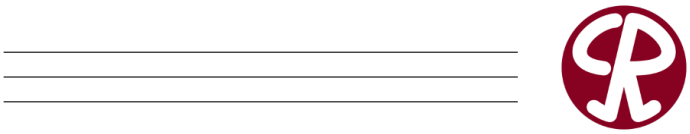
Projekt

Wizualizacja Danych Sensorycznych

Wizualizacja danych z drona

Jakub Delicat, 259326

Eryk Możdżeń, 259375



Prowadzący:
dr inż. Bogdan Kreczmer

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki, Fotoniki i Mikrosystemów
Politechniki Wrocławskiej

12 marca 2023

Spis treści

1 Charakterystyka tematu projektu

Projekt zakłada stworzenie oprogramowania aplikacji okienkowej, wizualizującej dane sensoryczne pochodzące z aktualnie rozwijanego bezzałogowego obiektu latającego „Goose” [2]. Podstawowym celem programu jest zapewnienie użytkownikowi/pilotowi szybkiego i przystępnego dostępu do danych telemetrycznych oraz ich zwizualizowanie w celu ich łatwiejszej weryfikacji. Z uwagi na to, że projekt dotyczy obiektu szybkozmiennego, należy zwrócić szczególną uwagę na wystarczająco wysoką częstotliwość transferu danych oraz ich wpływ na interfejs.

2 Specyfikacja finalnego produktu

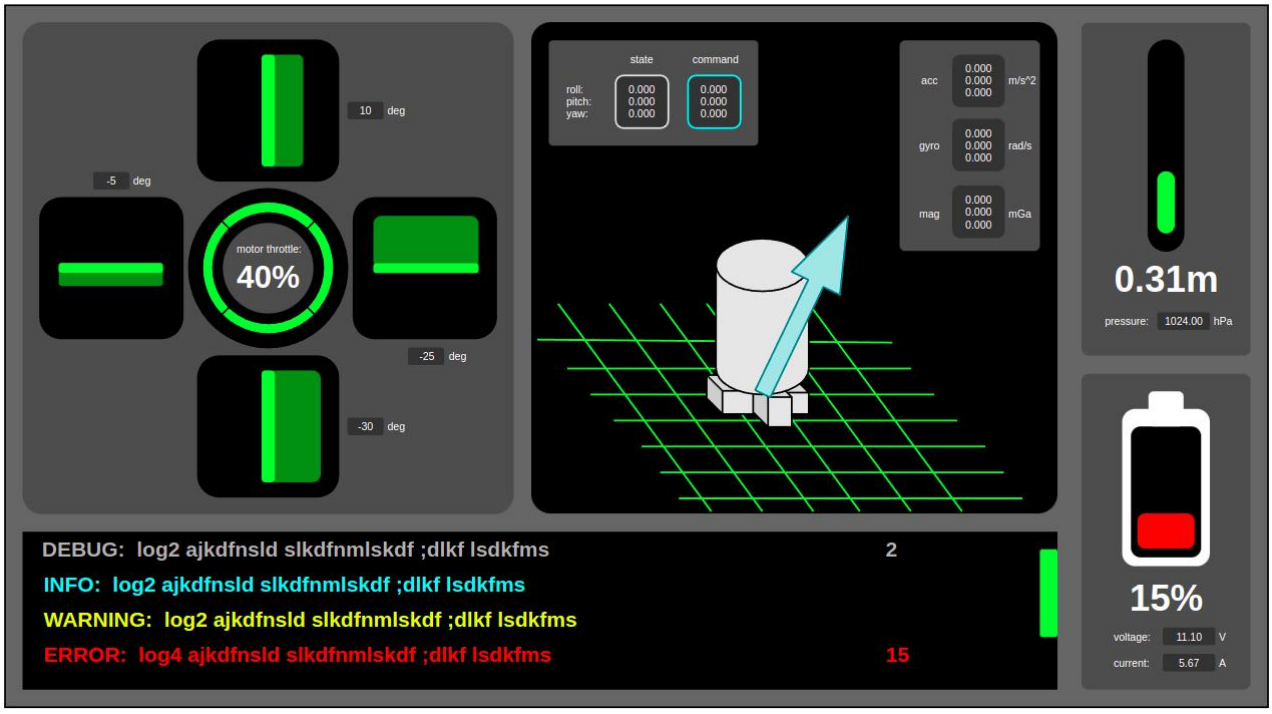
Wymagania sprzętowe

Finalna wersja aplikacji powinna pracować oraz kompilować się na komputerach wyposarzonych w system operacyjny Linux Ubuntu (wersja co najmniej 20.04). Proces kompilacji powinien być zautomatyzowany narzędziem Makefile wraz z CMake.

Interfejs użytkownika

Interfejs użytkownika powinien umożliwiać wgląd do kluczowych danych liczbowych odebranych od drona:

- komunikaty tekstowe
- odczyty czujników
- wartości zadawane na elementy wykonawcze (wirnik, łopatki)
- wartości estymowane (poziom baterii, wysokość nad ziemią, orientacja)



Rysunek 1: Docelowy układ elementów w oknie

Komunikaty tekstowe

Okienko wyświetlające komunikaty tekstowe powinno umożliwiać użytkownikowi swobodny wgląd wstecz do 50 różnych komunikatów. W przypadku, gdy dany komunikat pojawia się wielokrotnie, powinien pojawiać się przy nim licznik jego wystąpień. Kolor, w którym wyświetlany jest komunikat powinien umożliwiać jednoznaczne wskazanie jego typu.

Elementy wykonawcze

Rozkład elementów wizualizujących elementy wykonawcze powinien odzwierciedlać ich docelowe położenie względem siebie w fizycznej konstrukcji pojazdu. Dopuszczalnym polem pracy serwomechanizmów jest zakres $\pm 15\,^{\circ}$. Wychylenie każdego z nich powinno być reprezentowane zmianą szerokości prostokąta w taki sposób, aby naśladować wygląd w przekroju z góry. Wartość przepustnicy głównego silnika napędowego (prędkość wirnika) zadana jest wartością od 0 do 100 %. Rotor powinien być reprezentowany przez wirujący pierścień oznakowany tak, aby możliwy był zgrubny odczyt jego prędkości kątowej.

Scena 3D

Okno powinno prezentować model 3D w orientacji odpowiadającej danym otrzymanym z obiektu wraz z elementami ułatwiającymi orientacje w przestrzeni (elementy odniesienia). Okno powinno prezentować odczyty akcelerometru, żyroskopu, magnetometru z dokładnością do trzech miejsc po przecinku oraz zapewniać miejsce na trzy cyfry przed przecinkiem wraz z ciągłym wyświetlaniem znaku. Obecna powinna być także liczbowa reprezentacja aktualnej oraz zadanej orientacji w kątach RPY zaokrąglona do pełnych stopni z ciągłym wyświetlaniem znaku.

Wysokościomierz

Wysokościomierz przedstawiony powinien być jako słupek mogący zmieniać swoją wysokość. Pełne wypełnienie oznaczać będzie osiągnięcie lub przekroczenie 2.5 m. Wartość ciśnienia powinna być wyrażona w hPa z dokładnością do trzech miejsc po przecinku.

Wskaźnik poziomu baterii

Wskaźnik poziomu baterii reprezentowany przez słupek zmieniający swoją wysokość jak i kolor w zależności od stopnia naładowania. Wraz ze spadkiem poziomu naładowania kolor powinien stopniowo przechodzić z neutralnego w jaskrawy. Wartości prądu i napięcia powinny być podane z dokładnością do dwóch miejsc dziesiętnych odpowiednio w woltach i amperach.

Budowa programu

Wewnętrzna budowa programu powinna być zaprojektowana tak, aby umożliwiać łatwe dodawanie nowych funkcjonalności – innymi słowy powinna być łatwo skalowalna.

3 Podcele i etapy realizacji projektu

Lista podcelów (Eryk Możdżeń):

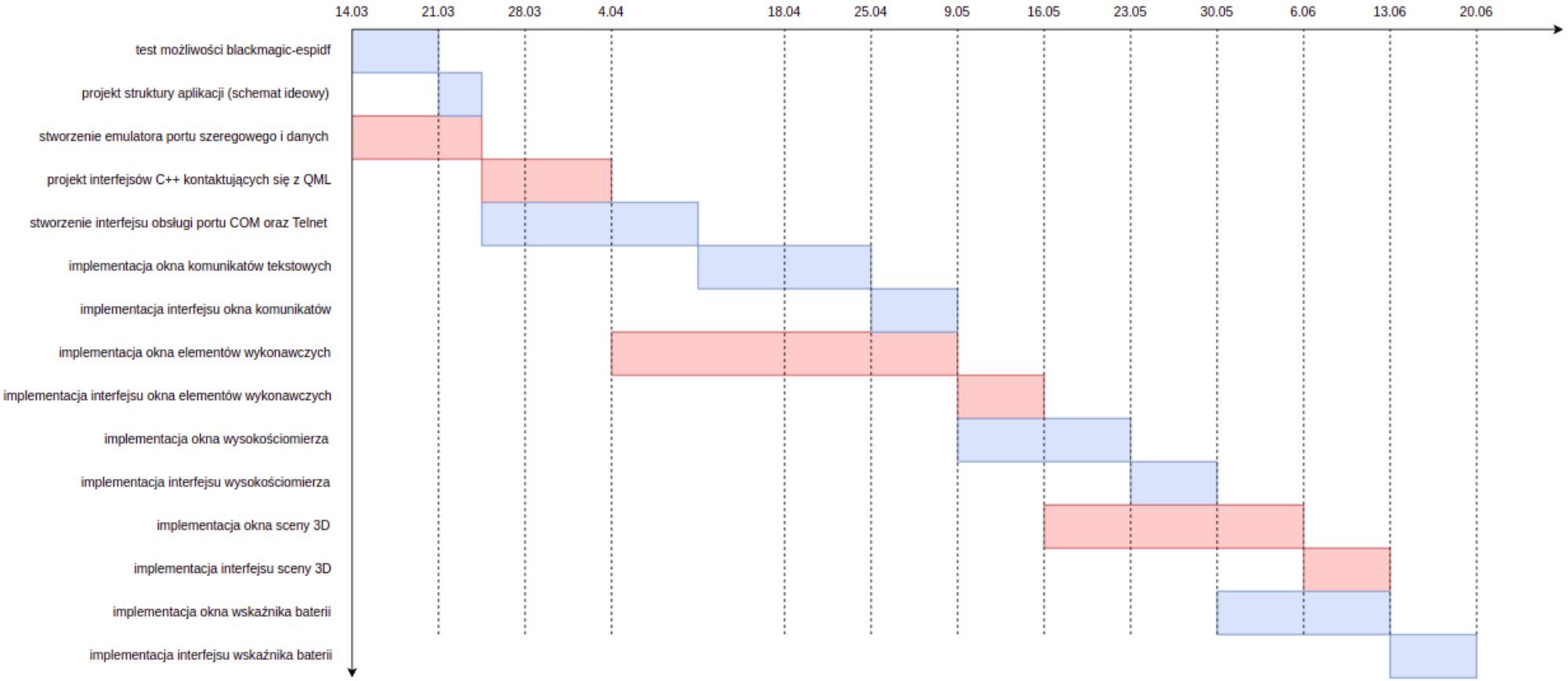
- zapoznanie i przetestowanie projektu „blackmagic-espidf”[1]
- stworzenie interfejsu obsługi portu COM oraz Telnet w C++
- projekt struktury aplikacji (schemat ideowy)
- implementacja okna komunikatów tekstowych w QML
- implementacja okna wskaźnika baterii w QML
- implementacja okna wysokościomierza w QML
- implementacja interfejsu okna komunikatów w C++
- implementacja interfejsu wysokościomierza i wskaźnikiem poziomu baterii w C++

Lista podcelów (Jakub Delicat):

- stworzenie emulatora portu szeregowego i danych
- implementacja okna sceny 3D w QML
- implementacja okna elementów wykonawczych w QML
- projekt interfejsów C++ kontaktujących się z QML
- implementacja interfejsu sceny 3D w C++
- implementacja interfejsu okna elementów wykonawczych w C++

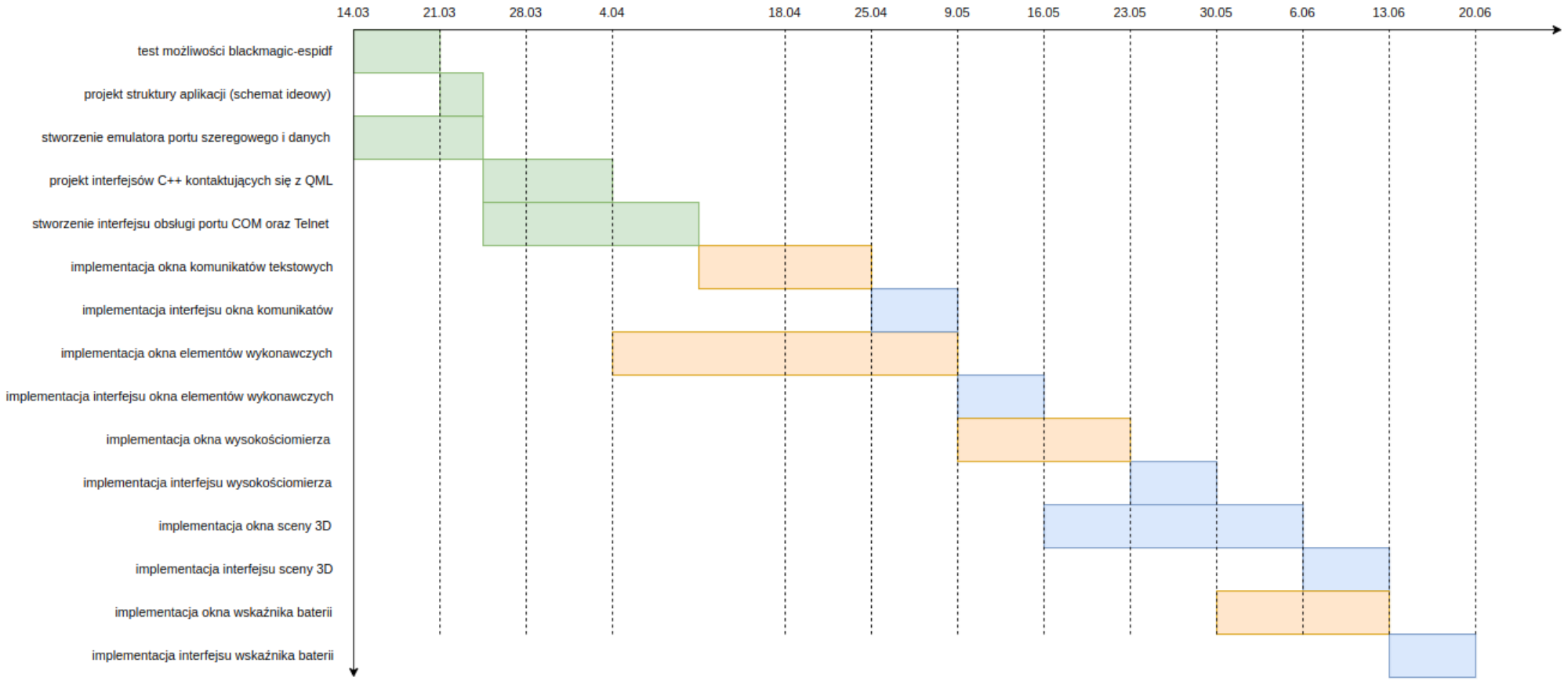
4 Terminarz realizacji poszczególnych podcelów

- 21 marca 2023 – zapoznanie i test możliwości projektu „blackmagic-espidf”[1]
- 28 marca 2023 – projekt struktury aplikacji (schemat ideowy),
  stworzenie emulatora portu szeregowego i danych
- 4 kwietnia 2023 – projekt interfejsów C++ kontaktujących się z QML
- 18 kwietnia 2023 – stworzenie interfejsu obsługi portu COM oraz Telnet
- 25 kwietnia 2023 – implementacja okna komunikatów tekstowych
- 9 maja 2023 – implementacja interfejsu okna komunikatów,
  implementacja okna elementów wykonawczych
- 16 maja 2023 – implementacja interfejsu okna elementów wykonawczych
- 23 maja 2023 – implementacja okna wysokościomierza
- 30 maja 2023 – implementacja interfejsu wysokościomierza
- 6 czerwca 2023 – implementacja okna sceny 3D
- 13 czerwca 2023 – implementacja interfejsu sceny 3D,
  implementacja okna wskaźnika baterii
- 20 czerwca 2023 – implementacja interfejsu wskaźnika poziomu baterii


14.03
21.03
28.03
4.04
18.04
25.04
9.05
16.05
23.05
30.05
6.06
13.06
20.06
test możliwości blackmagic-espidf
projekt struktury aplikacji (schemat ideowy)
stworzenie emulatora portu szeregowego i danych
projekt interfejsów C++ kontaktujących się z QML
stworzenie interfejsu obsługi portu COM oraz Telnet
implementacja okna komunikatów tekstowych
implementacja interfejsu okna komunikatów
implementacja okna elementów wykonawczych
implementacja interfejsu okna elementów wykonawczych
implementacja okna wysokościomierza
implementacja interfejsu wysokościomierza
implementacja okna sceny 3D
implementacja interfejsu sceny 3D
implementacja okna wskaźnika baterii
implementacja interfejsu wskaźnika baterii

5 Wstępne rezultaty

5.1 Przebieg prac



Rysunek 2: zielony – wykonane w terminie, żółty – wykonane przed czasem, niebieskie – do wykonania

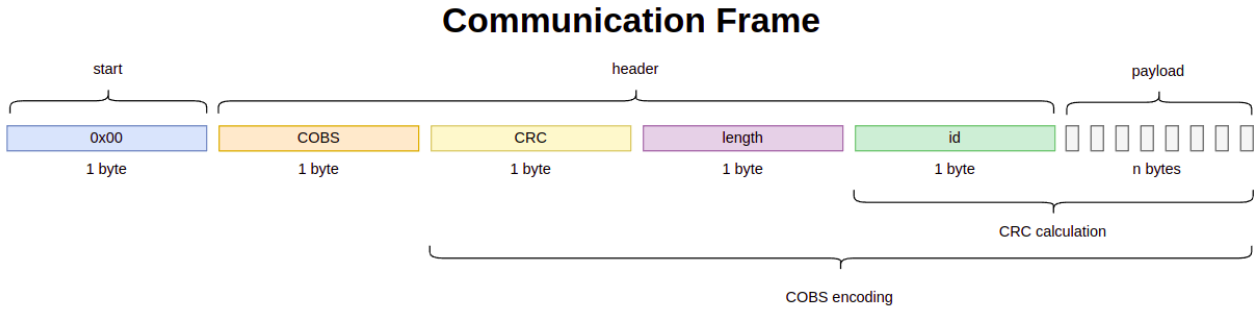
5.2 Komunikacja

Protokół komunikacji

W celu komunikacji umożliwiającej dwustronną komunikację pomiędzy stacją pilota (PC) a dronem został opracowany binarny protokół komunikacyjny. Ramka danych3 składa się z trzech części: startu, nagłówka oraz ładunku.

Bajt startu o wartości 0 definitywnie oznacza początek nadawania ramki danych. Doświadczenie z poprzednich projektów pokazuje, że jest to mechanizm konieczny do sprawnego dekodowania informacji. Dzięki zastosowaniu algorytmu COBS[3], w modelowym przypadku, zapewnione będzie występowanie wartości 0 wyłącznie na początku ramki danych.

Nagłówek o stałym rozmiarze mieści w sobie informacje takie jak nadmiarowy bit COBS, suma kontrolna CRC-8[4], długość ładunku danych oraz jednobajtowy identyfikator ramki. Dzieki zastosowaniu sumy kontrolnej będzie możliwa prosta walidacja odebranych danych.



Rysunek 3: Zaimplementowana ramka danych

Odczyt portu szeregowego USB

Aplikacja zdolna jest do dwustronnej komunikacji używając portu szeregowego. Rdzeniem zapewniającym funkcjonalność jest klasa QSerialPort. W przypadku niepowodzenia w otwarciu wskazanego portu lub jego przedwczesnego zamknięcia, następuję aktywacja licznika QTimer w celu cyklicznego sprawdzenia dostępności portu. Proces ten trwa do skutku. Taki mechanizm zapewnia automatyczne przywrócenie komunikacji w przypadku resetu mikrokontrolera jak i wypięcia przewodu z gniazda.

Klasa jest także odpowiedzialna za odkodowanie danych przychodzących, za pomocą klasy Transfer. Jest to klasa implementująca protokół 3, której kod źródłowy współdzielony jest między aplikację oraz program drona.

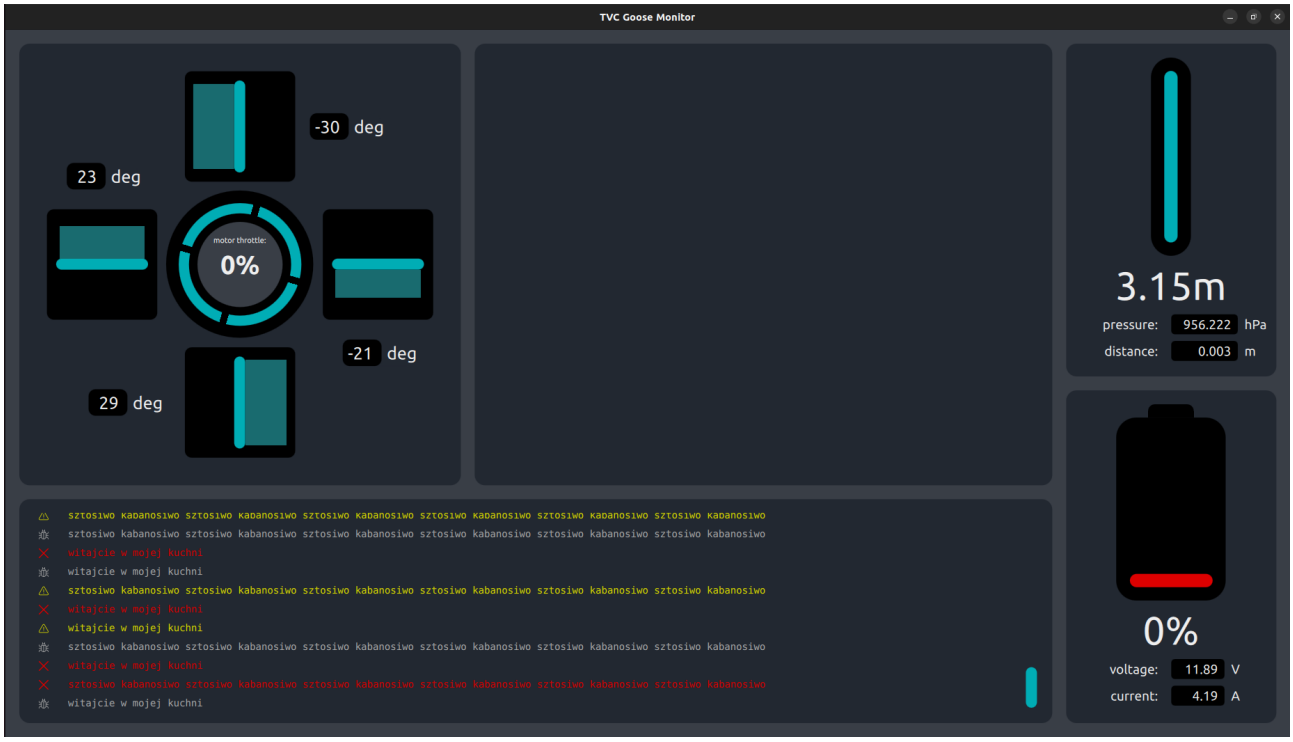
Odczyt portu Telnet

Główną drogą komunikacji docelowo będzie bezprzewodowa wymiana danych. W tym celu zostały przeprowadzone testy możliwości oprogramowania BlackMagic–ESPIDF [1], z którego na potrzeby tego projektu wykorzystana będzie komunikacja po usłudze Telnet. Tak zaprogramowany mikrokontroler ESP8288 automatycznie łączy się na stałe zadaną siecią, i stanowi rolę pośrednika w wymianie Telnet – UART.

Klasa komunikacji po Telnet oparta jest o klasę QTcpSocket, która posiada wbudowany mechanizm oczekiwania na port pod zadanym adresem. W przypadku rozłączenia, także następuję próba ponownego nawiązania połączenia. Tak jak w przypadku USB klasa zawiera własny obiekt klasy Transfer do odkodowywania przychodzących wiadomości.

5.3 Interfejs użytkownika

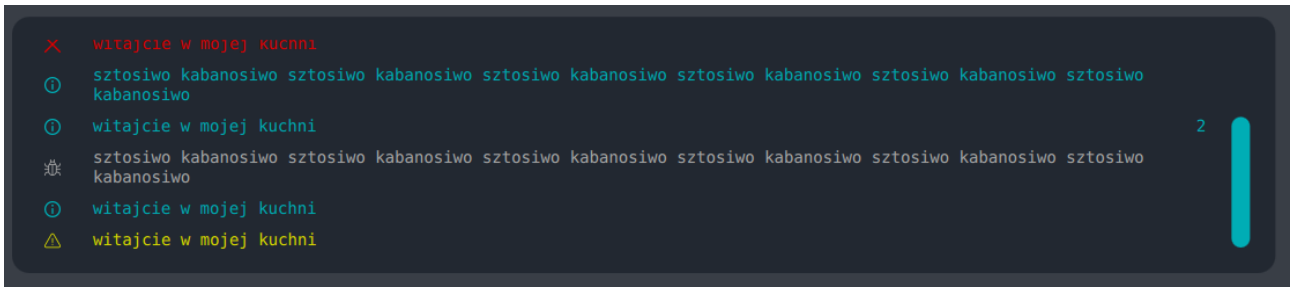
Rozkład komponentów składających się na okno aplikacji został ograniczony tak, aby utrzymać czytelność oraz walory estetyczne nawet przy skrajnych zmianach proporcji i rozmiarów okna. Wizualizacja elementów wykonawczych wpisana jest w okno zachowujące kształt kwadratu. Na scenę 3D zostało nałożone ograniczenie niepozwalające jej wysokości przekroczyć jej szerokości, czyli przy skrajnym skurczeniu szerokości okna staje się kwadratem. Przy rozciąganiu rozszerza się dowolnie, nie zmieniając swojej wysokości. Wysokościomierz oraz wskaźnik poziomu baterii wpisane są w prostokąty o złotej proporcji wysokości do szerokości, niezależnei od rozmiarów i proporcji okna.



Rysunek 4: Okno aplikacji – od lewej: elementy wykonawcze, scena 3D, wysokościomierz, wiadomości tekstowe, wskaźnik poziomu baterii

5.3.1 Okno wiadomości tekstowych

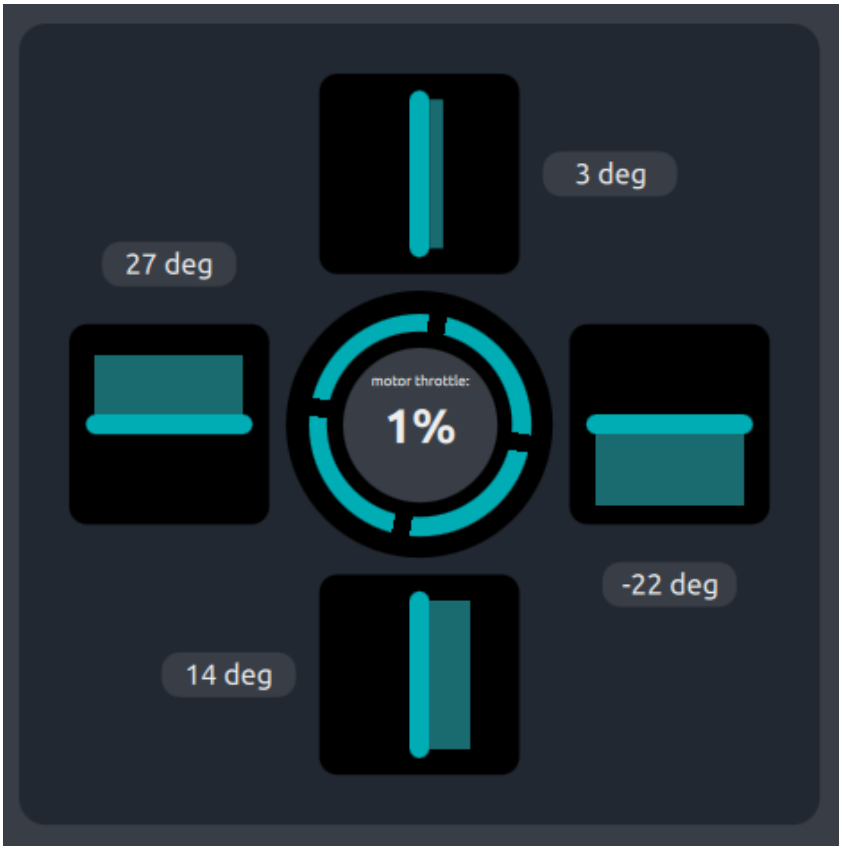
Wiadomości tekstowe pojawiają się na dole listy i wraz z ich przybywają wędrują w górę. Gdy ilość wiadomości przekroczy 50 najstarsza wiadomość jest usuwana. W zależności o typu wiadomości nadawany jest jej inny kolor oraz ikona wyświetlana po lewej stronie. W przypadku wystąpienia tego samego komunikatu więcej niż raz wyświetlany jest licznik jej wystąpień. Możliwy jest swobodny wgląd do wcześniejszych wiadomości dzięki belce po prawej stronie.



Rysunek 5: Okno komunikatów tekstowych – widoczny licznik oraz różne typy wiadomości

5.3.2 Wizualizacja elementów wykonawczych

Wychylenie czterech lotek reprezentowane jest przez zmianę szerokości odpowiadających prostokątów. Każda lotka jest wyposażona w liczbową reprezentacje wychylenia. Wartość przepustnicy liniowo przekształca się na wartość prędkości obrotu pierścienia w centrum (rotor).

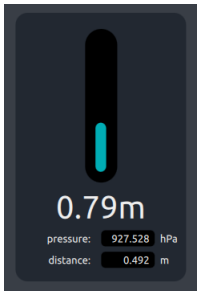


Rysunek 6: Wizualizacja stanu elementów wykonawczych

5.3.3 Wysokościomierz

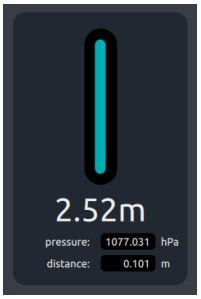
Zgodnie ze specyfikacją, wysokościomierz reprezentowany jest przez słupek o zmiennej wysokości. Maksymalne wypełnienie osiągane jest już przy 2.5 m (typowa wysokość stropu budynku), lecz nie ogranicza to wyświetlanej wartości w przypadku lotu na zewnątrz. Poniżej wyświetlane są także dane o pomiarze ciśnienia z barometru oraz dystans w czujnika laserowego. Z fuzji tych danych uzyskiwana będzie estymacja wysokości.

(a) wartość minimalna

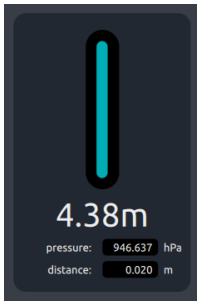


(b) opis

(c) opis



(d) granica wizualizacji

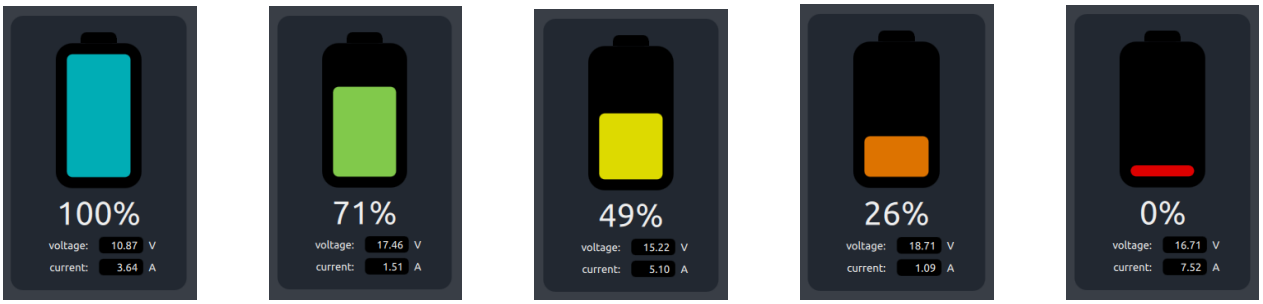


(e) przekroczenie wartości

Rysunek 7: Wygląd słupka dla danych z praktycznego zakresu

5.3.4 Wskaźnik poziomu baterii

Wskaźnik poziomu baterii oprócz zmiany wysokości słupka, zmienia również swój kolor. Kolor wynikowy obliczany jest poprzez interpolację liniową pomiędzy kolorem głównym a żółtym (dla poziomu $> 50\,\%$) oraz pomiędzy żółtym a czerwonym (dla poziomu $< 50\,\%$). Poniżej wyświetlane są mierzone parametry zasilania: napięcie baterii oraz chwilowy pobór.



(a) naładowana bateria (główny)

(b) lekko rozładowana bateria

(c) połowa pojemności (żółty)

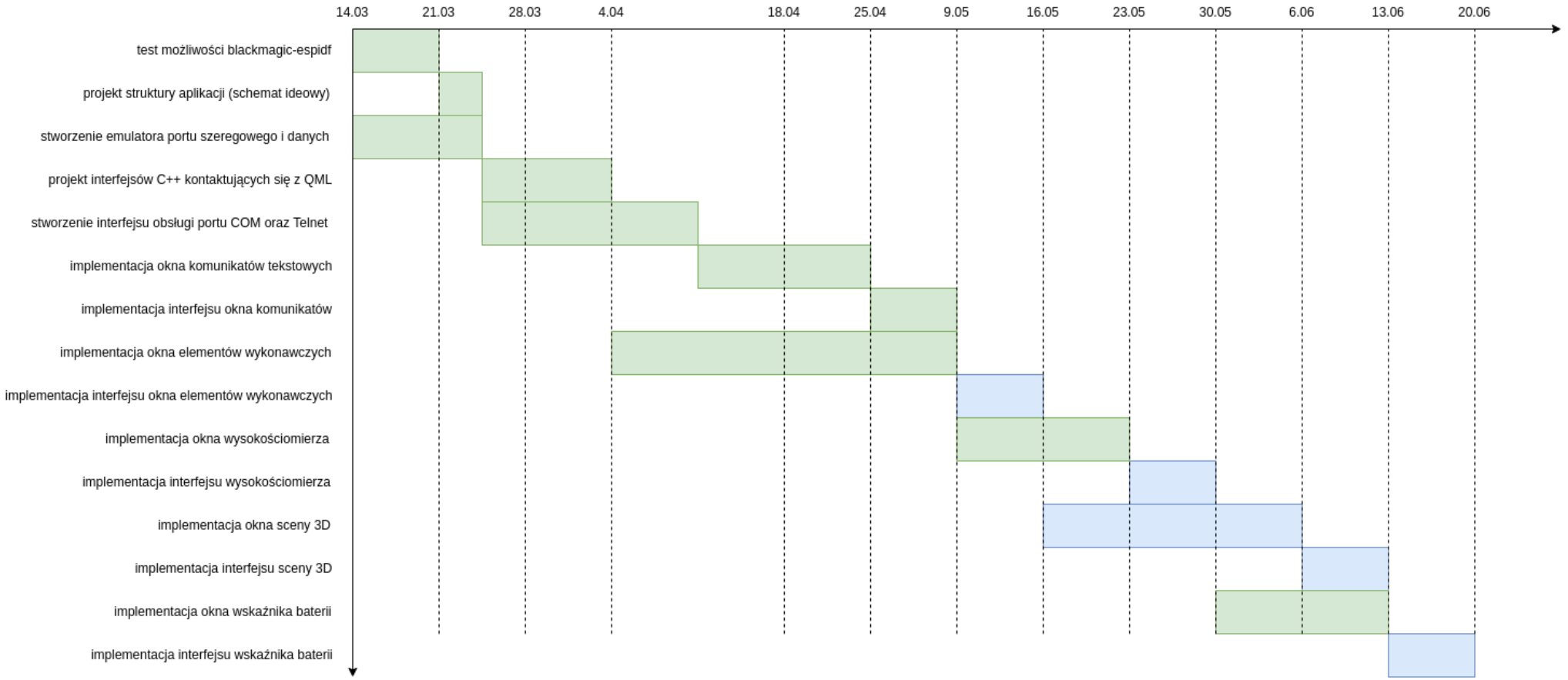
(d) na wyczerpaniu (pomarańczowy)

(e) rozładowana bateria (czerwony)

Rysunek 8: Reprezentacja baterii dla różnych stopni rozładowania

6 Rezultaty zaawansowane

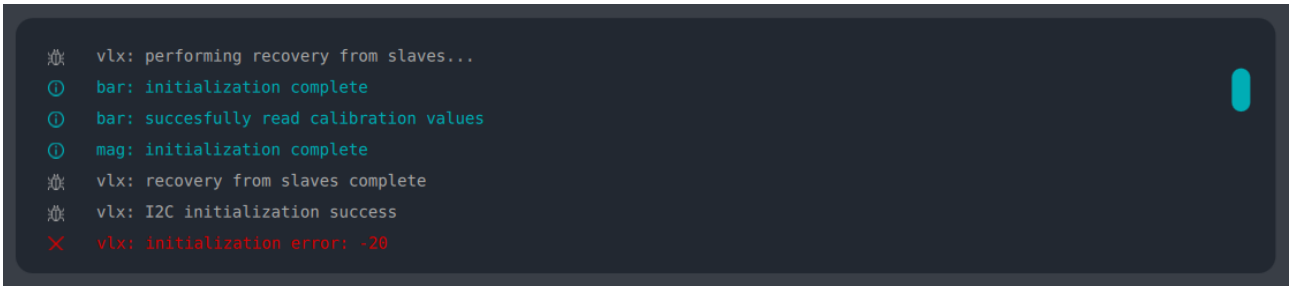
6.1 Przebieg prac



Rysunek 9: zielony – wykonane w terminie lub przed, niebieskie – do wykonania

6.2 Implementacja połączenia okna komunikatów

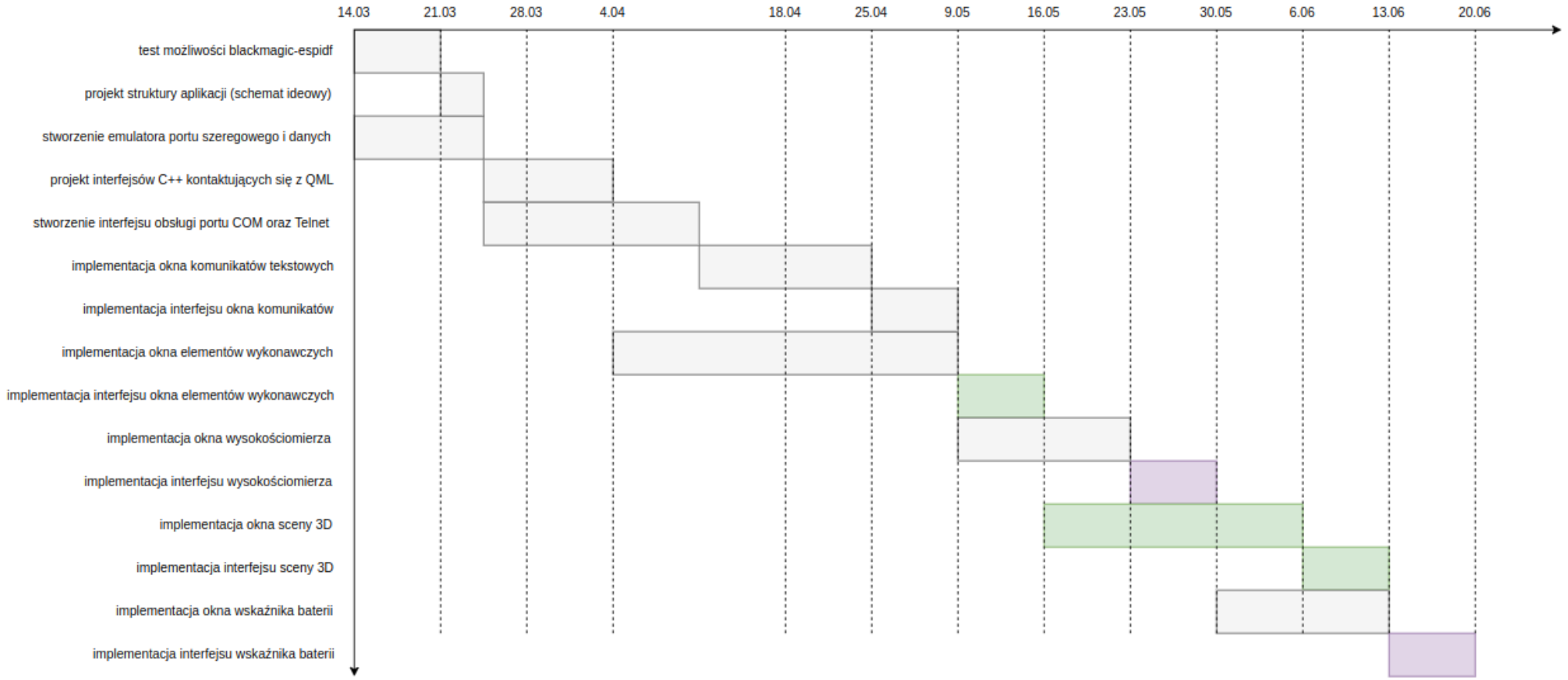
W celu wyświetlania komunikatów tekstowych wysyłanych przez sterownik lotu do komputera została zaimplementowana w języku C++ klasa Logger, która pośredniczy w transporcie danych. Slot „receive”, który odpowiada za dodawanie nowych wiadomości tekstowych do okna, został połączony z sygnałami emitowanymi przez klasy USB i Telnet w momencie otrzymania nowej ramki danych. Wiadomości są interpretowane jako ciągi znaków i w zależności od identyfikatora ramki nadawany jest im jeden z czterech typów: „debug”, „info”, „warning” lub „error”.



Rysunek 10: Wygląd działania okna podczas inicjalizacji sterownika lotu.

7 Rezultaty zaawansowane

7.1 Przebieg prac

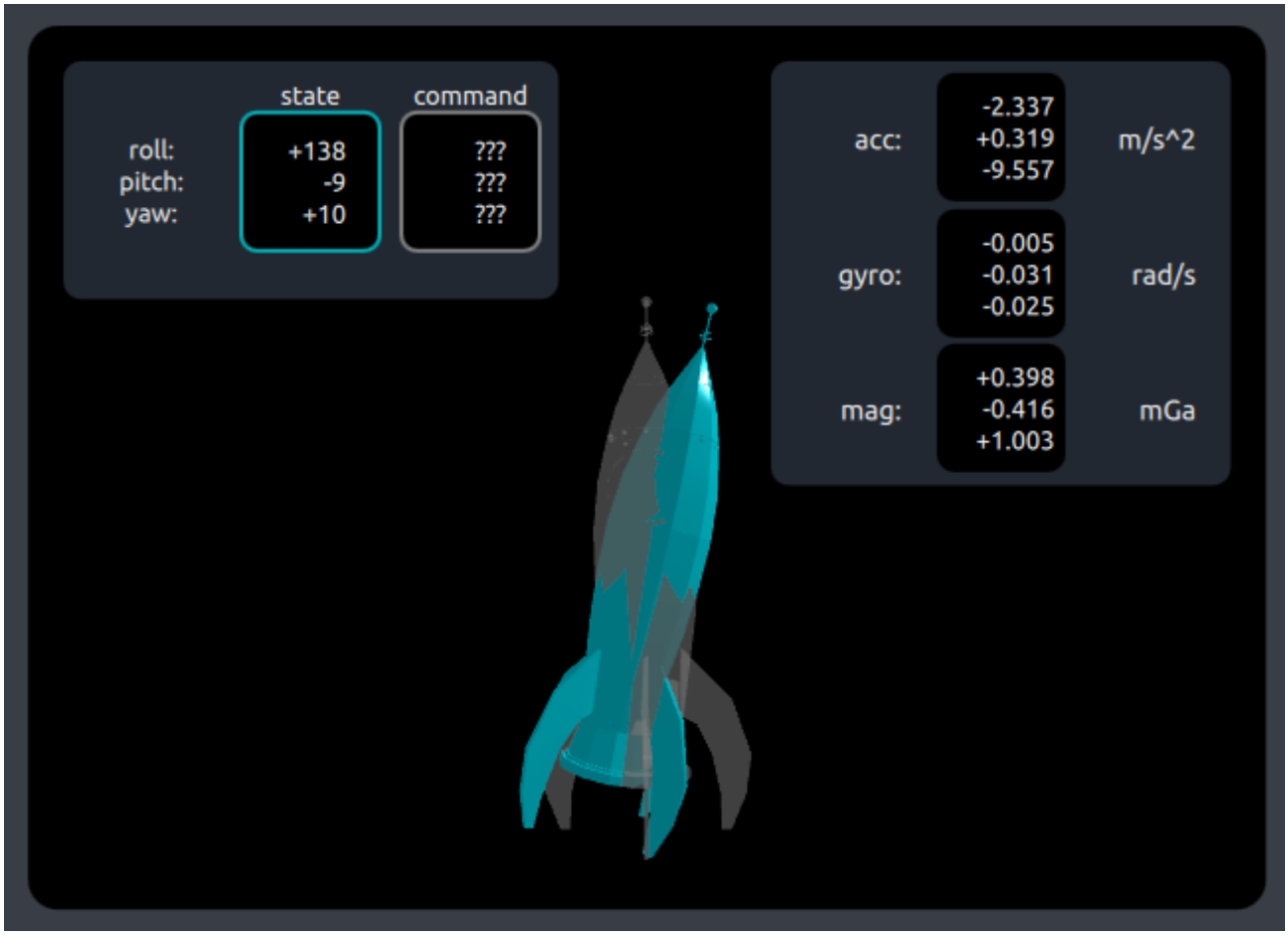


Rysunek 11: zielony – wykonane przez Jakuba Delicata, fioletowe – wykonane przez Eryka Możdżenia, szare – wykonane wcześniej

7.2 Implementacja okna sceny 3D

W celu wizualizacji orientacji w przestrzeni drona została zastał użyty obiekt „Scene3D” w QML tak jak na rys. 12. Na scenie są dwa obiekty: niebieski reprezentuje aktualny stan drona, natomiast szary to stan zadany. Stan zadany jest półprzeźroczysty, aby nie przysłaniać stanu obecnego. Liczbowa reprezentacja obrotu w kątach RPY wyświetlana jest w okienku powyżej.

Dodatkowo wyświetlane są odczyty z czujników wchodzących w skład fuzji do otrzymania orientacji. Wektory wskazań odpowiednio z akcelerometru, żyroskopu oraz magnetometru są sformatowane tak, aby miejsce dziesiętne nie zmieniało swojego położenia. Tak jak zostało opisane w specyfikacji, wyświetlają się trzy miejsca po przecinku, a znak jest cały czas widoczny.



Rysunek 12: Wygląd działania okna podczas obrotu drona.

7.3 Wyświetlanie rzeczywistych danych z obiektu

Aby wyświetlać rzeczywiste dane z obiektu, zostały usunięte tymczasowe oprogramowanie, które do tej pory generowało wartości odczytów (przebiegi sinusoidalne o różnych przesunięciach fazowych i częstotliwościach). W QML zostały napisane odpowiednie funkcje wpływające na interfejs, które następnie zostały za pomoca systemu slotów i sygnałów połączone z kodem w C++. Powstały odpowiednio udokumentowane w Doxygenie klasy, które odpowiadają za przechwytywanie przychodzących wiadomości i generowanie odpowiednich sygnałów do aktualizacji interfejsu w QML.

Oprócz tego zostały dodane liczniki, które w przypadku braku otrzymywania konkretnych danych przez dłużej niż 100 ms wyświetlają znaki „???”. Informuje to użytkownika o braku danych lub zbyt niskiej częstotliwości ich wysyłania.

Bibliografia

[1] *Blackmagic Wireless SWD Debug probe hosted on esp-idf SDK (for ESP8266) with UART on Telnet port and HTTP using xterm.js*. URL: https://github.com/walmis/blackmagic-espidf.

[2] Eryk Możdżeń. *UAV TVC Goose*. URL: https://github.com/Eryk-Mozdzen/uav-tvc-goose.git.

[3] Wikipedia. *Consistent Overhead Byte Stuffing*. URL: https://en.wikipedia.org/wiki/Consistent_Overhead_Byte_Stuffing.

[4] Wikipedia. *Cykliczny kod nadmiarowy*. URL: https://pl.wikipedia.org/wiki/Cykliczny_kod_nadmiarowy.