Dokumentacja projektu c++

Inteligenty dom

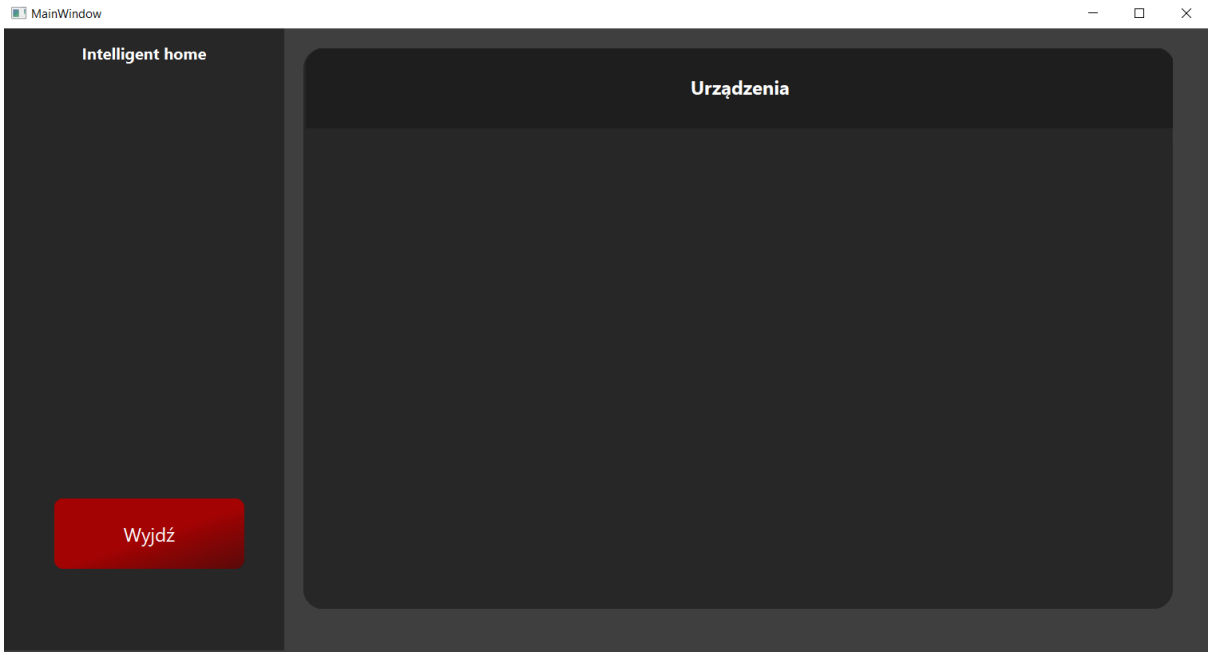
Bartłomiej Mazurkiewicz

2EF-DI, proj. 05, 173670

Spis treści

Przedstawienie programu

Po uruchomieniu programu wyświetla się utworzone okno głównej aplikacji. Jest to miejsce, gdzie dodawane będą nowo podłączone, jak i podłączone już urządzenia.

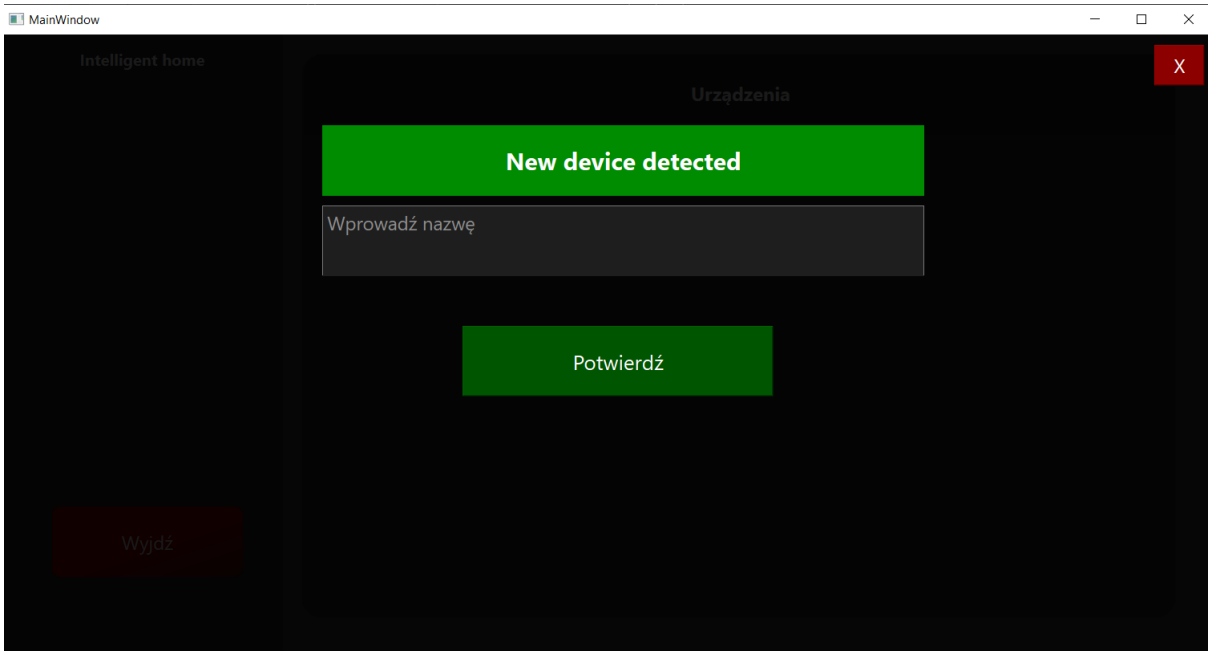


Rysunek 1 Aplikacja główna

Z lewej strony znajduje się panel boczny, w przypadku późniejszej rozbudowy aplikacji i podziale na różne pokoje, a także przycisk do wyjścia z aplikacji. W centralnej części znajduje się miejsce na kontrolery podłączonych urządzeń, które będą dodawane w trakcie uruchamiania klientów.

Podłączenie urządzeń

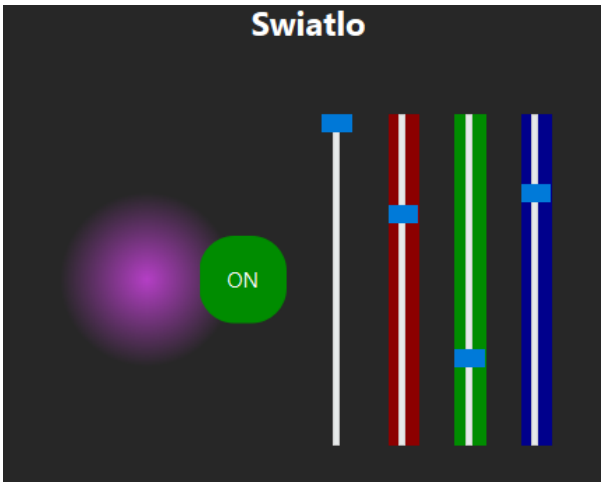
Po uruchomieniu klienta, czyli urządzenia, nasza aplikacja prosi nas o podanie nazwy nowego urządzenia.



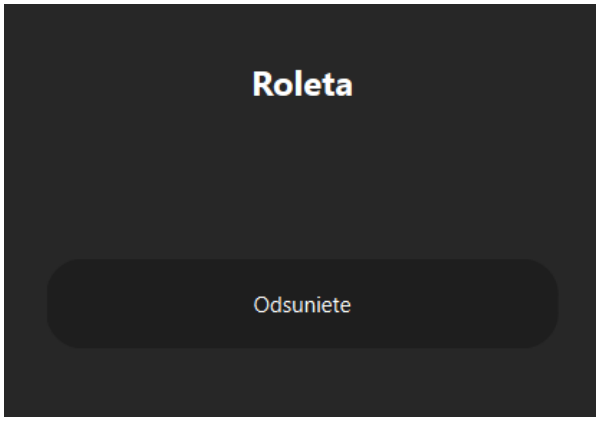
Rysunek 2 Podanie nazwy nowego urządzenia

Nazwa musi być unikalna, dlatego podanie używanej już nazwy jest niemożliwe. Po podaniu nazwy wyświetlony zostaje kontroler w panelu głównym, w zależności od rodzaju urządzenia: światła, temperatury, rolety.

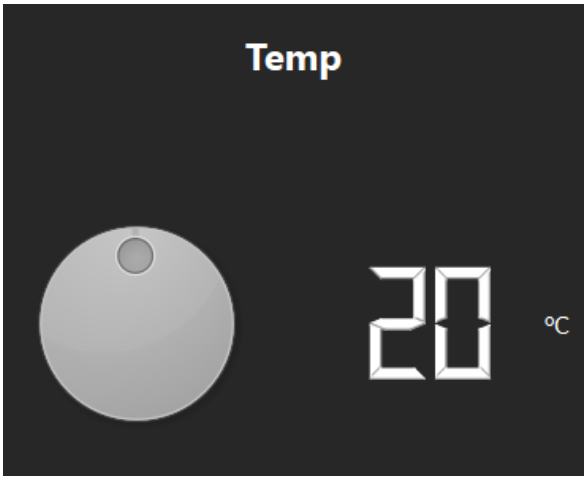
Urządzenia dodawane są jedno pod drugim, co umożliwia przejrzysty widok w aplikacji.



Rysunek 3 Kontroler światła



Rysunek 4 Kontroler rolety



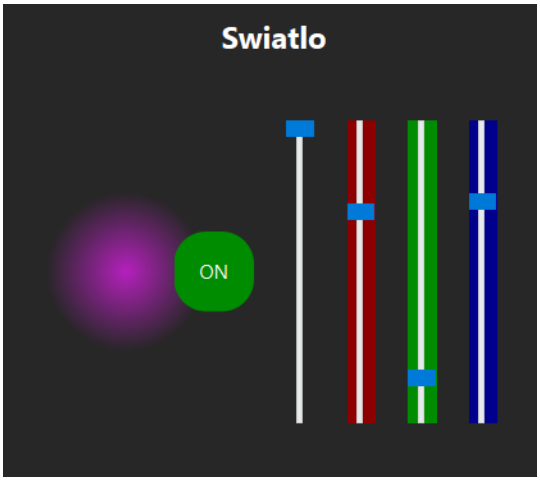
Rysunek 5 Kontroler temperatury

Symulacja urządzeń

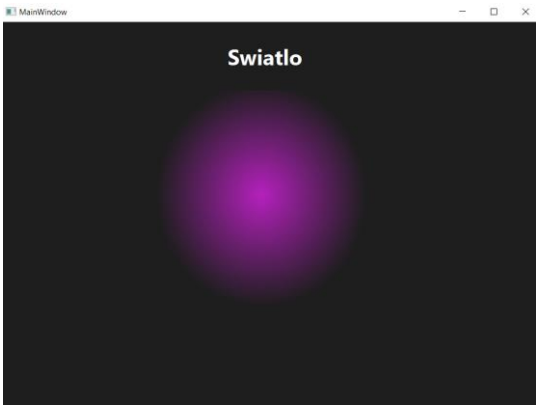
Każde z urządzeń komunikuje się z serwerem na tym samym porcie, a wiadomości wysyłane są broadcastem, czyli do wszystkich urządzeń. Serwer wysyłając wiadomość o zmianie wartości posługuje się za pomocą nazwy urządzenia, a nie jego typem, dlatego po podłączeniu kilku tych samych urządzeń i zmianie wartości w jednym z nich nie wpłynie na pozostałe.

Światło

Zmieniając wartości w oknie naszej aplikacji możemy podejrzeć w oknie symulacji klienta zmianę, jaka się wykonuje. W przypadku światła możliwa jest zmiana wartości intensywności, zmiana wartości światła czerwonego, zielonego i niebieskiego, oraz włączenie / wyłączenie urządzenia.



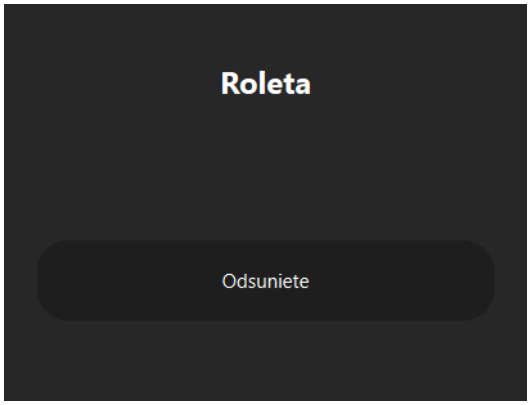
Rysunek 6 Zmiana wartości na serwerze



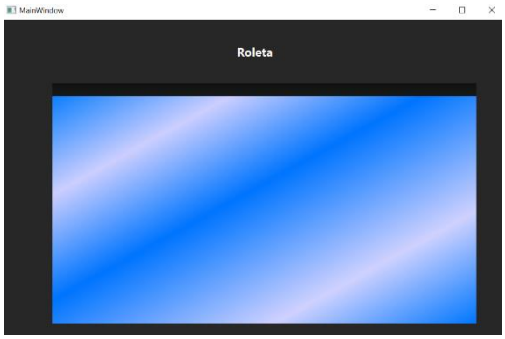
Rysunek 7 Symulacja urządzenia

Roleta

W przypadku rolety uruchamiana jest symulacja zasłaniania i odsłaniania się rolety. Na serwerze dostępna jest opcja do wykonania tej czynności.

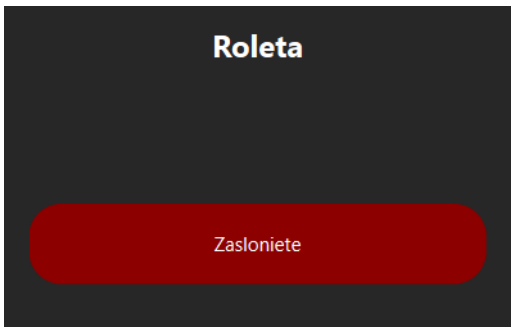


Rysunek 8 Stan początkowy rolety na serwerze

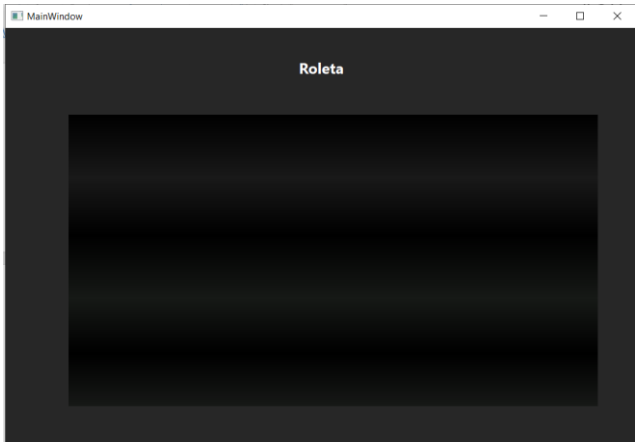


Rysunek 9 Stan początkowy rolety w symulacji

Po naciśnięciu przycisku na serwerze zostanie uruchomiona symulacja zasuwania lub odsuwania rolety, a przycisk zmieni swój napis i kolor tła.



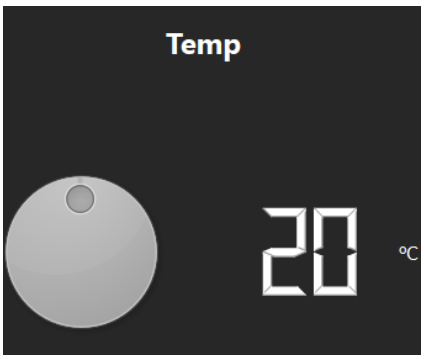
Rysunek 10 Zasunięta roleta



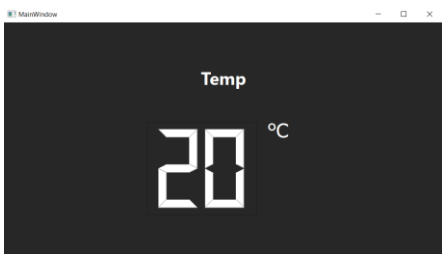
Rysunek 11 Zasunięta roleta w symulacji

Temperatura

W przypadku temperatury wyświetlane są ustawione stopnie Celsjusza, które są ustawiane za pomocą pokrętła.



Rysunek 12 Temperatura na serwerze



Rysunek 13 Temperatura w symulacji

Kod aplikacji

Serwer

W trakcie uruchamiania okna aplikacji tworzony jest server TCP na porcie 12345 oraz adresie w pętli zwrotnej 127.0.0.1. Połączone zostają sygnały do obsługi wydarzeń, takich jak: połączenie nowego urządzenia, odebrania wiadomości, odłączenie urządzenia, wysłanie nazwy do nowego urządzenia.

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    _server = new MyTCPServer();
    connect(_server, &MyTCPServer::newDeviceConnected, this, &MainWindow::newDeviceConnected);
    connect(_server, &MyTCPServer::dataReceived, this, &MainWindow::deviceDataReceived);
    connect(_server, &MyTCPServer::deviceDisconnect, this, &MainWindow::deviceDisconnected);
    connect(this, &MainWindow::deviceSubmitted, _server, &MyTCPServer::sendNameToDevice);

    gridLayout = new QGridLayout(ui->scrollAreaWidgetContents);
    ui->add_new_device->hide();
    ui->invalid_name->hide();
}
```

Rysunek 14 Główne okno aplikacji

```
MyTCPServer::MyTCPServer(QObject *parent)
    :QObject{parent}{

    _server = new QTcpServer(this);
    connect(_server, &QTcpServer::newConnection, this, &MyTCPServer::on_device_connecting);
    _isStarted = _server->listen(QHostAddress("127.0.0.1"), 12345);
    if(!_isStarted){
        qDebug() << "Server could not start";
    }else{
        qDebug() << "Server started";
    }
}
```

Rysunek 15 Serwer TCP

W przypadku połączenia nowego urządzenia zostaje dodany socket do listy i zostaje utworzone nowe połączenie. Tworzone jest połączenie do przypadku odłączenia urządzenia oraz odebrania wiadomości od urządzenia.

```
void MyTCPServer::on_device_connecting(){
    qDebug()<<"New device detected";
    auto socket = _server->nextPendingConnection();
    _socketsList.append(socket);
    socket->write("Connected");
    connect(socket, &QTcpSocket::readyRead, this, &MyTCPServer::deviceDataReady);
    connect(socket, &QTcpSocket::disconnected, this, &MyTCPServer::deviceDisconnected);
}

void MyTCPServer::deviceDisconnected()
{
    emit deviceDisconnect(device_name);
}

void MyTCPServer::deviceDataReady()
{
    auto socket = qobject_cast<QTcpSocket *>(sender());
    QString data = QString(socket->readAll());
    emit dataReceived(data);
}
```

Rysunek 16 Obsługa urządzenia

Serwer jest w stanie wysłać wiadomość broadcastem za pomocą funkcji sendToAll, oraz wysyła wiadomość do nowo podłączonego urządzenia za pomocą komendy write.

```
void MyTCPServer::sendToAll(QString message)
{
    foreach (auto socket, _socketsList) {
        socket->write(message.toUtf8());
    }
}

void MyTCPServer::sendNameToDevice(QString deviceName){
    QString message = QString("name=%1").arg(deviceName);
    device_name = deviceName;
    _socketsList.last()->write(message.toUtf8());
    _socketsList.last()->setObjectName(deviceName);
}
```

Rysunek 17 Wysyłanie wiadomości od serwera

W głównym oknie aplikacji dodawany jest kontroler w zależności od typu urządzenia, wykorzystując utworzone klasy do tworzenia widgetów.

```
if(device_type == "Temperatura"){
    TemperatureDeviceWidget *temperature = new TemperatureDeviceWidget(_server, device_name);
    temperature->setObjectName(device_name);
    gridLayout->addWidget(temperature);
    widgets.append(temperature);
}
```

Rysunek 18 Przykładowy widget

Zmienna device_type jest ustawiana przy połączeniu nowego urządzenia, odbierając od niego wiadomość o typie tego urządzenia.

Klient

Przy tworzeniu okna nawiązywane jest połączenie na porcie 12345 i adresie ip 127.0.0.1

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    auto ip = "127.0.0.1";
    auto port = 12345;

    _controller.connectToDevice(ip, port);
    _controller.send(device_type);
    setDeviceController();
}
```

Rysunek 19 Nawiązywanie połączenia

W kontrolerze urządzenia nawiązywane są połączenia do obsługi połączenia, odłączenia, zmiany stanu, błędu, odebrania danych.

```
DeviceController::DeviceController(QObject *parent)
    : QObject{parent}
{
    connect(&_socket, &QTcpSocket::connected, this, &DeviceController::connected);
    connect(&_socket, &QTcpSocket::disconnected, this, &DeviceController::disconnected);
    connect(&_socket, &QTcpSocket::stateChanged, this, &DeviceController::socket_stateChanged);
    connect(&_socket, &QTcpSocket::errorOccurred, this, &DeviceController::errorOccurred);
    connect(&_socket, &QTcpSocket::readyRead, this, &DeviceController::socket_readyRead);
}
```

Rysunek 20 Połączenia klienta

Po wywołaniu funkcji connectDoDevice w oknie głównym aplikacji nawiązane zostaje połączenie na podany adres i port.

```
void DeviceController::connectToDevice(QString ip, int port){

    if(_socket.isOpen()){
        if(ip == _ip && port == _port){
            return;
        }

        _socket.close();
    }

    _ip = ip;
    _port = port;
    _socket.connectToHost(_ip, _port);

}
```

Rysunek 21 Połączenie do serwera

Po udanym połączeniu klient otrzymuje nazwę od serwera, a ten ją odbiera i ustawia jako zmienną wewnątrz kodu.

```
void DeviceController::socket_readyRead(){
    auto data = _socket.readAll();

    QString message = QString::fromStdString(data.toStdString());
    QString command = QString("name=(\\w+)");
    QRegularExpression regex(command);
    QRegularExpressionMatch match = regex.match(message);      Δ Do
    if (match.hasMatch()) {
        QString nameStr = match.captured(1);
        _device_name = nameStr;
    }

    emit dataReady(data);
}
```

Rysunek 22 Ustawienie nazwy

Po zmianie wartości na serwerze dla danego urządzenia wysyłana jest wiadomość, którą klient odbiera i weryfikuje na podstawie swojej nazwy. Jeśli w komendzie występuje nazwa urządzenia, to komenda jest wykonywana, w przeciwnym wypadku urządzenie ignoruje sygnał.

```
void MainWindow::device_dataReady(QByteArray data){
    QString message = QString::fromStdString(data.toStdString());
    command = QString("name=(\\w+)");
    QRegularExpression regex_name(command);
    QRegularExpressionMatch match_name = regex_name.match(message);
    if (match_name.hasMatch()) {
        QString nameStr = match_name.captured(1);
        device_name = nameStr;
        ui->label->setText(device_name);
        ui->label->setStyleSheet("color: white; font-size: 14pt; tex-align: center; font-weight: bold;");
        ui->label->setAlignment(Qt::AlignCenter);
    }
```

Rysunek 23 Weryfikacja komendy