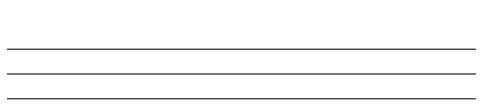


PROJEKT
WIZUALIZACJA DANYCH SENSORYCZNYCH

**Wizualizacja danych z czujników otoczenia
robota mobilnego "Wędrowiec"**

Michał Dołharz, 248943



Prowadzący:
dr inż. Bogdan Kreczmer

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki
Politechniki Wrocławskiej

Spis treści

1	Charakterystyka tematu projektu	1
2	Podcele i etapy realizacji projektu	1
3	Specyfikacja finalnego produktu	2
4	Harmonogram prac	3
4.1	Terminarz realizacji poszczególnych podcelów	3
4.2	Kamienie milowe	4
5	Projekt graficznego interfejsu użytkownika	4
6	Schemat ideowy	5
6.1	Czujniki HC-SR04	5
6.2	Zabezpieczenia zasilania	6
6.2.1	Odwrotna polaryzacja	6
6.2.2	Zbyt duże natężenie prądu	6
6.2.3	Filtracja zasilania	6
7	Obsługa jednego czujnika	8
7.1	Opis układu	8
7.2	Komunikacja z aplikacją	8
7.3	Protokół komunikacyjny OSOS	10
7.4	Działanie układu i testy	10
	Spis rysunków	12
	Bibliografia	12

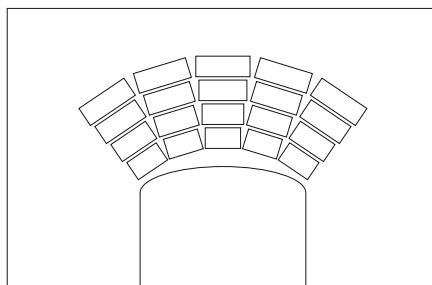
1 Charakterystyka tematu projektu

Celem projektu jest stworzenie aplikacji ostrzegającej o przeszkodach w otoczeniu robota mobilnego w oparciu o biblioteki Qt [12] języka C++, czujniki ultradźwiękowe HC-SR04 [5] oraz mikrokontroler Arduino Nano [1]. Wizualizacja ma przypominać wizualizację danych z czujników samochodu, wyświetlaną na wewnętrznym, wbudowanym ekranie.

Robot mobilny o kryptonimie "Wędrowiec" będzie budowany w ramach projektu z kursu Roboty mobilne. W założeniu ma obsługiwać dwa tryby pracy: autonomiczny, gdzie po wykryciu przeszkody na drodze zmienia trajektorię ruchu oraz zdalnie sterowany z pilota budowanego również w ramach tego kursu. Integracja obu projektów przewidziana jest jako ostatni etap ich realizacji.

Projekt samej aplikacji będzie zależał od wyników testów czujników. W najbardziej optymistycznej wersji aplikacja będzie obsługiwała trzy widoki: czujniki przednie (pięć czujników jak na rysunku 1), czujniki tylne (analogicznie jak przednie, ale widok odwrócony w poziomie) oraz całe otoczenie pojazdu. Odpowiednie widoki będą wywoływanie automatycznie przy wykryciu przeszkody.

Przy tworzeniu aplikacji zostaną wykorzystane materiały pomocnicze dostępne w Internecie [2, 3, 4, 10, 11]. Podobnie przy projektowaniu schematu ideowego oraz programowaniu mikrokontrolera zostaną wykorzystane internetowe źródła wiedzy [14, 15], a także materiały drukowane [7, 8].



Rysunek 1: Rysunek koncepcyjny okna aplikacji

2 Podcele i etapy realizacji projektu

Każdy etap budowy aplikacji lub platformy czujników wymieniony poniżej będzie podlegał serii testów. Po ich analizie korygowane będą rzeczy takie jak ustawienie czujników, oprogramowanie czy kod aplikacji. Wyniki niektórych testów będą dodatkowo podstawą do dalszych konfiguracji i doboru optymalnych parametrów.

Podział na główne etapy prezentuje się następująco:

1. Zebranie literatury na tematy powiązane z tematem projektu.
2. Projekt układu elektronicznego (schemat ideowy). Projekt interfejsu graficznego.
3. Tworzenie aplikacji oraz testowanie czujników jeszcze bez platformy:
 - (a) Stworzenie wstępnej wersji aplikacji komunikującej się z mikrokontrolerem.
 - (b) Rozbudowa aplikacji o obsługę jednego czujnika. Wyświetlanie tekstowe danych.

- (c) Rozbudowa aplikacji o obsługę pięciu czujników.
 - (d) Rozbudowa aplikacji o graficzną wizualizację danych.
 - (e) Testowanie modułu czujnika VL53L1X [13] pod kątem użycia jako czujnika wspomagającego.
4. Budowa platformy czujników.
 5. Rozbudowa aplikacji o obsługę przednich oraz tylnych czujników.
 6. Integracja platformy z robotem mobilnym.
 - (a) Prawdopodobne poprawki oprogramowania.
 - (b) Możliwe poprawki ustawienia czujników.
 - (c) Testy gotowego urządzenia.

3 Specyfikacja finalnego produktu

Finalna wersja aplikacja powinna charakteryzować się:

- poprawnym połączeniem z urządzeniem oraz poprawną komunikacją,
- poprawną interpretacją i graficzną wizualizacją danych pochodzących z czujników,
- obsługą dziesięciu niezależnych czujników HC-SR04,
- prostym, intuicyjnym i automatycznym interfejsem (ewentualnie z minimalną interakcją użytkownika) składającym się z trzech widoków: przedniego, tylnego, oraz całościowego.

Wykonana platforma czujników (bez integracji z robotem mobilnym) powinna charakteryzować się:

- zamocowaniem dziesięciu czujników HC-SR04,
- zasięgiem dobranym jako optymalny w trakcie testów czujników,
- możliwością przymocowania do robota mobilnego (a zatem dostosowanie do niego pod względem rozmieszczenia elementów),

Robot mobilny zintegrowany z platformą czujników powinien charakteryzować się:

- odmienną reakcją w zależności od trybu pracy i czujników wykrywających przeszkode,
- zredukowaną maksymalną prędkością w trakcie wizualizacji (kwestia bezpieczeństwa ze względu na komunikację przewodową).

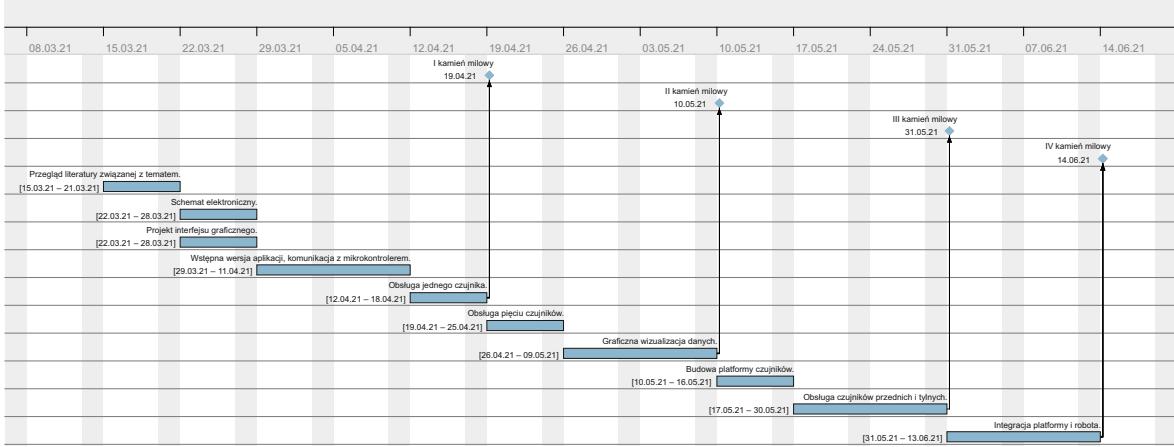
4 Harmonogram prac

4.1 Terminarz realizacji poszczególnych podcelów (z dokładnością do 1 tygodnia)

Każdy tydzień pracy w poniższym rozkładzie oprócz opisanych czynności zakłada testowanie wprowadzanych rozwiązań na bieżąco.

Harmonogram prac w rozkładzie tygodniowym prezentuje się następująco (podkreślone terminy oznaczają kolejne terminy oddania raportów z postępu prac nad projektem):

- 22 marca 2020 – zakończenie przeglądu materiałów związanych z tematem.
- 29 marca 2020 – zaprojektowanie schematu układu elektronicznego z uwzględnieniem wersji z jednym czujnikiem, pięcioma oraz dziesięcioma oraz zaprojektowanie interfejsu graficznego.
- 12 kwietnia 2020 – wstępna wersja aplikacji, komunikacja z mikrokontrolerem.
- 19 kwietnia 2020 – rozbudowanie aplikacji o obsługę jednego czujnika (tekstowe wyświetlanie danych).
- 26 kwietnia 2020 – rozbudowanie aplikacji o obsługę pięciu czujników (bez uwzględniania ich rozmieszczenia na platformie lub robocie).
- 4 maja 2020 – praca nad rozbudowaniem aplikacji o graficzną wizualizację danych (wersja uproszczona graficznie).
- 10 maja 2020 – stworzenie graficznej wizualizacji danych z uwzględnieniem przyszzej implementacji dodatkowych okien widoków oraz rozmieszczenia czujników na platformie.
- 17 maja 2020 – zbudowanie platformy czujników z uwzględnieniem przyszłego mocowania na robocie.
- 24 maja 2020 – praca nad rozbudowaniem aplikacji o obsługę czujników przednich i tylnych (łącznie dziesięć czujników).
- 31 maja 2020 – zaimplementowanie obsługi wszystkich dziesięciu czujników oraz trzech okien widoków i ich automatycznego wywoływania.
- 7 czerwca 2020 – praca nad integracją platformy i robota. Fizyczne mocowanie platformy do robota. Przeprojektowanie (połączenie) schematów ideowych.
- 14 czerwca 2020 – Przeprogramowanie mikrokontrolerów robota i platformy do komunikacji i współpracy. Testy gotowego urządzenia w obu trybach robota.



Rysunek 2: Diagram Gannta wykonywanych prac

4.2 Kamienie milowe

w harmonogramie prac nad projektem wyszczególnione są cztery kamienie milowe:

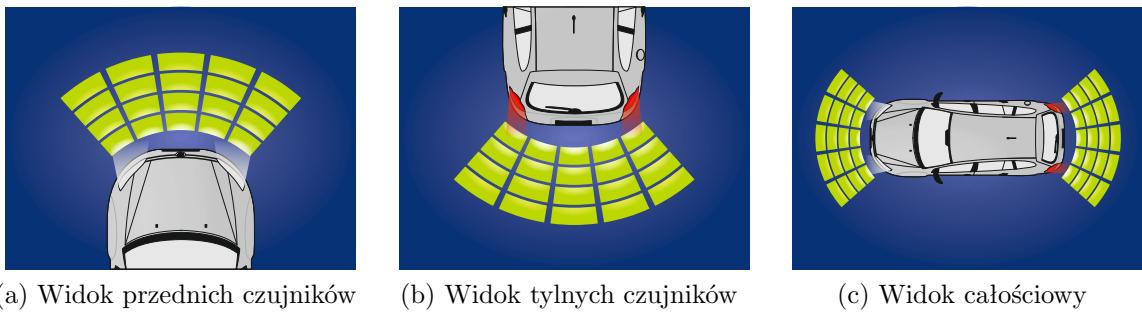
1. Pierwszy kamień milowy zakłada stworzenie aplikacji zdolnej do komunikacji z mikrokontrolerem oraz tekstowym wyświetlaniem danych pochodzących z jednego czujnika.
2. Drugi kamień milowy zakłada rozbudowanie aplikacji do obsługi pięciu czujników (nieprzymocowanych do platformy) i graficznego wyświetlania danych. Na tym etapie gotowe jest okno widoku przednich czujników.
3. Trzeci kamień milowy zakłada rozbudowę aplikacji o obsługę czujników tylnych oraz budowę właściwej platformy czujników. Na tym etapie aplikacja jest gotowa.
4. Czwarty kamień milowy zakłada połączenie robota mobilnego i platformy czujników. Robot jest odpowiednio przeprogramowany, aby jego reakcja zależała od odpowiednich czujników.

5 Projekt graficznego interfejsu użytkownika

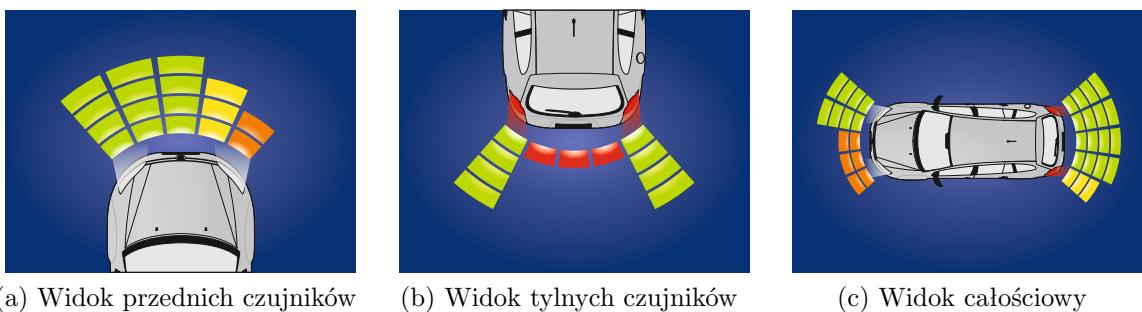
Parkując samochód kierowca trzyma obie ręce na kierownicy i rozgląda się przez okna samochodu badając otoczenie. Informacje wyświetlane na ekranie mają służyć pomocą, muszą więc być automatyczne.

Interfejs składa się z trzech okien, nazywanych widokami. Jest w pełni automatyczny, co oznacza, że odpowiednie widoki są włączane w zależności od czujników, które wykryją przeszkodę. Czujniki są podzielone na przednie (rys. 3a) i tylne (rys. 3b). Wystarczy jeden czujnik z grupy, aby aktywować odpowiadający mu widok. Jeżeli przeszkody są wykrywane zarówno przez czujniki przednie, jak i czujniki tylne, wtedy włącza się widok całościowy (rys. 3c).

Poszczególne rysunki z rysunku 3 przedstawiają widoki bez wykrycia żadnej przeszkody. Takie obrazy oczywiście nigdy nie zostaną wyświetcone, ponieważ warunkiem wyświetlania widoku jest wykrycie przeszkody. Przykładowe obrazy, które mogą zostać wyświetcone, są przedstawione na poszczególnych rysunkach 4a, 4b oraz 4c.



Rysunek 3: Trzy widoki (okna aplikacji)



Rysunek 4: Przykłady działania aplikacji

Zielony wskaźnik oznacza brak wykrytej przeszkody. Wskaźniki żółty, pomarańczowy oraz czerwony oznaczają wykryte przeszkody w programowo ustalonych zakresach (zakresy te zostaną dobrane w trakcie testów). Takie rozwiązanie zapewnia kierowcy możliwość natychmiastowego przyswojenia informacji wyświetlanych na ekranie.

W zamyśle aplikacja nie powinna wyświetlać żadnego okna, gdy żaden z czujników nie wykrywa przeszkody, umożliwiając działanie innych funkcjonalności ekranu samochodowego. Projekt ten nie zawiera jednak innych funkcjonalności, w związku z czym w trakcie bezczynności będzie wyświetlane czarne tło.

6 Schemat ideowy

Schemat ideowy układu prezentuje się na rysunku 5.

6.1 Czujniki HC-SR04

Wejścia Trig czujników są podpięte do osobnych wyprowadzeń mikrokontrolera. Wyjścia Echo są ze sobą zwarte poprzez diody przełączające 1N4148 [16] i podpięte pod jeden pin skonfigurowany jako przerwania zewnętrzne.

Impulsy zwrotne przychodzą na wspólny pin, ale wiedząc do którego czujnika został właśnie wysłany sygnał, można z łatwością przypisać sygnał zwrotny do odpowiedniego czujnika. Takie rozwiązanie pozwala na ograniczenie liczby wykorzystywanych pinów.

6.2 Zabezpieczenia zasilania

6.2.1 Odwrotna polaryzacja

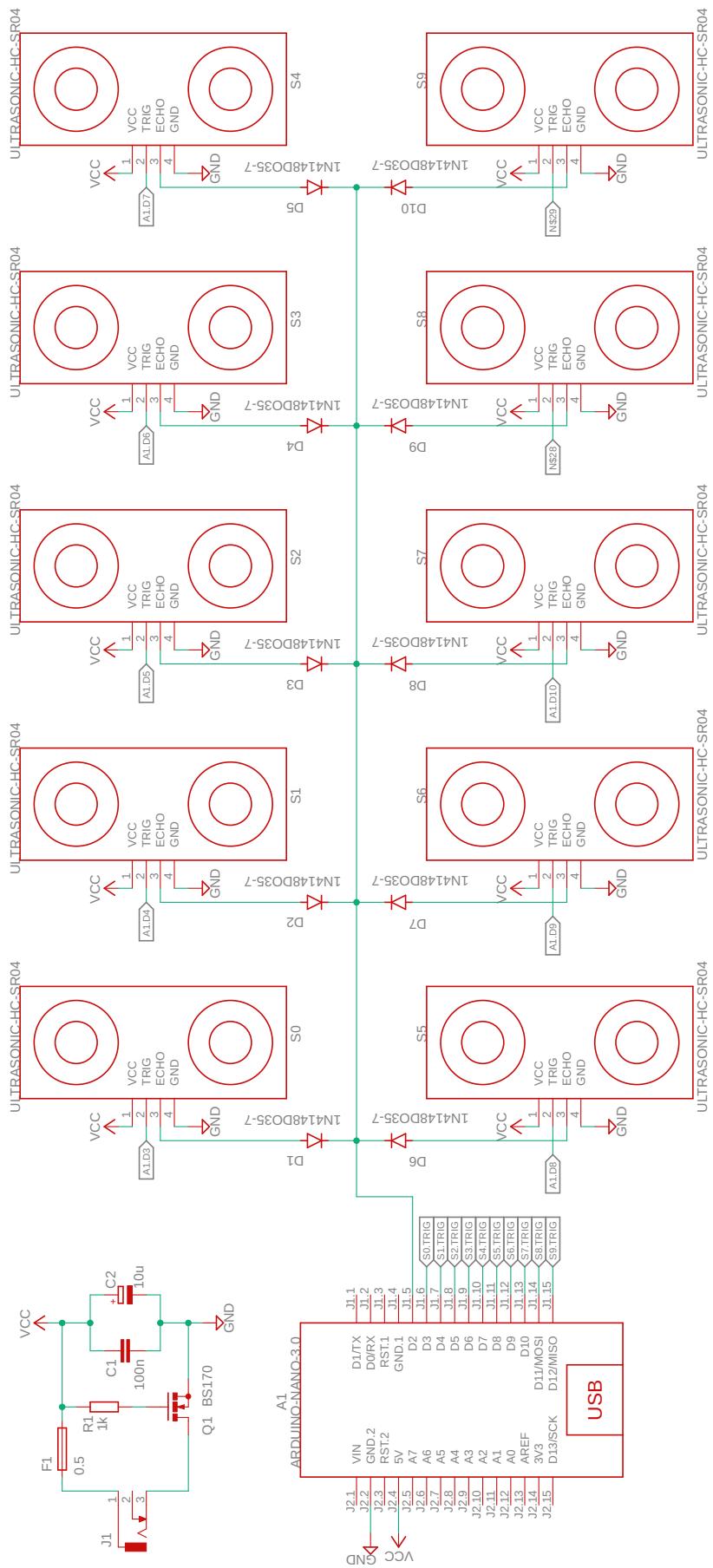
Do zabezpieczenia układu przed odwrotną polaryzacją [6] jest użyty tranzystor unipolarny MOSFET BS170 [9]. Tranzystor ten cechuje się prądem drenu 300 mA przy napięciach bramka-źródło oraz dren-źródło równych 5 V. Jest to wartość wystarczająca, ponieważ przewidywany pobór prądu dziesięciu czujników ($10 \cdot 15 \text{ mA} = 150 \text{ mA}$) wraz z Arduino Nano ($< 50 \text{ mA}$) nie powinien zbliżyć się do tej wartości.

6.2.2 Zbyt duże natężenie prądu

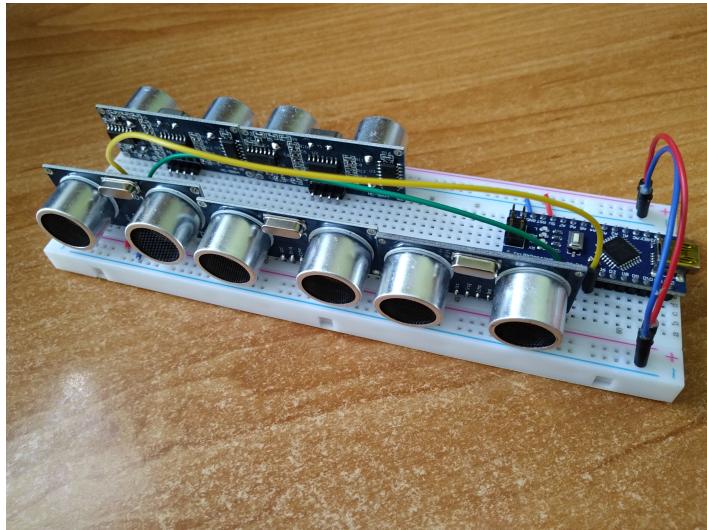
W celu ochrony układu przed zbyt dużym natężeniem prądu wykorzystany jest bezpiecznik topikowy 0.5 A.

6.2.3 Filtracja zasilania

Do filtrowania zasilania jest wykorzystywana popularna para kondensatorów ceramicznego 100 nF wraz z elektrolitycznym 10 μF .



Rysunek 5: Schemat ideowy układu



Rysunek 6: Układ zmontowany na potrzeby tego etapu projektu

7 Obsługa jednego czujnika

7.1 Opis układu

Omawiany układ składa się z mikrokontrolera Arduino Nano oraz jednego czujnika HCSR-04. Na prezentującym układzie zdj. 6 znajduje się więcej czujników, ale są niepodłączone, stanowią jedynie przygotowanie do kolejnego etapu projektu. Nie jest zastosowane również żadne zabezpieczenie zasilania, ponieważ zasilanie jest pobierane przez przewód micro USB, którym również jest wykonywana komunikacja UART z komputerem.

7.2 Komunikacja z aplikacją

Gdyby aplikacja była wgrana do komputera samochodowego lub innego urządzenia obsługującego wyświetlacz samochodowy, konfiguracja komunikacji mogłaby być zaprogramowana na stałe. W przypadku podłączenia urządzenia do komputera wielofunkcyjnego niezbędna jest możliwość wyboru portu, do którego podłączona jest platforma.

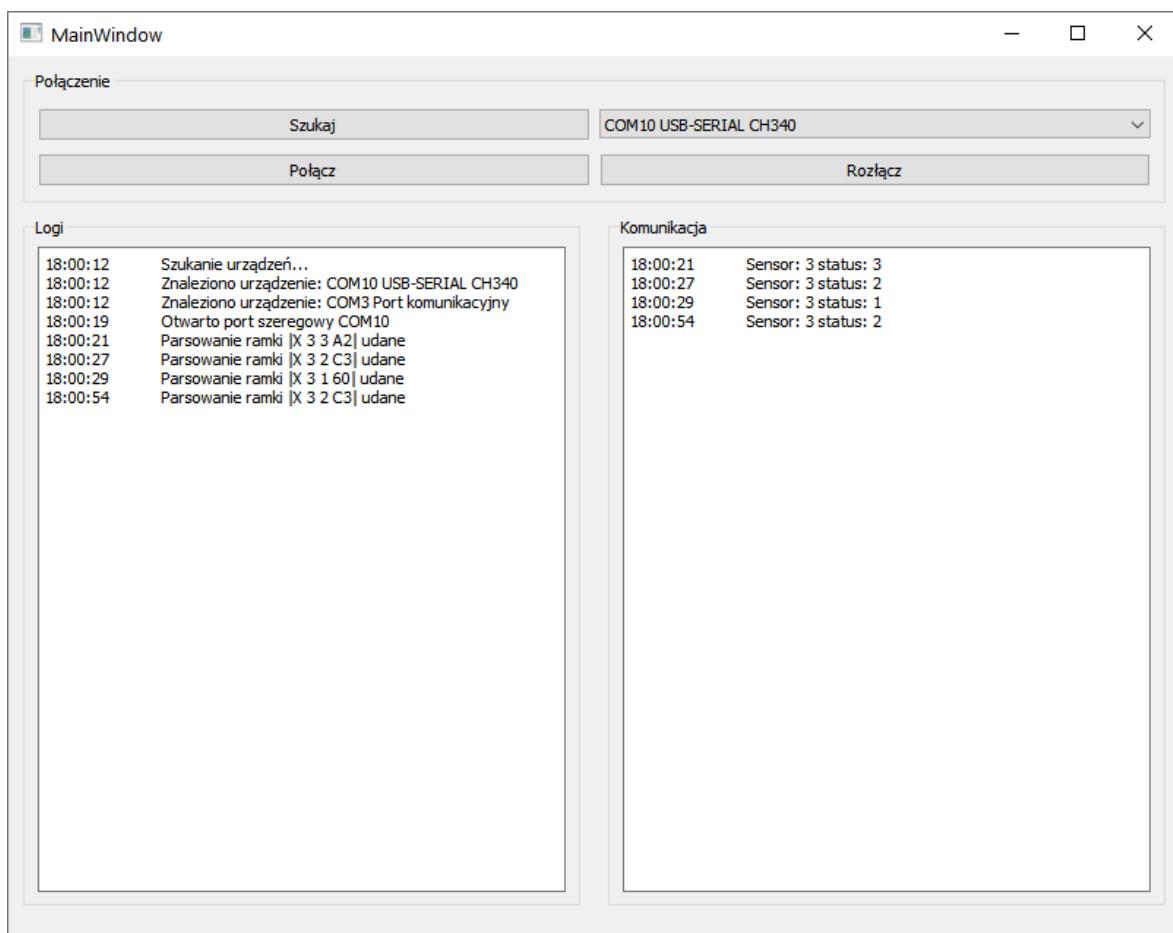
Na tym etapie projektu aplikacja pozwala na wyszukiwanie dostępnych portów COM, a także do łączenia i rozłączania się z nimi. Służą do tego trzy przyciski:

- **Szukaj** – wyszukuje dostępne urządzenia na portach COM. Znalezione urządzenia są wyświetlane w rozwijanej zakładce obok.
- **Połącz** – powoduje próbę połączenia z urządzeniem wybranym w zakładce.
- **Rozłącz** – zamyka połączenie z urządzeniem.

Informacje wyświetlane są w dwóch oknach tekstowych:

- **Logi** – informacje dotyczące działania komunikacji, jak np. udane lub nieudane połączenie, udane parsowanie ramki, itd.
- **Komunikacja** – informacje użytkowe, czyli jaki jest nowy status i na którym czujniku.

Okno programu prezentuje rysunek 7.



Rysunek 7: Okno aplikacji.

Tablica 1: Parametry układu

Status	Strefa	Zakres od [cm]	Zakres do [cm]
0	zielona	20	–
1	żółta	15	20
2	pomarańczowa	10	15
3	czerwona	5	10
4	zmiana tła	0	5

7.3 Protokół komunikacyjny OSOS

Platforma czujników komunikuje się z aplikacją za pomocą UART. Gdy dane pochodzące z czujnika wskazują na zmianę statusu, mikrokontroler wysyła odpowiedni komunikat. Dane wysyłane są w postaci ramki, której budowa znajduje się na rysunku 8.



Rysunek 8: Ramka protokołu komunikacji OSOS

Komunikacja jest znakowa. Ramka rozpoczyna się od znaku 'X'. Następnie, po spacji, znajduje się numer sensora, którego komunikat dotyczy (Sn). Po kolejnej spacji następuje numer statusu (St). Na koniec, również po spacji, znajduje się suma kontrolna CRC8 w zapisie heksadecymalnym, zajmująca dwa znaki w ramce.

Ze względu na zawartość pojedynczej ramki danych, protokół otrzymał nazwę OSOS (*One Sensor One Status*)

7.4 Działanie układu i testy

Do sprawdzania poprawności pomiarów czujnika używana jest plansza pomiarowa z zaznaczonymi odległościami (strefami) i odpowiadającymi im kolorami statusów. Parametry, przy których testowany jest czujnik, tzn. konfiguracja poszczególnych statusów czujnika znajduje się w tabeli 1.

Jak widać na zdjęciu 9 prezentującym planszę pomiarową, został wprowadzony dodatkowy, piąty status. Status ten nie wprowadza kolejnego kolorowego bloczku, jego działanie polega na zmianie tła aplikacji z niebieskiej na czerwoną w momencie krytycznym, a więc gdy przeszkoda zbliżała się do dowolnego czujnika bardzo blisko (w testowanym układzie jest to odległość od 0 cm do 5 cm, zgodnie z tabelą 1). Dokładniej zostanie to opisane w dalszej części.

Plansza pomiarowa daje dwie możliwości interpretacji informacji o umieszczeniu przeszkody, chociaż tak naprawdę obie mówią o tym samym. Z prawej strony znajduje się informacja, co zostanie wyświetlane w aplikacji, jeżeli w danej strefie znajdzie się przeszkoda. Z lewej strony znajduje się informacja o zasięgu, tzn. pierwsza strefa od góry, na której nie znajduje się przeszkoda, informuje strzałką o wyświetlanych blokach.

Wyświetlanie danych w aplikacji ma być intuicyjne, w związku z czym jest inspirowana działaniem światła. Oznacza to, że miejsce, w którym wykryto przeszkodę, jest już niedostępne (światło odbija się od przeszkody, nie wpada w nią). Innymi słowy, wyświetlany blok oznacza, że żadna przeszkoda się tam nie znajduje. Dopiero gdy zniknie



Rysunek 9: Układ z planszą pomiarową

oznacza to, że w tym miejscu znalazła się przeszkoda. Z tego powodu został dodany piąty status czujnika. Przykład: aplikacja wyświetla dwa pomarańczowe bloki. Oznacza to, że przeszkoda znajduje się w strefie pomarańczej (status czujnika 2), czyli w miejscu, gdzie znajdowałby się kolejny blok, a nie w miejscu, gdzie wyświetlany jest drugi pomarańczowy blok.

Rysunki 10a oraz 10b przedstawiają wizualizację wraz z przeszkodami. Przeszkody są dodane tylko na potrzeby przykładu, nie będą wyświetlane w gotowej aplikacji. Rysunek 10b dodatkowo przedstawia piąty status czujnika. W myśl inspiracji światłem, niewyświetlany blok oznacza obecność przeszkody. W przypadku statusu piątego samo zniknięcie bloku nie było zbyt ekspresywne. Zmiana tła z niebieskiego na czerwony pozwala na znaczne podkreślenie wagi wykrytej przeszkody (sytuacja krytyczna, przeszkoda bardzo blisko). Rysunek 10c prezentuje tę samą sytuację, co rysunek 10b, ale bez przeszkody.



Rysunek 10: Przykłady wykrywania przeszkód

Spis rysunków

1	Rysunek koncepcyjny okna aplikacji	1
2	Diagram Gantta wykonywanych prac	4
3	Trzy widoki (okna aplikacji)	5
4	Przykłady działania aplikacji	5
5	Schemat ideowy układu	7
6	Układ zmontowany na potrzeby tego etapu projektu	8
7	Okno aplikacji.	9
8	Ramka protokołu komunikacji OSOS	10
9	Układ z planszą pomiarową	11
10	Przykłady wykrywania przeszkód	12

Bibliografia

- [1] Arduino. [Arduino Nano \(V3.0\) User Manual](#).
- [2] Cairns B. [Qt 5 Core Advanced with C++](#). 2020. URL: www.udemy.com/course/qt-core-advanced.
- [3] Cairns B. [Qt 5 Core for Begginers with C++](#). 2020. URL: www.udemy.com/course/qt-core-for-beginners.
- [4] Cairns B. [Qt 5 Core Intermediate with C++](#). 2020. URL: www.udemy.com/course/qt-core-intermediate.
- [5] Elecfreaks. [HC-SR04 Datasheet](#).
- [6] elektroda.pl. [Zabezpieczenie przed odwróceniem polaryzacji zasilania #24](#) edu elektroda.pl. URL: <https://youtu.be/Yy-VR0PhQaQ>.
- [7] Kardaś M. [Język C. Pasja programowania mikrokontrolerów 8-bitowych](#). Wydanie II poprawione i uzupełnione. Szczecin: Atnel, 2014.
- [8] Kardaś M. [Mikrokontrolery AVR Język C – podstawy programowania](#). Wydanie II poprawione i uzupełnione. Szczecin: Atnel, 2013.
- [9] Motorola. [BS170 Datasheet](#).
- [10] Munteanu D. [Robust Qt & C++ Gui Programming 2D Graphics App Tutorial](#). 2020. URL: www.udemy.com/course/qt5-gui-cpp-programming-tutorial-2d-graphics.

- [11] Patyk M. Kurs Qt. 2021. URL: www.forbot.pl/blog/kurs-qt-1-czym-jest-qt-pierwsza-aplikacja-w-praktyce-id35549.
- [12] Qt Development Frameworks. Strona Internetowa biblioteki Qt. URL: www.qt.io.
- [13] STMicroelectronics. VL53L1X Datasheet.
- [14] Szymański D. Kurs Arduino. Poziom I. 2021. URL: www.forbot.pl/blog/kurs-arduino-podstawy-programowania-spis-tresci-kursu-id5290.
- [15] Szymański D. Kurs Arduino. Poziom II. 2021. URL: www.forbot.pl/blog/kurs-arduino-ii-wstep-spis-tresci-id15494.
- [16] Vishay Intertechnology. 1N4148 Datasheet.