Dokumentacja projektu końcowego kursu programowanie obiektowe

Temat projektu	Aplikacja do sterowania i analizy pomiarów z sensora w dwóch wymiarach z zewnętrznego urządzenia.	**Nazwa aplikacji**	Miernik2D
Wykonawca	Mikołaj Głos	**Indeks**	287095
Data wykonania	13.06.2025		

1 Założenia i wykonanie projektu

Aplikacja steruje obrotem sensora w płaszczyźnie pionowej i poziomej, a następnie wykonuje analizę i zobrazowanie danych w postaci pojedynczej wartości, tablicy jednowymiarowej lub tablicy dwuwymiarowej.

1.1 Założenia funkcjonalne

- Aplikacja pobiera kąt nastawy urządzenia w płaszczyźnie poziomej i pionowej oraz wartość mierzoną przez sensor i wizualizuje te dane do odczytu przez użytkownika.
- Aplikacja określa kąt nastawy w płaszczyźnie poziomej i pionowej urządzenia na podstawie danych wejściowych od użytkownika.
- Aplikacja oferuje tryb "skan 2D" dla stałego kąta poziomego lub pionowego określonego przez użytkownika w którym odczytuje i zapisuje wartości sensora dla wszystkich wartości nastawy kąta zmiennego i wizualizuje je dla użytkownika w postaci wykresu liniowego funkcji:
  $y(x)$ [x - kąt, y - wartość].
- Aplikacja oferuje tryb demo umożliwiający przetestowanie funkcjonalności na modelu symulującym urządzenie.

2 Wymagania systemowe i sprzętowe

2.1 System

Aplikacja została stworzona i przetestowana na komputerze o następujących parametrach:

Procesor	13th Gen Intel(R) Core(TM) i5-13500H, 2600 MHz, Rdzenie: 12, Procesory logiczne: 16
Pamięć RAM	16 GB
System operacyjny	Microsoft Windows 11 Home 64bit

Prawidłowe działanie na urządzeniu o gorszych parametrach i tym samym systemie operacyjnym nie jest gwarantowane, ale bardzo prawdopodobne.

2.2 Zewnętrzne urządzenie pomiarowe

Aplikacja jest w stanie współpracować z dowolnym urządzeniem spełniającym określone wytyczne komunikacji przez komputerowy port szeregowy USB:

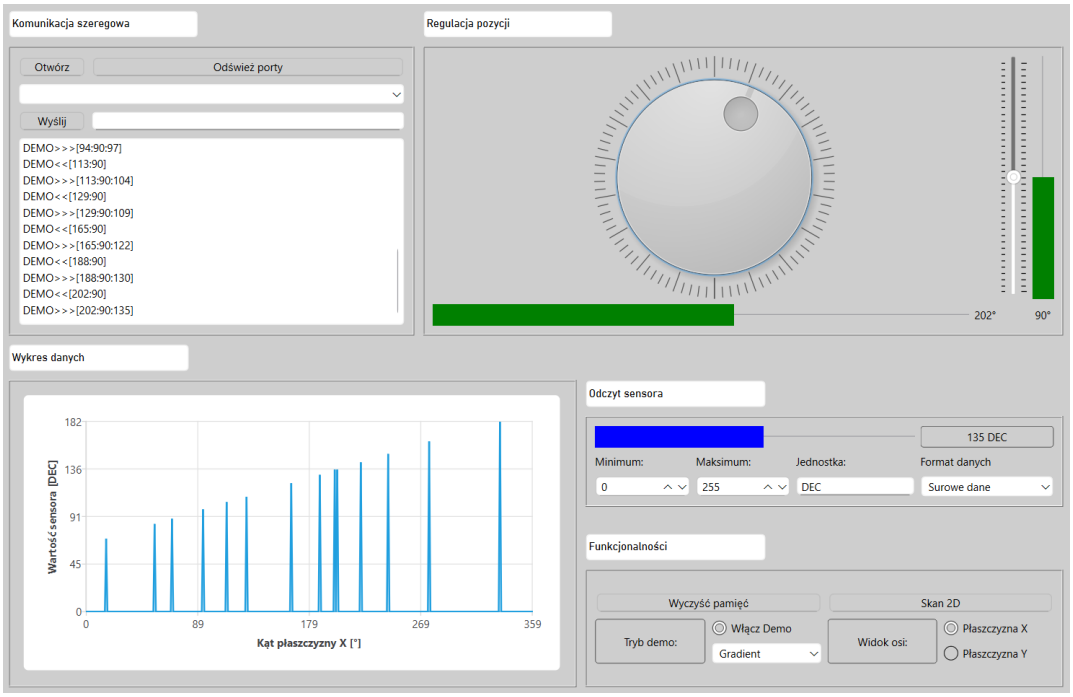
Wytyczne formatu wiadomości		
Wiadomość wejściowa:	x:y	x = pozycja "pozioma" rozdzielczość [0-360]
		y = pozycja "pionowa" rozdzielczość [180]
Wiadomość wyjściowa	x:y:z	z = wartość zwrotna rozdzielczość [0-255]
Kodowanie	UTF-8	Tekst w obie strony zakodowany w formie tablicy bajtów
Parametry komunikacji		
Baudrate		115200
Data bits		8
Parity		None
Stop bits		1

Niepoprawny format dowolnej wiadomości wysłanej do aplikacji spowoduje katastrofalne błędy. Jeżeli urządzenie w konstrukcji odbiega od intuicji zastosowanej w interfejsie graficznym, korzystanie z aplikacji może być utrudnione z perspektywy użytkownika.

3 Instrukcja użytkownika

3.1 Wstępne informacje

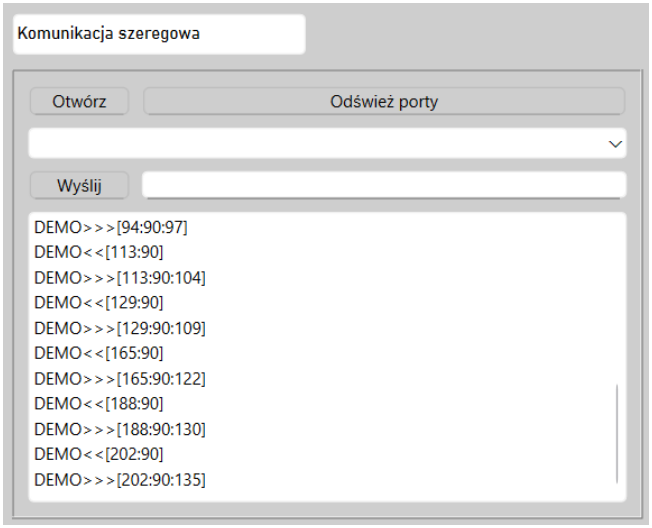
Aplikacja jest podzielona na moduły funkcjonalne, opisane szczegółowo w dalszej części instrukcji.



Rysunek 1: Interfejs aplikacji

- Jeżeli urządzenie pomiarowe nie jest dostępne, w celu przetestowania działania programu należy zaznaczyć opcję [*Włącz Demo*] w panelu [**Funkcjonalności**].
- Jeżeli urządzenie pomiarowe jest dostępne należy podłączyć je do jednego z portów szeregowych i zapoznać się z dalszą częścią instrukcji, zaczynając od modułu [**Komunikacja szeregowa**].

3.2 Komunikacja szeregowa



Rysunek 2: Moduł "Komunikacja szeregowa"

Moduł **[Komunikacja szeregowa]** służy do połączenia się z urządzeniem oraz monitorowania komunikacji między urządzeniem a aplikacją. W trybie demo moduł ma charakter jedynie informacyjny.

3.2.1 Przyciski

- *[Otwórz]* - Otwiera port szeregowy wybrany w menu poniżej. Otwarcie portu z podłączonym urządzeniem jest konieczne przed rozpoczęciem pracy.
- *[Odśwież porty]* - Odświeża listę wykrytych portów w menu poniżej. Konieczne w przypadku podłączenia urządzenia w trakcie działania aplikacji.
- *[Wyślij]* - Umożliwia ręczne przesłanie wiadomości wpisanej w pole obok do urządzenia. Przydatne w przypadku urządzenia obsługującego dodatkowe komendy lub diagnozowania problemów z komunikacją. Odradzane w przypadku standardowego użytkowania.

3.2.2 Okno komunikacyjne

Największa część modułu przechowująca wiadomości nadane i odebrane z urządzenia w następującym formacie:

- Wiadomość wysłana:

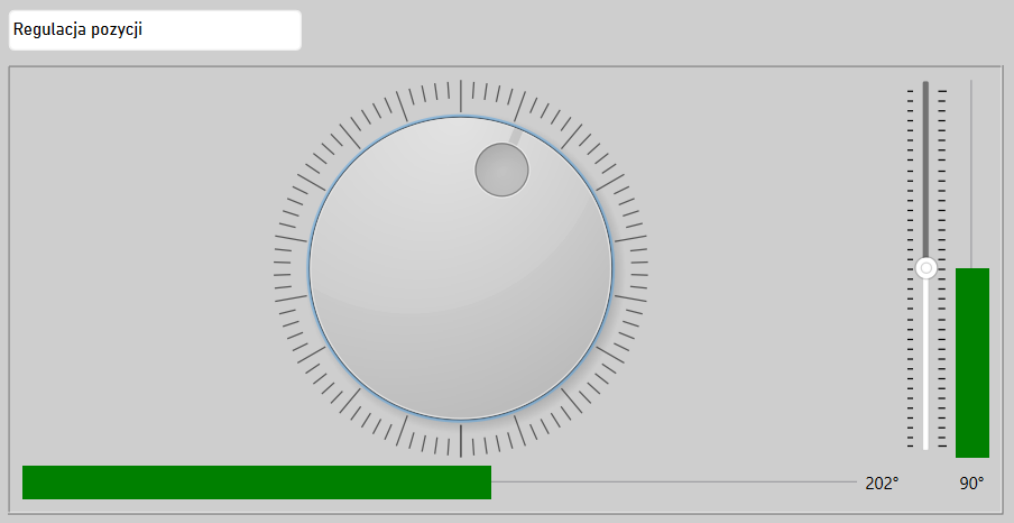
  $<<[\mathrm{Wiadomosc}] \quad <<[\mathrm{PozycjaX}:\mathrm{PozycjaY}] \quad \mathrm{np}:<<[100{:}50]$

- Wiadomość odebrana:

  $>>>[\mathrm{Wiadomosc}] \quad <<[\mathrm{PozycjaX}:\mathrm{PozycjaY}:\mathrm{OdczytSensora}] \quad \mathrm{np}:<<[100{:}50{:}255]$

W przypadku trybu demo logi pojawiające się w oknie posiadają przedrostek "DEMO". Te wiadomości nie są przesyłane na port szeregowy.

3.3 Regulacja pozycji



Rysunek 3: Moduł "Regulacja pozycji"

Moduł **[Regulacja pozycji]** służy do ustawiania i odczytywania pozycji urządzenia.

3.3.1 Odczytywanie pozycji

- *[Pasek pozycji X]* - Pozycję X reprezentuje poziomy pasek razem z wartością liczbową po prawej stronie.
- *[Pasek pozycji Y]* - Pozycję Y reprezentuje pionowy pasek razem z wartością liczbową pod nim.

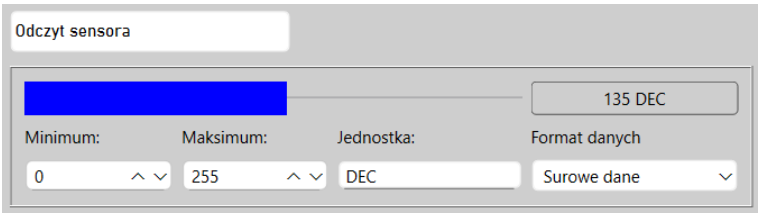
Gdy pasek jest zielony, wyświetlana pozycja odpowiada rzeczywistej pozycji urządzenia. Gdy pasek jest szary, wyświetlana pozycja odpowiada pozycji ustawionej przez użytkownika i aplikacja czeka na informację zwrotną o synchronizacji pozycji od urządzenia.

3.3.2 Ustawianie pozycji

- *[Pokrętło pozycji X]* - Służy do ustawiania poziomego kąta pozycji w zakresie $[0 - 359°]$.
- *[Suwak pozycji Y]* - Służy do ustawiania pionowego kąta pozycji w zakresie $[0 - 179°]$.

Informacja o ustawionej pozycji zostaje wysłana dopiero w momencie puszczenia elementu. W przypadku gdy odpowiadający elementowi pasek jest zielony, pozycja suwaka/pokrętła odpowiada rzeczywistemu nastawieniu urządzenia. W przypadku gdy po regulacji element przeskakuje na inną pozycję, należy zinterpretować to jako wystąpienie problemu z nastawą pozycji ze strony urządzenia i skontrolować logi komunikacyjne modułu **[Komunikacja szeregowa]**.

3.4 Odczyt sensora



Rysunek 4: Moduł "Odczyt sensora"

Moduł **[Odczyt sensora]** służy do odczytywania i formatowania wartości zmierzonej przez urządzenie dla określonej pozycji.

3.4.1 Odczytywanie danych

- *[Pasek odczytu sensora]* - Reprezentuje wartość wartość zmierzoną przez urządzenie ze skalą odpowiadającą ustawionemu formatowi danych. Kolor niebieski oznacza wartość aktualną a kolor szary wartość dla poprzedniej pozycji.
- *[Pole wartości liczbowej]* - Znajduje się po prawej stronie paska. Wyświetla zmierzoną wartość liczbową razem z jednostką odpowiadającą ustawionemu formatowi danych.

3.4.2 Format danych

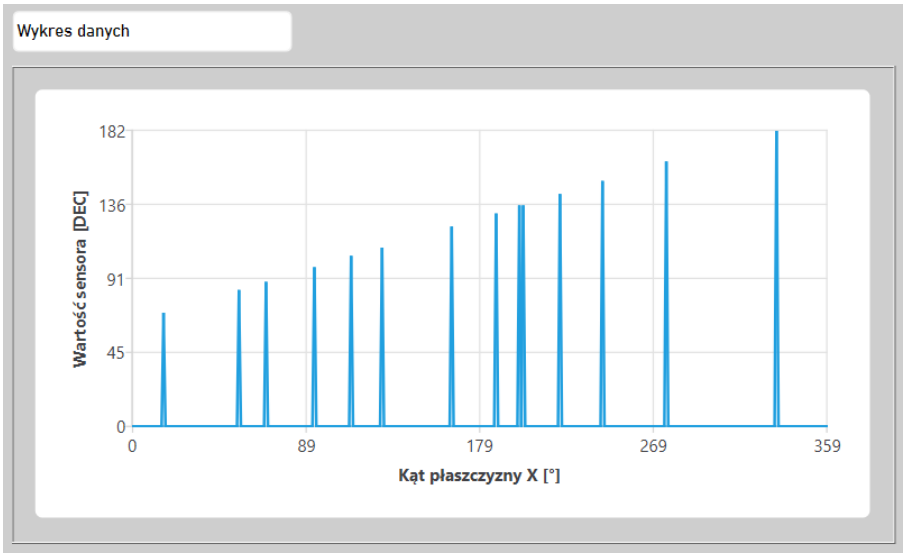
Menu podpisane *[Format danych]* umożliwia wybór sposobu konwersji i wyświetlania wartości odczytanej przez urządzenie pomiarowe dla aktualnie ustawionej pozycji:

- *[Surowe dane]* - Wyświetlane dane odpowiadają bezpośrednio wartości dziesiętnej 8-bitowego odczytu sensora - brak konwersji.
- *[Procent]* - Wyświetlane dane są przekonwertowane na wartość procentową.
- *[Napięcie 5V]* - Wyświetlane dane są przekonwertowane na zakres $0 - 5V$.
- *[Użytkownika]* - Odblokowuję możliwość ustawienia własnych parametrów konwersji:
  - *[Minimum]* Najmniejsza wartość mierzona. Możliwe jest ustawienie jedynie wartości dodatniej całkowitej mniejszej od maksymalnej.
  - *[Maksimum]* Największa wartość mierzona. Możliwe jest ustawienie jedynie wartości dodatniej całkowitej większej od minimalnej.
  - *[Jednostka]* Jednostka wyświetlana obok wartości. Liczba znaków jednostki jest dowolna ale ich zbyt duża ilość może wpłynąć na czytelność interfesju.

Uwaga: Przy zmianie formatu danych pamięć o zebranych pomiarach zostaje wyczyszczona. (Patrz **[Wykres danych]**).

Uwaga: Ustawienie wartości maksymalnej ponad 255 nie zwiększy dokładności pomiaru. (Rozdzielczość sensora pozostaje taka sama).

3.5 Wykres danych



Rysunek 5: Moduł "Wykres danych"

Moduł [**Wykres danych**] służy do zapamiętywania i wizualizacji dużej ilości pomiarów na jednej z osi pozycyjnych urządzenia.

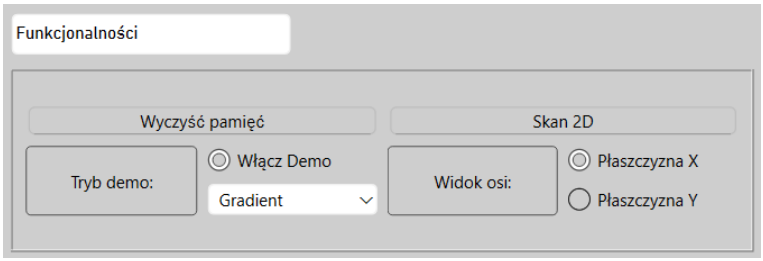
3.5.1 Oś pozioma

Przedstawia kąt płaszczyzny wybranej za pomocą modułu [**Funkcjonalności**] o odpowiadającym jej zakresie. Rysowany wykres prezentuje zmienne wartości kąta jednej z płaszczyzn przy stałej wartości kąta drugiej płaszczyzny. W przypadku nagłego zniknięcia wykresu należy upewnić się, że kąt płaszczyzny niebędącej na osi poziomej nie został zmieniony.

3.5.2 Oś pionowa

Przedstawia wartość zmierzoną dla kąta wybranej płaszczyzny. Wartość rysowana na wykresie jest sformatowana zgodnie z ustawieniami modułu [**Odczyt sensora**] przy czym wyświetlana wartość minimalna i maksymalna osi jest obliczana dynamicznie na podstawie uzyskiwanych pomiarów. Raz wykonany pomiar dla określonej pozycji urządzenia zostaje zapamiętany do momentu wyczyszczenia pamięci.

3.6 Funkcjonalności



Rysunek 6: Moduł "Funkcjonalności"

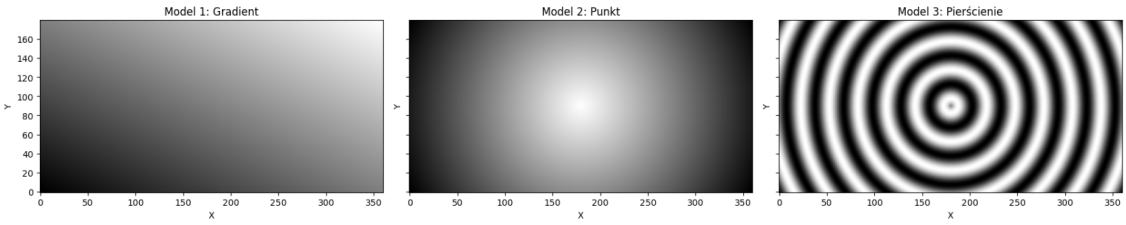
Moduł [**Funkcjonalności**] posiada funkcje związane z rysowaniem wykresu i trybem demo.

3.6.1 Wyczyść pamięć

Usuwa pamięć o wykonanych pomiarach dla wszystkich pozycji, jednocześnie czyści całą zawartość modułu [**Wykres danych**].

3.6.2 Tryb demo

- *[Włącz Demo]* - Przełącza aplikację w tryb demo. W trybie demo zamiast komunikacji z zewnętrznym urządzeniem, aplikacja wykonuje pomiary na podstawie zasymulowanego modelu wybranego w menu poniżej przycisku. Nazwy opcji reprezentują rozłożenie mierzonych wartości na płaszczyźnie XY:
  - *[Gradient]* Wartości rozłożone w formie gradientu rosnącego wraz z kątem poziomym i pionowym.
  - *[Punkt]* Punkt o epicentrum w $X = 180°$ oraz $Y = 90°$.
  - *[Pieścienie]* Pierścienie o równych odstępach przypominające obraz dyfrakcyjny.



Rysunek 7: Reprezentacja wizualna symulowanych modeli

3.6.3 Widok osi

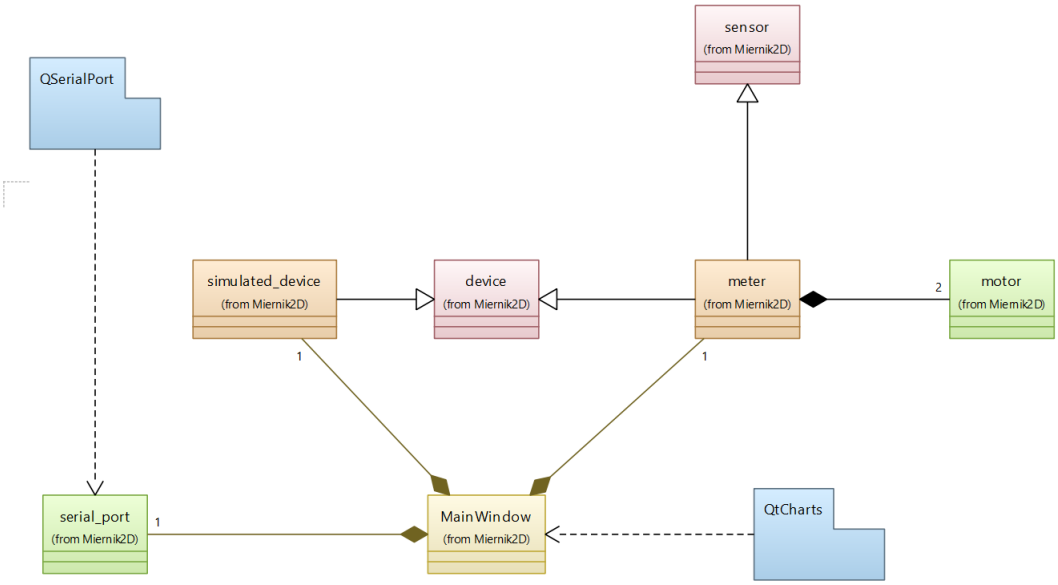
Umożliwia wybranie płaszczyzny widocznej w module [**Wykres danych**].

3.6.4 Skan 2D

Automatycznie wykonuje po jednym pomiarze na każdy stopień dla płaszczyzny wybranej w widoku osi. Aplikacja czeka na wiadomość zwrotną od urządzenia po każdej automatycznej nastawie. W trakcie skanu nie należy modyfikować interfejsu. Czas skanu może być bardzo duży w zależności od opóźnień w komunikacji z urządzeniem.

4 Dokumentacja techniczna

4.1 Zarys ogólny programu



Rysunek 8: Diagram UML

W przypadku modułów Qt należy sięgnąć do oficjalnej dokumentacji zewnętrznej.

4.2 Opis klas

4.2.1 Device

Klasa abstrakcyjna narzucająca metody do zarządzania pamięcią w postaci macierzy. Stworzona w celu oznaczenia klas urządzeń.

4.2.2 Motor

Klasa określająca element obracający się w określonym zakresie kątów.

- motor(int x, int y, int z) - obowiązkowa inicjalizacja obiektu o zakresie [x-y] kątów i startowej pozycji z.
- changePos(int x) - zmienia aktualną pozycję na wartość x. Posiada zabezpieczenie przed ustawieniem wartości poza zakres.
- readPos() - zwraca wartość aktualnej pozycji.

4.2.3 Sensor

Klasa określająca element mierzący wartość surową i konwertujący ją na wartość sformatowaną. Zawiera informację o jednostce mierzonej wartości.

- setVal(int x) - ustawia wartość wejściową. Posiada zabezpieczenie przed ustawieniem wartości poza zakres surowych danych wejściowych.
- readVal() - zwraca wartość wyjściową przekonwertowaną na podstawie pól OUTmin, OUTmax według następującego wzoru:

$$OUT_{min} + \frac{(value - IN_{min}) * (OUT_{max} - OUT_{min})}{IN_{max} - IN_{min}} \tag{1}$$

- setUnit(QString x) - ustawia jednostkę wartości wyjściowej. Użycie string zamiast char zwiększa swobodę w użyciu.
- readUnit() - zwraca jednostkę wartości wyjściowej.
- setOut[Min/Max]() - ustawia granice zakresu wartości wyjściowych.
- getOut[Min/Max]() - zwraca wartości granic zakresu wartości wyjściowych.

4.2.4 Meter

Główna klasa określająca urządzenie mierzące wartość w dwóch wymiarach. Stanowi złożenie dwóch elementów obracających (obiekty motor), elementu mierzącego (klasa przodek sensor) i dwuwymiarowej pamięci (klasa przodek device).

- setCords(int x, int y) - Ustawia pola pozycji obiektów typu motor na podane koordynaty x oraz y. Posiada zabezpieczenie przed wyjściem poza zakres dla obu obiektów.
- saveData() - zapisuje wartość zmierzoną dla aktualnej pozycji obiektów typu motor do macierzy data. Wartość zapisywana jest sformatowana według parametrów klasy sensor.
- getData(int x, int y) - zwraca wartość z macierzy data dla podanych koordynatów.
- clearData() - czyści pamięć wypełniając macierz data wartością 0.

4.2.5 Serial port

Klasa obsługująca dwustronną komunikację przez port szeregowy z wykorzystaniem modułu $< QSerialPort >$.

- connect(QString portName) - metoda określająca parametry komunikacji i otwierająca port o podanej nazwie. Zawiera zabezpieczenia sprzątające poprzednie połączenie i zwraca informacje o powodzeniu otwarcia.
- dataReady() - slot emitujący sygnał dataReceived(...) w przypadku odebrania wiadomości przez port.
- write(QByteArray data) - metoda wysyłające podane dane na port w formacie tablicy bajtów. Zawiera zabezpieczenia przed nadawaniem na pusty lub nieotwarty port
- serial port() - destruktor sprzątający połączenie w przypadku zamknięcia programu.

4.2.6 Simulated device

Klasa modelująca urządzenie zewnętrzne dla trybu demo. Umożliwia symulacje do pomiaru przez obiekt meter, w którym rolę otoczenia pełni macierz data wypełniona ustalonymi danymi.

- generateData(int x) - wypełnia macierz pamięci wartościami według formuły matematycznej wybranej za pomocą zmiennej x: (1-gradient, 2-punkt, 3-pierścienie).
- getData[(QString text)/(int x, int y)] - metoda przeciążona. Dla dwóch zmiennych typu int zwraca wartość z macierzy na koordynatach xy. Dla tekstu działa w sposób symulujący dynamikę komunikacji szeregowej: otrzymuje wiadomość w formacie wyjściowym aplikacji "x:y", konwertuje wiadomość na koordynaty, po czym zwraca wartość w macierzy w formacie wejściowym aplikacji "x:y:wartość".
- saveData() - w aktualnym stanie nieużywana, deklaracja wymuszona przez klasę abstrakcyjną device.
- clearData() - czyści pamięć wypełniając macierz data wartością 0.

4.3 Opis kluczowych metod klasy "mainwindow"

4.3.1 Konstruktor i Destruktor

- setupChart() - metoda ustawiająca dane inicjalizacyjne wykresu.
- killChart() - metoda usuwujące obiekty związane z wykresem.

4.3.2 Komunikacja i dodatkowa logika

- loadPorts() - ładuje porty do wyświetlenia w module [**Komunikacja Szeregowa**].
- readData() - obszerna metoda odpowiedzialna z odbieranie, dekodowanie i zapis wiadomości zwrotnej o formacie "x:y:wartość".
- sendPos() - koduje i wysyła wartości o nastawie pozycji na port szeregowy lub symulowane urządzenie demo.

4.3.3 Metody pomocnicze

- setFormat() - metoda do szybkiego uzupełniania elementów UI związanych z formatowaniem sensora.
- updatePos() - metoda do aktualizacji elementów UI związanych z odczytem pozycji i wartości sensora.
- refreshChart() - metoda odświeżająca wykres.