Kraków, 31.05.2022r

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Projekt wagi elektronicznej z wykorzystaniem tensometra

Autor pracy: Malwina Cieśla

Kierunek: Informatyka techniczna

Nr. Albumu: 306656

Projekt powstał w ramach przedmiotu: Inteligentne systemy pomiarowe

Spis treści

1. Wprowadzenie i cele 3
2. Wykorzystane komponenty i urządzenia 3
3. Użyte środowiska....4
3.1 STM32CUBEIDE....4
3.1.1 Perspektywa Device Configuration Tool 5
3.1.2 Perspektywa C/C++....5
3.2 Qt Creator IDE....7
3.2.1 Tryb Edycja....7
3.2.2 Tryb Design....8
4. Wykorzystane w projekcie elementy 9
4.1 Pomiar obciążenia....9
4.1.1 Belka tensometryczna 9
4.1.2 Wzmacniacz operacyjny 10
4.1.3 Płytka NUCLEO z mikrokontrolerem STM32L476RG....12
5. Komunikacja z komputerem, UART....14
5.1 Informacje wstępne....14
5.2 Konfiguracja UART na STM32....15
5.3 Wysyłanie i odbieranie danych 16
6. Aplikacja....16
6.1 Pomiar obciążenia....17
6.1.1 STM32CUBEIDE....17
6.1.2 Qt Creator....18
6.2 Zapis wagi do pliku tekstowego....20
7. Podsumowanie i dalsze kierunki rozwoju 20

1. Wprowadzenie i cele

Opracowany projekt ma na celu stworzenie wagi elektronicznej służącej przy użyciu belki tensometrycznej do pomiaru obciążenia nie większego niż maksymalna zapisana na tensometrze waga. Jednakże zważywszy na konstrukcję wagi i jej wymiary badane były obciążenia do 5 kg.

Waga, w dostępnym dla użytkownika panelu, po włączeniu programu przyciskiem ON/OFF od razu zostaje starowana do początkowej wartości „0”. Dzięki temu użytkownik nie musi pamiętać o dodatkowym tarowaniu po włączeniu wagi co może niejednokrotnie skrócić czas oczekiwania na gotowość wagi. Może on, po włączeniu i odczekaniu chwili, aż pojawi się waga „0”, od razu przystąpić do mierzenia wagi badanego przedmiotu.

W wykonanej w ramach projektu wadze elektronicznej można wyróżnić główne zadanie. Jest nim mierzenie obciążenia położonego na wagę obiektu. Jednakże program realizujący obsługę wagi został dodatkowo wzbogacony o możliwość zapisu wybranej wagi do pliku tekstowego. Dzięki tej funkcji wszystkie wykonane przez użytkownika pomiary mogą być dostępne nawet po wyłączeniu programu. Zapis do pliku wykonywany jest również przez osobę wykonującą pomiary, ponieważ funkcjonalność ta znajduje się również w tym samym panelu. Jednakże dla czytelniejszej formy panelu sterowania wagą zapis znajduje się w wyodrębnionej części o nazwie „Notes”.

2. Wykorzystane komponenty i urządzenia

Do wykonania projektu wykorzystano następujące komponenty:

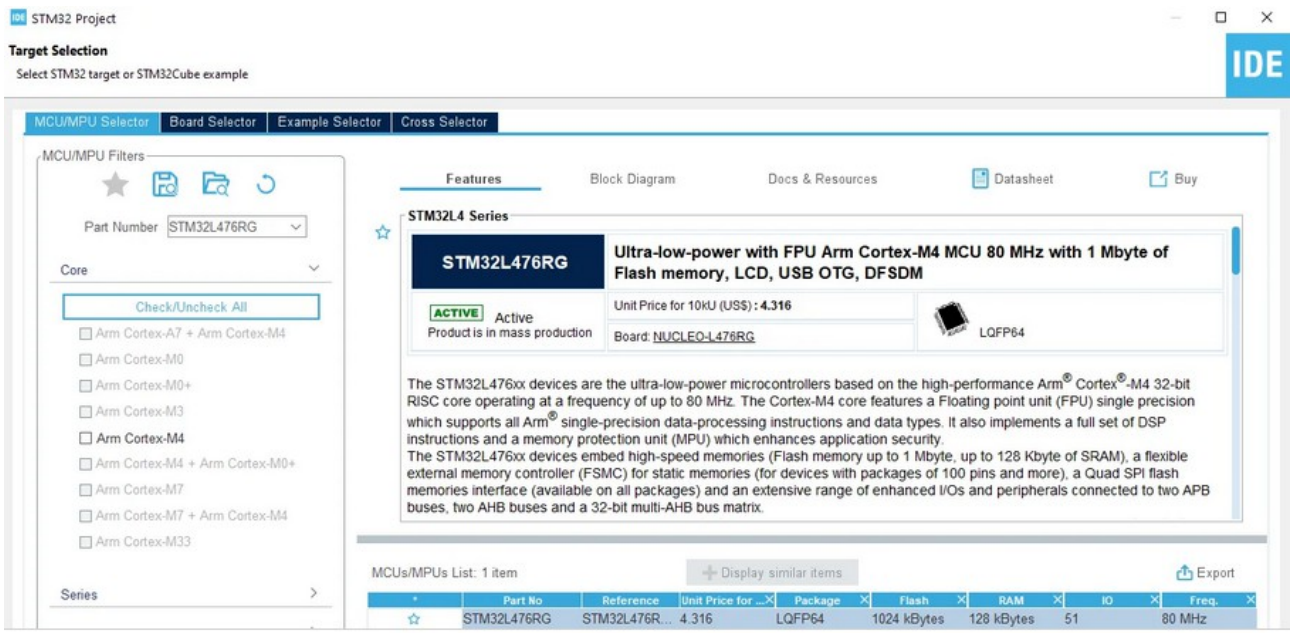
- Płytka NUCLEO z mikrokontrolerem STM32L476RG
- Wzmacniacz do belki tensometrycznej HX711
- Belka tenometryczna, maksymalna waga = 20 kg
- przewody, śrubki do łączenia elementów
- 2x drewniana płytka o wymiarach
- 5x filcowe podkładki pod krzesła
- Kabel USB A - mini-USB B

3. Użyte środowiska

Ze względu na specyfikację zadania oraz plan realizacji projektu, który zakładał użycie narzędzi publicznie dostępnych dla programistów używane były dwa środowiska: STM32CUBEIDE oraz Qt Creator IDE. W środowisku STM32CUBEIDE kod programu pisany był w języku C, a w środowisku Qt Creator IDE kod napisany został w języku C++.

3.1 STM32CUBEIDE

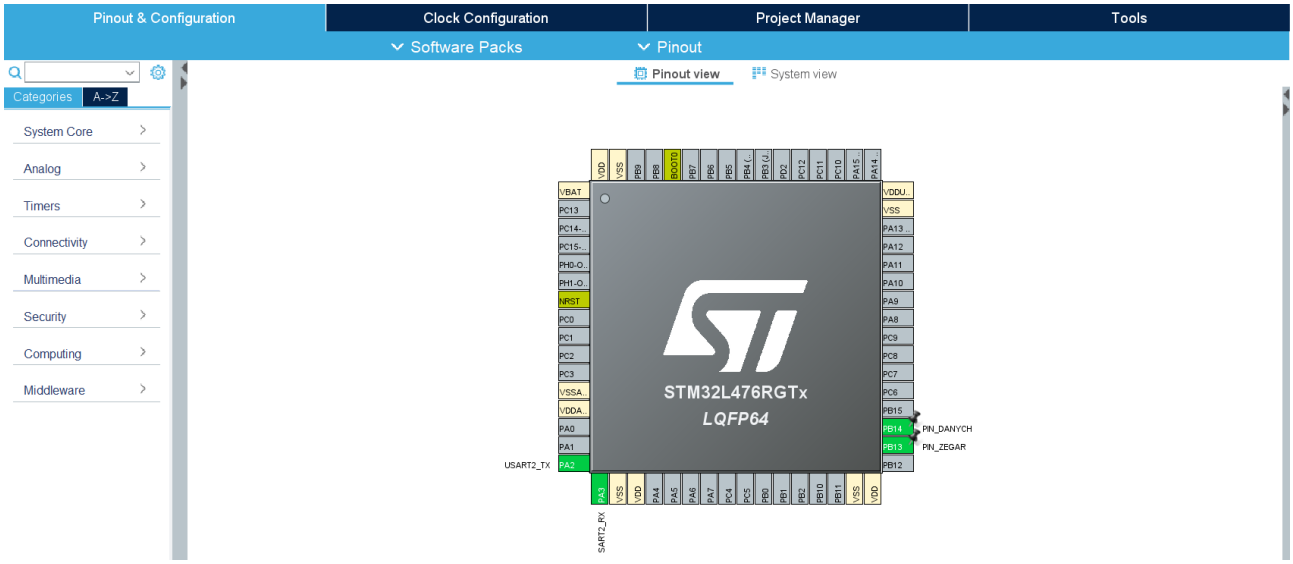
Używane w projekcie STM32CUBEIDE jest oficjalnym środowiskiem IDE, za którego rozwój odpowiada firma STMicroelectronics. Dzięki temu program ten jest przeznaczony do tworzenia oprogramowań pod układy STM32. W opcjach tworzenia programu występuje opcja wyboru używanego mikrokontrolera. Dzięki temu po wybraniu odpowiedniego symbolu widoczny będzie skrót informacji na temat wybranego mikrokontrolera, jego cena (dla zamówień hurtowych) wraz z informacjami o płytce rozwojowej. W oknie tym od razu widoczny jest też dostęp do dokumentów, które powiązane są z tym układem. Na ilustracji nr 1 przedstawiony został przykład wypisanych informacji dla używanego w projekcie mikrokontrolera:



Ilustracja 1: Wybór mikrokontrolera w STM32CUBEIDE

3.1.1 Perspektywa Device Configuration Tool

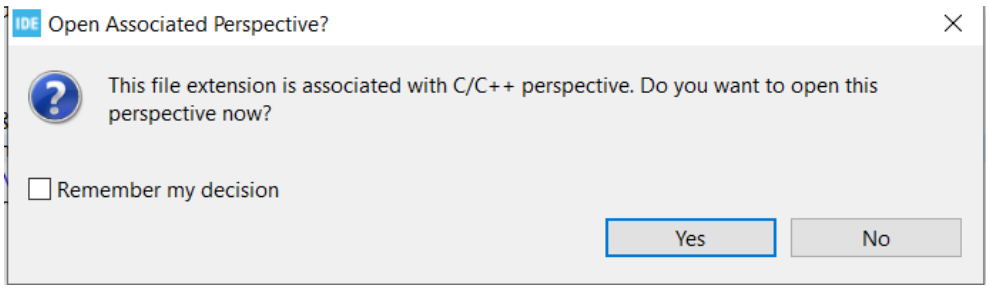
Perspektywy Device Configuration Tool, nazywana też CubeMX, jest narzędziem przygotowanym przez producenta mikrokontrolerów STM32. Narzędzie to przedstawia widok dostępnych pinów. Zawiera on wizualizację mikrokontrolera i jego wyprowadzeń oraz posiada pionowy boczny pasek narzędzi. Wybierane są tam wymagane urządzenia zewnętrzne. Wyboru sterowników można dokonać, wybierając pin bezpośrednio obsługujący urządzenie zewnętrzne lub wybierając konkretne urządzenie z paska narzędzi. Narzędzie automatycznie przypisze to urządzenie do odpowiednich pinów oraz rozwiąże konflikty pinów, przenosząc skonfliktowane urządzenie zewnętrzne do nieużywanych pinów, które również obsługują urządzenia. Na ilustracji nr 2 przedstawiony został schemat pinów używanych w projekcie:



Ilustracja 2: Perspektywa Device Configuration Tool

3.1.2 Perspektywa C/C++

W katalogu *Core* > *Src* dostępnym z poziomu eksploatora projektów po lewej stronie okna należy dwukrotnie kliknąć w plik *main.c* – środowisko automatycznie zaproponuje przełączenie na perspektywę C/C++. Należy zgodzić się na to, by ujrzeć kod źródłowy programu z przypisanymi nazwami odzwierciedlającymi przypisanie do pinów utworzone wcześniej w perspektywie Device Configuration Tool:



Ilustracja 3: Okno zmiany perspektywy

Po przełączeniu perspektywy otworzy się kod źródłowy, w którym występują miejsca przeznaczone do wpisywania przez programistę danych oraz miejsca już z utworzonymi częściami kodu. Należy pamiętać o wpisywaniu własnych linii kodu jedynie w miejscach do tego przeznaczonych, w przeciwnym razie środowisko po zapisaniu i zbudowaniu programu usunie linie, które znajdują się w niewłaściwym miejscu. Na ilustracji nr 4 przedstawiony został początek kodu źródłowego wraz z widocznymi miejscami przeznaczonymi dla programisty:

```
1  /* USER CODE BEGIN Header */
2  /**
3    ******************************************************************************
4    * @file           : main.c
5    * @brief          : Main program body
6    ******************************************************************************
7    * @attention
8    *
9    * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
10   * All rights reserved.</center></h2>
11   *
12   * This software component is licensed by ST under BSD 3-Clause license,
13   * the "License"; You may not use this file except in compliance with the
14   * License. You may obtain a copy of the License at:
15   *                        opensource.org/licenses/BSD-3-Clause
16   *
17   ******************************************************************************
18   */
19 /* USER CODE END Header */
20 /* Includes ------------------------------------------------------------------*/
21 #include "main.h"
22
23 /* Private includes ----------------------------------------------------------*/
24 /* USER CODE BEGIN Includes */
25 #include <stdio.h>
26 #include <string.h>
27 #include <math.h>
28 #include <hx711.h>
29 /* USER CODE END Includes */
30
31 /* Private typedef -----------------------------------------------------------*/
```

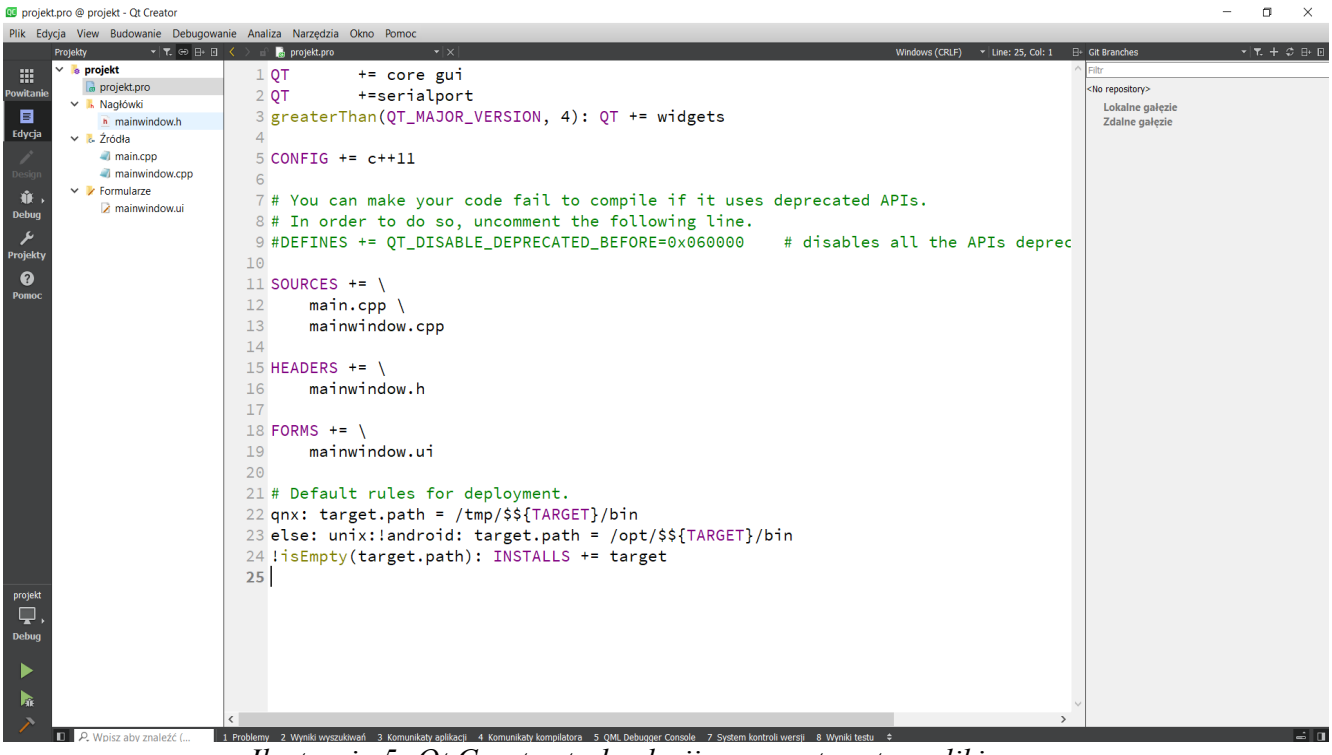
Ilustracja 4: Początek kodu źródłowego wraz z widocznymi miejscami przeznaczonymi dla programisty. Widoczny fragment przedstawia użyte w projekcie biblioteki

3.2 Qt Creator IDE

Qt Creator IDE jest responsywnym i intuicyjnym wieloplatformowym środowiskiem programistycznym ze zintegrowanymi narzędziami do projektowania interfejsu użytkownika, edytorem kodu z uzupełnianiem składni oraz wizualnymi narzędziami do debugowania i profilowania interfejsu. Do tworzenia projektu wykorzystywany był Qt Creator w wersji 4.15.1 (Community).

3.2.1 Tryb Edycja

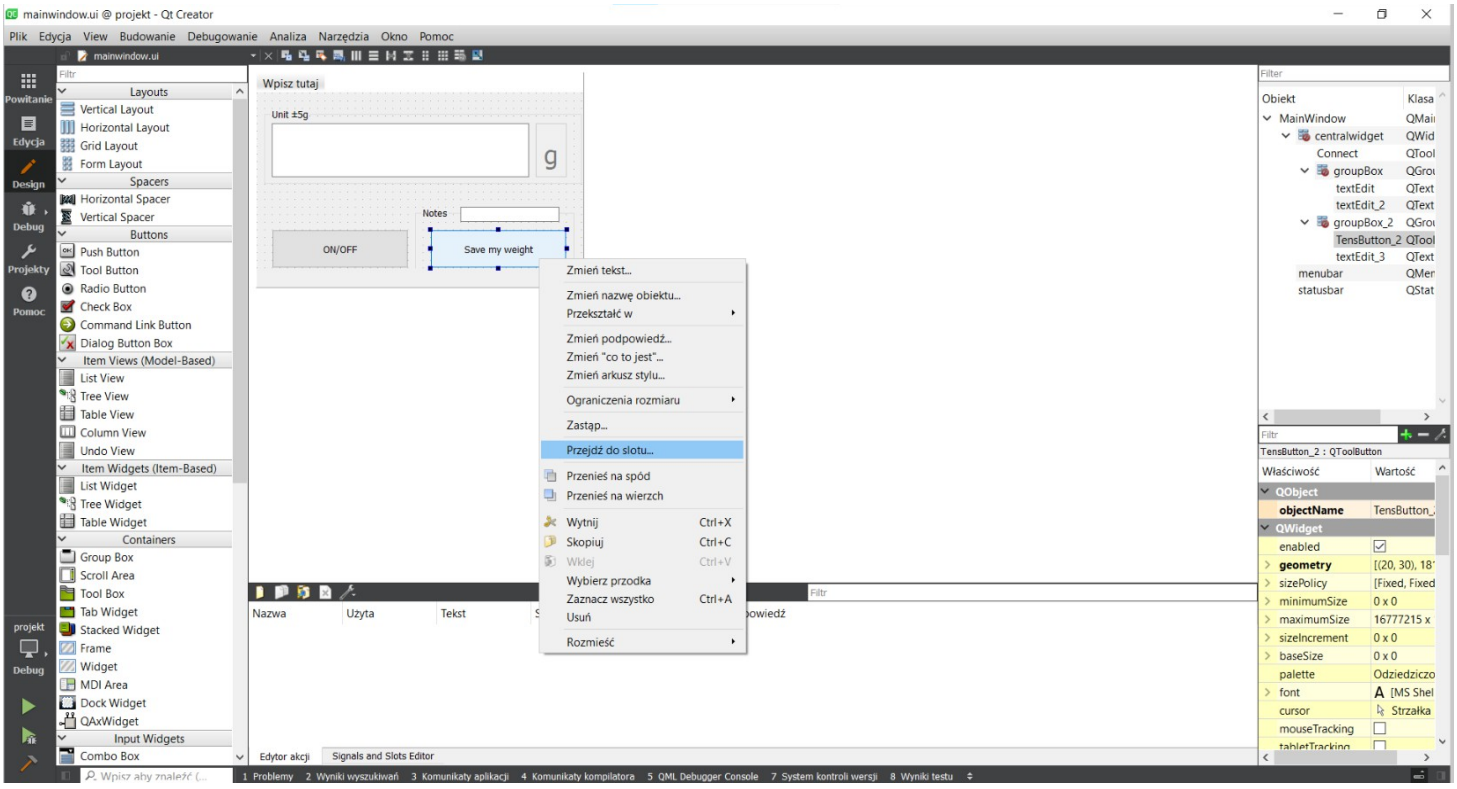
Poprzez otwarcie okna z linią poleceń w katalogu, w którym znajduje się plik .cpp z kodem programu i wykonanie polecenia: **qmake -project qmake make** uzyskany zostaje plik wykonywalny o rozszerzeniu .pro. Komenda qmake to narzędzie Qt wspierające budowanie aplikacji poprzez generację plików Makefile, które opisują proces budowania, na bazie specjalnych plików konfiguracyjnych. Dzięki temu w panelu widocznym po uruchomieniu środowiska w trybie edycji po lewej stronie dostępne będą zakładki z plikiem .pro, nagłówki, źródła i formularze. Na ilustracji nr 5 przedstawiony został widok zakładek wraz z otworzonym plikiem .pro dla stworzonego projektu:



Ilustracja 5: Qt Creator tryb edycji wraz z otwartym plikiem .pro

3.2.2 Tryb Design

Qt Creator zapewnia zintegrowany edytor wizualny do projektowania aplikacji opartych na widgetach w trybie projektowania. Integracja obejmuje zarządzanie projektami i uzupełnianie kodu źródłowego i bibliotek o dodane w trybie projektowania funkcjonalności. Dlatego też zakładka formularze widoczna w trybie edycja pozwala przejść do pliku .ui (user interface), który otwierany jest w edytorze wizualnym, w którym po lewej stronie widnieje kolumna możliwych do użycia widgetów. Po wprowadzeniu ich do obiektu MainWindow możliwe jest dodanie funkcjonalności poprzez wybranie prawym przyciskiem myszy opcji „Przejdź do slotu”. Po prawej stronie od głównego okna widoczna jest hierarchia użytych w projekcie widgetów wraz z ustawionymi dla nich nazwani, które po wprowadzeniu danego widgetu do slotu zostaną automatycznie prowadzone do kodu źródłowego. Na ilustracji nr 6 przedstawiony został widok trybu edycji wraz z utworzonym dla projektu oknem MainWidow. Przedstawiona została również możliwość ustawienia funkcjonalności widgetu „Push Button” poprzez użycie przycisku „Przejdź do slotu”:



Ilustracja 6: Tryb design wraz z dostępnymi widgetami i możliwością ustawienia funkcjonalności widgetu „Push Button” poprzez przycisk "Przejdź do slotu"

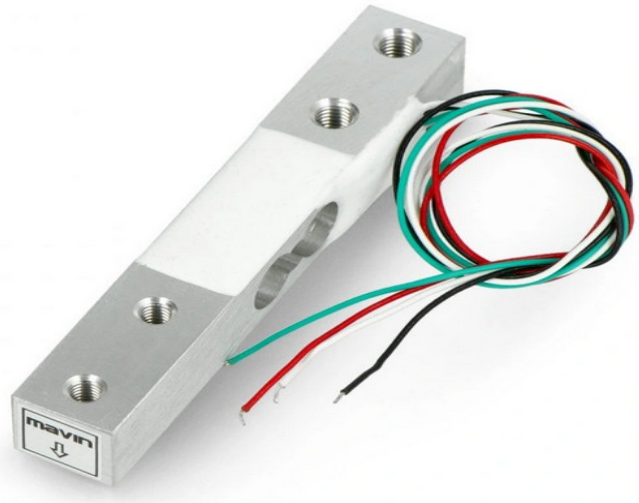
4. Wykorzystane w projekcie elementy

4.1 Pomiar obciążenia

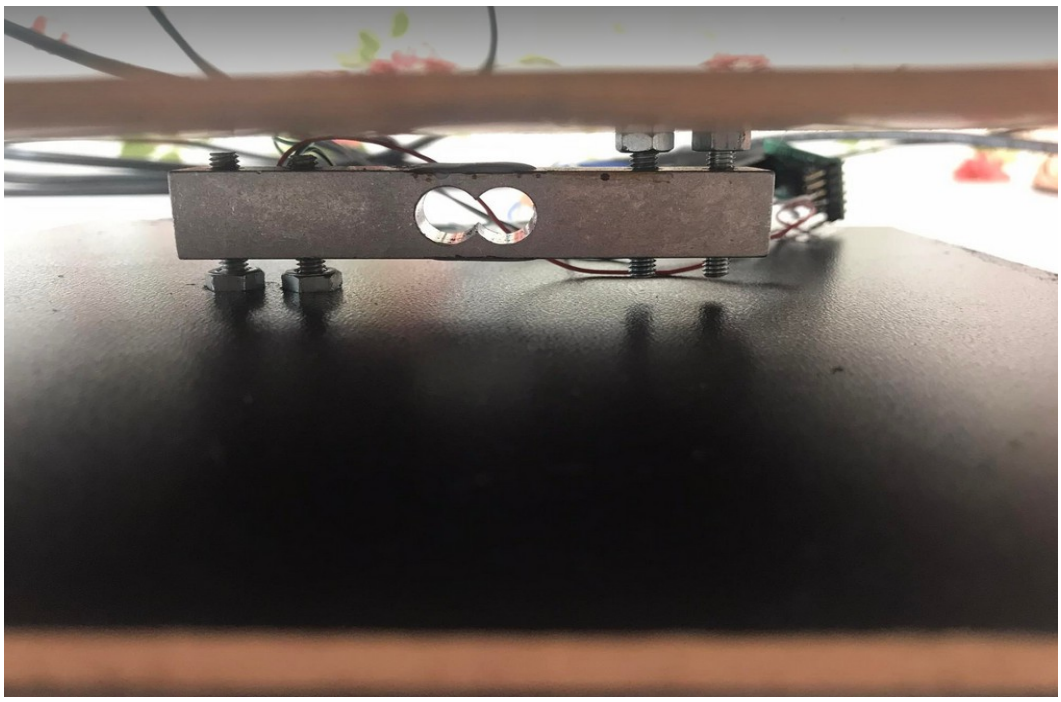
W skład najważniejszych elementów potrzebnych do utworzenia funkcjonalności pomiaru obciążenia wchodziły: belka tensometryczna, wzmacniacz do tensometra oraz płytka NUCLEO z miktrokontrolerem STM32L4. Poniżej zostały opisane te elementy oraz przedstawione zostały ilustracje opisujące wybrane elementy wraz z wykonanymi połączeniami między nimi.

4.1.1 Belka tensometryczna

Belka tensometryczna przeznaczona jest do statycznych i dynamicznych zadań pomiarowych. Jest to czujnik nacisku (wagi) stosowanym np. w wagach kuchennych. Belka tensometryczna działa w zakresie obciążeń do 20 kg, zakres ten podany jest na boku belki wraz ze strzałką przedstawiającą kierunek siły nacisku. Czujnik tensometryczny wykonany jest z aluminium i posiada wyprowadzone 4 przewody, które łączą się z wzmacniaczem do belek tensometrycznych HX711. W belce dodatkowo znajdują się gwintowane otwory montażowe potrzebne do montażu drewnianych płytek będących podstawami wagi.



Ilustracja 7: Belka tensometryczna



Ilustracja 8: Przymocowana beka tensometryczna do podstaw wagi

4.1.2 Wzmacniacz operacyjny

Wzmacniacz operacyjny przeznaczony jest do obsługi belek tensometrycznych. Układ zasilany jest napięciem od 2,6 V do 5,5 V, a komunikacja odbywa się poprzez dwuprzewodową magistralę: linię danych i linię zegarową. Czujniki tensometryczne obsługiwane są poprzez tzw. mostek Wheatstone'a, czyli przewody połączone w następujący sposób:

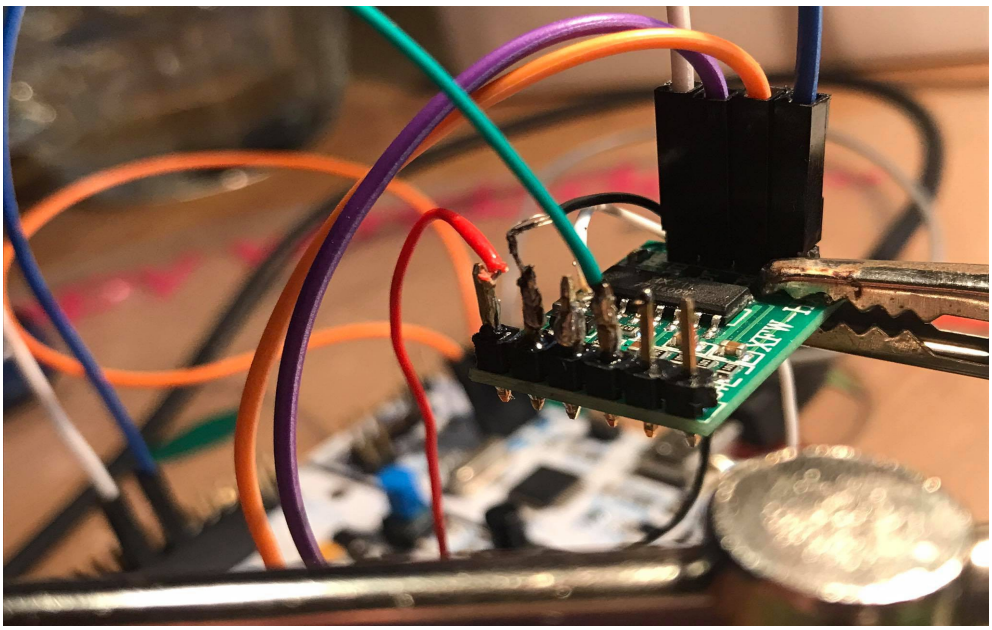
- **E+ (Excitation+)** - wzbudzenie dodatnie lub VCC (zasilanie)
- **E- (Excitation-) -** wzbudzenie ujemne lub GND (masa)
- **A- / A+** - wzmocnienie dodatnie, sygnał dodatni lub wyjście dodatnie
- **B- / B+** - wzmocnienie ujemne, sygnał ujemny lub wyjście ujemne

W projekcie wykorzystane były połączenia z E+. E-, A+ oraz A-.

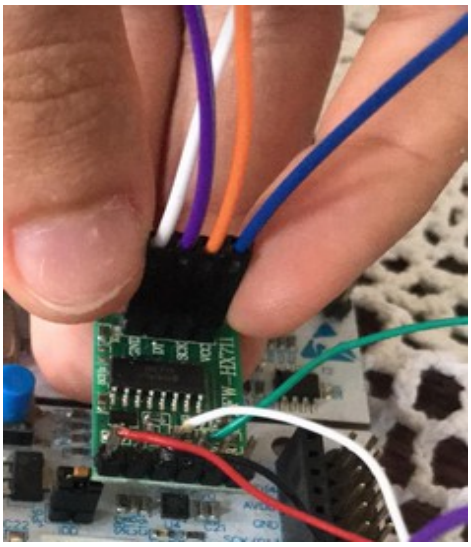
Po drugiej stronie wychodzą cztery piny. Opis ich wraz z użytymi w projekcie kolorami kabelków przedstawiony został poniżej:

- GND – masa układu (kolor biały)
- DT - linia danych magistrali komunikacyjnej (kolor fioletowy)
- SCK - linia zegarowa magistrali komunikacyjnej (kolor pomarańczowy)
- VCC - napięcie zasilania od 2,6 V do 5,5 V (kolor niebieski)

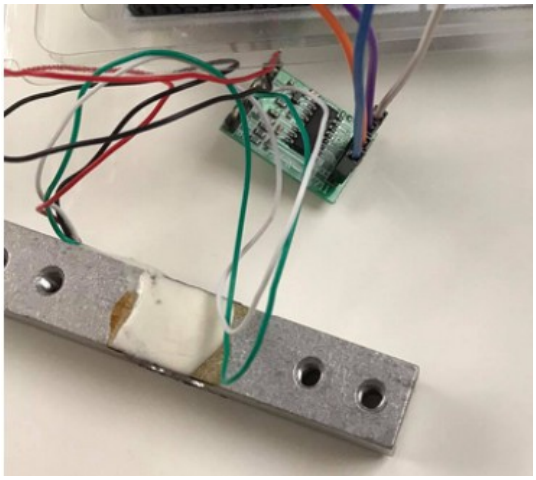
Przylutowane przewody tensometra oraz połączenia z płytką przedstawione zostały na ilustracjach 9, 10 oraz 11:



Ilustracja 9: Połączenie wzmacniacza do tensometra z przylutowanymi kabelkami tensometra oraz w tle przewody łączące wzmacniacz z płytką



Ilustracja 10: Widok z góry połączeń kabelków z odpowiednimi pinami na wzmacniaczu

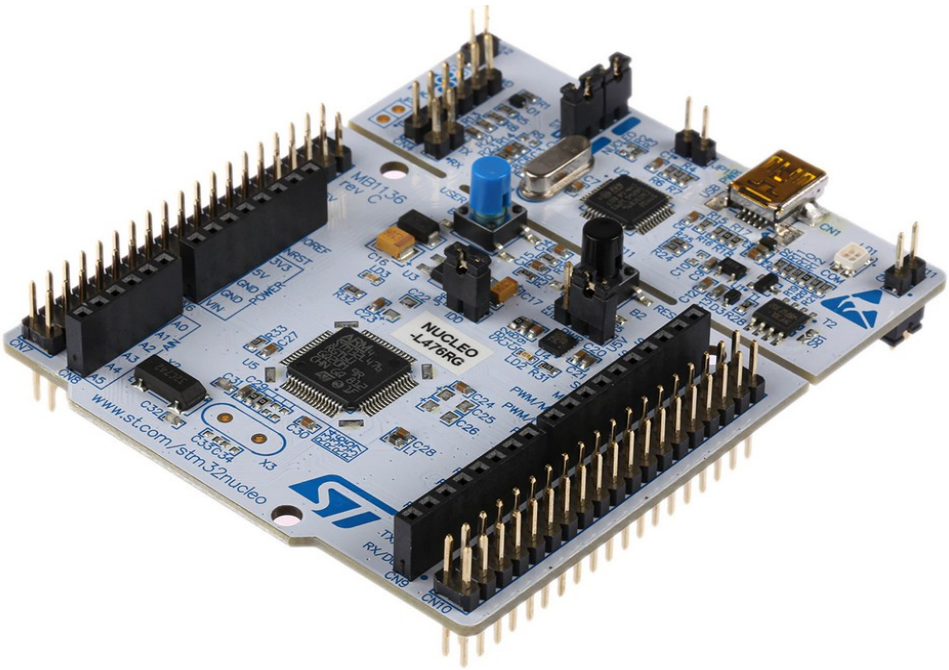


Ilustracja 11: Połączenie wzmacniacza z tensometrem, widok z góry

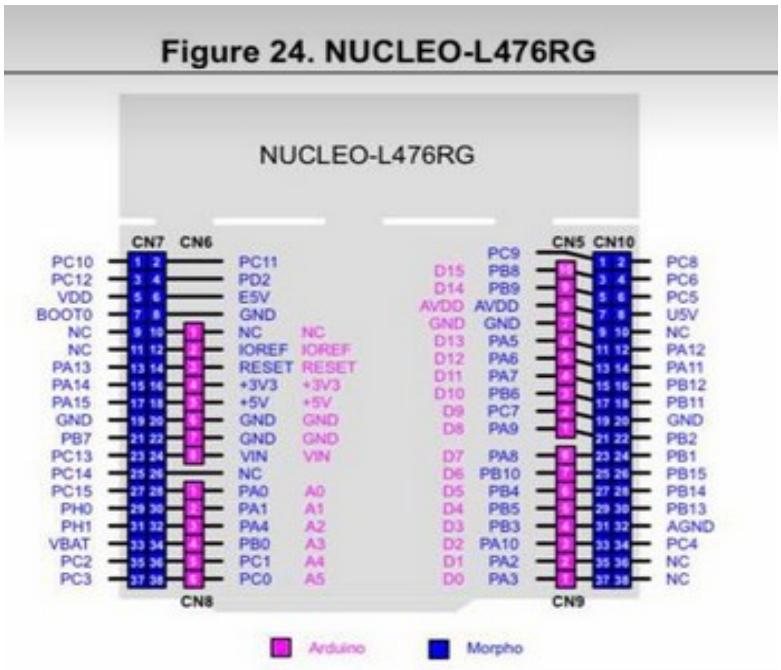
4.1.3 Płytka NUCLEO z mikrokontrolerem STM32L476RG

NUCLEO-L476RG to płytka rozwojowa Nucleo-64 STM32 z mikrokontrolerem STM32L476RG. Płytki z rodziny Nucleo firmy STMicroelectronics pozwalają na łatwe wykonywanie projektów układów wykorzystujących wydajny mikrokontroler z 32-bitowym rdzeniem ARM Cortex. Dostęp do wszystkich linii I/O mikrokontrolera uzyskamy jest dzięki złączu w standardzie ST Morpho, które pozwala na łatwe rozbudowywanie projektów. Dodatkowo płytka STM32 jest dostarczana z kompleksową biblioteką oprogramowania HAL STM32 wraz z różnymi pakietowymi przykładami oprogramowania, a także bezpośrednim dostępem do zasobów internetowych mbed. Biblioteki HAL (Hardware Abstraction Layer) stanowi wysokopoziomowy interfejs do części sprzętowej mikrokontrolera. Cube generuje kod konfiguracyjny oraz inicjalizujący dla włączanych przez programistę peryferiów wykorzystując do tego biblioteki HAL.

Budowa płytki oraz rozkład pinów wraz z ich nazwami przedstawione zostały na ilustracjach nr 12 oraz nr 13:

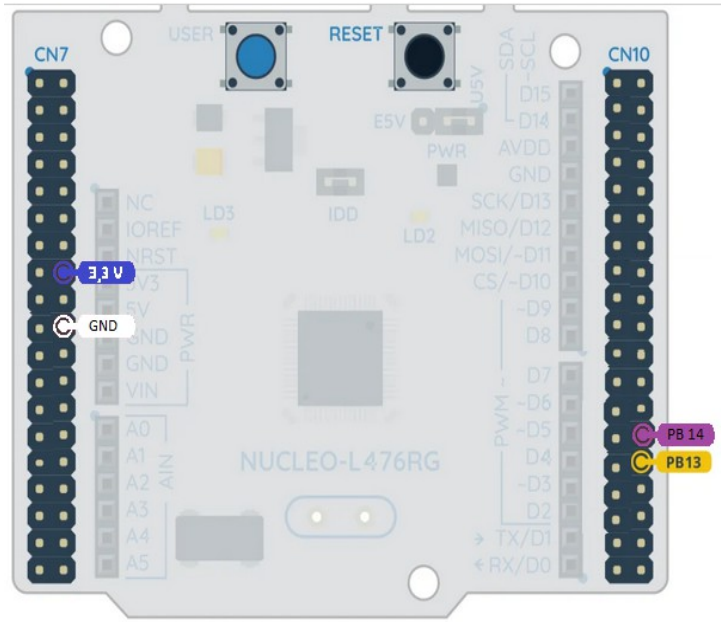


Ilustracja 12: Płytka NUCLEO z mikrokontrolerem STM32L476RG



Ilustracja 13: Widok płytki NUCLEO STM32L476RG wraz z rozpisanymi nazwami pinów. Ilustracja pochodzi ze specyfikacji płytki

Połączenie płytki ze wzmacniaczem i belką tensometryczną wykonane zostało przy użyciu czterech kabelków, które transportują informacje o masie układu (kolor biały), linii danych (kolor fioletowy), linii zegarowej (kolor pomarańczowy) i napięciu zasilania (kolor niebieski):

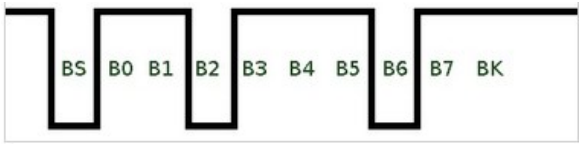


Ilustracja 14: Lokalizacja pinów używanych w projekcie

5. Komunikacja z komputerem, UART

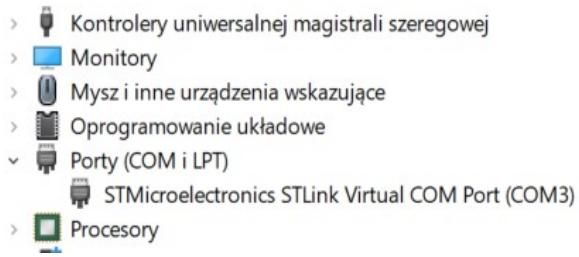
5.1 Informacje wstępne

W mikrokontrolerach stosuje się uproszczoną wersję interfejsu, działającą na takich napięciach jak 3.3V. Moduł odpowiedzialny za obsługę takiej komunikacji nazywany jest UART (ang. *Universal Asynchronous Receiver and Transmitter*). Transmisja rozpoczyna się od bitu startu (BS). Zawsze jest to bit będący logicznym zerem. Następnie, w zależności od użytej konfiguracji, następuje po sobie 7, 8 lub 9 bitów danych, które są wysyłaną informacją. Bit stopu (BK) to bit będący logiczną jedynką informujący o końcu transmisji. Przykładowa ramka UART przedstawiona została na ilustracji nr 15:



Ilustracja 15: Przykładowa ramka UART posiadająca bity danych B0-B7

Płytka Nucleo wyposażona jest również w konwerter z UART na USB, dzięki czemu po podłączeniu do komputera, przejściówka będzie widziana jako port COM (port szeregowy):



Ilustracja 16: Fragment Menedżera urządzeń

5.2 Konfiguracja UART na STM32

Pierwszym krokiem była konfiguracja zegara. STM32 posiada kilka modułów USART, jednak najczęściej wykorzystywany jest USART2, ponieważ jest on podłączony do przejściówki, która znajduje się na płytce. W części GPIO Ports Clock Enable znajdują się dwie instrukcję, z których pierwsza uruchamia zegar linii I/O portu A, druga uruchamia USART2:

```
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
```

Ilustracja 17: Fragment kodu z konfiguracją zegara

Linie USART2_TX oraz USART2_RX wystarczy wybrać w opcjach dostępnych w perspektywie *Device Configuration Tools.* Dzięki temu linia wyjściowa (TX) będzie dostępna na wyprowadzeniu PA2, a wejście (RX) na PA3. Na ilustracji nr 2 przedstawiony został widok użytych pinów.

Po skonfigurowaniu linii należało przygotować moduł USART2 wprowadzając początkowe dane takie jak prędkość transmisji i długość przesyłanego słowa. W projekcie użyta została standardowa wartość prędkości transmisji równa 115200. Zainicjowane wartości dla modułu USART2 przedstawione zostały na ilustracji nr 18:

```
huart2.Instance = USART2;
huart2.Init.BaudRate = 115200;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
if (HAL_UART_Init(&huart2) != HAL_OK)
{
  Error_Handler();
}
```

Ilustracja 18: Skonfigurowane urządzenie USART2

5.3 Wysyłanie i odbieranie danych

Do wysyłania danych wykorzystywana jest funkcja HAL_UART_Transmit(). Przyjmuje ona cztery parametry: wykorzystywany interfejs (USART2), adres i wielkość bufora danych do wysłania oraz timeout. Funkcja HAL_UART_Transmit() pozwala na wysyłanie danych dowolnego typu jednak w projekcie przesyłane były napisy. Funkcja HAL_UART_Transmit() sama ustali długość napisu oraz dokona niezbędnych konwersji typów (przesyłanie przez funkcję strlen()). Należy jednak pamiętać o różnicach używania znaków końca linii w różnych systemach. Ponieważ projekt uruchamiany był na komputerze z systemem Windows, wysyłane były dwa znaki CR LF (czyli \r\n). :

```
HAL_UART_Transmit(&huart2, (uint8_t*)message4, strlen(message4), HAL_MAX_DELAY);
```

Ilustracja 19: Fragment kodu z użytą funkcją HAL_UART_Transmit()

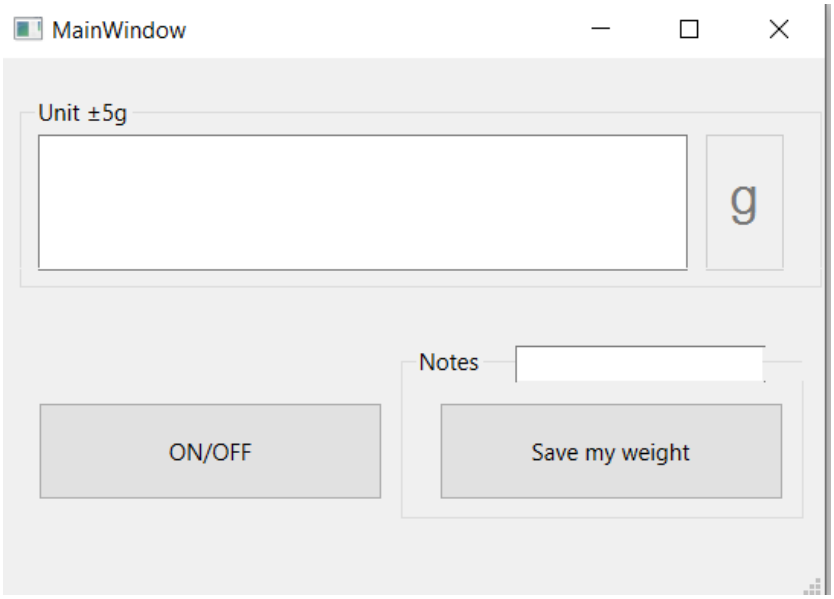
Do odbioru bajtu wykorzystywana była funkcja HAL_UART_Receive. Parametry funkcji odbierającej dane są bardzo podobne do funkcji HAL_UART_Transmit. Jedyna różnica to znaczenie bufora – w tym przypadku znajdują się dane odbierane przez port szeregowy:

```
HAL_UART_Receive(&huart2,&connector,3,HAL_MAX_DELAY);
```

Ilustracja 20: Fragment kodu z użytą funkcją HAL_UART_Receive()

6. Aplikacja

W celu wykonania sterowalnego panelu programu należało utworzyć w programie STM32CUBEIDE funkcję sczytującą dane z płytki NUCLEO i wczytującą je do programu Qt Creator. Natomiast w programie Qt Creator należało utworzyć funkcje odbierające dane z programu STM32CUBEIDE i wypisujące je w panelu tworzonym w trybie Design. Druga funkcjonalność programu, wpisywanie danych do pliku tekstowego, wymagała jedynie utworzenia dodatkowej funkcji z programie Qt Creator. Widoczna jest ona w panelu w części „Notes”. Widok aplikacji po włączeniu programu przedstawiony został na ilustracji nr 21:



Ilustracja 21: Widok aplikacji po włączeniu programu

6.1 Pomiar obciążenia

6.1.1 STM32CUBEIDE

W programie funkcjonalność sczytywania informacji z belki tensometrycznej wykonywana jest poprzez wzmacniacz. Wszystkie ważne funkcje dotyczące otrzymywanych danych ze wzmacniacza zapisane zostały w pliku hx711.c, który informacje sczytuje wykorzystując biblioteki „hx711.h" oraz "hx711Config.h". Dzięki temu w pliku main.c w głównej pętli while można początkową sczytywaną wartość zapisać do zmiennej, którą w kolejnych wykonaniach sczytywania odejmować od nowych pomiarów uzyskując w ten sposób wytarowaną wartość wagi. Tak uzyskaną wartość wagi należy przesłać poprzez funkcję HAL_UART_Transmit() do programu Qt Creator. Fragment kodu programu STM32CUBEIDE wraz z główną pętlą while przedstawiony został na ilustracji nr 22:

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
    HAL_UART_Receive(&huart2,&connector,3,HAL_MAX_DELAY);
   // HAL_UART_Transmit(&huart2,&connector,1,HAL_MAX_DELAY);
    if(strncmp(connector,message2,1)==0){
        hx711_init(&loadcell, PIN_ZEGAR_GPIO_Port, PIN_ZEGAR_Pin, PIN_DANYCH_GPIO_Port, PIN_DANYCH_Pin);
          hx711_coef_set(&loadcell, 354.5); // read afer calibration
          hx711_tare(&loadcell, 10);
          HAL_Delay(500);
          weight = hx711_weight(&loadcell, 10);
          weight2 = (int) (weight);
          while (1){
              HAL_Delay(500);
              weight = hx711_weight(&loadcell, 10);
              weight1 = (int) ((weight)-weight2)*3.5;
              sprintf(message4,"%d\r\n",weight1);
              HAL_UART_Transmit(&huart2, (uint8_t*)message4, strlen(message4), HAL_MAX_DELAY);
          }
        connector[5]="0\r\n";

    }
}
/* USER CODE END 3 */
```

Ilustracja 22: Fragment kodu programu STM32CUBEIDE wraz z główną pętlą while

6.1.2 Qt Creator

W programie wykorzystana została specjalna klasa Qt QserialPort, która pozwala na przechwytywanie informacji z mikrokontrolera. Dzięki temu po ustawieniu informacji podstawowych takich jak widoczne informacje na ilustracji nr 18 program gotowy był do wyszukiwania urządzeń spełniających dane wartości. Po znalezieniu takiego urządzenia program łączy się z nim używając obiektowej funkcji connect(), która przyjmuje cztery parametry: urządzenie wysyłające (w tym przypadku wybrane urządzenie QserialPort), sygnał, urządzenie połączone z urządzeniem wysyłającym oraz sposób połączenia. Sposób połączenia został napisany w dodatkowej funkcji readFromPort(). Funkcja ta oraz funkcja odpowiedzialna za połączenie do płytki NUCLEO przedstawione zostały na poniższych ilustracjach:

```
void MainWindow::on_Connect_clicked()
{
    if(switcher==0){

        QString message="1\r\n";
        devices = QSerialPortInfo::availablePorts();
        this->device->setPortName(devices.at(0).portName());
        if(!device->isOpen()) {
            // OTWÓRZ I SKONFIGURUJ PORT:
            if(device->open(QSerialPort::ReadWrite)) {
                this->device->setBaudRate(QSerialPort::Baud115200);
                this->device->setDataBits(QSerialPort::Data8);
                this->device->setParity(QSerialPort::NoParity);
                this->device->setStopBits(QSerialPort::OneStop);
                this->device->setFlowControl(QSerialPort::NoFlowControl);
                this->addToLogs("HELLO :)");
                this->device->write(message.toStdString().c_str());
                // CONNECT:
                connect(this->device, SIGNAL(readyRead()), this, SLOT(readFromPort()));
            }
        }
        switcher=1;
    }else if(switcher==1){
        if(this->device->isOpen()) {
            this->device->close();
            this->addToLogs("GOODBYE :)");
        }
        switcher=0;
    }
}
```

*Ilustracja 23: Funkcja o*dpowiedzialna za połączenie do płytki NUCLEO

```
void MainWindow::readFromPort() {
    while(this->device->canReadLine()) {
        QString line = this->device->readLine();
        //qDebug() << line;

        QString terminator = "\r";
        int pos = line.lastIndexOf(terminator);
        //qDebug() << line.left(pos);

        this->addToLogs(line.left(pos));
    }
}
```

Ilustracja 24: Funkcja readFromPort()

Widoczna na ilustracji nr 23 funkcja ma dwie funkcjonalności: po kliknięciu przycisku „ON/OFF”, przedstawionego na panelu sterowania (ilustracja nr 21), pozwala na połączenie płytki z programem, starowanie wagi i pomiar obciążenia. Po ponownym kliknięciu przycisku funkcja zamyka połączenie.

6.2 Zapis wagi do pliku tekstowego

Funkcjonalność ta wymagała utworzenia dodatkowej funkcji w programie Qt Creator. W celu czytelniejszej formy zapisu danych utworzony został oddzielny fragment panelu, w którym w oknie zapisu można wpisać wymaganą wartość wagi. Po naciśnięciu przycisku „Safe my weight" funkcja sprawdza czy plik jest otwarty i zdolny do zapisu danych, zapisuje wagę do pliku „result_storage.txt", a następnie w panelu usuwa wartość z okna zapisu. Na ilustracji nr 25 przedstawiony został kod funkcji dostępny w programie Qt Creator:

```
void MainWindow::on_TensButton_2_clicked()
{
    if(file.open(QIODevice::WriteOnly |QIODevice::Append |QIODevice::Text))
    {
        QTextStream stream(&file);
        stream << "Waga: " << ui->textEdit_3->toPlainText()<<'\n';
        ui->textEdit_3->clear();

        file.close();
        qDebug() << "Writing finished";
    }
}
```

Ilustracja 25: Funkcja odpowiedzialna za zapis obciążenia do pliku tekstowego

7. Podsumowanie i dalsze kierunki rozwoju

Głównym celem tego projektu było zaprojektowanie i utworzenie wagi elektronicznej wykorzystującej płytkę NUCLEO z mkrokontrolerem oraz belkę tensometryczną. Dodano również funkcjonalność zapisu najważniejszych pomiarów do pliku tekstowego tak, aby użytkownik nie po zamknięciu programu miał możliwość wglądu do wykonanych pomiarów.

W przyszłości projekt może zostać rozwinięty o dodatkowe funkcjonalności wspomagające pracę użytkownika. Głównym kierunkiem rozwoju wykonanej w ramach projektu wagi elektronicznej może być utworzenie bezprzewodowego mechanizmu pomiaru obciążenia np. poprzez użycie Wi-Fi, dzięki czemu waga stanie się bezprzewodowym i przenośnym sprzętem pomiarowym.