POZNAN UNIVERSITY OF TECHNOLOGY

SKN/SP/602030/2024
Raport końcowy

ZDALNY SYSTEM KONTROLI MISJI I AKWIZYCJI DANYCH DO ZASTOSOWANIA W RAKIETACH BADAWCZYCH

Koło naukowe: PUT Rocketlab
Kierownik projektu: dr inż. Bartosz Ziegler
Autorzy raportu: Marta Brzynska, Miłosz Krzyżanowski, Łukasz Kania, Paweł Babalski, Jakub Wicher, Krzysztof Martin, Maciej Łachut, Mikołaj Oterski, Wiktor Wojtkowiak, Wiktor Kowalewski, Wojciech Kubiak, Tomasz Krakowski

Poznań 2025

Spis treści

1 Wstęp

1.1 Cel projektu

Celem projektu jest rozwinięcie innowacyjnego systemu SCADA (ang. Supervisory Control And Data Acquisition), wykorzystującego nowoczesne technologie informatyczne Vue.js oraz typescript. Bazuje on na idei modularnego systemu rozproszonego, komunikującego się z wykorzystaniem interfaców sieciowych (WiFi i Ethernet) oraz magistrali CAN.

System ma za zadanie monitorować oraz kontrolować elementy systemu składających się na rakietę badawczą. Pozwala on na bezpieczne i efektywne przeprowadzenie prac rozwojowych związanych z rozwojem technologii silników rakietowych, a także zaawansowanych systemów awionicznych. Postawiono i spełniono następujące Wymagania projektowe:

- zdalne przeprowadzanie eksperymentu - odczyt danych z czujników oraz sterowanie aktuatorami (zawory, silnik)
- utrzymanie ciągłego radiowego połączenia telemetrycznego z rakietą sondażową - wymóg konieczny do zdalnej kontroli przeprowadzanej misji umożliwiający na zebranie danych kluczowych do weryfikacji uzyskanych parametrów technicznych układu
- sekwencyjny system wykonywania eksperymentu, zwiększający bezpieczeństwo przeprowadzanych eksperymentów
- wysoka modularność - skalowalność sprzętowa oraz programistyczna, zapewniająca łatwe rozszerzenie systemu w przypadku konieczności zmiany konfiguracji przeprowadzanych badań

Większość wymagań podyktowana jest koniecznością zapewnienia spełniania wymagań bezpieczeństwa pracy z utleniaczem - N2O (podtlenek azotu), poprzez zapewnienie bezpiecznej odległości od miejsca przeprowadzenia eksperymentu.

Możliwości systemu nie są ograniczone do pracy z jednym rodzajem rakiety sondażowej, czy też silnika rakietowego. Panel operatorski (Portabale Comand Center - PCC) daje użytkownikom zdolność do interaktywnego monitorowania i reagowania na różne aspekty lotu rakiety w czasie rzeczywistym oraz pobierania danych z wykorzystaniem rozproszonego systemu stacji odbiorczych. Kontrola eksperymentu zapewniana jest przez jednostki pomiarowe oraz wykonawcze, połączone z wykorzystaniem interfaców sieciowych. Kolejnym elementem zabezpieczeń jest system kamer sprzężony z układami RISC-V i FPGA, do automatycznego reagowania na anomalie w trakcie eksperymentu, zaimplementowany na w stanowiskach naziemnych, a także na pokładzie rakiety sondażowej.

Komunikacja między węzłami systemu uzyskana jest poprzez zastosowanie modułów radiowych LoRa oraz WiFi. W celu zmniejszenia zakłóceń, wykorzystano rozdział pasma radiowego. Na potrzeby łącza telemetrycznego z rakietą sondażową użyto częstotliwości z zakresu 410-440 MHz, natomiast komunikacja z oddalonymi węzłami prowadzi się na wyższej częstotliwości 2,4 GHz.

Układy awioniczne wyposażono w dodatkowe interfejsy (I2C, UART, CAN) poszerzające uniwersalność i skalowalność zaproponowanego systemu. Awionika zawiera czujniki inercyjne oraz moduły GNSS do określenia pozycji rakiety badawczej w trakcie misji eksperymentalnej.

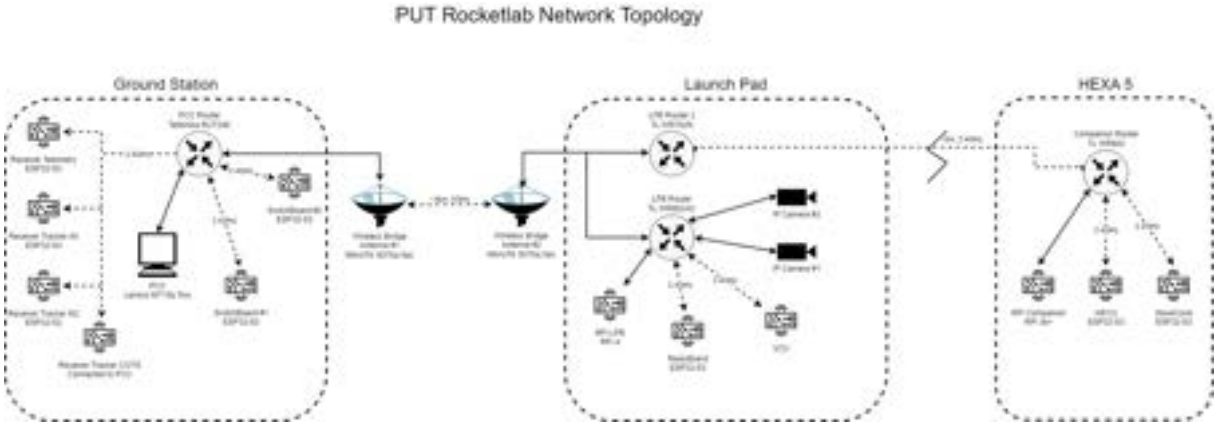
2 Komunikacja między komponentami systemu

2.1 Warstwa fizyczna

W celach komunikacyjnych między głównym panelem operatorskim, a rakietą badawczą został stworzony system komunikacji oparty o technologie sieciowe **LAN** (Ethernet i Wi-Fi) oraz LoRa [3]. Komunikacja poprzez WiFi odbywa się w trakcie przebywania rakiety na ziemi, przy wszelkich naziemnych testach silników, a także wewnątrz rakiety w trakcie samego lotu. Taka forma komunikacji została wybrana ze względu na kilka czynników:

- Prędkość - jest to jedna z najszybszych form komunikacji opierającej się o bezprzewodowe połączenia.
- Dostępność - urządzenia sieciowe są względnie tanie oraz szeroko dostępne.
- Skalowalność - dodawanie dodatkowych urządzeń klienckich oraz węzłów komunikacyjnych odbywa się bez potrzeby ingerencji w resztę systemu.

W przygotowanym projekcie sieci LAN (Rysunek 1) szczególnie duży nacisk został położony na odseparowanie poszczególnych jej części tak aby mogły one pracować niezależnie od siebie bez konieczności komunikacji z pozostałymi komponentami sieci. Sieć dzieli się na trzy główne podsieci (stacja naziemna, stacja znajdująca się przy rakiecie na ziemi oraz sieć w środku rakiety) z osobnym adresowaniem i serwerami DHCP. Urządzenia klienckie w zależności od tego w jakiej podsieci się znajdują posiadają inne adresowanie z puli adresów niezarezerwowanych dla poszczególnych urządzeń.



Rysunek 1: Architektura sieci LAN

Sercem każdej podsieci jest router do którego łączy się główny komputer z każdej sieci (przewodowo) oraz inne urządzenia przeważnie bezprzewodowo lub przez Switche.

Podsieci łączą się między sobą na dwa sposoby:

- Poprzez **most sieciowy** składający się z dwóch anten kierunkowych działających na częstotliwościach 5GHz aż do 12km co jest zdecydowanie wystarczające w przewidywanych warunkach testów. Zasilane są poprzez standard PoE (Power over Ethernet). Zapewniają one przepustowość do 300 Mbit/s oraz stabilne połączenie z niskimi opóźnieniami z punktu widzenia sieci są traktowane jak zwykłe połączenie kablowe. Anteny montowane są na specjalnych masztach antenowych o wysokości 10 m dzięki którym ewentualne zakłócenia z urządzeń naziemnych, a także wpływ przeszkód terenowych na sygnał jest znikomy. Działanie sieci zostało pomyślnie przetestowane na odległości około 700 m wraz ze sprawdzeniem opóźnień oraz stabilności połączenia.
- **Dodatkowy router** w strefie przy rakiecie służący do komunikacji z routerem w awionice ze względu na brak możliwości stosowania połączenia kierunkowego oraz przewodowego. Router ten działa jako extender routera podstawowego i działa na paśmie 2.4 GHz ze względu na jego większy zasięg.

Wszystkie routery zostały skonfigurowane przy użyciu oprogramowania OpenWRT [9] umożliwiającego łatwe archiwizowanie utworzonych konfiguracji.

Komunikacja przy użyciu protokołu **LoRa** odbywa się głównie w trakcie lotu rakiety pomiędzy stacjami odbiorczymi na ziemi, a awioniką. Jest to często używana technologia w tego typu zastosowaniach ze względu na niskie zużycie energii oraz daleki zasięg działania komunikacji. Znakomicie nadaje się do pracy na odległościach rzędu kilkunastu kilometrów nawet bez bezpośredniej widoczności urządzenia nadającego przez stację odbiorczą.

System wykorzystuje transceiver Semtech SX1262, który obsługuje rozszerzony zasięg i stabilne łącze komunikacyjne. Jest on połączony z modułem frontendowym, który zawiera wzmacniacz mocy (PA) i wzmacniacz niskoszumowy (LNA), znacznie zwiększając zarówno moc transmisji, jak i czułość odbiornika.

Komunikacja odbywa się w zakresie częstotliwości 410–470 MHz, z czego częstotliwość Telemetri to 433 MHz. Modem LoRa jest skonfigurowany z szerokością pasma 125 kHz, spreading factor (SF) regulowanym w zakresie od 7 do 12 (w zależności od celu, mocy nadawczej i wymagań dotyczących zasięgu urządzenia), szybkością kodowania 4/5 (CR = 1) i długością preambuły wynoszącą 8 znaków. Ustawienia te zapewniają typową szybkość transmisji danych wynoszącą około 4,8 kbps, chociaż rzeczywista szybkość zależy od wybranego spreading factor.

Parametry te stanowią kompromis między zasięgiem komunikacji, przepustowością i zużyciem energii, zapewniając niezawodną przesył danych telemetrycznych podczas całego lotu rakiety. LoRa jest dobrym rozwiązaniem do tego typu zastosowań w zastosowaniach lotniczych/rakietowych.

2.2 Warstawa aplikacji

2.2.1 MQTT

Do obsługi komunikacji między urządzeniami z punktu widzenia oprogramowania został wykorzystany protokół MQTT (Message Queue Telemetry Transport) [7] oparty o paradygmat publish-subscribe oraz działanie wokół brokerów (serwerów). Protokół ten umożliwia łatwy sposób wymiany danych i istnieją do niego biblioteki działające z zdecydowaną większością współczesnych języków programowania. Jest on jedynym protokołem przekazywania danych z urządzeń innych niż kamery wykorzystywanym w projekcie po sieci LAN ze względu na jego wszechstronność.

Przy odpowiednim ustawieniu parametrów wysyłania i odbierania danych (QoS) można zapewnić, że dane dotyczące procesów krytycznych zostaną przesłane w odpowiedni sposób. Dodatkowo zaimplementowany został dedykowany system acknowledgmentów, który pozwala na sprawdzenie czy komendy wysyłane z panela operatorskiego do rakiety i urządzeń naziemnych zostały poprawnie przetworzone co pozwala operatorowi mieć pewność o tym, że system poprawnie działa.

2.2.2 Protobuf

Dane przesyłane przez MQTT są wysyłane w formacie Protobuf [10] również ze względu na jego szerokie wsparcie przez języki programowania oraz przez jego stałą strukturę. Każda wiadomość na danym topicu MQTT ma przypisany do siebie konkretny format Protobuf. Każdy programista w systemie wie w jakim formacie oraz jak ma odbierać i wysyłać dane co jest znacznym ulepszeniem w stosunku do wielu systemów tego typu korzystających z formatu JSON. Umożliwia to wykrywanie błędów komunikacji ze względu na format danych i ich typ na bardzo wczesnym etapie.

Dodatkowo dane wysyłane są w formie binarnej co zmniejsza zużycie dostępnej przepustowości sieci lokalnej.

2.2.3 Node-RED

Odpowiednie przekierowywanie danych między aplikacjami, dekodowanie wiadomości, obliczanie dodatkowych parametrów do wyświetlania na profilach misji, zapis danych do bazy danych MongoDB, publikowanie niektórych danych jest obsługiwane przez platformę Low-code w postaci systemu Node-RED [8] umożliwia ona tworzenie dedytkowanych flow w formie graficznej. Na każdym urządzeniu z brokerem MQTT istnieje osobna instancja Node-RED'a dzięki czemu zachowuje się separację między elementami systemu.

2.2.4 MongoDB

Jako postawowy sposób przechowywania danych wykorzystywana jest baza danych NoSQL MongoDB [6]. Podobnie jak wcześniej opisane komponenty jest ona zahostowana na każdym głównym urządzeniu sieci.

Zapisuje ona dane z poszczególnych stref sieci poprzez odpowiednio skonfigurowany flow w Node-RED. Dodatkowo jako podwójne zabezpieczenie przed utratą danych PCC zapisuje dane ze wszystkich urządzeń. Rozwiązanie tego typu jest zdecydowanie lepszą alternatywą dla dużych zbiorów różnych danych z wielu urządzeń niż zapis do plików CSV (Comma-separated values) przez ich mniejszy rozmiar na dysku oraz większą elastyczność co do różnych formatów danych.

Dzięki narzędziu MognoDB Compass [5] użytkownicy mają dostęp do wszystkich danych z dowolnego miejsca w sieci.

2.2.5 NTP

Zbieranie danych z wielu urządzeń rodzi problem synchronizacji czasu między nimi. Wiele węzłów pomiarowych nie posiada wbudowanego RTC (zegar czasu rzeczywistego) co rodzi potrzebę stworzenia innego sposobu pobierania go. Zaimplementowany został protokół NTP (Network Time Protocol) [1], którego główny serwer znajduje się na PCC. Wszystkie urządzenia łączą się do jednego serwera i synchronizują swój czas względem niego. Pozwala to na łatwiejszą obróbkę danych po zakończonym teście oraz lepsze odtwarzanie błędów jeżeli jakieś wystąpiły.

Czas serwera NTP jest pobierany poprzez Linuxowy daemon GPSD, który bierze go z odbiornika GPS znajdującego się w PCC. Umożliwia on również wysyłanie aktualnej pozycji stacji kontroli misji w celu wyświetlania jej na mapie.

2.2.6 UART i MAVLink

Komunikacja między poszczególnymi komponentami awioniki odbywa się za pośrednictwem UART przy użyciu protokołu MAVLink [4]. MAVLink (Micro Air Vehicle Link) to protokół komunikacyjny przeznaczony do komunikacji między urządzeniami bezzałogowymi takimi jak np. drony, a naziemnymi stacjami kontroli. Obsługuje on definicje wiadomości w specjalnych plikach, opiera się o binarny przesył danych i wbudowaną serializację, dzięki czemu nadaje się do telemetrii w czasie rzeczywistym i wymiany poleceń w systemach wbudowanych.

Wszystkie płytki elektroniczne w koszu awionicznym są połączone w topologii gwiazdy, a Telemetria pełni rolę centralnego węzła komunikacyjnego. Płyta telemetryczna implementuje router MAVLink, który przekazuje komunikaty między węzłami i zapewnia prawidłowe kierowanie komunikatów telemetrycznych i poleceń konfiguracyjnych w całym systemie.

Każdy węzeł komunikuje się bezpośrednio z płytą telemetryczną za pośrednictwem UART, co pozwala na wydajny i zorganizowany przepływ danych bez konieczności tworzenia połączeń peer-to-peer między wszystkimi komponentami.

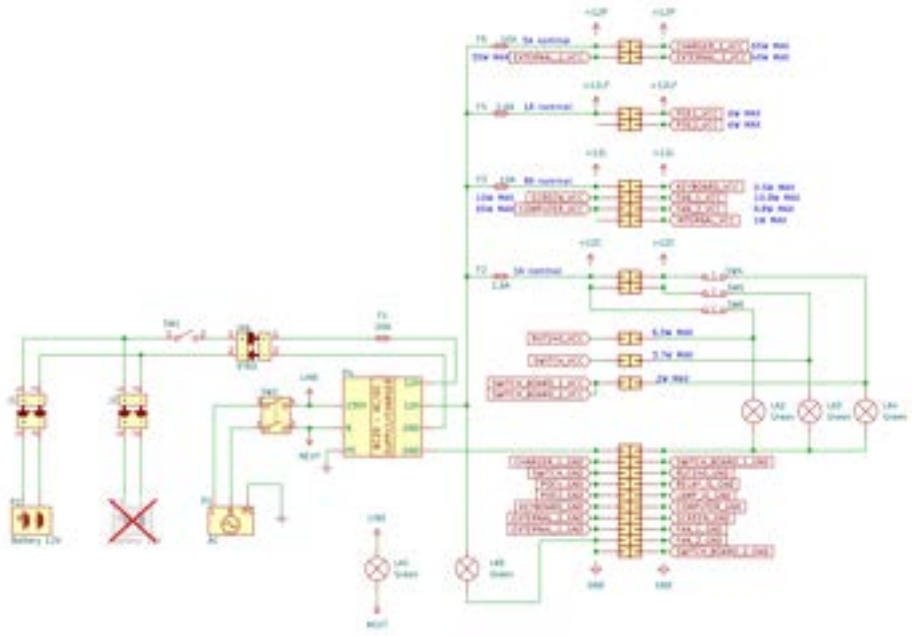
Wszystkie płytki są podłączone do RPi Companion za pośrednictwem UARTa i Mavlinka. Konfiguracja odbywa się za pomocą dedykowanej aplikacji Jest ona napisana w Pythonie i frameworku Flask (backend) oraz React.ts (frontend Rysunek 17). Aplikacja pozwala użytkownikowi wybrać docelową płytkę, polecenie, które ma zostać wysłane, oraz wartości parametrów dla tego polecenia. Dzięki temu, że RPi Companion jest podłączony do sieci Wi-Fi przed startem rakiety, polecenia mogą być wysyłane bezprzewodowo z dowolnego miejsca w sieci. Możliwe konfiguracje są importowane z konfiguracji Mavlinka, co eliminuje konieczność pisania dedykowanych plików konfiguracyjnych.

3 Panel operatorski - PCC

3.1 Opis konstrukcji

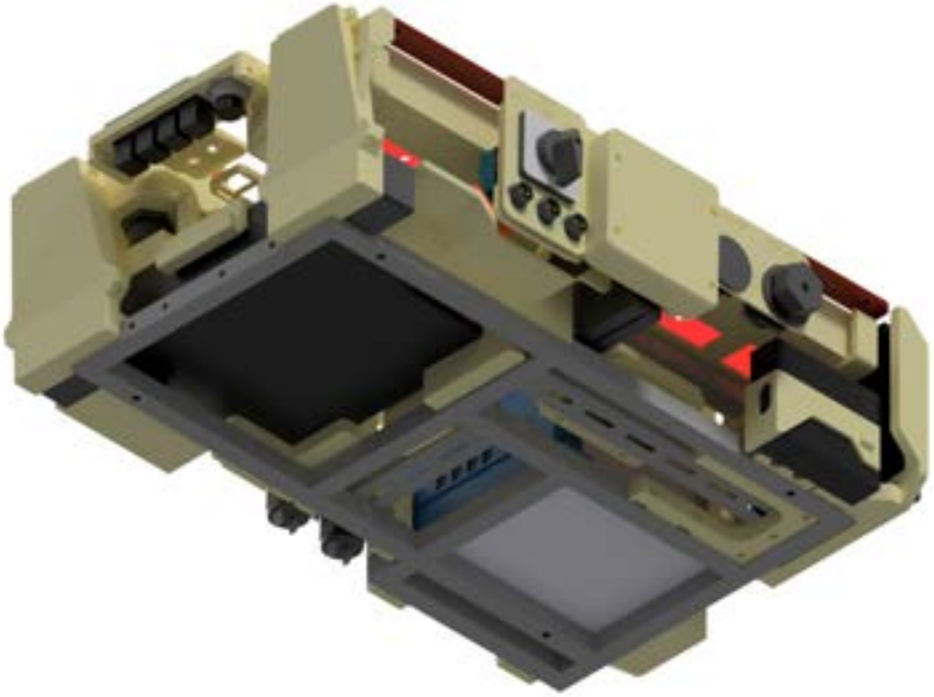
Panel operatorski składa się z komputera PC, monitora o wysokiej jasności, urządzeń sieciowych, a także dedykowanego urządzenia z przełącznikami opracowanego na potrzeby systemu (opis szczegółowy w sekcji 3.3). W celu zapewnienia zwiększonej mobilności złożenie przystosowana do zasilania z samochodowych akumulatorów kwasowo-ołowiowych, a także LiFePO4. W urządzeniu zaimplementowano ładowarkę pozwalającą doładowywać dołączone akumulatory za pomocy zasilania sieciowego (AC 110-220V), a także

na pracę bez dodatkowych akumulatorów. Dodatkowo zaimplementowano trzy izolowane przetwornice DC-DC do zasilania urządzeń pomocniczych i sieciowych. Szczegółowy schemat przedstawiono na rysunku 2.



Rysunek 2: Schemat elektryczny panelu operatorskiego - PCC

Konstrukcja nośna została zaprojektowana z myślą o zwiększonej odporności złożenia na skrajne warunki atmosferyczne. Jako obudowa transportowa została wykorzystana skrzynia Auer Packing CP 6422. Zabudowa wykorzystuje szkielet z kwadratowych profili aluminiowych (AW-6060), połączonych z wykorzystaniem nitów. Elementy mocujące, a także kierujące przepływ powietrza chłodzącego komponenty wykonano z wykorzystaniem przyrostowych technik druku z 3D z ASA (akrylonitryl-styren-akrylan). Wizualizacja głównego złożenia przedstawiona została na Rysunku 3, na rzucie izometrycznym od dołu, można dostrzec elementy ramy, a także wydruki odpowiedzialne za intensyfikację odbierania ciepła od urządzeń. Całość konstrukcji nośnej udostępniono na otwartej licencji CC BY-NC-SA 4.0 na publicznym repozytorium.



Rysunek 3: Wizualizacja ramy PCC

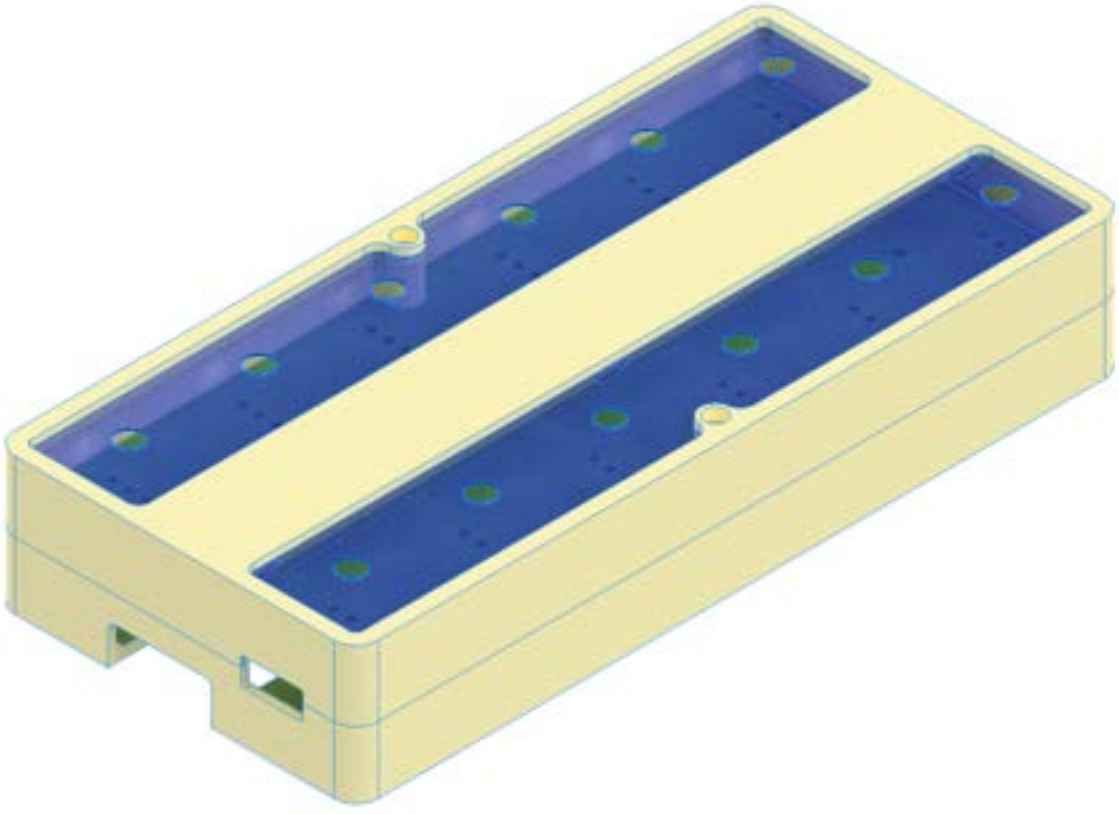
3.2 Projekt i architektura oprogramowania

Głównym przedmiotem projektu było stworzenie wygodnego panelu operatorskiego do zdalnego przeprowadzania misji rakiety sondażowej. Podstawową jego funkcją jest wyświetlanie wszystkich parametrów testu bądź lotu, przeprowadzenie bezpiecznie całej procedury zatankowania utleniacza do zbiornika rakiety, wydania komendy startu. PCC jest zlokalizowane w strefie stacji naziemnej i można je uznać uznać za interfejs typu człowiek-maszyna (HMI)[2]. MCU (główna jednostka obliczeniowa) to komputer Lenovo M710q Tiny. Jest on podłączony do routera Teltonika RUT200 za pośrednictwem Switcha. Router transmituje sygnał Wi-Fi na całym obszarze stacji naziemnej.

3.3 Switchboard

Switchboard to dedykowany kontroler, którego zadaniem jest przekazanie informacji o położeniu przełączników poprzez sieć bezprzewodową (WiFi - 2,4 GHz) z PCC z wykorzystaniem MQTT. Służą one do sterowania elektrozaworami na liniach tankujących, zaworem upustowym i innymi mniej istotnymi funkcjami.

Wybrany mikrokontroler - Espresiff ESP32-S3 jest dedykowanym rozwiązaniem do systemów wykorzystujących sieci bezprzewodowe. Połączony został z przełącznikami poprzez dedykowane filtry dolnoprzepustowe w celu eliminacji wpływu drgania styków. Stan przełączników wizualizowany jest z wykorzystaniem diod adresowalnych. W celu zapewnienia mobilności, zasilanie zrealizowane jest z wykorzystaniem ogniwa Li-ion w formacie 18650. Dla zapewnienia stabilności zasilania, główna linia 5V podłączona jest do PCC i doładowuje ogniwo z wykorzystaniem zintegrowanej ładowarki. W celu zwiększenia odporności na czynniki atmosferyczne płytka drukowana otoczona jest obudową wykonaną z ASA i PCTG (Rysunek 4)



Rysunek 4: Wizualizacja Switchboarda w obudowie

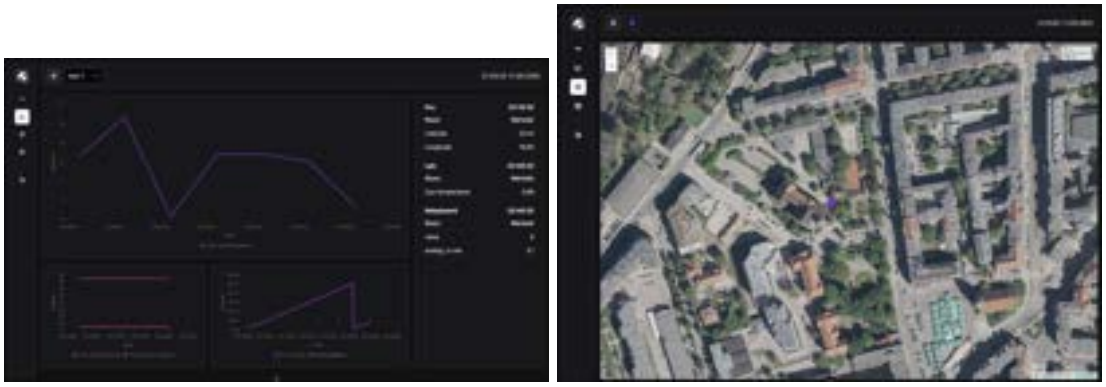
3.4 Aplikacja systemu SCADA

3.4.1 Frontend

HMI (front-end) napisany został w vue.js. Składa się on z poszczególnych podstron realizujących różne cele:

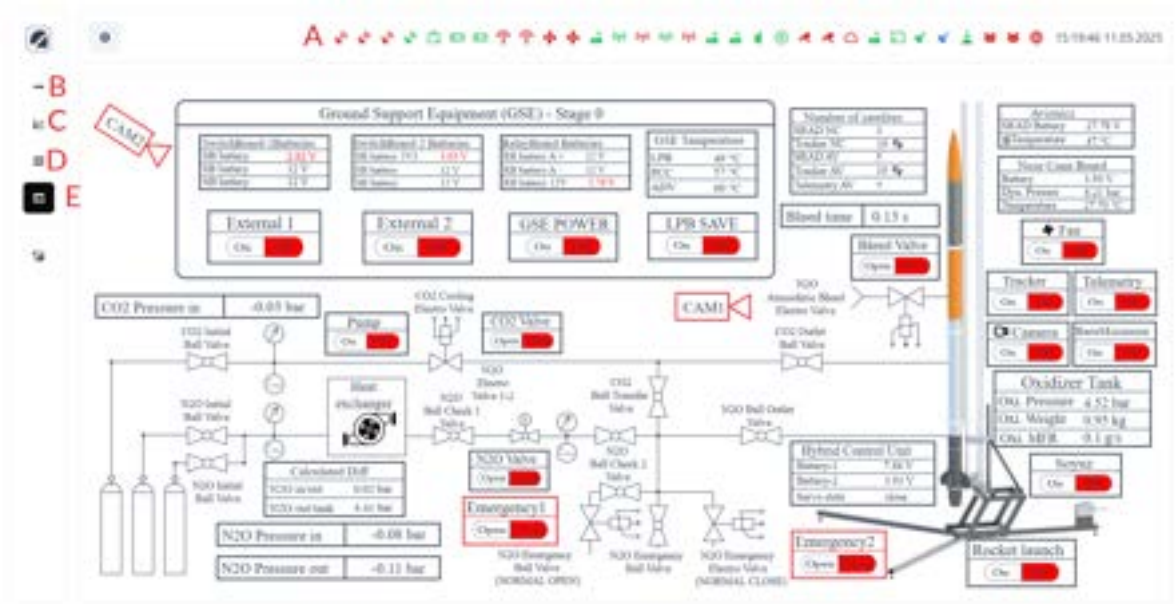
- Widok Raw - wyświetla aktualne wartości przysłane przez wszystkie urządzenia w sieci.
- Wykresy (Rysunek 5a)- pozwalają na wykreślenie zmiany wartości w czasie. Możliwe jest jednoczesne stworzenie wielu wykresów oraz przełączanie pomiędzy zapisanymi wykresami.
- Profile misji (Rysunek 6)- Obraz svg z nałożonymi zmieniającymi się wartościami z urządzeń. Umożliwia przedstawienie całego systemu i wszystkich urządzeń jako spójna całość. Możliwe jest przełączanie się pomiędzy wieloma skonfigurowanymi profilami
- Statusy urządzeń (Rysunek 6)- pokazują na żywo statusy wszystkich urządzeń sieci (Czy jest podłączone, czy wysyła dane).
- Mapy offline (Rysunek 5b) - pokazują na żywo pozycję wszystkich urządzeń wysyłających dane pozycyjne na mapie.

Wszystkie wyświetlane dane konfigurowalne są po stronie backendu. Front-end komunikuje się z backendem przy użyciu SOCKET.io



(A) Widok wykresów PCC (B) Widok map offline PCC

Rysunek 5: Interface graficzny PCC



Rysunek 6: Widok profilu misji PCC, A - statusy urządzeń, B - widok raw data, C - widok wykresów, D - mapy offline, E - profile misji

Funkcje używane do zmieniania wyglądu obiektów na profilu misji w zależności od podanej wartości. Umożliwia to na przykład zaznaczenie ciśnienia na czerwono w przypadku przekroczenia bezpiecznej wartości. Dzięki temu można szybko i sprawnie kontrolować ewentualne anomalie podczas testu.

```
const modifiers: Record<
  ModifierName,
  ModifierFunction<string> | ModifierFunction<number>
> = {
  highlightIfLT: (
    element: HTMLElement,
    value: number,
    threshold: number,
    color: string = "#ff0000"
  ) => {
    if (value < threshold) {
      element.style.fill = color;
    }
  },
  highlightIfMT: (
    element: HTMLElement,
    value: number,
    threshold: number,
    color: string = "#ff0000"
  ) => {
    if (value > threshold) {
      element.style.fill = color;
    }
  },
  highlightIfBitEq: (
    element: HTMLElement,
    value: number,
    bit: number,
    expectedValue: boolean,
    color: string = "#cccccc"
  ) => {
    if (isBitSet(value, bit) == expectedValue) {
      element.style.fill = color;
    } else {
      element.style.fill = "transparent";
    }
  },
};
```

3.4.2 Backend

Back-end systemu został stworzony w formie mikroserwisów - pomniejszych aplikacji komunikujących się wzajemnie. Główna aplikacja, napisana w go służy do zbierania wszystkich danych przesyłanych przy użyciu MQTT, przetwarzania ich i wysyłania ich do front-endu. Umożliwia ona konfigurację oraz przechowuje i wysyła pliki związane z mapami oraz profilami misji. Zapewnia ona również możliwość przetwarzania danych (downsampling, filtrowanie) przed wysłaniem ich do wyświetlenia. Zapewnia to nieprzerwane zbieranie danych oraz zmniejsza ilość obliczeń robionych podczas wyświetlania danych. Zarządza on również sesjami użytkowników, co umożliwia wielu osobom jednoczesne korzystanie z aplikacji oraz wyświetlanie różnych danych

Aplikacja przyjmuje konfigurację podczas włączania. Konfiguracja najpierw pobierana jest z pliku

yaml, po czym nadpisywana jest argumentami konsolowymi.

Format YAML został wybrany ze względu na jego czytelność oraz prostą strukturę.

```
func GetConfig() RunConfig {
    config := getDefaultConfig()
    flag.StringVar(&config.ConfigFile, "config-file", "app-config.yaml", "File with the
        ↪ configuration for the app.")
    config.fromFile()

    flag.IntVar(&config.HTTPPort, "http-port", config.HTTPPort, "Port to host the web server
        ↪ on.")
    flag.IntVar(&config.SocketioPort, "socketio-port", config.SocketioPort, "Port to host the
        ↪ socketio server on.")

    flag.StringVar(&config.MqttHost, "mqtt-host", config.MqttHost, "Address of the mqtt
        ↪ broker.")
    flag.StringVar(&config.MqttTopic, "mqtt-topic", config.MqttTopic, "Topic to listen to for
        ↪ mqtt data.")

    flag.StringVar(&config.DeviceConfigFile, "device-config", config.DeviceConfigFile, "File
        ↪ with the configuration for the devices.")
    flag.StringVar(&config.StatusAppURL, "status-app", config.StatusAppURL, "URL of the status
        ↪ app.")

    flag.StringVar(&config.ProfilesConfigFile, "profiles-config", config.ProfilesConfigFile,
        ↪ "File with the configuration for the profiles.")
    flag.BoolVar(&config.WithFrontend, "with-frontend", config.WithFrontend, "Whether to serve
        ↪ frontend from './frontend' directory")
    flag.StringVar(&config.Services, "services", config.Services, "Which services to use (maps,
        ↪ raw, profiles, plots, *) comma separated and case insensitive.")

    flag.Parse()

    log.Default().Println("Read run config ", config)
    return config
}
```

Z powodu ograniczeń bibliotek służących do wykresów, dane przeznaczone na wykresy muszą zostać poddane downsamplingowi. Zdecydowaliśmy się na metodę Largest Triangle Three Buckets, gdyż zapewnia ona zachowanie krztałtu wykresu przy znacznym zmniejszenu punktów.

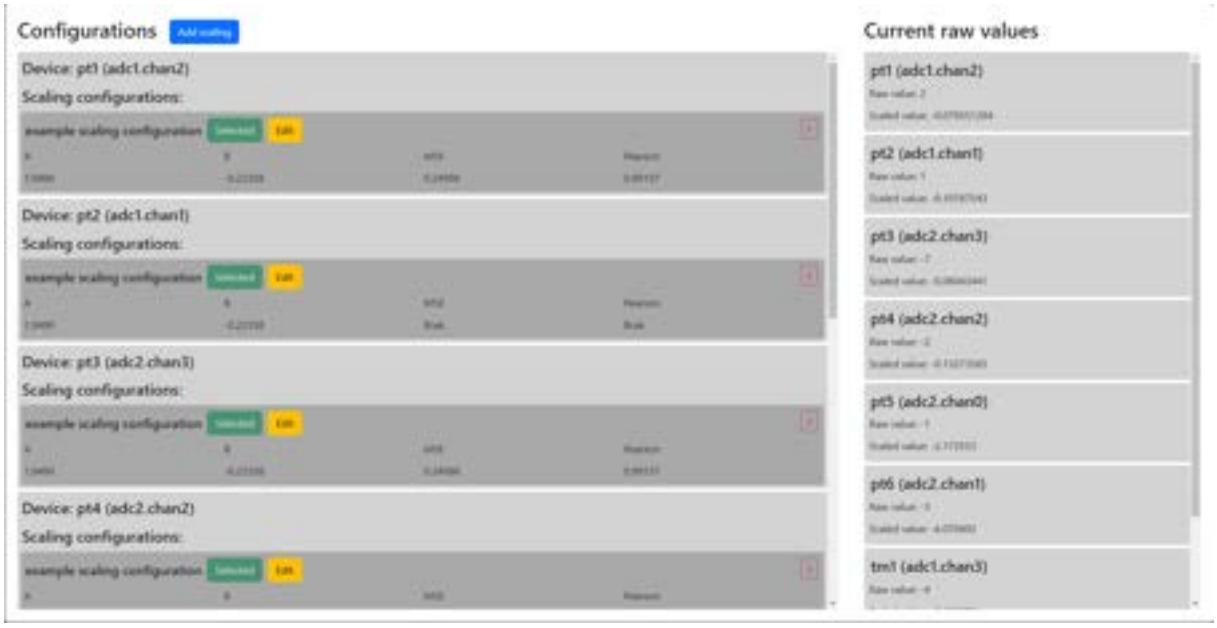
```
func LargestTriangleThreeBuckets(data []DataPoint, threshold int) []DataPoint {
    dataLength := len(data)
    if threshold >= dataLength || threshold == 0 {
        return data // Nothing to do
    }

    bucketSize := float64(dataLength-2) / float64(threshold-2)
    sampled := make([]DataPoint, 0, threshold)

    sampled = append(sampled, data[0])

    for i := 0; i < threshold-2; i++ {
        avgX, avgY := 0.0, 0.0
        for j := int(math.Floor(float64(i)*bucketSize)) + 1; j <=
```

```
            ↪ int(math.Floor(float64(i+1)*bucketSize)); j++ {
            avgX += data[j].Timestamp
            avgY += data[j].Value
        }
        avgX /= bucketSize
        avgY /= bucketSize

        rangeOffs := int(math.Floor((float64(i)+0.5)*bucketSize)) + 1
        nextA := int(math.Floor(float64(i+1.0)*bucketSize)) + 1

        maxArea := -1.0
        var maxAreaPoint = data[rangeOffs]

        for j := rangeOffs; j < nextA; j++ {
            area :=
                ↪ math.Abs(float64(maxAreaPoint.Timestamp*(data[j].Value-avgY)+data[j].Timestamp*(avgY-maxAreaP
                ↪ / 2.0)
            if area > maxArea {
                maxArea = area
                maxAreaPoint = data[j]
            }
        }

        sampled = append(sampled, maxAreaPoint)
    }

    sampled = append(sampled, data[dataLength-1])

    return sampled
}
```

Aplikacja do statusów umożliwia stałe monitorowanie urządzeń w sieci. Odbiera ona dane z wszystkich urządzeń oraz monitoruje połączenie internetowe, co ułatwia szybkie diagnozowanie problemów. Wszystkie zmiany stanu są zapisywane w logach, co umożliwia analizę w przypadku problemów. Aplikacja ta jest sposobem na natychmiastowe wykrywanie oraz diagnozowanie anomalii występujących podczas działania systemu oraz zapewnia niezwłoczne poinformowanie użytkowników w przypadku ich wystąpienia.

3.4.3 Aplikacja do skalowania

Aplikacja do skalowania pozwala na ustawianie konfiguracji urządzeń zbierających dane fizyczne (tensometry, przetworniki ciśnienia, konwertery ADC) oraz na znajdowanie przy użyciu regresji liniowej współczynników zmieniające jednostkę z wartości mierzonej na docelową (np. V, Kg). Dzieli się ona na front-end napisany przy użyciu technologii React.ts oraz backend napisany w języku python. Front-end aplikacji pozwala na zaczęcie oraz przerwanie zbierania nowych punktów, wybieranie oraz edycję wcześniejszych skalowań (Rysunek 7), dodanie nowego skalowania oraz wybór punków które powinny zostać odrzucone ze względu na błędy pomiarowe (Rysunek 8). Backend aplikacji służy do przetwarzania danych, zarządzania sesjami oraz jako interfejs między plikami konfiguracyjnymi oraz front-endem. Ze względów bezpieczeństwa, tylko jedna osoba może jednocześnie tworzyć nowe skalowanie.



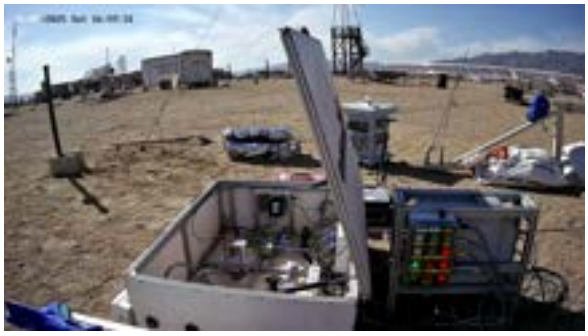
Rysunek 7: Aplikacja do skalowania - widok podstawowy

Rysunek 8: Aplikacja do skalowania - widok skalowania

3.5 System kamer

W celu bezpiecznego zdalnego kontrolowania eksperymentu został wykorzystany system kilku kamer znajdujących się na ziemi oraz jednej w rakiecie. Obraz z nich przekazywany jest w czasie rzeczywistym i dostępny z każdego miejsca w sieci co umożliwia podgląd z wielu urządzeń naraz.

Dwie kamery naziemne (model Zintronic B4) podłączone są przewodowo do Launch Pad Boxa. Mają one za zadanie obserwować zawór upustowy zbiornika rakiety oraz sytem Ground Support Equipement (GSE). Obserwacja zaworu upustowego jest niezbędna ze względu na wizualną metodę pomiaru poziomu zatankowania zbiornika rakiety, a GSE aby upewnić się czy zawroy tankujące działają poprawnie. Oprócz metody wzrokowej została zaimplementowana sieć neuronowa analizuje stan zaworów w GSE (ich położenie) oraz zmieniające się światła na Relayboardzie (również umieszczonym obok GSE) 9. Dane te analizowane są przez mikrokomputer z procesorem w architekturze RISC-V, a następnie przesyłane jako dodatkowy acknowledgment do PCC.



(A) Włączony zawór tankujący

(B) Wyłączony zawór tankujący

Rysunek 9: Obraz z kamer naziemnych

Kamera znajdująca się w rakiecie to Walksnail Moonlight 4K FPV sprzężona z układem FPGA. Zaprojektowany został do niej system OSD (On screen display) pokazujący w czasie rzeczywistym parametry lotu takie jak przyśpieszenia rakiety, jej pozycję oraz dane na temat napięcia baterii wraz z automatycznym systemem powiadamiania o przegrzewaniu urządzenia 10. Dodatkowo cały system jest w stanie wykrywać aktualną fazę lotu o czym automatycznie informuje w postaci stosownej wiadomości na ekranie.

Rysunek 10: Kamera z rakiety w trakcie lotu wraz z OSD

4 Naziemny system akwizycji danych

4.1 Wprowadzenie:

W nowoczesnych systemach rakietowych ogromne znaczenie odgrywa infrastruktura naziemna, która obsługuje i wspiera przygotowanie rakiety do startu. Jednym z najważniejszych elementów tej infrastruktury jest zestaw urządzeń odpowiedzialnych za tankowanie, monitorowanie ciśnienia, kontrolę parametrów środowiskowych oraz zapewnienie komunikacji z systemami pokładowymi rakiety. System Launch Pad Box (LPB) stanowi kompleksowe rozwiązanie inżynieryjne, które umożliwia przeprowadzenie wszystkich kluczowych operacji przygotowawczych w sposób bezpieczny, szybki i efektywny. Jest to szczególnie istotne w środowisku testowym oraz podczas eksperymentalnych startów rakiet hybrydowych, gdzie operacje przy zbiornikach ciśnieniowych i substancjach chemicznych wymagają maksymalnej redukcji ryzyka.

4.2 Potrzeba wdrożenia systemu LPB

System LPB wprowadza zupełnie nową jakość poprzez pełne zautomatyzowanie i przeniesienie obsługi do bezpiecznej odległości. W praktyce pozwala to na:

- redukcję obecności personelu w strefie niebezpiecznej,
- uproszczenie i przyspieszenie operacji logistycznych,
- zwiększenie powtarzalności i niezawodności procedur,
- obniżenie kosztów związanych z obsługą infrastruktury.

4.3 Warunki środowiskowe i konstrukcja fizyczna

LPB został zaprojektowany z myślą o działaniu w ekstremalnych warunkach środowiskowych. System musi funkcjonować niezawodnie na terenach pustynnych, gdzie panują wysokie temperatury, obecny jest pył zawieszony oraz silne nasłonecznienie. Obudowy wszystkich komponentów wykonane są z wytrzymałych materiałów odpornych na promieniowanie UV i uszkodzenia mechaniczne. Wszystkie połączenia i złącza są odpowiednio uszczelnione, aby zapobiec przedostawaniu się pyłu oraz wilgoci do wnętrza urządzeń. Ponadto elementy LPB są łatwo transportowalne – konstrukcja przewiduje możliwość szybkiego przeniesienia systemu w inne miejsce, co jest istotne z punktu widzenia testów terenowych. Modułowa budowa umożliwia również łatwą rozbudowę lub wymianę poszczególnych komponentów bez konieczności demontażu całego systemu.

4.4 Komponenty systemu LPB

System LPB składa się z trzech głównych jednostek funkcjonalnych, które pracują wspólnie jako spójny system obsługi platformy startowej.

4.4.1 Power Supply Box

Jest to moduł zasilającym, który służy jako główne źródło zasilania dla pozostałych części systemu. Wewnątrz znajduje się instalacja umożliwiająca zasilanie bezpośrednio z sieci lub akumulatora samochodowego, który został dobrany ze względu na dużą pojemność, trwałość i możliwość wielokrotnego ładowania. Obecność 2 różnych źródeł zasilania została zaimplementowana w celu zminimalizowania ryzyka wystąpienia zaniku napięcia. System zasilania wyposażono w odpowiednie zabezpieczenia przeciążeniowe oraz systemy kontroli napięcia, co zapewnia stabilną pracę wszystkich urządzeń.

4.4.2 Data Acquisition Box

Pełni funkcję centrum zbierania i przesyłania danych z czujników rozmieszczonych na platformie startowej. W jej wnętrzu znajdują się nowoczesne komponenty elektroniczne, w tym mikrokomputer, przełącznik sieciowy Ethernet oraz zestawy filtrów i przetworników umożliwiających integrację z szerokim zakresem czujników tensometrycznych, ciśnienia i temperatury. Istotną rolę odgrywa także możliwość rejestrowania danych w czasie rzeczywistym, co pozwala na pełną dokumentację każdego cyklu tankowania i testu.

4.4.3 Relay Box

Zawiera zestaw przekaźników elektromagnetycznych, które umożliwiają zdalne sterowanie wszystkimi elementami wykonawczymi systemu. Są to między innymi elektrozawory, pompy, przełączniki i układy bezpieczeństwa. Każdy przekaźnik może być uruchamiany niezależnie, co umożliwia dużą elastyczność operacyjną oraz precyzyjne dostosowanie zachowania systemu do konkretnego scenariusza startowego.

4.5 Komunikacja, infrastruktura sieciowa i integracja

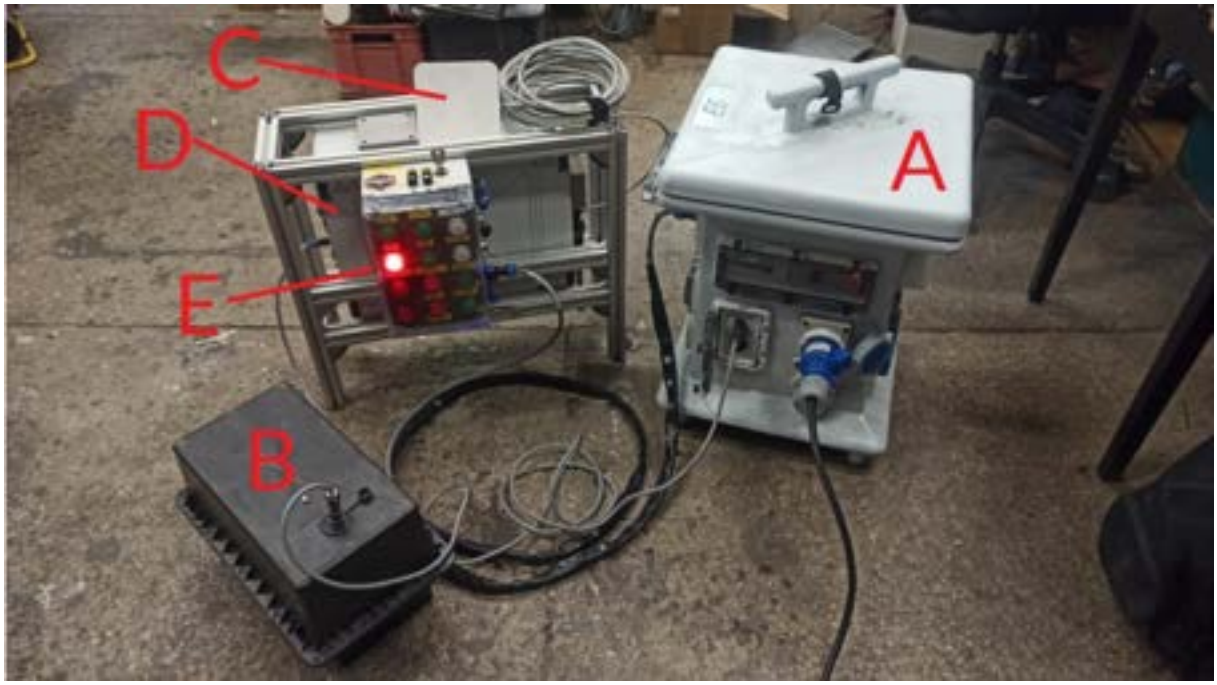
Wszystkie jednostki LPB są połączone za pomocą przewodowej sieci Ethernet oraz Wi-Fi, co zapewnia wysoką niezawodność oraz odporność na zakłócenia elektromagnetyczne, które mogą występować podczas uruchamiania silników rakietowych lub pracy urządzeń o dużej mocy. Dodatkowo zastosowano zewnętrzną antenę MikroTik, która łączy się bezpośrednio z przełącznikiem wewnątrz Data Acquisition Box. Antena ta umożliwia transmisję danych z dużych odległości, co ma kluczowe znaczenie w warunkach testów terenowych, gdzie zasięg standardowych urządzeń może być niewystarczający. Dzięki temu operatorzy mogą zachować pełną kontrolę nad systemem z bezpiecznej lokalizacji oddalonej nawet o kilkaset metrów.

4.6 Efektywność operacyjna i skrócenie czasu przygotowań

Zastosowanie LPB w praktyce skróciło czas przygotowania rakiety do testu lub startu z kilku godzin do około 30 minut. Kluczowe znaczenie miało tutaj zastosowanie dwóch wielożyłowych przewodów zbiorczych, które zastąpiły wcześniejsze indywidualne podłączanie każdego z czujników i zaworów. Zintegrowana logistyka systemu sprawia, że jego instalacja i uruchomienie mogą być przeprowadzane przez dwuosobowy zespół techniczny bez konieczności korzystania ze specjalistycznych narzędzi. Każdy komponent został wyposażony w znormalizowane szybkozłącza, co dodatkowo upraszcza procedury montażowe.

4.7 Wnioski i znaczenie systemu LPB

Zintegrowany system LPB to innowacyjne i unikalne rozwiązanie, które znacząco podnosi poziom bezpieczeństwa i profesjonalizmu w operacjach rakietowych. Jego wdrożenie pozwoliło nie tylko usprawnić logistykę startów i testów, ale także ograniczyć wpływ czynników środowiskowych oraz błędów ludzkich na końcowy przebieg eksperymentów. System ten może stanowić wzorcowy przykład dla przyszłych inwestycji w infrastrukturę startową, zarówno w ramach działalności akademickiej, jak i przemysłowej. Zastosowane rozwiązania są skalowalne i mogą być łatwo adaptowane do większych platform startowych oraz bardziej złożonych systemów nośnych.



Rysunek 11: Gotowy do użytku setup LPB

Launch Pad Box, A - Power Supply Box, B - Akumulator C - Antena mostu sieciowego, D - Data Acquisition Box, E - Relay Box

4.8 System tankowania

Valve Control Unit (VCU) jest odpowiedzialny za obsługę serwosterowanych zaworów w GSE. System bazuje na zmodyfikowanej wersji HECU. Stan zaworów jest zależny od stanu odpowiednich przełączników na switchboard-zie. Zawory zmieniają swoje stany tylko wtedy, gdy płytka wykryje zmianę wartości sygnału z płytki przekaźników, taki mechanizm daje możliwość realizacji dwóch ważnych funkcji. Po pierwsze, w celu zmniejszenia zużycia energii, może on wyłączyć zasilanie serwomechanizmów po upływie określonego czasu, w którym nie zostanie wykryta żadna zmiana sygnału sterującego. Po drugie, ze względów bezpieczeństwa, w przypadku nieoczekiwanego resetu płytki, zawory zostaną zamknięte, a ich otwarcie będzie możliwe dopiero po przełączeniu odpowiednich przełączników ze stanu 0 do stanu 1. Podobnie jak w HECU, kąty otwarcia i zamknięcia można skonfigurować ze strony internetowej. Takie rozwiązanie pozwala na kontrole tankowania po przez głowne centrum dowodzenia, bez konieczności przełączania fizycznie zaworów na GSE.

4.9 Aplikacja do zbierania danych z urządzenia Advantech

Opracowane w ramach działalności koła naukowego rozwiązanie służy do monitorowania ciśnień silnika rakietowego. System składa się z dwóch głównych części.

4.9.1 Część główna

Uruchamiana na komputerze z dedykowaną wersją jądra systemu Linux oraz specjalnie skonfigurowanym kompilatorem gcc. Zakres funkcjonalności:

- Inicjalizacja urządzeń – wszystkie podłączone komponenty są uruchamiane podczas startu systemu.
- Zbieranie danych – odczyty napięć z urządzenia Advantech, odpowiadających wartościom ciśnienia.
- Skalowanie sygnałów – przeliczanie napięć na rzeczywiste wartości ciśnienia na podstawie wcześniej wczytanych charakterystyk dla 16 kanałów.
- Wysyłanie danych – przesył danych w czasie rzeczywistym za pośrednictwem protokołu MQTT.
- Zapis danych – możliwość zapisu danych do bazy po naciśnięciu przycisku (lokalnie lub zdalnie).
- Monitoring urządzeń – stały nadzór nad stanem komponentów oraz informowanie użytkownika o awariach (np. rozłączenie kabla) za pomocą komunikatów głosowych.
- Modularna architektura – ułatwia rozbudowę systemu o nowe funkcjonalności.
- Udostępnione API – umożliwia integrację z interfejsem użytkownika.

4.9.2 Interfejs użytkownika (front-end + serwer pośredni)

- **Serwer pośredni (Python)** – zapewnia bezpieczną i ujednoliconą komunikację pomiędzy częścią główną a interfejsem.
- **Interfejs webowy** – dostępny poprzez hotspot z karty sieciowej komputera, obsługiwany na dowolnym urządzeniu (np. smartfonie).

4.9.3 Dostępne funkcje interfejsu:

- Uruchamianie i zatrzymywanie zapisu danych.
- Wybór lokalizacji zapisu plików.
- Podgląd danych w czasie rzeczywistym.
- Zmiana ustawień systemowych, w tym przeglądanie katalogów.

5 Systemy Awioniczne

5.1 Hybrid Engine Control Unit

HECU (Hybrid Engine Control Unit) to kontroler silnika hybrydowego SRAD. Jest on odpowiedzialny za kontrolowanie pozycji serwomechanizmów głównego zaworu utleniacza. ESP32-S3 jest używany jako centralna jednostka przetwarzająca. Ze względu na odległość HECU od reszty awioniki zdecydowano się na połączenie bezprzewodowe między nimi, z wykorzystaniem WIFI dostępnego właśnie na mikrokontrolerach z serii ESP32-S3.
HECU wykorzystuje algorytm sterowania oparty na czasie. Wykrycie sygnału zapłonu uruchamia licznik czasu, który mierzy opóźnienie zapłonu, po tym czasie HECU wysyła sygnał otwarcia serwomechanizmu, po odpowiednim czasie wysyła sygnał zamknięcia do serwomechanizmów.
Pomiar napięcia akumulatora i stanu otwarcia serwomechanizmu jest wysyłany do PCC (komputera naziemnego) za pośrednictwem MQTT. To rozwiązanie pozwala na ciągłe monitorowanie stanu tak kluczowego elementu rakiety, jakim jest właśnie zawór utleniacza.
HECU hostuje stronę internetową używaną do procesu konfiguracji, do której można uzyskać dostęp pod określonym adresem IP w przeglądarce internetowej. Użytkownicy mogą decydować o czasie otwarcia i zamknięcia zaworu oraz o tym, jaki stopień otwarcia jest interpretowany jako stan otwarcia i zamknięcia. Pozwala to na zdalną częściową kontrole przebiegu pracy, bez potrzeby przeprogramowania czy też jakiekolwiek fizycznego wpinania się w układ sterujący.

5.2 Głowica

Głowica rakiety Hexa 5 pełni kluczową funkcję modułu badawczego, w którym umieszczony jest payload naukowy oraz elektronika pomiarowo-nawigacyjna. Została zaprojektowana jako niezależny, wymienny segment konstrukcyjny, umożliwiający integrację różnorodnych ładunków użytecznych w zależności od misji.

5.2.1 Nose Cone board

Moduł Nose Cone (Rysunek 12), oparty na mikrokontrolerze ESP32-S3, został zaprojektowany do zbierania danych z rurki Pitota, wykorzystując pomiar różnicy ciśnień. W celu zapewnienia wysokiej przepustowości danych, czujnik Honeywell HSCMRRD100PDSA3 jest połączony z mikrokontrolerem za pomocą magistrali SPI (Serial Peripheral Interface). Na podstawie odczytów ciśnienia możliwe jest obliczenie rzeczywistej prędkości rakiety względem powietrza (TAS – True Airspeed).

Rysunek 12: Model płytki Nose Cone

Dodatkowo, moduł może rejestrować dane dotyczące przyspieszeń oraz prędkości kątowych dzięki czujnikowi Bosch BMI088, a także wektor pola magnetycznego przy użyciu czujnika Bosch BMM150.

Oba czujniki komunikują się przez magistralę I2C (Inter-Integrated Circuit).

Poza czujnikami MEMS, napięcie baterii mierzone jest przez wbudowany przetwornik ADC mikrokontrolera ESP32-S3. Wszystkie dane z czujników zapisywane są w formacie binarnym (.bin) w wlutowanym module pamięci flash. Dane te można pobrać poprzez serwer HTTP uruchomiony na ESP32-S3. Interfejs webowy pozwala na ściągnięcie plików binarnych zapisanych w pamięci modułu, które można następnie przekonwertować do formatu .csv. Dodatkowo, interfejs umożliwia konfigurację płytki oraz usuwanie wcześniej zapisanych plików binarnych, aby zwolnić miejsce na nowe dane.

Płytka Nose Cone Board komunikuje się z jednostką RPi Companion poprzez sieć WiFi. Dane z czujników są przesyłane do RPi jako zapasowe miejsce przechowywania, na wypadek niepowodzenia pobrania danych z serwera HTTP. Dzięki połączeniu WiFi, zespół naziemny może monitorować status płytki Nose Cone w trakcie procedury tankowania oraz samego lotu.

5.2.2 GNSS Tracker

Nowy moduł GNSS Tracker (Rysunek 13) to płytka do akwizycji i transmisji danych lotu, oparta na mikrokontrolerze STM32WL55 z wbudowanym transceiverem LoRa® o mocy 100 mW. Ścieżka radiowa korzysta z przełącznika RF, który kieruje sygnał pomiędzy nadajnikiem LoRa a siecią antenową pracującą na częstotliwości 433 MHz. Wszystkie cyfrowe interfejsy (SPI, I²C, UART) zostały wyposażone w układy dopasowujące poziomy logiczne i bufory sygnałowe, co zapewnia niezawodne działanie w warunkach drgań i skrajnych temperatur.

Pozycjonowanie opiera się na odbiorniku GNSS U-blox M10s, pracującym w trybie lotniczym z częstotliwością do 10 Hz, co pozwala na ciągłe rejestrowanie trajektorii nawet podczas szybkiego wznoszenia. Dane inercyjne dostarcza IMU Bosch BMI088 (akcelerometr + żyroskop), który umożliwia krótkotrwałą nawigację inercyjną (dead-reckoning) w przypadku utraty sygnału GNSS. Wysokość barometryczna oraz dane środowiskowe są mierzone przez czujnik Bosch BMP581, którego pomiary ciśnienia uzupełniają dane z GNSS, zapewniając płynniejsze profile wysokości.

Wszystkie strumienie danych – współrzędne GNSS, wektory IMU oraz odczyty barometryczne – są znacznikowane czasowo (timestampowane) za pomocą zegara czasu rzeczywistego mikrokontrolera STM32 i zapisywane bezpośrednio na karcie SD. Po zakończeniu misji dane mogą zostać pobrane z dużą prędkością przez interfejs Segger RTT bezpośrednio z karty pamięci.

Rysunek 13: Model płytki Trackera

5.3 Kosz awioniczny

Kosz awioniczny Hexy 5 to zaawansowany, modularny system odpowiadający za kontrolę, pomiary i komunikację w trakcie lotu rakiety. Został zaprojektowany z myślą o niezawodności i elastyczności, co osiągnięto poprzez podział funkcji na osobne, wyspecjalizowane podsystemy – każdy z nich ma własną płytkę PCB oraz niezależny moduł zasilania. Taka architektura ułatwia diagnostykę oraz zwiększa odporność na awarie.

Sam kosz znajduje się w lekkiej, ale wytrzymałej obudowie z kompozytu epoksydowego z włóknem szklanym, zainstalowanej bezpośrednio nad mocowaniem silnika. Konstrukcja zawiera komponenty z dru-

karki 3D wykonane z tworzywa PET-G oraz elementy ze wzmacnianej sklejki, co pozwala ograniczyć masę do 1,5 kg.

Całość została przemyślana pod kątem łatwego montażu i obsługi – baza przykręcona jest do rakiety za pomocą sześciu śrub M5, a płyty elektroniki są stabilnie mocowane na nylonowych dystansach. Baterie są łatwo dostępne, co ułatwia ich wymianę i ładowanie między lotami.

Rysunek 14: Model kosza awionicznego w rakiecie HEXA 5

5.3.1 Bare Minimum

Bare Minimum (Rysunek 15) to komputer pokładowy SRAD (Student Researched and Developed), którego nazwa pochodzi od minimalistycznej konstrukcji – zawiera jedynie niezbędne komponenty służące do estymacji położenia przestrzennego oraz stanu lotu rakiety. Układ oparty jest na mikrokontrolerze STM32F411. Korzysta z trzech czujników połączonych przez magistralę I2C: barometru MS5611-01BA03, jednostki inercyjnej ICM-20948, oraz czujnika dużych przyspieszeń H3LIS331DL.

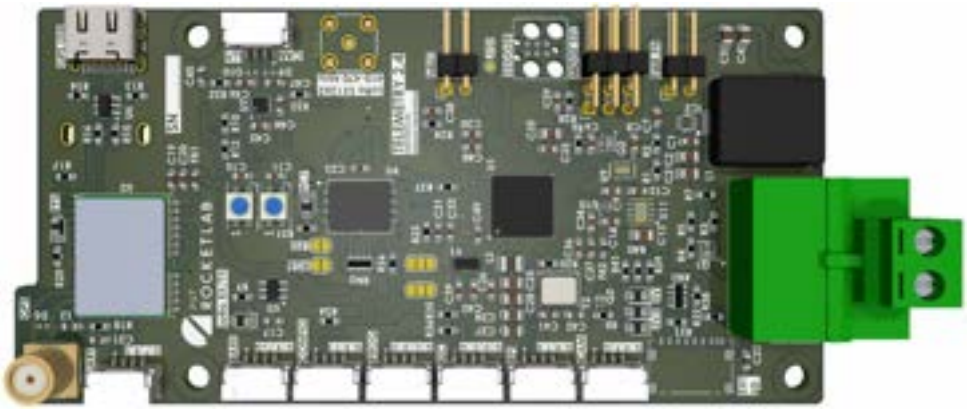
Rysunek 15: Model plytki Bare Minimum

Dane z czujników są przetwarzane przy użyciu algorytmu Madgwicka, służącego do fuzji danych inercyjnych i estymacji orientacji rakiety. Wyniki oraz stan lotu są przesyłane do płytki Telemetry Board za pomocą interfejsu UART w standardzie komunikacyjnym MAVLink. Dodatkowo, wszystkie wysyłane komunikaty są zapisywane lokalnie w pamięci flash (karta SD w obudowie SMT).

Płytka może także służyć jako zapasowy system uwalniania spadochronów, z możliwością testowania ciągłości obwodów zapłonników (e-match). Status lotu komunikowany jest również akustycznie – poprzez generowanie dźwięku za pomocą buzzera, który po lądowaniu działa w trybie ciągłym, ułatwiając zespołowi odzyskiwanie rakiety.

5.3.2 Moduł telemetrii

Telemetry Board (Rysunek 16) pełni rolę głównego węzła routingu danych MAVLink w stosie awioniki SRAD. Zbudowana w oparciu o mikrokontroler STM32H725RG, realizuje zadania agregacji, multipleksowania i przekazywania komunikatów między urządzeniami peryferyjnymi a komputerem pokładowym (Raspberry Pi Companion).

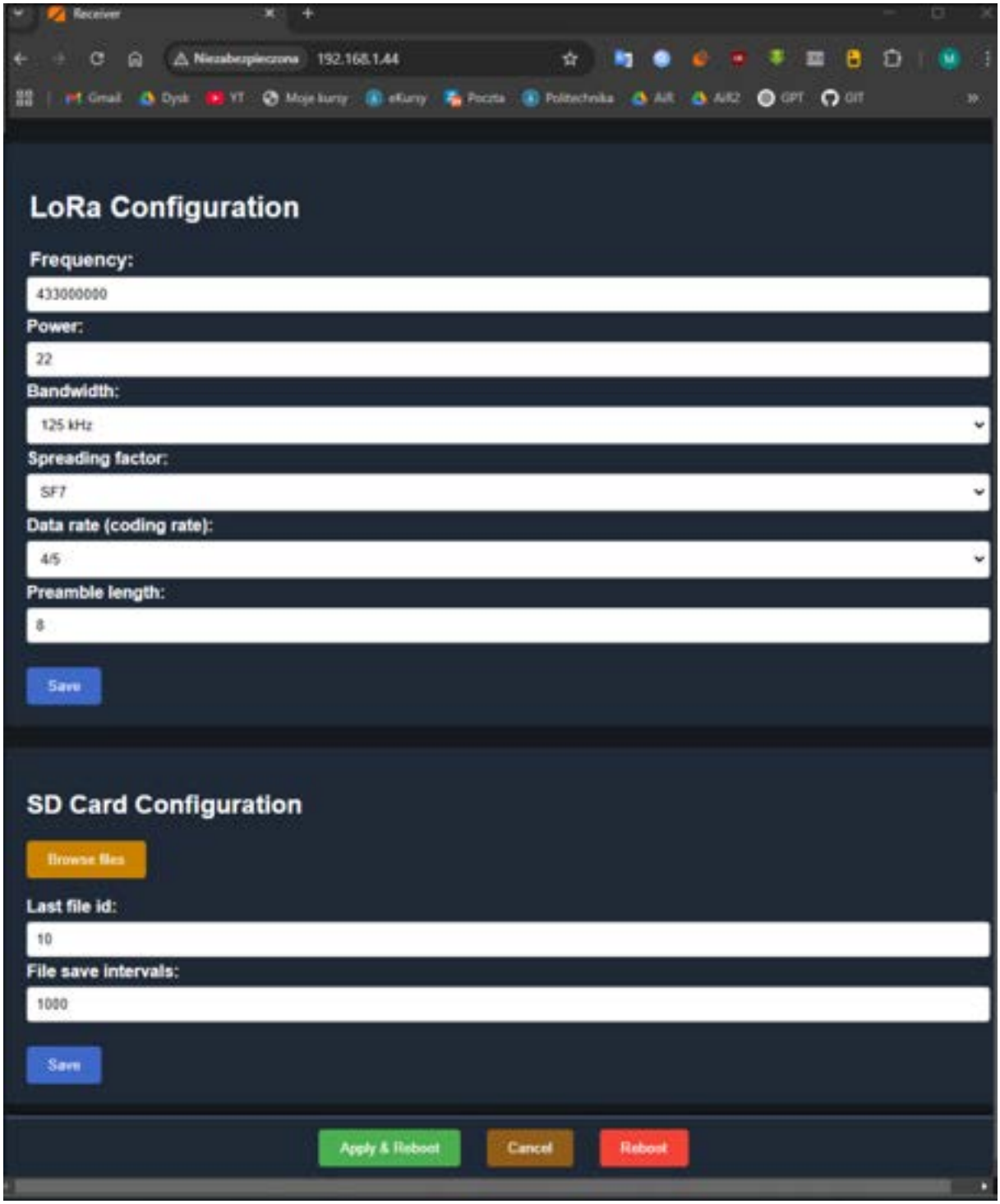


Rysunek 16: Model płytki Telemetry

Szybki interfejs UART (do 921600 baud) łączy płytkę z Raspberry Pi, umożliwiając dwukierunkową wymianę danych telemetrycznych i komend ze stacją naziemną. Pozycja GNSS jest pozyskiwana przez odbiornik u-blox NEO-M9N, pracujący w trybie lotniczym (10 Hz). Każdy pakiet GNSS (UBX) jest znacznikowany czasowo i zapisywany do pamięci za pośrednictwem interfejsu SDMMC.

Moduł Ebyte E22 (z chipem Semtech SX1262, LNA i PA), połączony przez SPI, odpowiada za bezpośredni downlink danych awioniki na ziemię, zapewniając zasięg rzędu kilkudziesięciu kilometrów. Telemetria wideo dostarczana jest przez kamerę Walksnail Moonlight 4K FPV, z nałożonymi danymi telemetrycznymi (OSD - on screen display).

Płytka umożliwia wysokoprzepustowy zapis na SD, a dzięki integracji transmisji Wi-Fi, LoRa i FPV, stanowi centrum komunikacyjne całego systemu SRAD.



Rysunek 17: Strona konfiguracyjna odbiornika LoRa

5.3.3 RPi companion

RPi Companion (Rysunek 18) to nakładka (shield) dla mikrokomputera Raspberry Pi 4B, pełniące rolę źródła zasilania dla niego oraz zaworu upustowego pracującego z napięciem 24V wykorzystywanego w procesie tankowania. Raspberry Pi subskrybuje wiadomości MQTT przychodzęce z Switchboardów z PCC

i za ich pomocą steruje zaworem upustowym, wentylatorami, kamerą pokładową oraz resetami innych płytek awioniki poprzez odpowiednie zmienianie stanów na pinach GPIO.

Obsługa wentylatorów jest niezbędna ze względu na wysokie temperatury występujące w koszu awionicznym.

Zintegrowany przetwornik ADC ADS1115 (połączony przez I2C) służy do pomiaru ciśnienia w zbiorniku utleniacza oraz monitorowania napięcia baterii. Dane z ADC są przesyłane do PCC przez MQTT.

Podczas tankowania komunikacja odbywa się wyłącznie przez Wi-Fi. Po starcie rakiety dane przesyłane są przez UART do jednostki Telemetry, która również przekazuje dane z powrotem do Pi. Wszystkie dane są lokalnie zapisywane w bazie MongoDB, co umożliwia późniejsze przetwarzanie i analizę.

Rysunek 18: Model plytki RPi Companion

5.4 Optymalizacja zasilania bateryjnego

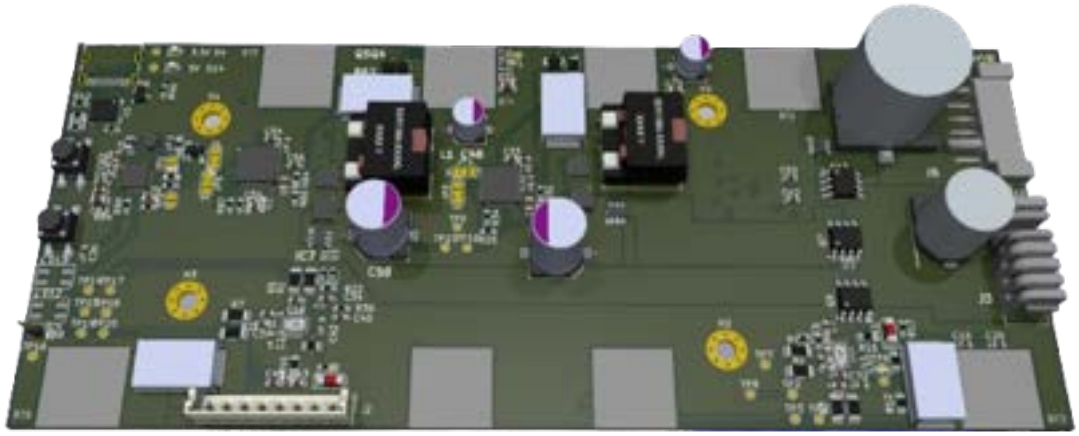
Jedną z ważniejszych cech opracowywanego systemu jest praca w warunkach terenowych bez dostępu do zasilania sieciowego. W celu oszacowania zapotrzebowania na energię przeprowadzono testy złożenia awioniki. Zasilanie komputerów pokładowych zostało przeprowadzone w celu przetestowania możliwych maksymalnych okien pracy dla operacji startu. Oba komputery pokładowe są zasilane ogniwami litowo-jonowymi 18650 o prądzie rozładowania 30A. Wyniki przedstawiono w Tabeli **??**, testy na zimno zostały przeprowadzone w warunkach zimowych (średnia temperatura -10°C) jako oddzielne urządzenia do prawie całkowicie wyczerpanych baterii. Testy na gorąco zostały przeprowadzone w pełnym stosie, z ogrzewaną komorą (40°C) i trwały 10 godzin, testowano działanie serwozaworu HECU (Hybrid Engine Control). Test na gorąco wykazał, że RPi Companion i RPi potrzebują zmian w strategi zasilania

Urządzenie	Szacowany czas pracy [h]	Test na zimno [h]	Test gorący	Test gorący - napięcie przed-po
Blue Raven - komputer komercyjny	70	30	Zdany	8.2V-8.0V
Altimax G4 - komputer komercyjny	120	60	Zdany	8.2V-8.1 V
GPS tracker - komercyjny	22	14	Zdany	4.1V-3.6V
HECU	18	12	Zdany	8.2V-7.1V
RPi Companion + RPi 4 + Kamera	5	3	Niezdany	12.3V-10.2V
GPS tracker	20	13	Zdany	4.1V-3.5V

Tabela 1: Wyniki testów zasilania komponentów systemu

Opracowany system zasilania składa się z dwóch niezależnych pakietów akumulatorów $LiFePO_4$ w konfiguracji 3S1P (trzy ogniwa połączone szeregowo, jeden pakiet równoległy), co zapewnia redundancję

oraz elastyczność zasilania. Każdy pakiet akumulatorów jest wyposażony we własny układ monitoringu stanu ogniw BQ76905 oraz przetwornicę buck-boost w oparciu o sterownik LM51772, która stabilizuje napięcie wyjściowe niezależnie od stanu naładowania akumulatorów. Całość komunikuje się z mikrokontrolerem ESP8684 za pośrednictwem interfejsu I^2C, umożliwiając centralny nadzór nad pracą wszystkich modułów oraz integrację z systemem awioniki (wizualizacja przedstawiona na Rysunku 19). Dodatkowo układ przewiduje możliwość zasilania z zewnętrznego źródła, co zwiększa bezpieczeństwo podczas testowania i uruchamiania systemu, a także pełne działania podczas fazy przygotowania do startu rakiety już na padzie bez rozładowywania pakietów akumulatorów.



Rysunek 19: Model 3D góry zaprojektowanego PCB z komponentami

6 Podsumowanie

Projekt zakończył się sukcesem, wszystkie założone cele zostały spełnione. Całe centrum zdalnej kontroli misji oraz awionika została ponadto skutecznie przetestowana na międzynarodowych zawodach Friends of Amateur Rocketry 2025 na pustyni Mojave w Kalifornii w Stanach Zjednoczonych.

System został sprawdzony w trudnych warunkach terenowych (pył oraz wysokie temperatury) i bez większych problemów umożliwił przeprowadzenie procedury przygotowania rakiety do lotu, zatankowania jej, wydania komendy startu, zbierania danych telemetrycznych w trakcie lotu aż do jej znalezienia po lądowaniu i odczytu wszystkich danych.

Wygodne zbieranie danych miało miejsce z 14 węzłów pomiarowych (wobec zakładanych 10) oraz 7 wykonawczych (wobec zakładanych 5) ze względu na założoną na wstępie bardzo modularną architekturę systemu opartą o urządzenia sieciowe z dostępem do Wi-Fi.

Kosz awioniczny został eksperymentalnie przetestowany w trakcie lotu rakiety i cały czas nadawał wszystkie informacje o położeniu, przyśpieszeniu, prędkości oraz ciśnieniach w zbiorniku rakiety. Dodatkowo nagranie z kamery umieszczonej w rakiecie umożliwiało podgląd przebiegu całego lotu, otwarcia systemów Recovery oraz miejsca lądowania rakiety w czasie rzeczywistym wraz z komunikatami o anomalii w postaci lekkiego przegrzewania się kamery. Wraz z systemem kamer naziemnych obserwujących ewentualne anomalie w działaniu GSE nie wystąpiły problemy z ich identyfikacją oznaczając pełną skuteczność w ich detekcji.

System SCADA okazał się bardzo wygodnym sposobem prezentacji danych oraz umożliwił podgląd wszystkich parametrów z wielu urządzeń. Dodatkowo ze względu na łatwość tworzenia poszczególnych profili misji i całościowej jego konfiguracji świetnie nadaje się on do wielu innych projektów pomiarowych.

Dodatkowo oprogramowanie oraz projekty zostały udostępnione w serwisie GitLab dostępna pod linkiem: `https://gitlab.com/putrocketlab1/portablecommandcenter` i licencją CC BY-NC-SA 4.0 wraz z dokumentacją techniczną.

Projekt "Systemu Kontroli Misji i Akwizycji Danych do Zastosowania w Rakietach Badawczych"okazał się spełniać wszystkie postawione założenia, a nawet wykraczać poza ich przyjęte ramy dodatkowymi funkcjonalnościami. Kluczowym było postawienie początkowych założeń oraz stworzenie spójnego systemu rozwiązań komunikacyjnych, uniwersalnego dla wszystkich komponentów. System można uznać za w pełni przetestowany oraz gotowy do ewentualnych modyfikacji przez innych użytkowników. Świetnie nadaje się on do przeprowadzania testów silników rakietowych oraz ich startów, a także do innych eksperymentów.

Bibliografia

[1] Network time protocol version 4: Protocol and algorithms specification. `https://datatracker.ietf.org/doc/html/rfc5905`, 2010.

[2] What is hmi? `https://www.inductiveautomation.com/resources/article/what-is-hmi`, 2018.

[3] Lora alliance. `https://lora-alliance.org/`, 2023.

[4] Mavlink protocol. `https://mavlink.io/en/`, 2025.

[5] Mongodb compass. `https://www.mongodb.com/products/tools/compass`, 2025.

[6] Mongodb database. `https://www.mongodb.com/`, 2025.

[7] Mqtt: The standard for iot messaging. `https://mqtt.org/`, 2025.

[8] Node-red. `https://nodered.org/`, 2025.

[9] Openwrt. `https://openwrt.org/`, 2025.

[10] Protocol buffers. `https://protobuf.dev/`, 2025.