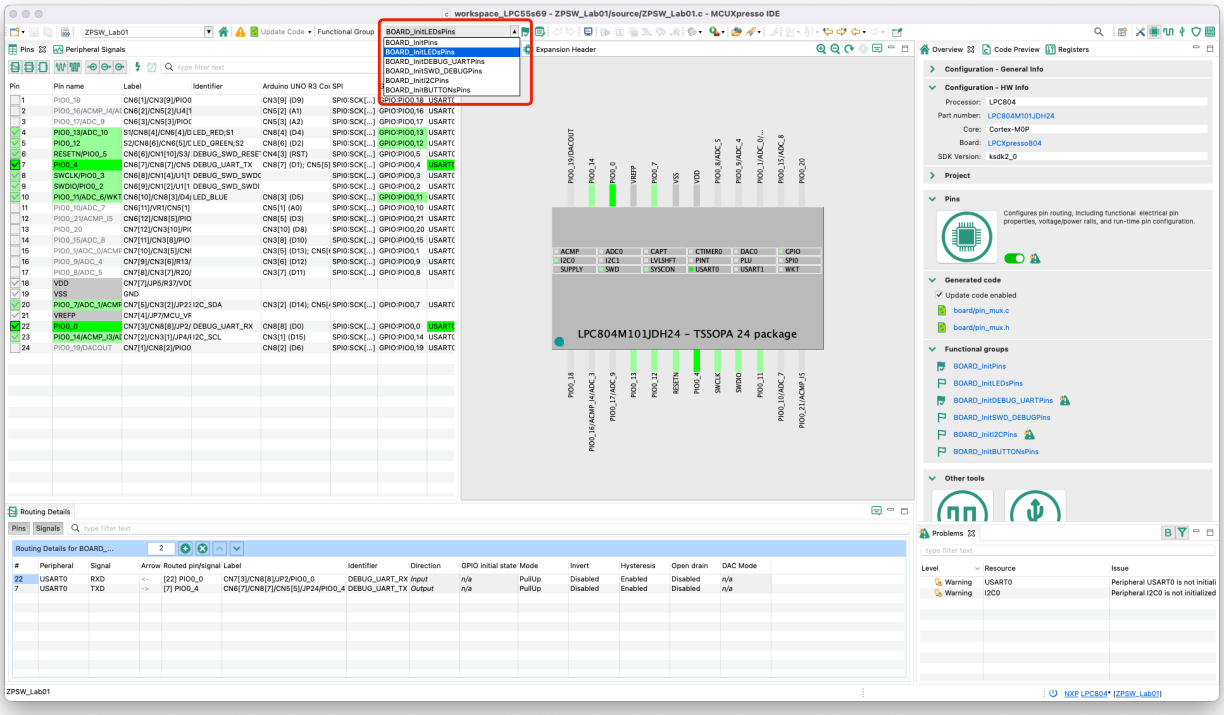
This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Programing of embedded systems

10. Parent application - virtual serial port

I. Sterownik LED

1. Należy skonfigurować 3 linie *GPIO* do sterowania poszczególnymi diodami *RGB*, tak jak na pierwszych zajęciach. W tym celu kliknij prawym przyciskiem na nazwie projektu i wybierz *MCUXpresso Config Tools -> Open Pins*. Z menu *Functional Group* wybierz preset *BOARD_InitLEDsPins*, a następnie aktywuj go zaznaczając ikonę flagi po lewej stronie:



2. Wybierz *Update Code* i zaakceptuj zmiany przyciskiem *OK*.
3. Zmodyfikuj kod w funkcji *main*, tak aby odebranie odpowiedniego znaku odpowiadało sterowaniu poszczególnymi diodami LED:
   - *a: Red-On*
   - *z: Red-Off*
   - *s: Green-On*
   - *x: Green-Off*
   - *d: Blue-On*
   - *c: Blue-Off*

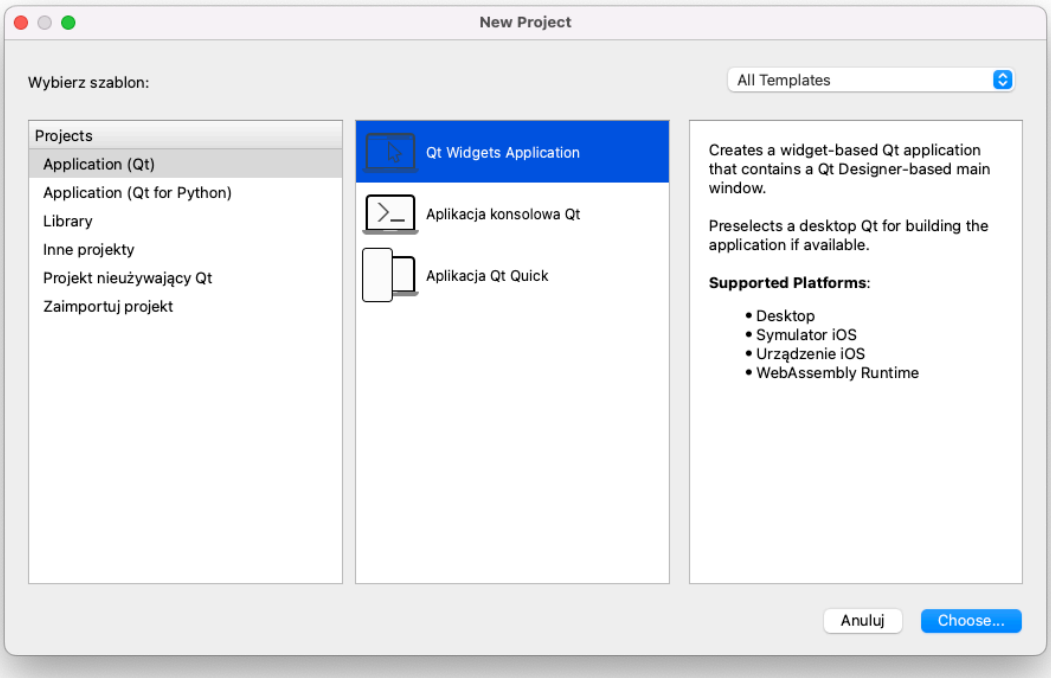
```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif
```

```
        PRINTF("LPC804 Start...\r\n");

        char c;

        while(1) {

                c=GETCHAR();

                if( c== 'a') {
                        // On
                        GPIO_PinWrite(BOARD_INITLEDSPINS_LED_RED_GPIO,
                                      BOARD_INITLEDSPINS_LED_RED_PORT,
                                      BOARD_INITLEDSPINS_LED_RED_PIN,
                                      0);

                }
                if(c == 'z') {
                        // Off
                        GPIO_PinWrite(BOARD_INITLEDSPINS_LED_RED_GPIO,
                                      BOARD_INITLEDSPINS_LED_RED_PORT,
                                      BOARD_INITLEDSPINS_LED_RED_PIN,
                                      1);
                }
                if( c== 's') {
                        // On
                        GPIO_PinWrite(BOARD_INITLEDSPINS_LED_GREEN_GPIO,
                                      BOARD_INITLEDSPINS_LED_GREEN_PORT,
                                      BOARD_INITLEDSPINS_LED_GREEN_PIN,
                                      0);

                }
                if(c == 'x') {
                        // Off
                        GPIO_PinWrite(BOARD_INITLEDSPINS_LED_GREEN_GPIO,
                                      BOARD_INITLEDSPINS_LED_GREEN_PORT,
                                      BOARD_INITLEDSPINS_LED_GREEN_PIN,
                                      1);
                }
                if( c== 'd') {
                        // On
                        GPIO_PinWrite(BOARD_INITLEDSPINS_LED_BLUE_GPIO,
                                      BOARD_INITLEDSPINS_LED_BLUE_PORT,
                                      BOARD_INITLEDSPINS_LED_BLUE_PIN,
                                      0);

                }
                if(c == 'c') {
                        // Off
                        GPIO_PinWrite(BOARD_INITLEDSPINS_LED_BLUE_GPIO,
                                      BOARD_INITLEDSPINS_LED_BLUE_PORT,
                                      BOARD_INITLEDSPINS_LED_BLUE_PIN,
                                      1);
                }
        }

        return 0 ;
}
```

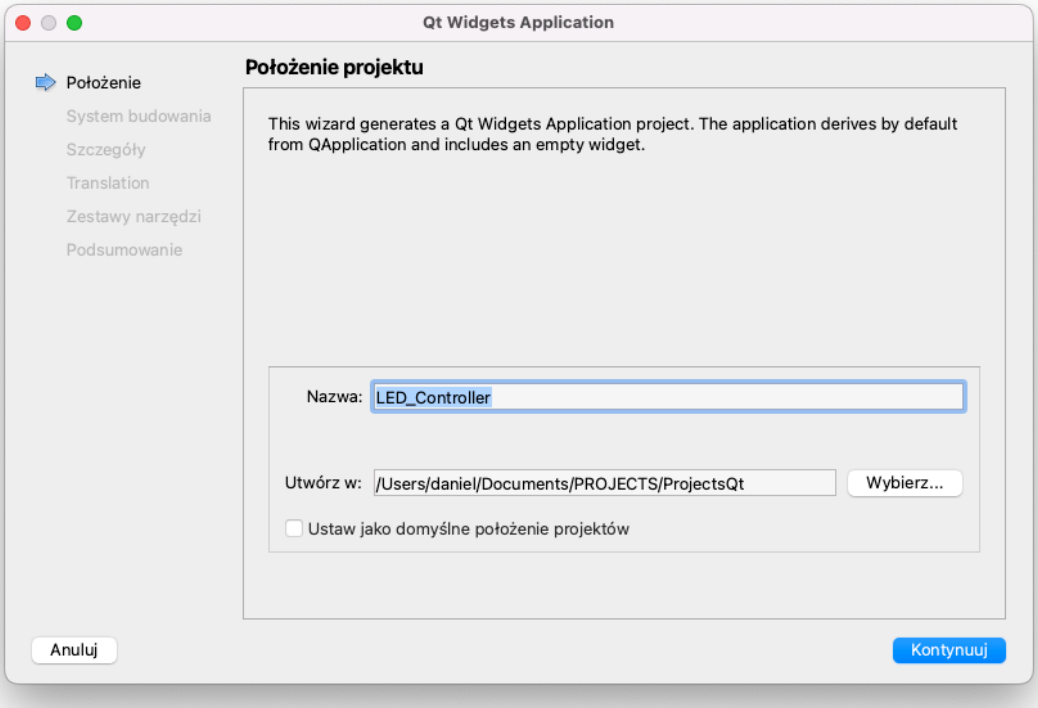
Zbuduj projekt i zaprogramuj układ.

II. Aplikacja nadrzędna

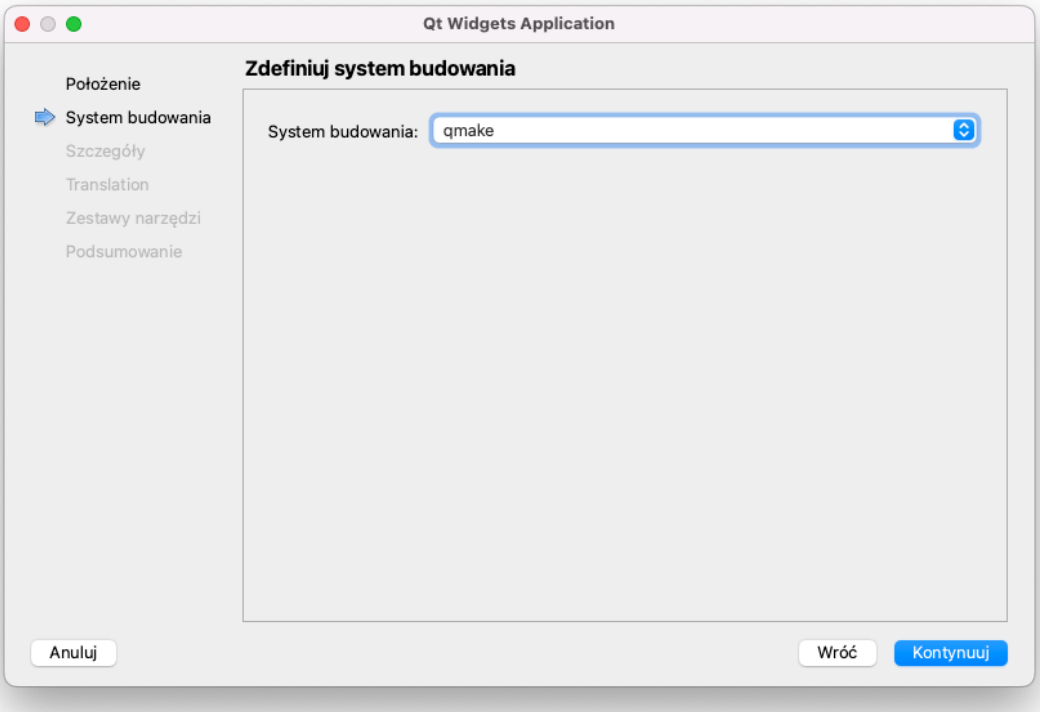
1. Uruchom środowisko *Qt Creator* i utwórz nowy projekt *Qt Widgets Application*:



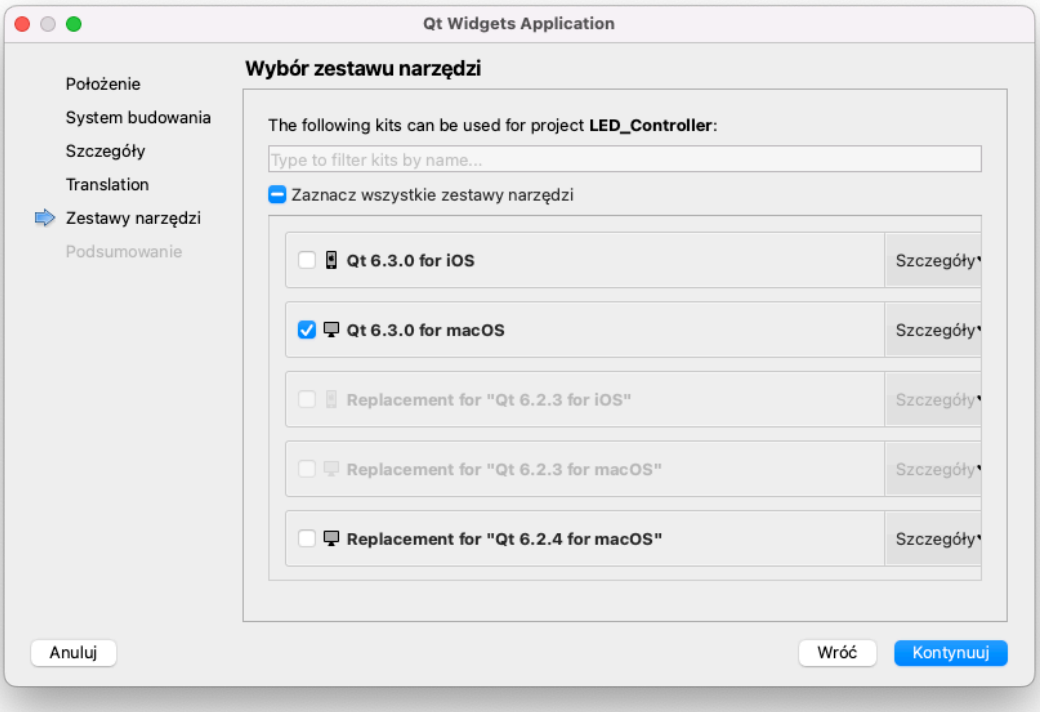
2. Nazwij go *LED_Controller*:



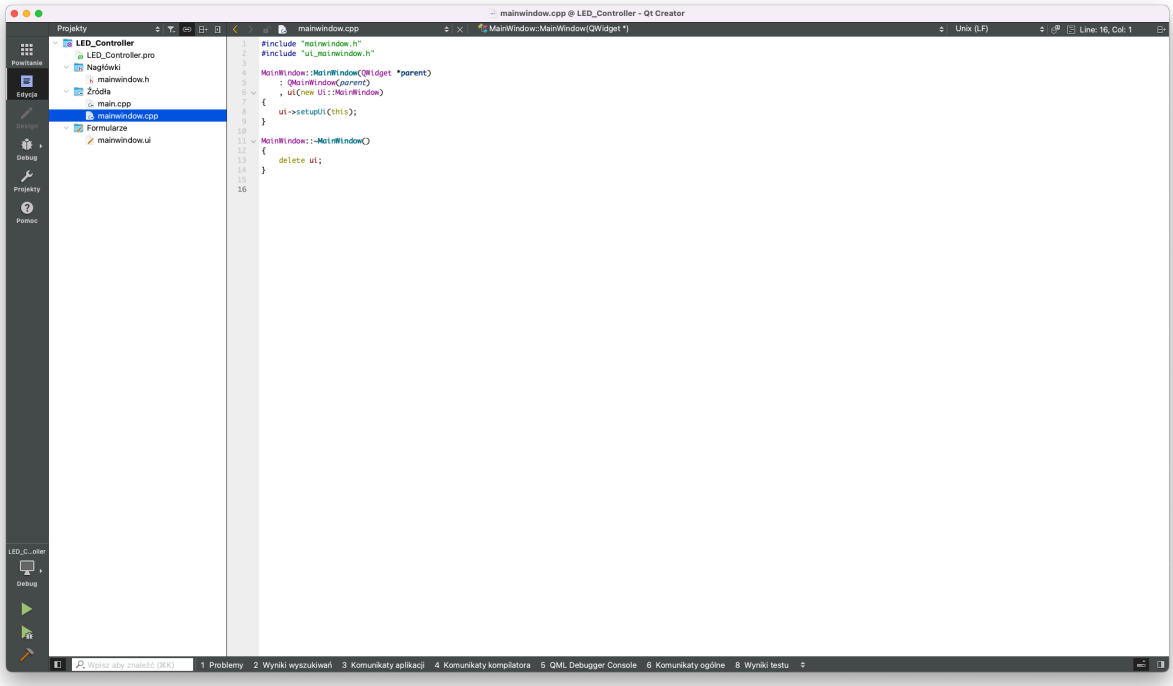
3. Jako system budowania wybierz *qmake*:



4. W kolejnych oknach pozostaw domyślne ustawienia.
5. W oknie wyboru zestawu narzędzi wybierz *Qt 6.3 for macOS* (*MinGW* na Windows):



6. Widok struktury wygenerowanego projektu:



7. W pliku projektu *LED_Controller.pro* dodaj bibliotekę *serialport*:

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets serialport

CONFIG += c++17

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

8. Przejdź do plik *mainwindow.h* i zmodyfikuj kod:

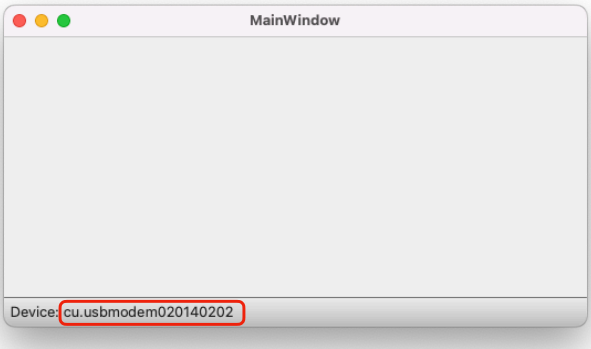
```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private:
```

```
    Ui::MainWindow *ui;
    QSerialPort serial;
    QByteArray cmdData;
    QByteArray rawData;

private slots:
    void readData();
    void error(QSerialPort::SerialPortError error);

};
#endif // MAINWINDOW_H
```

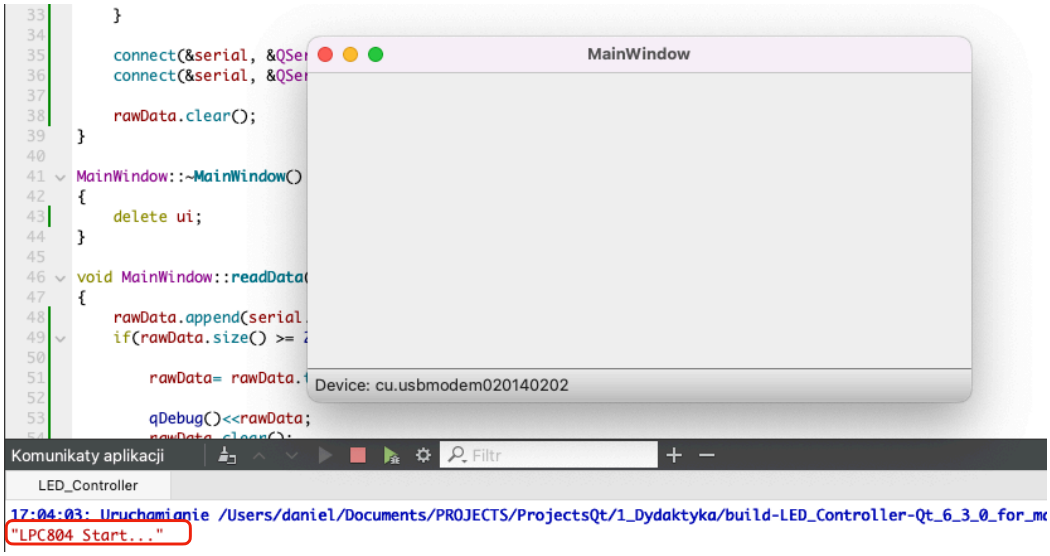
9. Przejdź do plik *mainwindow.cpp* i zmodyfikuj kod:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    ui->statusbar->showMessage("No device");
    QString portname;
    const auto infos = QSerialPortInfo::availablePorts();
    for (const QSerialPortInfo &info : infos) {

        // if(info.portName()=="cu.usbmodem020140202") { // OS Windows e.g "COM1"
        // if(info.description() =="LPC11U3x CMSIS-DAP v1.0.4") {
        if(info.manufacturer() =="NXP Semiconductors") {

            portname=info.portName();
            serial.setPortName(portname);
            if (serial.open(QIODevice::ReadWrite)) {

                ui->statusbar->showMessage(tr("Device: %1").arg(info.portName()));
                serial.setBaudRate( serial.Baud9600,  serial.AllDirections);
                serial.clear();
            } else {

                ui->statusbar->showMessage(tr("Can't open %1, error code
%2") .arg(serial.portName()).arg(serial.error()));
                return;
            }
            break;
        }
    }

    connect(&serial, &QSerialPort::readyRead, this, &MainWindow::readData);
    connect(&serial, &QSerialPort::errorOccurred, this, &MainWindow::error);

    rawData.clear();
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::readData()
{
    rawData.append(serial.readAll());
    if(rawData.size() >= 2 && (rawData.last(2)) == "\r\n") {

        rawData= rawData.trimmed();  // delete \r\n

        qDebug()<<rawData;
        rawData.clear();
    }
}

void MainWindow::error(QSerialPort::SerialPortError error)
{
    qDebug()<<error;
    qDebug()<<"---------";
}
```

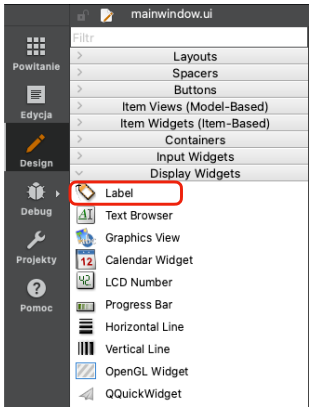
10. Podłącz płytkę z mikrokontrolerem do portu USB.
11. Zbuduj projekt i uruchom aplikację. Na pasku statusu powinien pojawić się tekst z nazwą wirtualnego portu szeregowego:

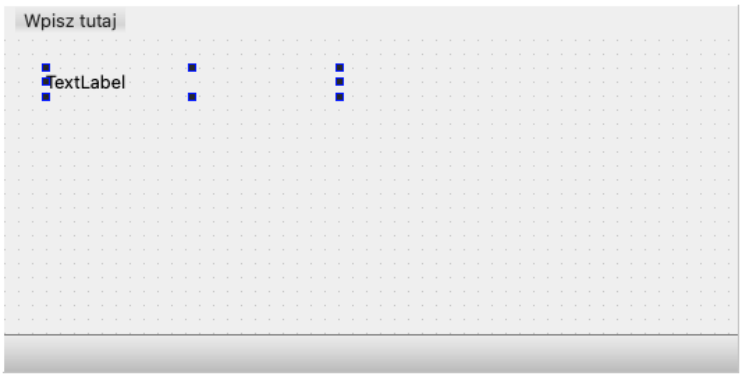


12. Wciśnij reset na płytce mikrokontrolera. W oknie *Komunikaty aplikacji*, w *Qt Creator* powinien pojawić się tekst wysłany przez mikrokontroler:



13. Zamknij aplikację.
14. Przejdź do *Formularze* -> *mainwindow.ui* i wstaw (przeciągnij) na formatkę *label*:



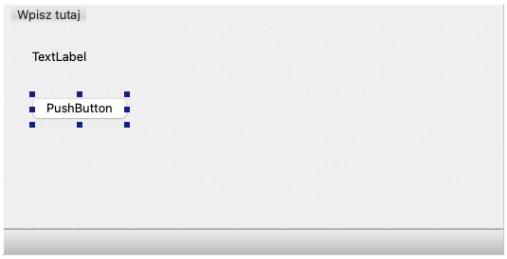


15. Przejdź do metody *readData* w pliku *mainwindow.cpp* i dodaj kod:

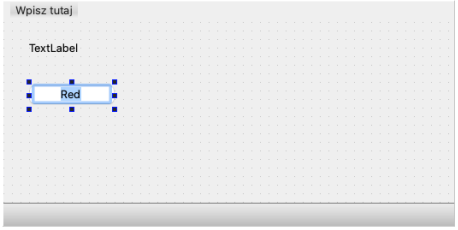
```
void MainWindow::readData()
{
    rawData.append(serial.readAll());
    if(rawData.size() >= 2 && (rawData.last(2)) == "\r\n") {

        rawData= rawData.trimmed();  // delete \r\n

        ui->label->setText(rawData);
        qDebug()<<rawData;
        rawData.clear();
    }
}
```

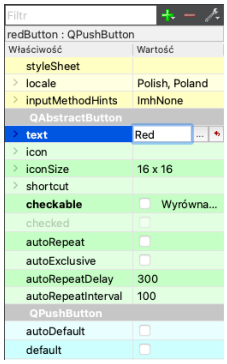
16. Zbuduj i uruchom aplikację a następnie wciśnij reset na płytce z mikrokontrolerem. Odebrany tekst powinien wyświetlić się na widżecie *label*:

17. Zamknij aplikację, przejdź do *Formularze -> mainwindow.ui* i wstaw na formatkę przycisk *PushButton*:

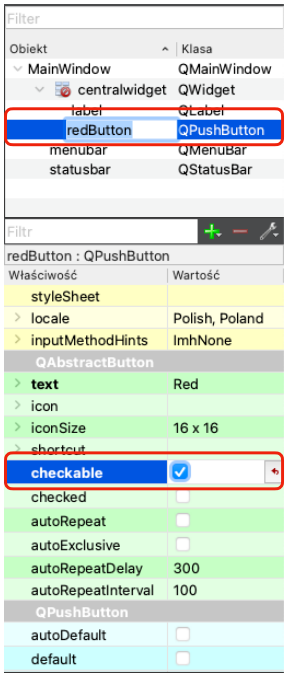


18. Zmień etykietę na *Red* przez podwójne kliknięcie lub we właściwościach, w kolumnie po prawej stronie okna *Qt Creator*:

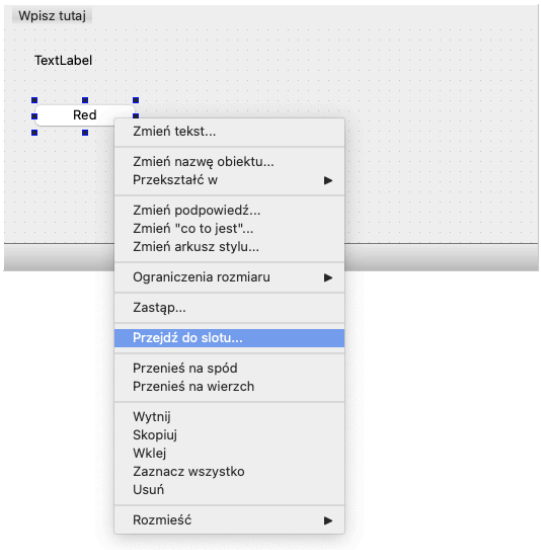


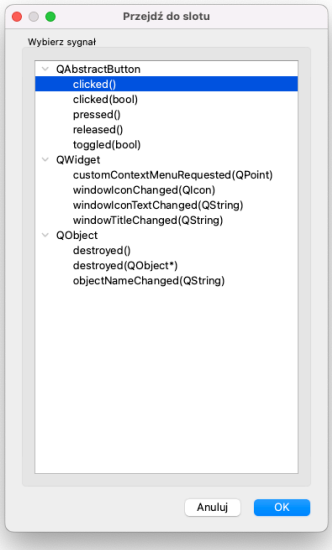


19. Zmień nazwę obiektu *pushButton* na *redButton* oraz ustaw właściwość *checkable* na *true* we właściwościach obiektu po prawej stronie okna *Qt Creator*:



20. Klikając prawym przyciskiem, z menu kontekstowego, wybierz *Przejdź do slotu…* Następnie wybierz sygnał *clicked*:





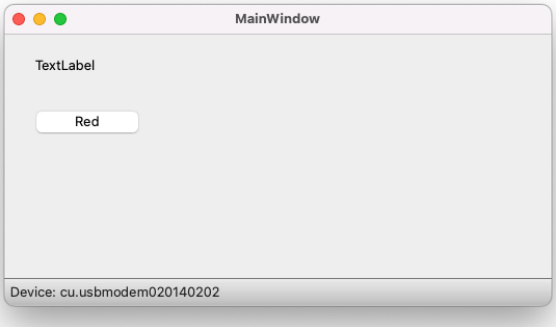
21. W pliku *mainwindow.cpp* pojawi się slot (definicja w pliku *mainwindow.h*) *on_redButton_clicked*:

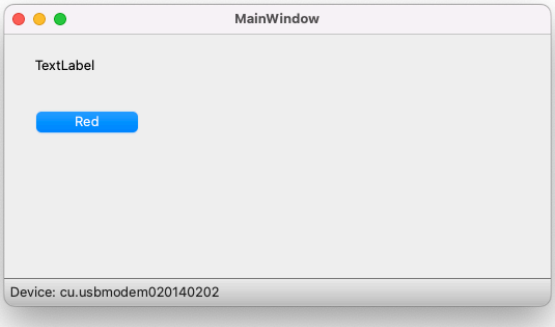
```
void MainWindow::on_redButton_clicked()
{

}
```

22. Dodaj kod wysyłający dane do mikrokontrolera:

```
void MainWindow::on_redButton_clicked()
{
    cmdData.clear();
    cmdData.append(ui->redButton->isChecked() ? 'a' : 'z');
    serial.write(cmdData);
}
```

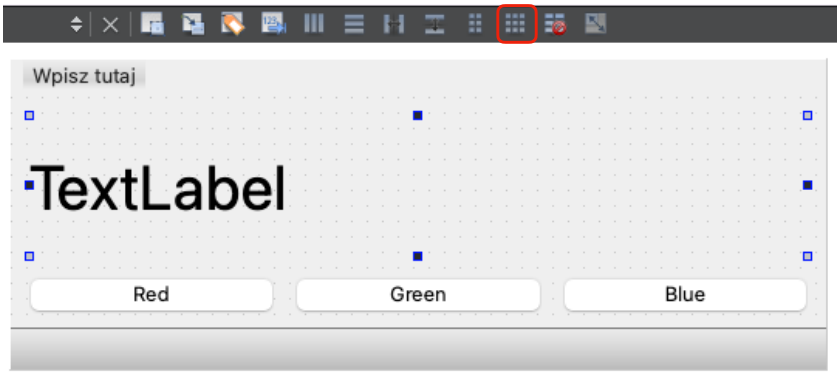
23. Zbuduj i uruchom aplikację a następnie wciśnij przycisk *Red*. Czerwona dioda LED powinna świecić gdy przycisk jest wciśnięty i gasnąć gdy jest wyciśnięty:

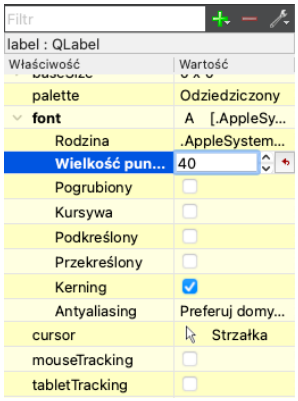




III. Zadania

1. Dodaj dodatkowe przyciski do sterowania zieloną i niebieską diodą LED.
2. Rozmieść widżety w siatce formatki i zwiększ rozmiar czcionki w widżecie *label* na 40:





3. Dodaj wysyłanie informacji z mikrokontrolera o zapaleniu lub zgaszeniu poszczególnych diod LED i wyświetlanie ich na widżecie *label:*