AKADEMIA GÓRNICZO – HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

PRACA DYPLOMOWA
inżynierska

Implementacja panelu HMI na zestawie uruchomieniowym zawierającym mikrokontroler z serii STM32

Implementation of the HMI panel on a discovery kit containing a STM32 microcontroller

Autor: **Szymon Artur Kwieciński**
Kierunek studiów: Automatyka i Robotyka
Opiekun pracy: **dr inż. Łukasz Jastrzębski**

……..
podpis

Kraków, rok 2019/2020

OŚWIADCZENIA STUDENTA

Kraków, dnia 15.01.2020 r.

Szymon Artur Kwieciński
Imiona i nazwisko studenta

Automatyka i Robotyka, pierwszego stopnia, stacjonarne, ogólnoakademicki

Kierunek, poziom, forma studiów i profil

Wydział Inżynierii Mechanicznej i Robotyki
Nazwa Wydziału

dr inż. Łukasz Jastrzębski
Imiona i nazwisko opiekuna pracy dyplomowej

ja niżej podpisany(-a) oświadczam, że:

jako twórca / współtwórca* pracy dyplomowej inżynierskiej / licencjackiej / magisterskiej* pt.
Implementacja panelu HMI na zestawie uruchomieniowym zawierającym mikrokontroler z serii STM32

1. **uprzedzony(-a) o odpowiedzialności karnej** na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2018 r. poz. 1191, z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, **a także uprzedzony(-a) o odpowiedzialności dyscyplinarnej** na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.” **niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy;**
2. praca dyplomowa jest wynikiem mojej twórczości i nie narusza praw autorskich innych osób;
3. wersja elektroniczna przedłożonej w wersji papierowej pracy dyplomowej jest wersją ostateczną, która będzie przedstawiona komisji przeprowadzającej egzamin dyplomowy;
4. praca dyplomowa nie zawiera informacji podlegających ochronie na podstawie przepisów o ochronie informacji niejawnych ani nie jest pracą dyplomową, której przedmiot jest objęty tajemnicą prawnie chronioną;
5. [TAK]** udzielam nieodpłatnie Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie licencji niewyłącznej, bez ograniczeń czasowych, terytorialnych i ilościowych na udostępnienie mojej pracy dyplomowej w sieci Internet za pośrednictwem Repozytorium AGH.

……
czytelny podpis studenta

Jednocześnie Uczelnia informuje, że:

1. zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony,

autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. – Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem Jednolitego Systemu Antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych;

2. w świetle art. 342 ust. 3 pkt 5 i art. 347 ust. 1 ustawy Prawo o szkolnictwie wyższym i nauce minister właściwy do spraw szkolnictwa wyższego i nauki prowadzi bazę danych zwaną repozytorium pisemnych prac dyplomowych, która obejmuje:tytuł i treść pracy dyplomowej; imiona i nazwisko autora pracy dyplomowej; numer PESEL autora pracy dyplomowej, a w przypadku jego braku – numer dokumentu potwierdzającego tożsamość oraz nazwę państwa, które go wydało; imiona i nazwisko promotora pracy dyplomowej, numer PESEL, a w przypadku jego braku – numer dokumentu potwierdzającego tożsamość oraz nazwę państwa, które go wydało; imiona i nazwisko recenzenta pracy dyplomowej, numer PESEL, a w przypadku jego braku – numer dokumentu potwierdzającego tożsamość oraz nazwę państwa, które go wydało; nazwę uczelni; datę zdania egzaminu dyplomowego; kierunek, poziom i profil studiów. Ponadto, zgodnie z art. 347 ust. 2-5 ustawy Prawo o szkolnictwie wyższym i nauce ww. dane wprowadzają do Zintegrowanego Systemu Informacji o Szkolnictwie Wyższym i Nauce POL-on (System POL-on) rektorzy. Dostęp do danych przysługuje promotorowi pracy dyplomowej oraz Polskiej Komisji Akredytacyjnej, a także ministrowi w zakresie niezbędnym do prawidłowego utrzymania i rozwoju repozytorium oraz systemów informatycznych współpracujących z tym repozytorium. Rektor wprowadza treść pracy dyplomowej do repozytorium niezwłocznie po zdaniu przez studenta egzaminu dyplomowego. W repozytorium nie zamieszcza się prac zawierających informacje podlegające ochronie na podstawie przepisów o ochronie informacji niejawnych.

* - niepotrzebne skreślić;
** - należy wpisać TAK w przypadku wyrażenia zgody na udostępnienie pracy dyplomowej, NIE – w przypadku braku zgody; nieuzupełnione pole oznacza brak zgody na udostępnienie pracy.

STUDENT'S DECLARATIONS

Kraków, 15.01.2020

Szymon Artur Kwieciński
Student's full name

Automatics Control and Robotics, first cycle study programme, Full-time Course of Study, general academic
Field of study, level, form of studies and educational profile

Faculty of Mechanical Engineering and Robotics
Name of the Faculty

dr inż. Łukasz Jastrzębski
Full name of thesis supervisor

I, the undersigned, hereby certify that:

as the author/co-author of the engineering thesis/undergraduate final diploma thesis/graduate final diploma thesis with the title of
Implementation of the HMI panel on a discovery kit containing a STM32microcontroller

1. **warned of the criminal liability** as set forth in Article 115 section 1 and 2 of the Act of 4th February 1994 on copyright and related rights (uniform text Polish Journal of Law *[Dziennik Ustaw]* from 2018 item 1191 as amended): „Who appropriates the authorship or misleads as to the authorship of the entirety or part of somebody else's work or artistic work shall be subject to a fine, restriction of freedom penalty or of imprisonment up to 3 years. The same penalty shall be imposed on one who distributes (without acknowledging the author by their name or pseudonym) someone else's work in the original version or in an adapted form, the artistic work or who publicly alters such work, artistic work, sound record, videogram or transmission ", **and also warned of the disciplinary responsibility** as set forth in Article 307 section 1 of the Act of 20th July 2018 - Law on higher education and science (Polish Journal of Law *[Dziennik Ustaw]* from 2018 item 1668 as amended) „A student shall be subject to disciplinary responsibility for the infringement of university regulations or for acts detrimental to a student's dignity." **I created this final diploma thesis by myself and independently and I have not used sources other than the ones acknowledged in the final diploma thesis;**
2. the final diploma thesis is a result of my work and does not infringe the copyright of other people;
3. the electronic version of the submitted paper version of my final diploma thesis is the final version, which will be presented in front of the Commission conducting the final diploma examination;
4. this final diploma thesis does not contain information subject to protection as set forth in the regulation on the protection of classified information and does not contain legally protected secret information;
5. [YES]** I grant Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie a non-exclusive license free of charge without time, territorial and quantity limitations to provide access to my final diploma thesis on the Internet network through the medium of Repozytorium AGH [AGH Repository].

……
student's legible signature

At the same time the University informs that:

1. pursuant to Article 15a of the above mentioned act on copyright and related rights the university has the priority right to publish the student's final diploma thesis. If the university has not published the final diploma thesis within 6 months of the final diploma examination, the author is allowed to publish it unless the final diploma thesis is a part of collective work. Furthermore, the university as an entity referred to

in Article 7 section 1 item 1 of the Act of 20th July 2018 – Law on higher education and science (Polish Journal of Law *[Dziennik Ustaw]* from 2018 item 1668 as amended), is allowed to use (without renumeration and without attaining the author's consent) the work created by the student resulting from fulfilling the duties connected with his studies, as well as to make the work available to the minister in charge of higher education and science, and to make use of works located in databases kept by the minister in order to verify the thesis with the usage of Jednolity System Antyplagiatowy [the Uniform Anti-plagiarism System]. The minister in charge of higher education and science is allowed to make use of final diploma theses located in databases kept by him to the extent necessary to ensure the correct maintenance and development of these databases and IT systems working with them;

2. pursuant to Article 342 section 3 item 5 and Article 347 section 1 of the act - Law on higher education and science the minister in charge of higher education and science maintains a database called the repository of written final diploma theses, which includes: the title and content of the final diploma thesis; full name of the author of the final diploma thesis; PESEL number of the author of the final diploma thesis and if they do not have PESEL – the number of the document confirming their identity and the name of the country that issued the document; full name of the thesis supervisor, PESEL number and if they do not have PESEL – the number of the document confirming their identity and the name of the country that issued the document; full name of the thesis reviewer, PESEL number and if they do not have PESEL – the number of the document confirming their identity and the name of the country that issued the document; the name of the university; the date of the final diploma examination; the field of study, level and educational profile. Furthermore, pursuant to Article 347 sections 2-5 of the Act - Law on higher education and science the above mentioned data is entered into Zintegrowany System Informacji o Szkolnictwie Wyższym i Nauce POL-on (System POL-on) [the Integrated Information System on/governing Higher Education and Science POL-on (System POL-on)] by rectors. The access to the data is available to the final diploma thesis supervisor and PKA [Polish Accreditation Committee], as well as the minister to the extent necessary to ensure the correct maintenance and development of the repository and IT systems working with this repository. The Rector enters the content of the final diploma thesis into the repository immediately after the student has passed his final examination. The repository does not contain theses including information protected pursuant to regulations governing the protection of classified information.

* - delete as necessary;
** - enter TAK/YES if you agree to provide access to final diploma thesis, NIE/NO – if you do not agree; if you do not fill this in you are not consenting to share your work.

Akademia Górniczo-Hutnicza im. Stanisława Staszica Kraków, 15.01.2020
Wydział Inżynierii Mechanicznej i Robotyki

Kierunek studiów: Automatyka i Robotyka

Specjalność:

Szymon Kwieciński

Praca dyplomowa inżynierska
Implementacja panelu HMI na zestawie uruchomieniowym zawierającym mikrokontroler z serii STM32.

Opiekun: *dr inż. Łukasz Jastrzębski*

STRESZCZENIE

Celem pracy inżynierskiej było stworzenie panelu dotykowego, za pomocą którego będzie można sterować robotem mobilnym. W pracy kolejno omawiam zagadnienia związane z panelami HMI, takie jak: podanie podstawowych definicji, omówienie oprogramowania, przedstawienie zastosowań, oraz komercyjnych rozwiązań. Następnie przechodzę do tematyki mikrokontrolerów. Podając podstawowe definicje oraz możliwe zastosowania. Szczegółowo analizuję ofertę mikrokontrolerów i mikroprocesorów firmy STMicroelectronic opisując cztery główne rodzinny ich produktów. Przedstawiam budowę i parametry techniczne dwóch zestawów uruchomieniowych z mikrokontrolerami STM32 wyposażonymi w dotykowe panele LCD. Na koniec opisuję oprogramowanie i biblioteki przeznaczone dla mikrokontrolerów STM32, które użyłem w swoim projekcie. Po części teoretycznej w której opisuje zagadnienia związane z mikrokontrolerami i panelami HMI przechodzę do części projektowej i praktycznej. Przedstawiam założenia projektowe, oraz na ich podstawie dobieram niezbędne komponenty. Prezentuję za pomocą schematu blokowego logikę działania panelu HMI sterującego robotem mobilnym. Projektuje graficzny interfejs do panelu i opisuje jego funkcjonalności. Następnie dokładnie opisuje program napisany przeze mnie w języku C na mikrokontroler STM32. Program napisałem z użyciem bibliotek systemu czasu rzeczywistego, dzięki czemu mogłem podzielić zasoby mikrokontrolera na dwa główne wątki. Jeden odpowiedzialny za obsługę ekranu dotykowego. Drugi za obsługę peryferiów i całą logikę programu. Po czym wykonałem trzy testy funkcjonalne zaimplementowanego panelu HMI, sprawdzając komunikacje przewodową i bezprzewodową, sterowanie robotem, oraz odbieranie informacji zwrotnych od robota.

AGH University of Science and Technology
Faculty of Mechanical Engineering and Robotics

Kraków, the 15.01.2020

Field of Study: Automatics Control and Robotics

Profile/speciality*:

Szymon Kwieciński

Engineer Diploma Thesis

Implementation of the HMI panel on a discovery kit containing a STM32 microcontroller.

Supervisor: *dr inż. Łukasz Jastrzębski*

SUMMARY

The purpose of the Engineering Thesis was to create a touch panel, which can be used to control the mobile robot. In my work, I discuss issues related to HMI panels, such as: providing basic definitions, discussing software, presenting applications, and commercial solutions. The next part of my thesis is focused on the subject of microcontrollers. Then, I present their basic definitions and possible applications. In details, I discuss the offer of STMicroelectronics microcontrollers and microprocessors, introducing the four main family products. I present two development kits with STM32 microcontrollers. Finally, I describe the software and libraries for STM32 microcontrollers that I used in my project. After a theoretical part, in which I describe issues related to microcontrollers and HMI panels, I go to the design and practical part. I present the design assumptions based on which I selected the components for the project. Then, by the use of a block diagram, I clearly explain the logic of functioning the HMI panel, which controls the mobile robot. In the next part, I design a graphical interface of the panel and describe its practicality. Subsequently, I thoroughly describe the program, which I wrote in C language for the STM32 microcontroller. For creating the program I used real-time system libraries, so I could divide microcontroller resources into two main groups. The first one is responsible for operating the touch screen. The second one for peripheral support and all program logic. Afterwards, I performed three functional tests: checking wired and wireless communication, controlling the robot and receiving feedback from the robot.

Kraków, dnia 15.01.2020

AKADEMIA GÓRNICZO-HUTNICZA
WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

TEMATYKA PRACY DYPLOMOWEJ

Szymon Kwieciński
imię i nazwisko studenta

Tytuł pracy dyplomowej:
Implementacja panelu HMI na zestawie uruchomieniowym z serii STM32

Promotor pracy: *dr inż. Jastrzębski Łukasz*

Recenzent pracy: *dr hab inż. Andrzej Sioma*

…….
Podpis dziekana

PLAN PRACY DYPLOMOWEJ (Należy ustalić z promotorem, np.:)

1. Omówienie tematu pracy i sposobu realizacji z promotorem.
2. Zebranie i opracowanie literatury dotyczącej paneli HMI i mikrokontrolerów.
3. Implementacja panelu HMI na zestawie uruchomieniowym zawierającym mikrokontroler z serii STM32.
4. Przeprowadzenie testów funkcjonalnych zaimplementowanego panelu HMI.
5. Opracowanie redakcyjne.

Kraków, ... ……..
data *podpis dyplomanta*

…….
podpis promotora

Termin oddania do dziekanatu: _____ 20____ r.

Spis treści:

1.Wstęp

W pracy inżynierskiej chce zaimplementować panel HMI na zestawie uruchomieniowym STM32F746G-DISCOVERY, który jest wyposażony w panel dotykowy LCD.

Współcześnie technologia związana z dotykowymi panelami LCD rozwinęła się w takim stopniu, że każdy z nas ma przy sobie taki panel. Mianowicie ekrany dotykowe w naszych smartfonach lub w laptopach. poszerzenie rynku o segment konsumencki spowodowało ogromny skok technologiczny jak i drastyczny spadek cen paneli LCD, co wpłynęło znacznie na branżę przemysłową. W której niska cena ekranów sprawiła, że panele HMI stały się standardem. Panele HMI, pełnią funkcje interfejsu między człowiekiem a maszyną. Umożliwiają śledzenie parametrów pracy maszyny, oraz zmianę ich, w łatwy, ergonomiczny sposób. Dawniej operator wykonywał te czynności przy pomocy panelu sterowniczego, zbudowanego z przycisków, przełączników, klawiatur, diod sygnalizujących. Niosło to ze sobą znaczne ograniczenia. Raz zaprojektowany panel operatorski bardzo trudno było zmodyfikować. Wiązało się to z dużą ingerencją w strukturę mechaniczną i elektroniczną panelu. Nowoczesne panele HMI charakteryzują się ergonomicznym interfejsem graficznym i dzięki odpowiedniemu oprogramowaniu, które jest dostarczane przez producentów paneli HMI można łatwy i szybki sposób stworzyć zaawansowaną aplikację, jak i jej późniejszą zmianę w zależności od zmian w procesie produkcyjnym. Zaprogramowany panel HMI wraz z sterownikami przemysłowymi PLC tworzą kręgosłup dzisiejszych linii produkcyjnych. Panele pozwalają pracownikom nieposiadającym wiedzy programistycznej na intuicyjną obsługę wybranych maszyn.

Motywacja do podjęcia danej pracy była chęć zbudowania własnego panelu HMI i zapoznania się z programowaniem nowoczesnych 32 bitowych mikrokontrolerów.

1.1.Cel pracy inżynierskiej

Celem pracy inżynierskiej było zaprojektowanie oraz implementacja panelu HMI (ang. *Human Machine Interface*) na zestawie uruchomieniowym wyposażonym w dotykowy panel LCD i zawierający mikrokontroler serii STM32. Panel HMI ma być przenośnym urządzeniem sterującym robotem mobilnym z niezależnym napędem czterech kół. Komunikacja między panelem HMI i robotem ma się odbywać bezprzewodowo dzięki modułom ZigBee. Panel ma umożliwiać zadanie 7 rodzajów ruchu, przesyłając do robota

informacje o prędkości każdego z czterech kół oraz ma umożliwiać zmianę wartość tej prędkości. Ponadto panel HMI ma prezentować informacje na temat stanu aktualnego połączenia z robotem oraz o prędkości każdego z czterech kół (enkoder wbudowany w każdy z silników napędowych).

1.2.Zakres pracy inżynierskiej

Zakres pracy obejmuje:

- Przedstawienie podstawowych informacji dotyczących paneli HMI
- Przegląd mikrokontrolerów z serii STM32
- Dobór mikrokontrolera oraz panelu dotykowego
- Stworzenie interfejsu graficznego dla panelu dotykowego
- Stworzenie oprogramowania dla mikrokontrolera

1.3.Plan pracy

Praca jest podzielona na 7 rozdziałów. Pierwszy rozdział stanowi wstęp, zakres pracy oraz założenia projektowe. Drugi rozdział zawiera podstawowe informacje dotyczące paneli HMI. Przedstawiono w nim definicje, najważniejsze parametry paneli oraz przegląd ich rozwiązań komercyjnych. Przedstawia również czołowych producentów i niektóre zastosowania paneli HMI. Rozdział trzeci dotyczy przeglądu mikrokontrolerów firmy STMicroelectronics z rodziny STM32 oraz środowisk programistycznych użytych do stworzenia oprogramowania. Podaje w tym rozdziale najważniejsze definicje dotyczące mikrokontrolerów. Przedstawiam komercyjna ofertę różnych mikrokontrolerów i zestawów uruchomieniowych firmy STMicroelectronics. Opisuje środowiska programistyczne użyte do napisania oprogramowania na mikrokontroler, środowiska użytego do stworzenia interfejsu graficznego oraz użyte biblioteki programistyczne. W rozdziale czwartym dobieram mikrokontroler i panel dotykowy na podstawie zdefiniowanych założeń projektowych. Opisuję budowę i parametry zastosowanego do komunikacji bezprzewodowej modułu radiowego bazującego na protokole ZigBee. Przedstawiam schemat blokowy działania części sprzętowej projektu panelu HMI, na którym pokazuję sposób komunikacji robota z panelem. Przedstawiam również projekt interfejsu graficznego stworzonego dla panelu dotykowego. W piątym rozdziale szczegółowo opisuje poszczególne sekcje kodu programu. W szóstym rozdziale

opisuję przeprowadzone testy funkcjonalne panelu HMI połączonego z robotem mobilnym. Siódmy rozdział jest rozdziałem podsumowującym.

2.Panele HMI

Panele HMI są coraz bardziej powszechne i stają się obecnie standardem stosowanym zarówno w przemyśle jak również w urządzeniach codziennego użytku. Pozwalają w efektywny sposób komunikować się człowiekowi z maszyną. Są również często stosowane w "inteligentnych domach", pozwalając użytkownikowi zadawać i nadzorować parametry użytkowe jak np. temperatura panująca w pomieszczeniach i stopień ich oświetlenia. Ponadto prezentują informacje z czujników znajdujących się w budynku w sposób czytelny dla człowieka. W branży przemysłowej panele HMI pozwalają w łatwy i przejrzysty sposób odczytywać dane z maszyny lub kontrolować jej parametry pracy. Wszystko to stało się możliwe dzięki znacznemu spadkowi cen paneli HMI w ostatnich latach jak i dostępności oprogramowania, które pozwala osobom nieposiadającym zaawansowanej wiedzy informatycznej tworzyć rozbudowane aplikacje na panele HMI.

2.1.Definicja panelu HMI

HMI (ang. *Human Machine Interface*) jest to system, który umożliwia komunikację człowieka z maszyną, poprzez graficzny interfejs. Zapewnia wizualizację danych w czasie rzeczywistym oraz możliwość zadawania parametrów pracy. Oprogramowanie zazwyczaj od razu jest dostępne z panelem dotykowym [1].

Panele HMI są montowane bezpośrednio przy jednej maszynie lub przy jednym obiekcie automatyki i służą do sterowania, zadawania parametrów, a także do informowania operatorów o bieżącym stanie tej maszyny. Działają w czasie rzeczywistym. Powinny być intuicyjne w obsłudze, zapewniać czytelność informacji, a także cechować trwałością i niezawodnością. Ich użycie ogranicza się do małych aplikacji o ograniczonych wymaganiach, ale mogą być częścią większych systemów SCADA (Supervisory Control And Data Acquisition) lub MES (Manufacturing Execution System). Aktualnie systemy HMI posiadają wiele dodatkowych możliwości dostępnych wcześniej tylko dla systemów SCADA takich jak alarmowanie, śledzenie produkcji, obsługa receptur, harmonogramy. Coraz częściej panele HMI można spotkać jako małe urządzenia przenośne pozwalające sprawdzać parametry pracy z każdego miejsca [2].

2.2.Parametry paneli HMI

Panele HMI cechują się następującymi parametrami użytkowymi:

● Liczba dostępnych portów komunikacyjnych np.(RS232/422/485, Ethernet, USB Client/Host)
● Parametry obrazu np.(wielkość panelu, rozdzielczość ekranu, rodzaj matrycy dotykowej, obsługa multi-touch)
● Oprogramowanie do tworzenia i obsługiwania graficznego interfejsu
● Cena
● Odporność środowiskowa
● Wydajność (możliwości obliczeniowe)
● Łatwość czyszczenia
● Jakość wykonania
● Jakość oraz czas wsparcia producenta
● Integracja z systemami wyższego rzędu
● Ilość obsługiwanych protokołów komunikacyjnych [3]

2.3.Przegląd komercyjnych rozwiązań

Na dzień dzisiejszy istnieje szeroka gama producentów mających w swoim portfolio panele HMI. Są to takie firmy jak:

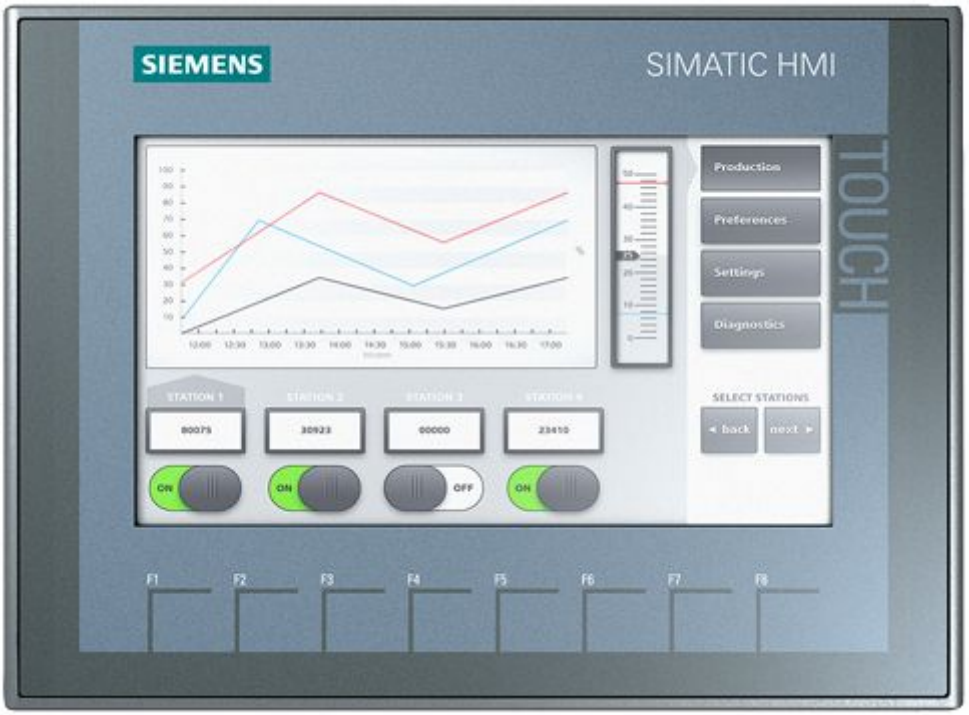
- Siemens
- Mitsubishi
- Omron
- Rockwell Automation
- ADVANTECH
- EATON
- KINCO
- PHOENIXCONTACT
- Shneider Electric
- Weintek
- Astor
- AAEON Europe
- Amtek
- ANIRO
- AT Control System
- Beckhoff Automation
- Bosch Rexroth
- B&R
- Dacpol
- Eaton Electric
- Elfa Distrelec
- Elhurt
- Elmark Automatyka
- Fatek Polska

Warto też zwrócić uwagę że takie firmy jak Siemens, Mitsubishi, Omron, Schneider Electric, B&R, Phoenix Contact, Beckhoff Automation, Bosch Rexroth, Rockwell Automation dostarczają pełen pakiet produktów automatyki przemysłowej, Od oprogramowania, przez sterowniki PLC po szeroką gamę urządzeń peryferyjnych, dzięki którym możemy zbudować układ automatyki korzystając z produktów tylko jednego producenta.

Z powodu wielkości i różnorodności oferty wszystkich producentów na potrzeby tej pracy przedstawiłem poniżej trzy interesujące rozwiązania paneli HMI. Ich opis ma charakter bardzo ogólny i nie przedstawia szczegółowo wybranych produktów.

2.3.1.Standardowy panel HMI

Najpowszechniej używany rodzaj paneli HMI. Zazwyczaj umiejscowionych przy maszynie lub linii produkcyjnej. Jednym z jego przykładów jest panel KTP700 BASIC (Rys.2.1) firmy Siemens wyposażony w panoramiczny dotykowy ekran 7 cali TFT (ang. *thin-film-transistor*) o rozdzielczości 480x800 px (65536 kolorów). Posiada 8 przycisków funkcyjnych i interfejsy komunikacyjne PROFIBUS/MPI. Programowalny za pomocą oprogramowania producenta TIA PORTAL WinCC V13/STEP7 V13(lub nowszych wersji). Panel jest kompatybilny ze wszystkimi sterownikami PLC firmy Siemens lecz producent zaleca stosowanie modelu PLC S-1200. Warto wspomnieć że procesor tego panelu HMI jest oparty na architekturze firmy ARM. Kosztuje około 3500zł [4].



Rys.2.1.Panel HMI KPTBasic700 Simens
Źródło:https://www.automation.siemens.com/bilddb/interfaces/InterfaceImageDB.asmx/GetImageVariant?objectkey=P_ST80_XX_02329&imagevariantid=19&lang=XX&interfaceuserid=MALL

2.3.2.Tablet przemysłowy

Coraz częściej zamiast stacjonarnie umiejscowionych paneli HMI używa się tabletów z systemem windows lub android. Tablet zazwyczaj łączy się z serwerem jako klient i wyświetla tylko najpotrzebniejsze informacje. Dzięki modułom GPS dane na panelu HMI są uzależnione od miejsca gdzie znajduje się operator. Jednym z takich przykładów jest SIMATIC ITP1000 (Rys.2.2). Tablet z 10 calowym wyświetlaczem o rozdzielczości 1280 x 1920 px, obsługującym multi-touch. Jest wyposażony w procesor intel core i5, ko-procesor Celeron G3902, 8 gb pamięci ram, 252gb pamięci wewnętrznej. Obsługuje interfejsy komunikacyjne Ethernet (RJ45), USB_V3.0, RS232, mini display port, USB C. Zapewnia ochronę przed pyłem i wodoszczelność o klasie IP40. Cena waha się w zależności od doboru podzespołów, ale średnia cena wynosi 9000zł [5].

Rys.2.2.Tablet przemysłowy SIMATIC ITP1000
Źródło:https://www.automation.siemens.com/bilddb/interfaces/InterfaceImageDB.asmx/GetImageVariant?objectkey=P_ST80_XX_03021&imagevariantid=19&lang=XX&interfaceuserid=MALL

2.3.3.Panele HMI oparte na mikrokontrolerach STM32

Wraz z rozwojem branży mikrokontrolerów firmy takie jak STMicroelectronics zaczęły rozszerzać swoją ofertę o rozwiązania programistyczne i urządzenia, dzięki którym jesteśmy w stanie stworzyć panele HMI dla systemów wbudowanych. Jednym z przykładów jest zestaw uruchomieniowy STM32F746G-DISCOVERY (Rys.2.3). Zestaw zawiera mikrokontroler z rdzeniem ARM Cortex-M7, pojemnościowy ekran LCD-TFT o przekątnej 4.3 cala i rozdzielczości 480x272 px. Posiada 1 Mbit pamięci FLASH oraz 340 Kbitów pamięci RAM, dodatkowo pamięć można rozszerzyć o 128 Mbitów pamięci FLASH. Jest wyposażony w Ethernet RJ45, wyjścia ARDUINO oraz USB. Kosztuje około 250zł [6].

Rys.2.3.Zestaw uruchomieniowy STM32F746G-DISCOVERY
Źródło:https://www.st.com/bin/ecommerce/api/image.PF261641.en.feature-description-include-personalized-no-cpn-medium.jpg

2.4.Oprogramowanie dedykowane panelom HMI

Interfejs użytkownika panelu HMI można przygotować za pomocą fabrycznego oprogramowania producenta, lecz jego możliwości są zazwyczaj bardzo ubogie. Ograniczenie dotyczy ilości obsługiwanych zmiennych. Dla bardziej zaawansowanych zastosowań producenci oferują osobne pakiety rozbudowanych programów SCADA/HMI, które instalowane na osobnych komputerach przemysłowych zapewniają większe możliwości i nie są uzależnione od fizycznego sprzętu.

W przypadku systemów wbudowanych producenci udostępniający oprogramowanie oferują jedynie niewielkie programy, które pomagają tworzyć interfejs graficzny, lecz całą logikę i programowanie musi napisać programista.

2.4.1.Wonderware InTouch

Przykładowym rozbudowanym oprogramowaniem HMI/SCAD jest WonderwareInTouch (Rys.2.4), który umożliwia projektowanie aplikacji oraz ich wizualizację. Ponadto zapewnia ogromną bazę gotowych obiektów graficznych co usprawnia budowanie aplikacji. W oprogramowaniu zawarte są również programy służące do komunikacji z różnego rodzaju sterownikami, co również znacznie ułatwia pracę. Program ponadto można rozbudować o oprogramowanie Historian Client, który jest przeznaczony do zbierania i analizy danych procesowych. Oprogramowanie umożliwia udostępnianie aplikacji na serwerze i odczytywanie ich na dowolnym urządzeniu podłączonym do lokalnej sieci [7].

Rys.2.4.Oprogramowanie Wonderware InTouch
Źródło: https://www.astor.com.pl/images/InTouchWonderware_website/main_intouch.png

2.4.2.SIMATIC WinCC

Oprogramowania nadrzędnego sterowania i akwizycji danych SCADA oraz interfejs operatorski HMI, produkowany przez firmę Siemens. System ten pozwala na tworzenie aplikacji służących do monitorowania i sterowania fizycznymi procesami przemysłowymi i infrastrukturalnymi obejmującymi proste, złożone jak i rozproszone struktury konfiguracji. SIMATIC WinCC może być wykorzystywany w ramach systemów DCS SIMATIC PCS7 oraz Teleperm. WinCC zostało opracowane pod kątem pracy w środowisku Microsoft Windows. System wykorzystuje bazę danych Microsoft SQL Server do długoterminowej archiwizacji danych procesowych oraz pozwala na wykorzystanie aplikacyjnych interfejsów programistycznych VBScript i ANSI C [8].
Warto nadmienić że firma Siemens ma w swoim portfolio oprogramowanie do kompleksowego tworzenia systemów Automatyki od sterowników PLC, oprogramowania, po panele HMI i dużą ilość urządzeń peryferyjnych.

2.4.3.TouchGFX

Oprogramowanie stworzone przez firmę STMicroelectronic. TouchGFX (Rys.2.5) jest darmowym oprogramowaniem zoptymalizowanym dla mikrokontrolerów STM32. Korzystając z zaawansowanych funkcji graficznych i architektury mikrokontrolerów STM32 przyspiesza tworzenie interfejsów graficznych. TouchGFX zawiera w sobie program TouchGFX Designer, który w łatwy sposób pozwala stworzyć GUI, dzięki gotowym obiektom graficznym, które upuszczamy na ekran, oraz wygenerować gotowy kod w języku C++. Całą logikę i połączenia peryferiów z panelem dotykowym programista musi napisać sam w języku programistycznym C/C++ [9].

Rys.2.5.Oprogramowanie TouchGFX
Źródło:https://g6h4c7w3.stackpathcdn.com/wp-content/uploads/2018/12/Screen-Shot-2018-12-19-at-11.26.09-AM.png

2.5.Zastosowania paneli HMI

Panele HMI są używane w fabrykach wielu sektorach przemysłu. Mogą być używane jako terminale przy pojedynczej maszynie lub być zastosowane do znacznie większych aplikacji, wraz z sterownikami PLC są kręgosłupem dzisiejszych linii produkcyjnych. W przemyśle często są używane do kontroli procesu przemysłowego, łącząc maszyny, czujniki i urządzenia wykonawcze z całej hali produkcyjnej. Panele HMI mogą przesyłać dane do chmur obliczeniowych lub serwerów danej firmy umożliwiając integrację danych z całej fabryki lub całej produkcji firmy w celu bardziej efektywnego i scentralizowanego monitoringu i kontroli nad produkcją. W fabrykach, panele HMI dostarczają cennych danych, umożliwiając operatorom optymalizację wydajności poprzez dostosowanie ustawień maszyny w celu poprawy jej wydajności.

Human-machine interface są używane dla niezliczonych aplikacji w prawie każdej gałęzi gospodarki. Portal branżowy "AutomatykaB2B.pl" przeprowadził ankiete (Rys.2.5) dotyczącą najważniejszych w Polsce grup odbiorców HMI. Rezultaty które otrzymał i zestawił w postaci grafu dają pogląd na temat wykorzystania paneli HMI w Polsce.

Rys.2.5.Ankieta popularności paneli HMI w danych branżach
Źródło https://automatykab2b.pl/i/images/9/8/7/dz03MzAmaD00NTM=_src_155987-46297panele_rys.jpg

Przykłady zastosowań

- **Przemysł i automatyka:**

W fabrykach stali (Rys.2.6), panele HMI mogą kontrolować jak metal jest cięty i zginany, oraz jak szybko ma się to odbywać. HMI pozwala na efektywniejszą kontrolę nad używanym materiałem do produkcji, co pozwala zmniejszyć straty jak i lepiej zaplanować dostawy .

Rys.2.6.Zdjęcie z fabryki stali
Źródło:https://ik.imagekit.io/horizons/pannam/wp-content/uploads/2017/03/stockfresh_6856180_worker-in-manufacturing-plant-at-machine-control-panel_sizeXS.jpg

- **Pojazdy i transport:**

Już od dawna producenci samochodów wyposażają droższe modele w komputery pokładowe (Rys.2.7), zintegrowane z panelami HMI, które pozwalają monitorować wszystkie dane dotyczące samochodu oraz ich intuicyjną zmianę podczas jazdy.

Rys.2.7.Panel HMI umieszczony w samochodzie
Źródło:https://st3.depositphotos.com/11452492/14586/v/600/depositphotos_145866063-stock-video-the-interior-of-a-tesla.jpg

3.Mikrokontrolery STM32

W rozdziale przedstawiono podstawowe informacje dotyczące mikrokontrolerów serii STM32. Zawiera opis najważniejszych rodzin mikrokontrolerów produkowanych przez firmę STMicroelectronics, zestawów uruchomieniowych, środowisk programistycznych, bibliotek, wraz z określeniem obszarów ich zastosowań.

3.1.Mikrokontrolery z rdzeniem ARM

Procesor (CPU, ang. *central processing unit*) – "*sekwencyjne urządzenie cyfrowe, które pobiera dane z pamięci operacyjnej, interpretuje je i wykonuje jako rozkazy. Procesory wykonują ciągi prostych operacji matematyczno-logicznych ze zbioru operacji podstawowych.*

Procesory wykonywane są zwykle jako układy scalone zamknięte w hermetycznej obudowie. Sercem procesora jest monokryształ krzemu, na który naniesiono techniką fotolitografii szereg warstw półprzewodnikowych, tworzących, w zależności od zastosowania, sieć od kilku tysięcy do kilku miliardów tranzystorów. Jego obwody wykonywane są z metali o dobrym przewodnictwie elektrycznym, takich jak aluminium czy miedź.

Jedną z podstawowych cech procesora jest określona długość (liczba bitów) słowa, na którym wykonuje on podstawowe operacje obliczeniowe. Jeśli przykładowo słowo tworzą 32 bity, to taki procesor określany jest jako 32-bitowy. Innym ważnym parametrem określającym procesor jest szybkość, z jaką wykonuje on rozkazy. Przy danej architekturze procesora, szybkość ta w znacznym stopniu zależy od czasu trwania pojedynczego taktu, a więc głównie od częstotliwości jego taktowania."[10]

Mikrokontroler (MCU, ang. *microcontroller unit*) – "*scalony system mikroprocesorowy, zrealizowany w postaci pojedynczego układu scalonego zawierającego jednostkę centralną (procesor, CPU), pamięć RAM (ang. random-access memory) oraz rozbudowane układy wejścia-wyjścia i na ogół pamięć programu jako FRAM (ang. ferroelectric random-access memory), MRAM (ang. Magnetoresistive Random Access Memory), ROM (ang. read-only memory) lub pamięci FLASH (ang. flash memory). Mikrokontrolery są zaprojektowane głównie z myślą o aplikacjach systemów wbudowanych (ang. embedded system), w przeciwieństwie do mikroprocesorów używanych w komputerach osobistych lub innych*

aplikacjach ogólnego zastosowania. Stanowią użyteczny i całkowicie autonomiczny system mikroprocesorowy, nie wymagający użycia dodatkowych elementów. Mikrokontrolery wykorzystuje się powszechnie w sprzęcie AGD i RTV, układach kontrolno-pomiarowych, w przemysłowych układach automatyki, w telekomunikacji, podzespołach i urządzeniach podłączanych do komputerów."[11]

Poza procesorem w każdym mikrokontrolerze można wyróżnić następujące bloki funkcyjne:

- jednostka obliczeniowa ALU (ang. *Arithmetic Logic Unit*)
- pamięć danych RAM
- pamięć programu FRAM, MRAM, ROM lub Flash
- uniwersalne porty wejścia-wyjścia
- układy czasowo-licznikowe
- kontrolery przerwań

Dodatkowo wybrane modele mogą zawierać:

- kontrolery transmisji szeregowej:
  - UART (ang. *universal asynchronous receiver-transmitter*)
  - SPI (ang. *Serial Peripheral Interface*)
  - I2C (ang. *Inter-Integrated Circuit*)
  - USB (ang.*Universal Serial Bus*)
  - CAN (ang. *Controller Area Network*)
- przetworniki analogowo-cyfrowe, cyfrowo-analogowe
- obszar nieulotnej pamięci danych EEPROM (ang. *electrically erasable programmable read-only memory*)
- zegar czasu rzeczywistego RTC (ang. *Real-Time Cloc*k)
- układ kontroli poprawnej pracy (ang. *watchdog*)
- wewnętrzne czujniki wielkości nieelektrycznych, np.czujnik temperatury

"Jako języki programowania mikrokontrolerów najczęściej wykorzystywane są asemblery, język C, oraz uproszczone dialekty języka BASIC. Asembler pozwala lepiej wykorzystać możliwości mikrokontrolera (w tym również ograniczone zasoby pamięci), jednak z reguły wymaga większego nakładu pracy programisty."[11]

Mikroprocesor – "*układ cyfrowy wykonany jako pojedynczy układ scalony o wielkim stopniu integracji zdolny do wykonywania operacji cyfrowych według dostarczonego ciągu instrukcji.*"[12] Mikroprocesor łączy funkcje centralnej jednostki obliczeniowej (procesora, CPU) w pojedynczym półprzewodnikowym układzie scalonym. Mikroprocesor komunikuje się z otoczeniem za pomocą szyny danych i szyny adresowej.

"*W każdym mikroprocesorze można wyróżnić następujące bloki funkcyjne:*

- *jednostka arytmetyczno logiczna ALU*
- *układ sterowania (CU, ang. control unit), zwany też dekoderem rozkazów*
- *Rejestry, umieszczone wewnątrz mikroprocesora komórki pamięci o niewielkich rozmiarach służące do przechowywania tymczasowych wyników obliczeń* "[12]

Przedstawienie firm STMicroelectronics oraz ARM Holding:

Na wstępie należy zaznaczyć że firma STMicroelectronics wytwarza swoje produkty w oparciu o architekturę procesorów firmy ARM Holdings. Na rysunku 3.1 przedstawiam logo firmy ARM Holdings.

Rys.3.1.Logo firmy ARM Holding
Źródło:https://upload.wikimedia.org/wikipedia/commons/thumb/7/77/Arm_logo_2017.svg/330px-Arm_logo_2017.svg.png

ARM Holding jest globalną firmą zajmująca się głównie rynkiem układów scalonych oraz wytwarzaniem oprogramowania. Ich główne produkty to licencje architektury swoich własnych procesorów oraz narzędzia programistyczne takie jak: DS-5, RealView, Keil. Co wyróżnia tą firmę od innych to, że nie zajmują się produkcją swoich produktów, lecz sprzedają licencję na ich produkcję innym firmom. Procesory oparte na architekturze firmy ARM są wykorzystywane w Systemach wbudowanych, Systemach biometrycznych, smart TVs, Smartwatchach, Smartfonach, Tabletach, Laptopach. W 2018 roku procesory firmy ARM stanowiły 33% rynku światowego i wyprodukowano 22,9 biliony procesorów architektury ARM [13].

Rys.3.2.Logo firmy STMicroelectronics
Źródło:https://upload.wikimedia.org/wikipedia/en/thumb/f/f3/STMicroelectronics_logo.svg/225px-STMicroelectronics_logo.svg.png

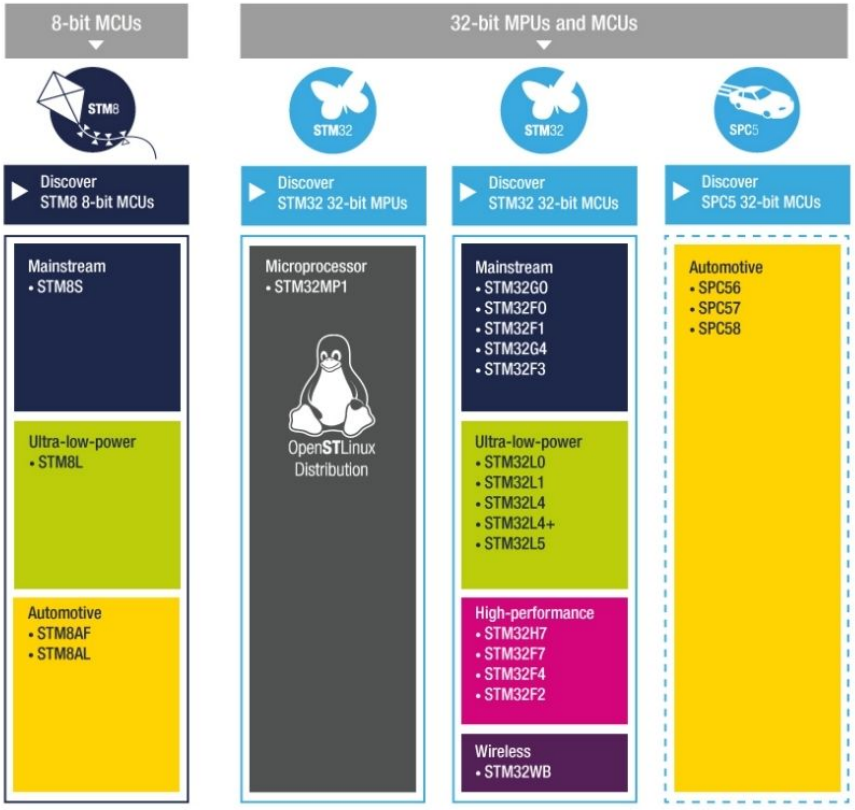
STMicroelectronics (Rys.3.2) jest globalną spółką z kapitałem francusko-włoskim działająca na rynku elementów elektronicznych i układów scalonych. Jest największym europejskim sprzedawcą układów scalonych. Jak już wcześniej wspomniałem produkuje ona w swoich fabrykach mikroprocesory na licencji firmy AMR. Produkują oni mikrokontrolery z rodziny STM32 [14].

3.2.Mikrokontrolery firmy STMicroelectronics

W tym rozdziale przybliżę aktualną ofertę firmy STMicroelectronics (Rys.3.3). Szeroki wachlarz produktów firmy oraz mnogość ich specyfikacji nie pozwalają w krótki i zwięzły sposób ich przedstawić. Dlatego też ograniczę się do podania najważniejszych, charakterystycznych cech głównych gałęzi produktów. Szczegółowe dane dla poszczególnego produktu można znaleźć na głównej stronie producenta. Architektura 8-bitowa oznacza, że słowa, dane, adresy mieszczą się na najwyżej 8 bitach. Analogicznie dla architektury 32 bitowej.

W ofercie producenta można wyróżnić cztery główne gałęzie produktów.

1. STM8 MCUs - Mikrokontrolery o architekturze 8-bitowej
2. STM32 MPUs - Mikroprocesory o architekturze 32-bitowej
3. STM32 MCUs - Mikrokontrolery o architekturze 32-bitowej
4. SPC5 - Mikrokontrolery o architekturze 32-bitowej, dla aplikacji automotive



Rys.3.3.Drzewo produktów firmy STMicroelectronics
Źródło:https://www.st.com/content/ccc/fragment/product_related/family_information/family_diagram/42/f8/83/2d/4a/8e/46/1d/microcontrollers_FM141.jpg/files/microcontrollers_FM141.jpg/_jcr_content/translations/en.microcontrollers_FM141.jpg

3.2.1.Mikrokontrolery o architekturze 8-bitowej

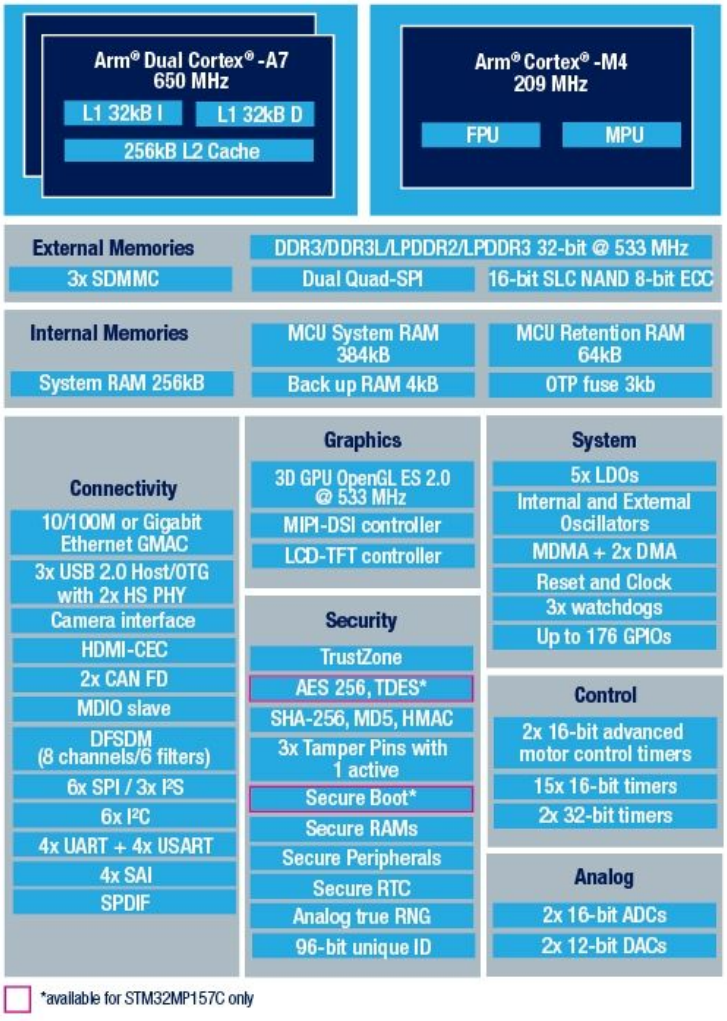
W tej gałęzi został zaimplementowany wydajny 8-bitowy procesor taktowany przebiegiem o częstotliwości do 24 MHz oraz najnowocześniejsze bloki peryferyjne. Pamięć Flash wykonana została w procesie technologicznym 130 nm [15]. W tabeli 3.1 przedstawiam szczegółowo tę rodzinę.

Tab.3.1.Rodzina 8-bitowych mikrokontrolerów

STM8S Mainstream	STM8L Ultra-Low-power	STM8A Automotive
Mikrokontrolery ogólnego przeznaczenia. Zasilane napięciem od 2.95V do 5.5V. Są to najtańsze mikrokontrolery generalnego przeznaczenia w ofercie producenta.	Mikrokontrolery o obniżonym poborze prądu, przeznaczone zwłaszcza dla urządzeń przenośnych. Zasilane napięciem od 1.8V do 3.6V. Najniższy pobór prądu wynosi 0.30µA. Dynamiczny pobór mocy wynosi 150µA/MHz	Mikrokontrolery przeznaczone dla branży automotive, w których kluczową rolę stanowi niezawodność działania. Zwiększenie temperatury pracy 150ºC
Flash memory: 4-128 Kbytes	Flash memory: 2-64 Kbytes	Flash memory: 4-128 Kbytes
RAM: 1-6 Kbytes	RAM: 1-6 Kbytes	RAM: 1-6 Kbytes
Date EEPROM: 128-2048 bytes	Date EEPROM: 128-2048 bytes	Date EEPROM: 640-2048 bytes
USART, SPI, I2C	USART, SPI, I2C	USART, SPI, I2C
10-bit ADC	12-bit ADC, 12-bit DAC	10-bit ADC
8-bit timers, 16-bit timers	8-bit timers, 16-bit timers	8-bit timers, 16-bit timers
16 MHz crystal oscillator, 128 kHz internal RC oscillator	RTC with 32 kHz oscillator	16 MHz crystal oscillator, 128 kHz internal RC oscillator
SWIM debug module	SWIM debug module	SWIM debug module
Dodatkowo poszczególne produkty mogą zawierać: CAN 2.0 B, 2nd UART, Additional analog channels, LNB firmware	Dodatkowo poszczególne produkty mogą zawierać: Comparators, Temperature sensor, Four DMA channels, LCD interface, AES 128-bit Crypto	Dodatkowo poszczególne produkty mogą zawierać: CAN 2.0B, LIN 2.1, Additional analog channels, 12-bit ADC, 12-bit CDA, AES 128-bit Crypto

3.2.2.Mikroprocesory o architekturze 32-bitowej

Mikroprocesory tej serii wyposażone są w dwa rodzaje procesorów: Arm Cortex-A7 oraz ko-procesor Arm Cortex-M4. Również z opcjonalnym procesorem graficznym GPU (ang. *graphics processing unit*). Cortex-A7 jest jednym z najwydajniejszych procesorów firmy Arm, który wspierają i pozwalają na zainstalowanie systemów operacyjnych, takich jak np. Linux. Cortex-M4 są to procesory do ogólnego przeznaczenia, charakteryzujące się dobrym stosunkiem ceny do osiągów. Głównie stosowane do projektowania systemów wbudowanych (embedded system). Opcjonalne GPU pozwala między innymi na stworzenie zaawansowanego wyświetlacza dotykowego, o funkcjonalności panelu HMI [16]. Informacje na temat najbardziej zaawansowanego produktu z tej serii przedstawia grafika z rysunku 3.4.



Rys.3.4. Specyfikacja produktu STM32MP157
Źródło:https://www.st.com/content/ccc/fragment/product_related/rpn_information/product_circuit_diagram/group0/0d/31/b3/fc/07/61/4b/e7/bd_stm32mp157/files/bd_stm32mp157.jpg/_jcr_content/translations/en.bd_stm32mp157.jpg

3.2.3.Mikrokontrolery o architekturze 32-bitowej

Mikrokontrolery z tej rodziny są oparte na procesorach ARM Cortex-M, ich możliwości zależą od rodzaju procesora z rodziny Cortex-M. Cechują się wysokimi osiągami, możliwością zastosowania ich w systemach czasu rzeczywistego (RTOS, ang. *real-time operating system*) oraz systemach wbudowanych, przetwarzaniem sygnału cyfrowego, niskim poborem mocy przy niskim napięciu, oraz rozbudowanymi interfejsami łączności. Przy zachowaniu pełnej integralności i łatwością rozwoju aplikacji. Szeroka gama produktów oraz narzędzi programistycznych czyni tę rodzinę idealnym wyborem dla każdego projektu. Producent wydzielił z rodziny 4 podgrupy [17]. Na rysunku 3.5 przedstawiam poglądową tę rodzinę.

- **STM32WB Wireless**

  Oparte na procesorze Arm Cortex-M4 pracującym z częstotliwością 64 MHz oraz posiadający ko-procesor odpowiedzialny za łączność bezprzewodową Arm Cortex-M0+ (network processor) pracującym z częstotliwością 32 MHz. Mikrokontroler wspiera Bluetooth 5 i IEEE 802.15.4 wireless standards.

- **STM32 Ultra-Low-power**

  Mikrokontrolery o bardzo niskim poborze mocy, który wynosi w zależności od wybranego urządzenia od 0,3µA do 1,2µA.

    - STM32L0 series
        - 32-bit procesor Arm Cortex-M0 (32MHz)
        - Dostępny z pamięcią Flash od 8 do 192 Kbytes
        - Najniższy pobór prądu (+RAM,+RTC): 0.67µA
    - STM32L1 series
        - 32-bit procesor Arm Cortex-M3 (32MHz)
        - Dostępny z pamięcią Flash od 32 do 512 Kbytes
        - Najniższy pobór prądu (+RAM,+RTC): 1.2µA
    - STM32L4 series
        - 32-bit procesor Arm Cortex-M4 (80MHz)
        - Dostępny z pamięcią Flash od 64 Kbytes do 1 Mbyte
        - Najniższy pobór prądu (+RAM,+RTC): 0.34µA
    - STM32L4+ series
        - 32-bit procesor Arm Cortex-M4 (120MHz)
        - Dostępny z pamięcią Flash od 1 do 2 Mbyte
        - Najniższy pobór prądu (+RAM,+RTC): 1µA
    - STM32L5 series
        - 32-bit procesor Arm Cortex-M33 (110MHz)
        - Dostępny z pamięcią Flash od 256 do 512 Kbytes
        - Najniższy pobór prądu (+RAM,+RTC):0.35µA

- **STM32 Mainstream**

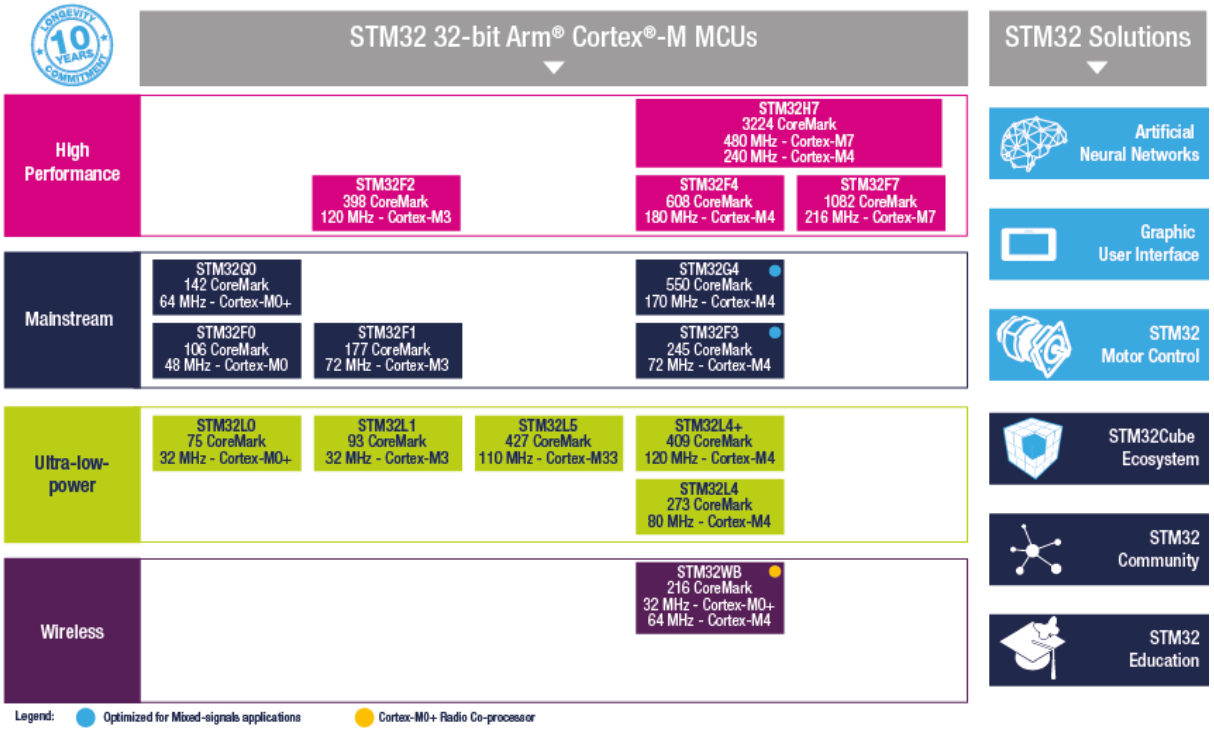
  Mikrokontrolery o najbardziej zoptymalizowanych parametrach, aby można było ich użyć w szerokiej liczbie aplikacji. Rodzaj dostępnych interfejsów komunikacyjnych oraz obsługiwanych peryferiów zależy od indywidualnego produktu.

  - STM32F0 series
    - 32-bit procesor Arm Cortex-M0 (48MHz)
    - Dostępny z pamięcią Flash od 16 do 256 Kbytes
  - STM32G0 series
    - 32-bit procesor Arm Cortex-M0+ (64MHz)
    - Dostępny z pamięcią Flash od 16 do 512 Kbytes
  - STM32F1 series
    - 32-bit procesor Arm Cortex-M3 (72MHz)
    - Dostępny z pamięcią Flash od 16 Kbytes do 1 Mbyte
  - STM32F3 series
    - 32-bit procesor Arm Cortex-M4 (72MHz)
    - FPU (ang. Floating-Point Unit)
    - Dostępny z pamięcią Flash od 16 do 512 Kbytes
  - STM32G4 series
    - 32-bit procesor Arm Cortex-M4 (170MHz)
    - FPU (ang. Floating-Point Unit)
    - Dostępny z pamięcią Flash od 32 do 512 Kbytes

- **STM32 High performance**

  Mikrokontrolery, których głównym założeniem jest maksymalizacja wydajności i osiągnięć. Najlepsza ze wszystkich serii wydajność systemu do wykonywania programu, przesyłania danych oraz ich przetwarzania. Posiadają największą pamięć wbudowaną oraz zaawansowane urządzenia peryferyjne. Odznaczają się wysoką wydajnością energetyczną.

  - STM32F2
    - 32-bit procesor Arm Cortex-M3 (120MHz)
    - Dostępny z pamięcią Flash od 128 Kbytes do 1 Mbyte
  - STM32F4
    - 32-bit procesor Arm Cortex-M4 (180MHz)
    - FPU (ang. Floating-Point Unit)
    - Dostępny z pamięcią Flash od 256 Kbytes do 2 Mbytes
  - STM32F7
    - 32-bit procesor Arm Cortex-M7 (200MHz)
    - FPU (ang. Floating-Point Unit)
    - Dostępny z pamięcią Flash od 256 Kbytes do 2 Mbytes
  - STM32H7
    - 32-bit procesor Arm Cortex-M7 (480MHz)
    - 32-bit ko-procesor Arm Cortex M-4
    - FPU (ang. Floating-Point Unit)
    - Dostępny z pamięcią Flash od 1 do 2 Mbytes



Rys.3.5.Drzewo przedstawiające serie 32-bitowych mikrokontrolerów
Źródło:https://www.st.com/content/ccc/fragment/product_related/class_information/class_level_diagram/group0/c7/8f/0b/3c/5f/f0/43/04/stm32_cl1734/files/stm32_cl1734.png/_jcr_content/translations/en.stm32_cl1734.png

3.2.4.Mikrokontrolery o architekturze 32-bitowej, dla aplikacji automotive

Rodzina 32-bitowych mikrokontrolerów, zaprojektowana dla szerokiego spektrum aplikacji z branży automotive. SPC5 zapewnia do 3 procesorów pracujących z częstotliwością do 200 MHz. Maksymalnej temperatury pracy wynosi 165 stopni Celcjusza. Mikrokontrolery zapewniają niski pobór mocy, dużą ilość interfejsów komunikacyjnych (CAN-FD, Ethernet, LIN, DSPI, FlexRay). Nacisk ze strony producenta został nałożony na bezpieczeństwo, dzięki spełnieniu normy ISO 26262 ASIL-D. Posiada zaawansowane timery, w tym GTM (ang. *Generetic Timer module*), FlexPWM, ulepszony modułowy system wejścia-wyjścia (eMIOS) oraz ulepszony Time Processing Unit (eTPU). Rodzina dzieli się na dwie główne podgrupy SPC5 G (General Purpose) i SPC5 P (Perfotmance) [18].

3.3.Zestawy uruchomieniowe z mikrokontrolerem STM32 i dotykowymi panelami LCD

Zestawy uruchomieniowe (ang. *discovery kit*) są to płytki przygotowane przez producenta tak aby dany mikrokontroler był natychmiast gotowy do użycia, w celu na przykład przetestowania go przez inżynierów lub aby był platformą do nauki. Discovery Kit zawiera mikrokontroler oraz urządzenia peryferyjne (moduły) zależne od danego modelu zestawu (np: diody led, przetworniki cyfrowo-analogowe, mikrofon, akcelerometr, żyroskop, ekran dotykowy). Najważniejszym elementem wyposażenia jest programator/debugger ST-Link, dzięki któremu będziemy mogli wgrać program do mikrokontrolera oraz podgląd wnętrza układu podczas jego pracy. Na płytce są umieszczone listwy typu goldpin, do których doprowadzono część lub w niektórych produktach wszystkie linie mikrokontrolera, dzięki czemu można podłączyć zewnętrzne układy bezpośrednio do mikrokontrolera. Pozwala to na dołączenie dowolnych modułów.

Poniżej przedstawię dwa interesujące zestawy uruchomieniowe firmy STMicroelectronics, dostępne na polskim rynku.

3.3.1. Zestaw uruchomieniowy STM32F746G-Discovery

Jeden z najbardziej rozbudowanych zestawów z serii Discovery bazującym na rdzeniu Cortex-M7, posiadający wszystkie cechy rodziny STMF7. Można go użyć do szerokiej gamy zastosowań, takich jak: aplikacji audio, obsługa wielu czujników, grafika, ochrona (bezpieczeństwo), video, wysokiej jakości funkcje łączności. Warto dodać że procesory z rdzeniem Cortex-M7 są najefektywniejsze z całej rodziny Cortex-M. Posiada ponadto dotykowy ekran [6]. Jego wygląd przedstawia rysunek 3.6.

Figure 1. STM32F746G-DISCO board (top view)

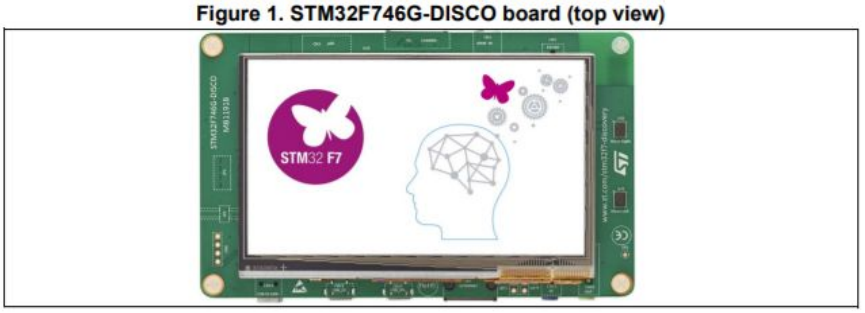
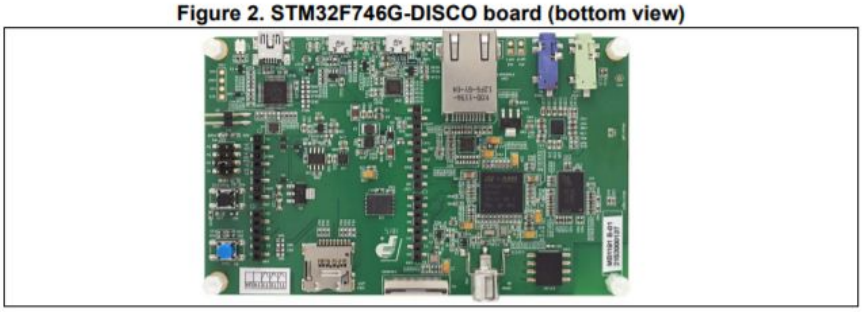


Figure 2. STM32F746G-DISCO board (bottom view)



Rys.3.6. Zestaw uruchomieniowy STM32F746G-DISCOVERY
Źródło:https://www.st.com/bin/ecommerce/api/image.PF261641.en.feature-description-include-personalized-no-cpn-large.jpg

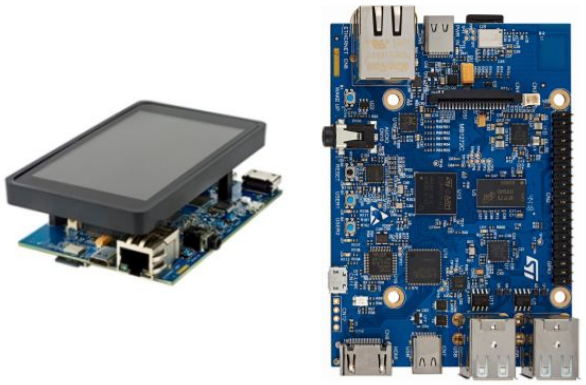
Specyfikacja:

- Mikrokontroler STM32F746NGH6 (1 MB Flash, 340 kB SRAM, 216MHz, Cortex-M7)
- Wyświetlacz 4,3 cala LCD-TFT WQVGA 480x272 pikseli z pojemnościowym touch-panelem
- Kodek audio ze wzmacniaczem mocy
- Dwa mikrofony MEMS
- Ethernet 10/100 Mb/s
- 2 x USB-OTG (FS i HS)
- Złącza zgodne z Arduino (3,3V)
- Złącze MicroSD
- Złącze kamery CCD
- Wejście SPDIF
- Pamięć SDRAM 64 Mb
- Pamięć NOR Flash z QSPI 128 Mb
- Programator-debugger ST-Link/V2-1 (zgodny z mbed.org)

3.3.2. Zestaw uruchomieniowy STM32MP157C-DK2

Zestaw bazuje na serii STM32MP1. Jest to pierwszy zestaw firmy ST z mikroprocesorem o heterogenicznej architekturze, która zawiera dwa procesory z rdzeniem Cortex-A7 oraz koprocesor z rdzeniem Cortex-M4. Dzięki procesorom z rdzeniem Cortex-A7, które wspierają 32-bitowe systemy operacyjne jesteśmy w stanie operować na systemach takich jak Linux, a dzięki procesorowi z rdzeniem Cortex-M4 jesteśmy w stanie zbudować aplikację

RTOS. Seria MP1 jest wyposażona w zintegrowany procesor graficzny. Posiada ponadto dotykowy ekran [19]. Jego wygląd przedstawia rysunek 3.7.



Rys.3.7.Zestaw uruchomieniowy STM32MP157C-DK2
https://www.st.com/bin/ecommerce/api/image.PF267415.en.feature-description-include-personalized-no-cpn-large.jpg

Specyfikacja:

- STM32MP157A oparty na podwójnym rdzeniu Cortex-A7 32 bity oraz Cortex-M4 32 bity MPU w obudowie TFBGA361
- Układ zasilający ST PMIC STPMIC1A
- 4-Gbit DDR3L, 16 bitów, 533 MHz
- 1 Gb/s Ethernet (RGMII) zgodny z IEEE-802.3ab
- USB OTG HS
- Kodek audio
- 4 diody LED użytkownika
- 2 przyciski użytkownika i resetowania, 1 przycisk wake-up
- Wejście zasilania 5 V/3 A USB typu C
- Wyświetlacz TFT 4" 480 × 800 pikseli z podświetleniem LED, interfejsem MIPI DSISM i pojemnościowym panelem dotykowym
- Wi-Fi 802.11b / g / n
- Bluetooth® Low Energy 4.1
- Wbudowany debugger/programator ST-LINK/V2-1 z możliwością wykorzystania jako USB:
- wirtualny port COM i port debugowania
- STM32CubeMP1 i pełne wsparcie open-source Linux STM32 MPU OpenSTLinux Distribution
- Obsługa zintegrowanych środowisk programistycznych (IDE), w tym IAR, Keil, GCC IDE
- Złącza:
    - Ethernet RJ45
    - 4 x Host USB typu A
    - USB Type-C DRP
    - MIPI DSISM
    - HDMI
    - Złącze Jack (gniazdo słuchawkowe z analogowym wejściem mikrofonowym)
    - Gniazdo kart microSD
    - Złącze rozszerzenia GPIO (kompatybilne z nakładkami Raspberry Pi)
    - Złącza rozszerzające Arduino Uno V3

3.4. Środowisko programistyczne

W tym podrozdziale opisuje programy dostarczane przez producenta mikrokontrolerów SMT32. Zestaw programów pozwala w szybki sposób opracowywać aplikacje na mikrokontrolery. Rysunek 3.8 przedstawia pakiet programów dostarczany przez firmę STMicroelectronic.

Rys.3.8.Oprogramowanie dostarczane przez STMicroelectronic
Źródło:https://www.st.com/content/ccc/fragment/product_related/subclass_information/subclass_level_diagram/group0/e5/58/73/a4/2c/da/4b/6c/stm32_software_development_tools_sc2106/files/stm32_software_development_tools_sc2106.jpg/_jcr_content/translations/en.stm32_software_development_tools_sc2106.jpg

3.4.1.Aplikacja STM32CubeMX

To narzędzie graficzne pozwalające bardzo łatwą na konfigurację mikrokontrolerów i mikroprocesorów STM32, oraz generowanie kodu w języku C dla procesora z rdzeniem Cortex-M lub częściowo drzewo urządzeń w systemie Linux dla rdzenia Arm Cortex-A. Poprzez metodyczne przechodzenie krok po kroku przez kolejne okna w programie [20].

3.4.2.Środowisko Athollic TRUEStudio for STM32

Jest zintegrowanym środowiskiem programistycznym dla mikrokontrolerów z serii STM32. Wykorzystującym otwarty framework Eclipse, kompilator GCC, oraz

debuger oparty o standardzie GBD. Jest kompatybilny z programem CubeMX, z którego po wstępnej konfiguracji i wygenerowaniu kodu w płynny sposób zaczyna się programować w tym środowisku [20].

3.4.3.Aplikacja STM32CubeProgrammer

Program zapewnia łatwe w użyciu i wydajne środowisko do odczytu, zapisu i weryfikacji pamięci urządzenia poprzez debug interface(JTAG and SWD) i bootloader interface (UART, USB DFU, I2C, SPI, CAN). Zawiera szeroki zakres funkcji do programowania pamięci wewnętrznej (Flash, RAM, OTP), a także pamięci zewnętrznej [20].

3.4.4.Aplikacja STMStudio

Program pomaga w debagowaniu i analizie działających aplikacji na mikrokontrolerze, poprzez odczytywanie i wyświetlanie w czasie rzeczywistym zmiennych. Program działa na komputerze jako nieinwazyjne narzędzie programistyczne, które nie zakłóca pracy aplikacji i po przez ST-LINK komunikuje się z mikrokontrolerem [20].

3.4.5.Środowisko TouchGFX

Na bardziej szczegółowe opisanie zasługuje oprogramowanie do tworzenia GUI (ang. *Graphical User Interface*). TouchGFX jest zaawansowanym oprogramowaniem stworzonym przez firmę STMicroelectronics do tworzenia grafiki, program jest zoptymalizowany dla mikrokontrolerów STM32. Wykorzystując funkcje graficzne oraz architekturę mikrokontrolerów STM32. TouchGFX pozwala na łatwe i szybkie budowanie interfejsów graficznych używanych do implementacji panelu HMI na mikrokontrolerze połączonym z dotykowym wyświetlaczem LCD (ang. *liquid-crystal display*). Program zawiera łatwe w użyciu narzędzie do budowania grafiki oparte na technologii drag-and-drop (przeciągnij i upuść), oraz silnik graficzny, który optymalizuje pracy procesora graficznego (GPU). TouchGFX ułatwia tworzenie GUI łącząc “WYSIWYG” (What You See Is What You Get) symulator oraz automatyczne generowanie kodu. TouchGFX jest sprzedawany razem w pakiecie programów STM32Cube i jest w pełni kompatybilny z programem STM32CubeMX [9].

3.5.Biblioteki programistyczne

Firma STMicroelectronics dostarcza lub wspiera dla każdej serii mikrokontrolerów i mikroprocesorów spersonalizowany zestaw rozbudowanych bibliotek (STM32Cube MCU and MPU packages) dla łatwego i szybkiego opracowywania aplikacji systemów wbudowanych.

3.5.1.STM32Cube hardware abstraction layer

Biblioteki, które tworzą uniwersalny interfejs programowy umożliwiający obsługę układów peryferyjnych i rdzenia Cortex dla układów firmy ST, wykorzystując do tego celu standaryzowane funkcje, makra i definicje. Funkcje wspierają użycie systemów operacyjnych czasu rzeczywistego, middleware, oraz aplikacji używających interfejsów komunikacyjnych np. I2C, UART, SPI, Ethernet. Ideą stworzenia tych bibliotek była łatwość przenoszenia programów między różnymi platformami sprzętowymi, aby to umożliwić funkcje charakteryzują się pewnym stopniem abstrakcji. To znaczy, że ukrywają przed programistą działania na rejestrach [21].

3.5.2.STM32Cube low layer

Biblioteki, które oferują niskopoziomowy interfejs programowy (API, ang. *application programming interface*) na poziomie rejestrów, są bardzo dobrze zoptymalizowane. Programy są praktycznie nie przenośne między różnymi platformami sprzętowymi, ponieważ są stworzone dla konkretnych platform. W porównaniu do HAL, interfejsy programowe LL nie są dostarczane dla urządzeń peryferyjnych, w których zoptymalizowany dostęp nie jest kluczową cechą, lub wymagają zaawansowanej konfiguracji programistycznej. Biblioteki HAL i LL uzupełniają się nawzajem [21].

3.5.3.STM32Cube middleware

Szeroka gama programów stworzonych przez firmę ST lub wspieranych produktów innych firm w znacznym stopniu upraszczająca programowanie mikrokontrolerów i mikroprocesorów. Są to na przykład:

- File system (STM32Cube - FatTS)

  System plików, który definiuje w jaki sposób są nazywane i w jaki sposób są umieszczane w katalogach.

- RTOS system (FreeRTOS)

  Jest biblioteką służącą do dzielenia czasu między kilkoma zadaniami na jednym rdzeniu. Zapewnia przełączanie zadań w znanym i ograniczonym czasie.
- USB (STM32Cube - Host & Device)

  Biblioteki pozwalające na wymianę danych portem szeregowym USB.
- TCP/IP (STM32Cube - LWwP)

  Biblioteka pozwalająca na wymianę danych poprzez protokoły TCP/IP.
- Display library (STM32Cube - STemWin i STM32Cube - TouchGFX)

  Biblioteki pozwalające obsługiwać wyświetlacze poprzez interfejsy komunikacyjne szeregowe lub równoległe.

3.6.Zastosowania mikrokontrolerów

W dzisiejszym świecie mikrokontrolery możemy spotkać praktycznie w każdym elektrycznym urządzeniu. Bardzo dobry podgląd zastosowań mikrokontrolerów pokazuje przetłumaczona przeze mnie grafika promocyjna firmy STMicroelectronics (Rys.3.9).

Rys.3.9.Zastosowania mikrokontrolerów
Źródło:Materiał własny

4.Projekt panelu HMI

Rozdział zawiera opis projektu panelu HMI obejmujący określenie założeń projektowych, dobór elementów składowych, schemat blokowy przedstawiający działanie panelu HMI, oraz projekt interfejsu graficznego.

4.1.Założenia projektowe

W celu realizacji panelu HMI przyjąłem następujące założenia:

- projekt oparty jest na mikrokontrolerze z serii STM32 firmy STMicroelectronics,
- rozmiar panelu HMI powinien wynosić między 4 a 6 cali,
- ekran panelu HMI powinien być dotykowy,
- mikrokontroler powinien posiadać liczne interfejsy do komunikacji przewodowej i bezprzewodowej,
- mikrokontroler powinien posiadać peryferia, które wspomagają obliczenia graficzne,
- mikrokontroler i panel dotykowy powinny być wspierane przez oprogramowanie TouchGFX,
- Obiektem, który steruje panel HMI jest robot mobilny z niezależnym napędem czterech kół.

4.2.Realizacja sprzętowa

W podrozdziale przedstawiam elementy składowe panelu HMI oraz opisuje obiekt sterowania.

4.2.1.Zestaw uruchomieniowy STM32F746G-DISCOVERY

Zestaw uruchomieniowy został wstępnie przedstawiony w rozdziale 3.4.1 . Wybrałem dany zestaw do mojej pracy z uwagi na spełnienie wszystkich założeń projektowych.

- oparty na mikrokontrolerze STM32F746NG,
- posiada dotykowy ekran 4.3’’ RGB 480x272 LCD-TFT,
- posiada Chrom-ART Accelerator, który wspiera grafikę 2D,
- posiada wszystkie podstawowe interfejsy komunikacyjne,
- zestaw uruchomieniowy jest wspierany przez program TouchGFX.

4.2.2.Moduł radiowy ZigBee Core2530

Twórca robota wyposażył go w moduł radiowy ZigBee do komunikacji radiowej [22]. Z tego powodu postanowiłem do komunikacji bezprzewodowej wykorzystać już gotowe rozwiązanie i też zaimplementować moduł radiowy ZigBee (Rys. 4.1) do panelu HMI. Moduł ZigBee został użyty z uwagi na założenie projektowe, mówiące że robot będzie poruszał na odległości rzędu kilkunastu metrów w pomieszczeniach zamkniętych. Protokoł ZigBee, cechuje się dużą opornością na zakłócenia, kosztem szybkości transferu, co pozwala mu spełnić powyższe założenie. Jeden moduł ZigBee połączono interfejsem UART z panelem HMI, drugi połączono interfejsem UART ze sterownikiem robota mobilnego.

Rys.4.1.Moduł ZigBee C2530
Źródło:https://kamami.pl/54996-thickbox_default/cc2530-eval-kit-zestaw-ewaluacyjny-zigbee.jpg

W tabeli 4.3 przedstawiłem sposób połączenia (z oznaczeniami wyprowadzeń) zestawu uruchomieniowego STM32F746G-DISCOVERY na którym realizuję panel HMI z modułem radiowym ZigBee Core 2530. Analogicznie w tabeli 4.2 przedstawiłem sposób połączenia zestawu uruchomieniowego STM32F4DISCOVERY będącym sterownikiem robota z modułem radiowym ZigBee Core 2530.

Tab.4.2. Połączenie STM32F4DISCOVERY i Core 2530

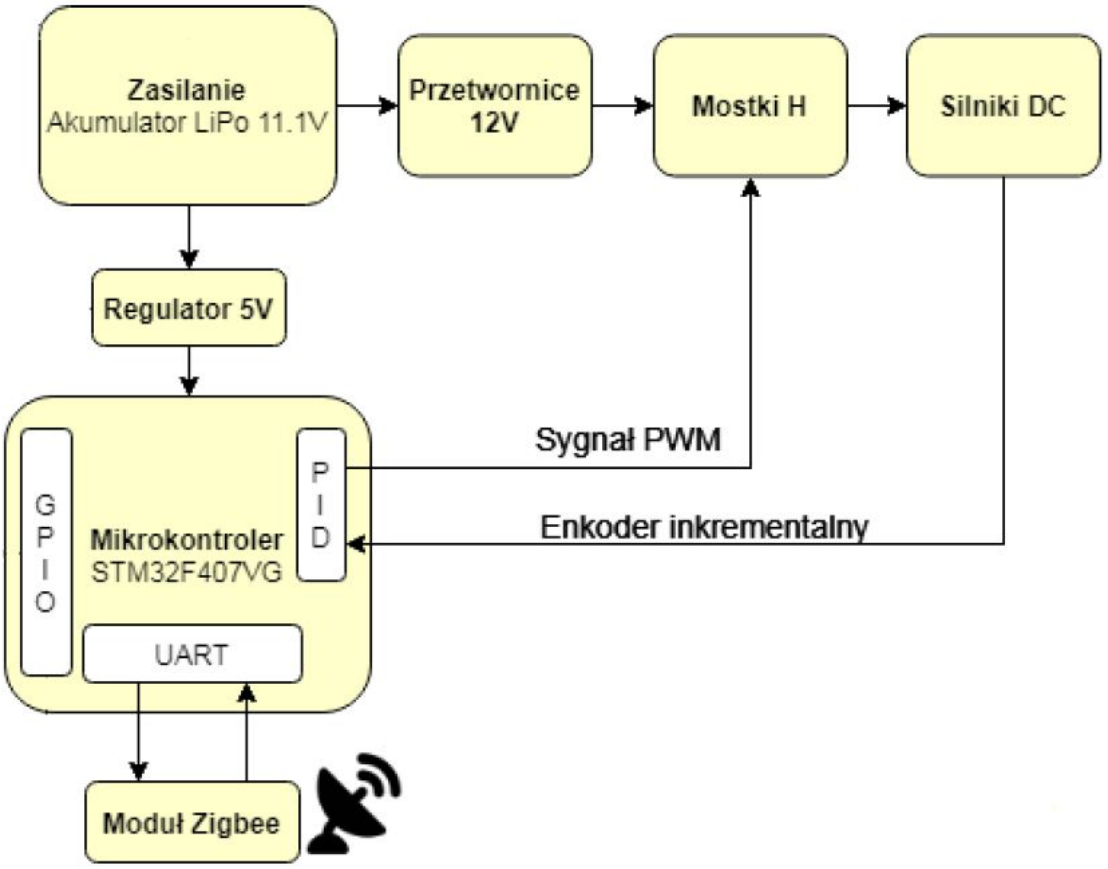
GND	3V	PA2	PA3
GND	VDD	P0_2	P0_3

Tab.4.3. Połączenie STM32F746G-DISCOVERY i Core 2530

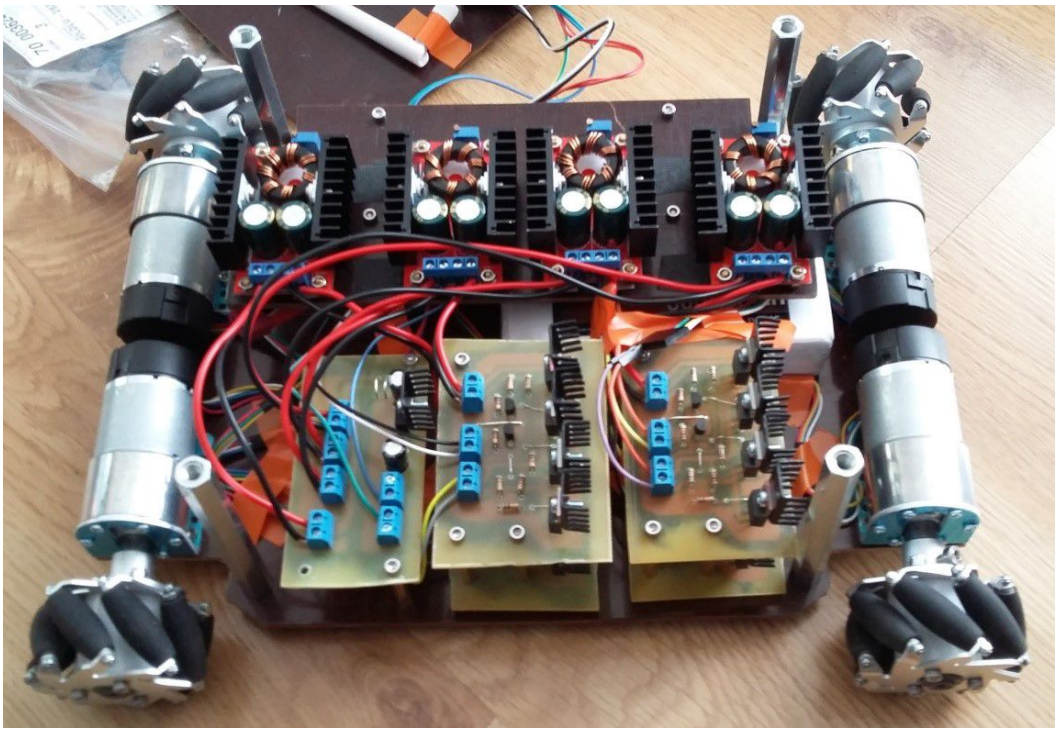
GND	3V	PC7	PC6
GND	VDD	P0_2	P0_3

4.2.3.Robot mobilny z niezależnym napędem czterech kół

Obiektem, który wybrałem jest robot mobilny (Rys.4.3) opisany w pracy inżynierskiej [22] autorstwa Bartłomieja Piwowarczyka. Robot mobilny dzięki czterem kołom typu mecanum ma zapewnioną możliwość wielokierunkowego ruchu po powierzchni płaskiej. Dzięki takiemu rozwiązaniu, sterujemy ruchem robota poprzez zmianę kierunku i prędkości obrotowej kół. Autorski układ elektroniczny zawierający mostki H, oraz enkodery wbudowane w silniki pozwala kontrolować prędkość i kierunek poszczególnych silników. Sterownik robota zbudowany jest na zestawie uruchomieniowym STM32F4DISCOVERY. Mikrokontroler steruje silnikami za pomocą sygnałów PWN podawanym na wejścia sterujące mostków H. Komunikuje się przez UART z układem radiowym ZigBee, który umożliwia bezprzewodowy przesył komend sterujących [22]. Schemat blokowy działania robota przedstawia rysunek 4.2.



Rys.4.2.Schemat robota mobilnego
Źródło:[23]



Rys.4.3.Zdjęcie robota mobilnego
Źródło: [22]

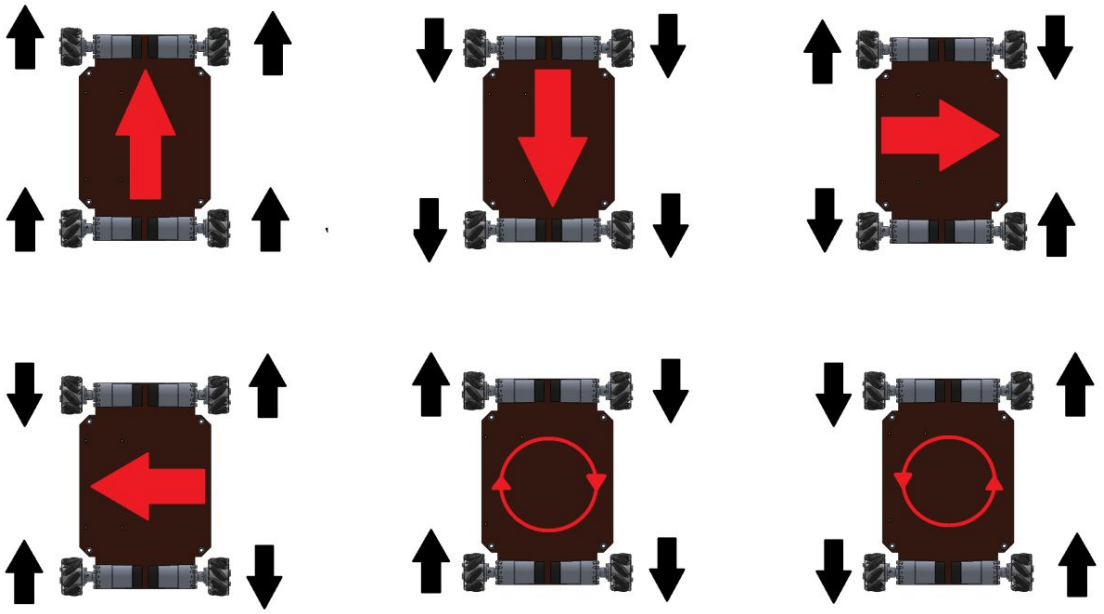
Ponadto robot ma wgrane oprogramowanie napisane przez Nikodema Kastelika, w ramach jego pracy dyplomowej pt. “Implementacja algorytmów sterowania robotem mobilnym z niezależnym napędem czterech kół”. Zaimplementował on między innymi cyfrowy algorytm PID do układu sterowania silnikami poszczególnych kół robota. Szczegółowy opis tego oprogramowania znajduje się w pracy [22]. Tutaj krótko opisze tylko komendy sterujące, które wykorzystuje do komunikacji panelu HMI z robotem (Tab.4.1). Komenda jest tablicą znaków wysłaną interfejsem UART do mikrokontrolera sterującego robotem.

Tab.4.1.Lista obsługiwanych komend przez robota mobilnego

Komenda	Interpretacja
“sp=FR=FL=RR=RL\n” przykłady: jazda do przodu sp=1000=1000=1000=1000\n obrót w lewo sp=500=-500=500=-500\n	Ustawianie prędkości każdego z kół robota w jednej komendzie, gdzie FR=Front Right, FL=Front Left, RR=Right Rear, LR=Left Rear. Prędkość podajemy w impulsach enkodera na sekundę, a kierunek definiujemy znakiem ‘-’, bądź jego brakiem.
“setpoint_**KOŁO**=**WARTOŚĆ**\n” koło przednie prawe jazda do przodu setpoint_FR=1000\n	Ustawienie prędkości wybranego koła. Zamiast **KOŁO**: wpisujemy FR,FL,RR,LR. Zamiast **WARTOŚĆ**: wartość prędkości w impulsach enkodera na sekundę

“pidkp_**KOŁO=WARTOŚĆ**\n”	Ustawienie wartości parametru Kp regulatora PID dla wybranego koła. Zamiast **KOŁO**: wpisujemy FR,FL,RR,LR. Zamiast **WARTOŚĆ**: wartość parametru Kp pomnożoną przez 100.
“pidki_**KOŁO=WARTOŚĆ**\n”	Ustawienie wartości parametru Ki regulatora PID dla wybranego koła. Zamiast **KOŁO**: wpisujemy FR,FL,RR,LR. Zamiast **WARTOŚĆ**: wartość parametru Ki pomnożoną przez 100.
“pidkd_**KOŁO=WARTOŚĆ**\n”	Ustawienie wartości parametru Kd regulatora PID dla wybranego koła. Zamiast **KOŁO**: wpisujemy FR,FL,RR,LR. Zamiast **WARTOŚĆ**: wartość parametru Kd pomnożoną przez 100.

Rysunek 4.4 przedstawia możliwe rodzaje ruchu robota, które zaimplementowałem na panelu HMI.

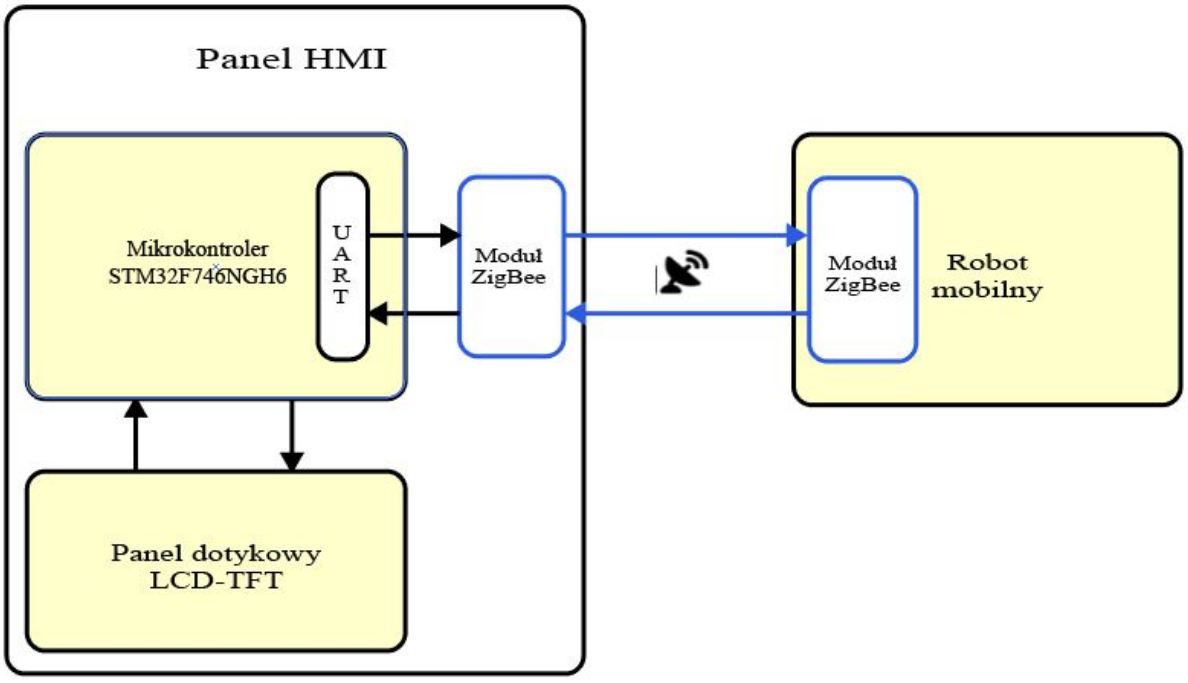


Rys.4.4.Opis możliwości ruchu robota mobilnego
Źródło: [22]

4.3.Schemat blokowy

Na rysunku 4.5 przedstawiony został schemat blokowy pokazujący sposób łączenia się projektowanego panelu HMI z robotem mobilnym. Projektowany panel HMI został zrealizowany na zestawie uruchomieniowym STM32F746G-DISCOVERY. Do komunikacji panelu HMI z robotem wykorzystano moduł radiowy w standardzie ZigBee.

Poprzez interakcje człowieka z interfejsem graficznym na panelu dotykowym mikrokontroler wysyła odpowiednią komendę poprzez UART do modułu ZigBee panelu HMI. Drugi moduł ZigBee znajdujący się w robocie mobilnym odbiera informacje przesłane drogą radiową i przesyła komendę po przez UART do mikrokontrolera znajdującego się w sterowniku robota. Reaguje on na przesłaną komendę poprzez ustawienie prędkości poszczególnych silników robota. Program wgrany do robota przetwarza komendę i wykonuje odpowiednią akcję. Sterownik robota cyklicznie wysyła informację zwrotną do panelu HMI o prędkości i kierunku obrotu silników. Informacja zwrotna w postaci ciągów znakowych jest przetwarzana w mikrokontrolerze panelu HMI i wyświetlana na panelu dotykowym. Ponadto na panelu dotykowym wyświetlana jest informacja czy robot jest połączony z panelem HMI czy nie.



Rys.4.5.Schemat blokowy komunikacji panelu HMI z robotem
Źródło:Materiał własny

4.4.Interfejs graficzny dla panelu HMI

Obecnie nie trzeba zaawansowanych umiejętności programistycznych aby tworzyć interfejsy graficzne. Producenci udostępniają programy generujące kod, które pozwalają użytkownikowi ograniczyć się do stworzenia grafiki. Takim programem jest TouchGFX firmy STMicroelectronics, który generuje kod w C++.

Model generowania kodu opiera się na głównych trzech klasach (Rys.4.6).

- Model

  Szkielet na którym opiera się całe GUI. Zawiera wszystkie najważniejsze informacje dla interfejsu graficznego. Służy jako łącznik między poszczególnymi ekranami oraz jako łącznik między peryferiami (w moim przypadku robotem mobilnym) a dotykowym interfejsem graficznym. Zawiera wskaźnik na adres w komórce pamięci do aktualnie działającego ekranu.

- Presenter

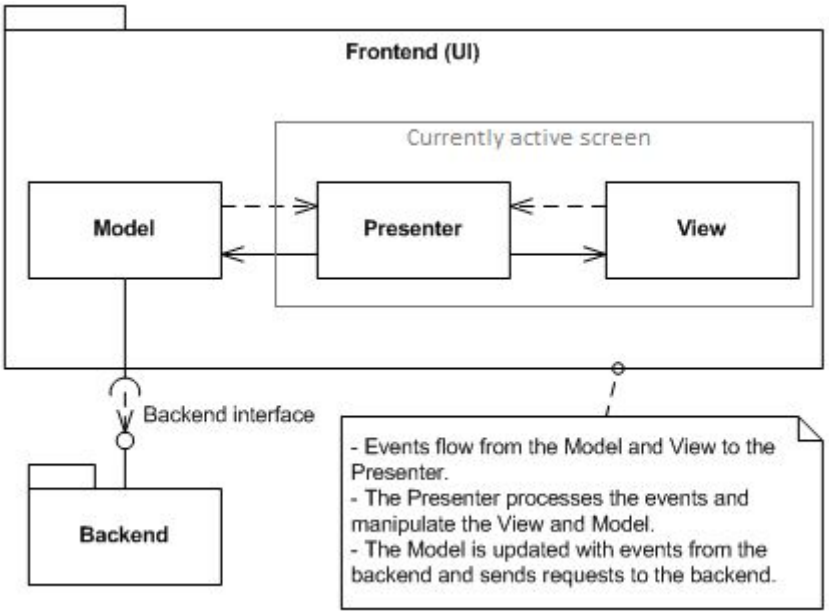
  Ta klasa jest odpowiedzialna za całą logikę dla pojedynczego ekranu. Otrzymuje informacje z klasy View dotyczących interakcji użytkownika z ekranem oraz inforamcje z klasy Model dotyczących peryferiów.

- View

  Zawiera cały kod dotyczący elementów graficznych oraz ich zachowania. Zawiera wskaźnik na adres odnoszący się do klasy Presenter, dzięki czemu może wysyłać informacje takie jak np. wciśnięcie przycisku na ekranie dotykowym. Zawiera dwie ważne funkcje:

  setupScreen, która jest wywoływana automatycznie po włączeniu ekranu, oraz

  tearDownScreen, która jest wywoływana automatycznie po wyłączeniu ekranu.

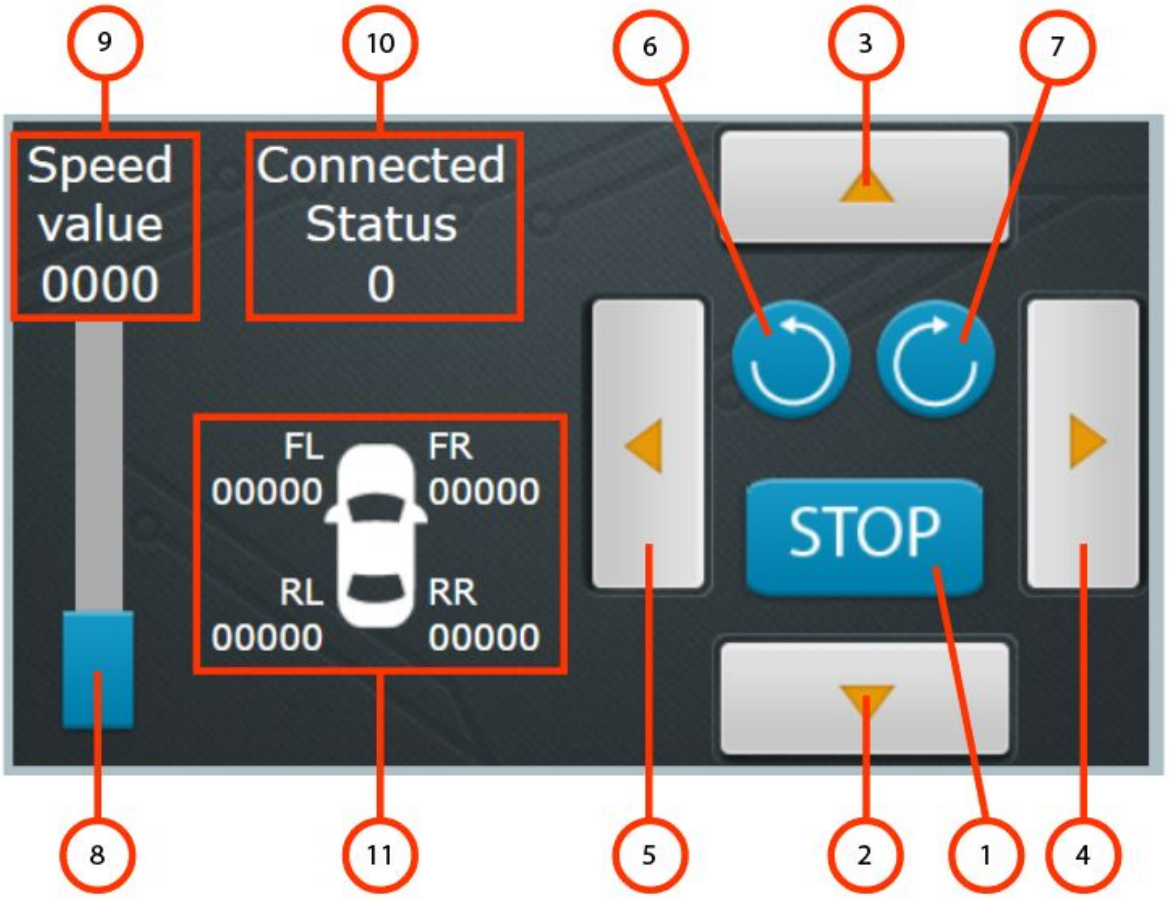


Rys.4.6.Model działania systemu: Model/Presenter/View
Źródło:https://touchgfx.zendesk.com/hc/en-us/article_attachments/202241562/mvp.png

Rozmieszczenie i funkcjonalność ikon graficznych:

Na rysunku 4.7 przedstawiony jest zaprojektowany przeze mnie interfejs graficzny. Spełnia następujące funkcje:

- Sterowanie kierunkiem ruchu (1,2,3,4,5,6,7)
- Sterowanie prędkością (8,9)
- Wyświetlanie statusu połączenia z robotem (10)
- Wyświetlanie informacji zwrotnej o prędkości kół od robota (11)



Rys.4.7.Opis funkcji na panelu dotykowym
Źródło: Materiał własny

Poszczególne ikony, pola oznaczają:

1. Zatrzymanie się robota
2. Jazda do tyłu
3. Jazda do przodu
4. Jazda w prawo
5. Jazda w lewo
6. Obrót w lewo
7. Obrót w prawo
8. Suwak służący do ustawiania prędkości
9. Wyświetlenie ustawionej wartości prędkości przez suwak
10. Wyświetlenia statusu połączenia z robotem “ON” lub “OFF”
11. Aktualna prędkość poszczególnych kół (informacja zwrotna z enkoderów

5.Oprogramowanie panelu HMI

Oprogramowanie dla mojego projektu napisałem z korzystając z dwóch pozycji literatury technicznej. Aby nauczyć się programowania w języku C sięgnąłem do “The C Programming Language” [24]. W celu zgłębienia wiedzy na temat mikrokontrolerów i ich programowania sięgnąłem do książki “Mikrokontrolery STM32 dla początkujących”[25].

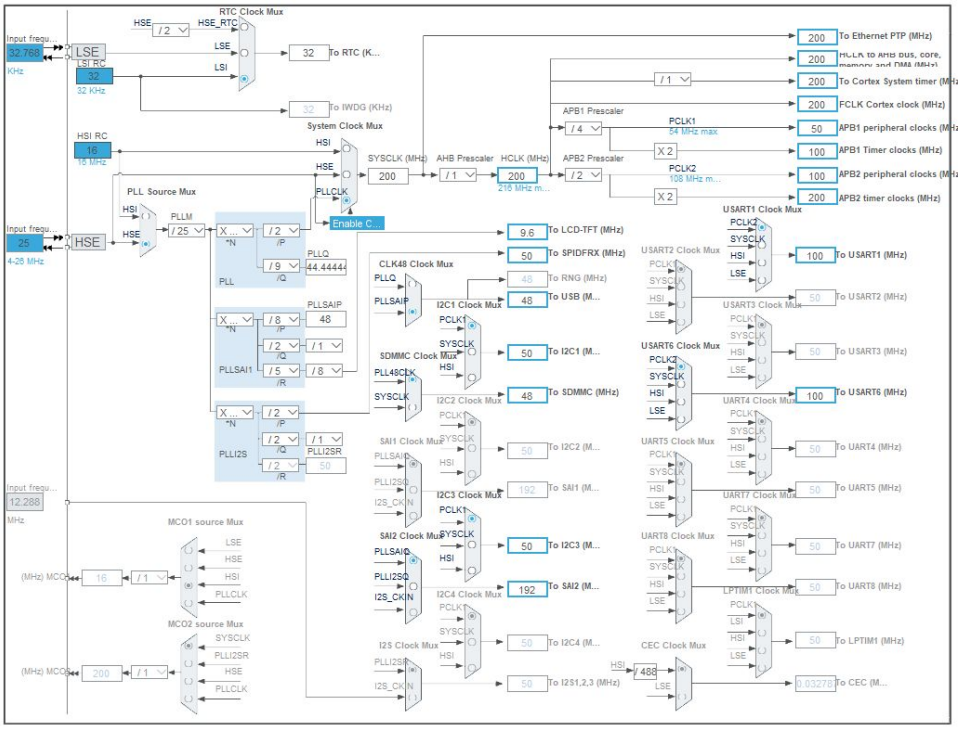
5.1.Inicjalizacja mikrokontrolera

Inicjalizacje peryferiów i dostępnego oprogramowania wykonujemy w programie STM32CubeMX. Po wyszukaniu zestawu uruchomieniowego SMT32F746G-DISCOVERY większość z ustawień (Rys.5.1) jest od razu dodana do inicjalizacji przy generowaniu kodu. Z mojej strony musiałem zaznaczyć dodatkowo w Middleware opcje GRAPHICS i w Application GFXSIMULATOR, aby zainicjować możliwość używania programu TouchGFX. W moim projekcie skorzystałem z:

- CORTEX_M7 - biblioteki dla procesora z Corem M7
- GPIO - aktywacja pinów arduino na płytce
- NVIC - aktywacja przerwań
- RCC - konfiguracja zegarów dla poszczególnych peryferiów
- RTC - aktywacja zegara czasu rzeczywistego
- TIM6 - baza dla czasu systemowego SYS
- QUADSPI - podłączenie dodatkowej pamięci FLASH (do 128Mb)
- USART1 - interfejs do komunikacji z komputerem podczas testów
- USART6 - interfejs do komunikacji z robotem mobilnym
- LTDC - aktywacja panelu dotykowego TFT-LCD
- FREERTOS - biblioteki systemu czasu rzeczywistego
- GRAPHICS i GFXSIMULATOR - aktywuje możliwość korzystania z programu TouchGFX

Rys.5.1.Ekran Inicjalizacji wstępnej CubeMX
Źródło: Materiał własny(zdjęcie ekranu oprogramowania CubeMX)

Przechodząc do zakładki szczegółowej konfiguracji częstotliwości pracy poszczególnych peryferiów (Rys.5.2) nic nie zostało zmienione przeze mnie. Zostawiłem ustawienia domyślne.



Rys.5.2.Ekran ustawienia zegarów CubeMX
Źródło:Materiał własny(zdjęcie ekranu oprogramowania CubeMX)

Po tych dwóch łatwych krokach miałem gotowy projekt z wygenerowanym kodem inicjalizującym i dodane wszystkie potrzebne biblioteki.

5.2.System czasu rzeczywistego

Przy opisie kodów źródłowych programu przyjmuje notacje komentowania poleceń umieszczając komentarz po prawej stronie lub nad komendą. Komentarz zaczyna się od '//' i jest pisany pogrubioną kursywą, zielonym kolorem.

Aby umożliwić jednoczesną obsługę panelu dotykowego i realizację kodu odpowiedzialnego za logikę i peryferia podzieliłem zasoby mikrokontrolera na dwa główne wątki. Jeden obsługuje kod źródłowy programisty, drugi obsługuje wyświetlacz LCD i ekran dotykowy.

W listingu 5.1 przedstawiono inicjalizację systemu czasu rzeczywistego, oraz dwóch głównych wątków, na których będzie opierał się program. Definiuje wskaźniki dla dwóch wątków oraz funkcje je wywołujące. Przyporządkowuje zasoby sprzętowe (pamięć RAM) dla poszczególnych wątków oraz ustawiam ich priorytet działania.

```
//wskaźniki do wątków
osThreadId defaultTaskHandle;
osThreadId UartTaskHandle;
//wątek odpowiedzialny za obsługę panelu dotykowego
void StartDefaultTask(void const * argument);
//wątek odpowiedzialny za obsługę reszty kodu/programu
void StartUartTask(void const * argument);
//Inicjalizacja wątków oraz przyporządkowanie zasobów im przeznaczonych
osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 4096);
defaultTaskHandle= osThreadCreate(osThread(defaultTask), NULL);
osThreadDef(UartTask, StartUartTask, osPriorityNormal, 0, 2048);
defaultTaskHandle= osThreadCreate(osThread(UartTask), NULL);
//funkcja inicjalizująca system czasu rzeczywistego
osKernelStart();
```

listning.5.1

5.2.1.Wątek główny programu

Wątek główny (Listing 5.2) został napisany w oparciu o 4 instrukcję warunkowe if oznaczone w komentarzach do kodu źródłowego jako "if_1", "if_2". "if_3", "if_4" oznaczone kolorem jasno niebieskim.

Instrukcja "if_1" zostaje wywołana jeśli nastąpi odbiór komendy z wątku obsługującego panel LCD. To znaczy jeśli operator naciśnie jeden z 7 przycisków odpowiedzialnych za ruch robota (przyciski nr. 1,2,3,4,5,6,7 widoczne na rysunku 4.7). Wtedy w zależności od wybranego przycisku mikrokontroler wysyła daną komendę ruchu do robota mobilnego.

Instrukcja "if_2" zostaje wywołana po zmianie wartości prędkości robota, czyli po przesunięciu suwaka na ekranie dotykowym (suwak nr. 8 widoczne na rysunku 4.7). Następnie wewnątrz instrukcji if następuje konwersja otrzymanej wartości prędkości na odpowiednie tablice znakowe, tworzące komendy, które są wykorzystywane do sterowania robotem.

Instrukcja "if_3" zostaje wywołana po otrzymaniu informacji zwrotnej od robota, dotyczącej aktualnej prędkości kół. Instrukcja otrzymana w postaci tablicy znaków jest konwertowana do tablicy czterech liczb całkowitych odpowiadającym prędkości czterech kół. Dalej tablica liczb całkowitych jest przesyłana do wątku obsługującego panel LCD, po czym wyświetlana na ekranie (grafika nr. 11 widoczna na rysunku 4.7).

Instrukcja "if_4" zostaje wywołana przy odbiorze dowolnego znaku od robota mobilnego. Służy to sprawdzania połączenia z robotem. Gdyby żaden znak nie był odbierany, a wiadomo, że robot wysyła regularnie informacje na temat prędkości kół, to panel HMI nie byłby połączony z robotem mobilnym. Po pomyślnym odbiorze dowolnego znaku sygnał o prawidłowym połączeniu jest wysyłany do wątku obsługującego panel LCD i następnie wyświetlany na ekranie (grafika nr. 10 widoczna na rysunku 4.7).

```
void StartUartTask(void const * argument)
{
int sliderCurrenValue = 0;//zmienna przechowująca wartość prędkości
//tablica przechowująca informacje zwrotną z enkoderów
int encoderValue[]={0,0,0,0};
//tablica pomocnicza przechowująca wartość prędkości jako string
char sliderCurrentValueChar[4];
uint8_t sterowanieSend=0;//zmienna przechowująca rodzaj ruchu robota
uint8_t command1[]="sp=0000=0000=0000=0000\n";//komenda jazdy do przodu
uint8_t command2[]="sp=-0000=-0000=-0000=-0000\n";//komenda jazdy do tyłu
uint8_t command3[]="sp=0000=-0000=-0000=0000\n";//komenda jazdy w prawo
uint8_t command4[]="sp=-0000=0000=0000=-0000\n";//komenda jazdy w lewo
uint8_t command5[]="sp=0000=-0000=0000=-0000\n";//komenda obrotu w prawo
uint8_t command6[]="sp=-0000=0000=-0000=0000\n";//komenda obrotu w lewo
uint8_t command7[]="sp=0000=0000=0000=0000\n";//komenda zatrzymania (stop)
//tablica przechowująca odpowiedź od robota
uint8_t bufferReceiveBack[]="Speed=-0000=-0000=-0000=-00000000000\n";
//tablica przechowująca obrobioną odpowiedź od robota
uint8_t bufferReceiveBackFiltr[]="Speed=-0000=-0000=-0000=-00000000000\n";
//zmienna pomocnicza(używana do sprawdzania czy jest połączenie z robotem)
uint8_t helpValue1 = 8;
//zmienna pomocnicza(używana do sprawdzania czy jest połączenie z robotem)
uint8_t helpValue2 = 1;
uint8_t uBuffer = 0;//zmienna przechowujące liczby całkowite do obliczeń
BSP_LED_Init(LED_GREEN);//inicjalizacja zielonej diody na płytce
```

```
  for(;;)//pętla nieskończona wątku
  {

      sterowanieSend = 0;
//Ta część programu odbiera od wątku odpowiedzialnego za ekran dotykowy którą
komendę odpowiedzialną za jeden z 7 rodzaju ruchu wysłać do robota interfejsem
UART
      if((xQueueReceive(QueueSend, &sterowanieSend, 1))==pdTRUE)// if_1
      {
//przy każdym odbiorze komendy dioda zmienia stan
       BSP_LED_Toggle(LED_GREEN);
       if(sterowanieSend == 1)//wysłanie komendy jazdy do przodu
            {
            HAL_UART_Transmit_IT(&huart6, command1, sizeof(command1)-1);
            }
       if(sterowanieSend == 2)//wysłanie komendy jazdy do tyłu
            {
            HAL_UART_Transmit_IT(&huart6, command2, sizeof(command2)-1);
            }
       if(sterowanieSend == 3)//wysłanie komendy jazdy w prawo
            {
            HAL_UART_Transmit_IT(&huart6, command3, sizeof(command3)-1);
            }
       if(sterowanieSend == 4)//wysłanie komendy jazdy w lewo
            {
            HAL_UART_Transmit_IT(&huart6, command4, sizeof(command4)-1);
            }
       if(sterowanieSend == 5)//wysłanie komendy obrotu w prawo
            {
            HAL_UART_Transmit_IT(&huart6, command5, sizeof(command5)-1);
            }
       if(sterowanieSend == 6)//wysłanie komendy obrotu w lewo
            {
            HAL_UART_Transmit_IT(&huart6, command6, sizeof(command6)-1);
            }
       if(sterowanieSend == 7)//wysłanie komendy stopu
            {
            HAL_UART_Transmit_IT(&huart6, command7, sizeof(command7)-1);
            }
      }

//Ta część kodu odbiera od drugiego wątku wartość prędkości, którą ustawiamy
poruszając suwakiem na ekranie dotykowym
 if((xQueueReceive(QueueSendSliderValue, &sliderCurrenValue, 1))==pdTRUE)// if_2
       {
//zmiana typu zmiennej z int na tablice char
        itoa(sliderCurrenValue,sliderCurrentValueChar,10);
//zmienia liczbę na 4 elementową tablicę charów i wypełnia początek zerami kiedy
trzeba
        IntToString(sliderCurrentValueChar, sliderCurrenValue);
```

```
//funkcje aktualizujące wartość prędkości dla każdej komendy
         UpdateSpeedValue((char*)command1, sliderCurrentValueChar);
         UpdateSpeedValue((char*)command2, sliderCurrentValueChar);
         UpdateSpeedValue((char*)command3, sliderCurrentValueChar);
         UpdateSpeedValue((char*)command4, sliderCurrentValueChar);
         UpdateSpeedValue((char*)command5, sliderCurrentValueChar);
         UpdateSpeedValue((char*)command6, sliderCurrentValueChar);
        }

//Część kodu odpowiedzialna za odbiór informacji zwrotnej od robota na temat
prędkości kół z enkoderów.
       if( HAL_UART_Receive_IT(&huart6, bufferReceiveBack,
sizeof(bufferReceiveBack))== 0)// if_3
       {
              bufferReceiveBack[sizeof(bufferReceiveBack)-1] = '\n';
              while(!(bufferReceiveBack[uBuffer] == '\n'))
              {
               bufferReceiveBackFiltr[uBuffer] = bufferReceiveBack[uBuffer];
                uBuffer = uBuffer + 1;
              }
              bufferReceiveBackFiltr[sizeof(bufferReceiveBackFiltr)]= '\n';
              uBuffer = 0;
//funkcja wydobywająca z tablicy charów cztery liczby całkowite, które
przechowują wartości odczytane z poszczególnych enkoderów
              StringToInt(bufferReceiveBackFiltr, encoderValue);
//przesłanie wartości z enkoderów, aby były wyświetlone na ekranie
              xQueueSend(QueueReceiveEncoderValue,encoderValue,0);
       }

//Część kodu która sprawdza połączenie z robotem. Gdy nie ma połączenie UART nic
nie odbiera. Resultat wysyłany jest do drugiego wątku
       if(HAL_UART_Receive_IT(&huart6, &helpValue1, 1) == 0)// if_4
       {
              xQueueSend(QueueReceive,&helpValue2,0);
       }
  }
}
```

listning.5.2

5.2.2.Wątek obsługujący dotykowy panel LCD

W strukturze kodu dla ekranu dotykowego warto przybliżyć dwa występujące scenariusze.

1. Dane są przesyłane z głównego wątku (np. dane z enkoderów robota) odbierane i przetwarzane w klasie Model po czym przesyłane do klasy Presenter a potem do klasy View, gdzie następuje wyświetlenie ich na ekranie LCD. Tabela 5.1 przedstawia funkcje zaimplementowane do tego przypadku.

Tab.5.1.Funkcje przypadku pierwszego

Model→	→ Presenter →	→ View
`Model::tick()`	`MainPresenter::DisplayConnectedStatus(status)`	`MainView::updateConnectedStatus(uint8_t status)`
	`MainPresenter::DisplayEncoderValue(encoderValueResponse[0],encoderValueResponse[1],encoderValueResponse[2],encoderValueResponse[3])`	`MainView::updateEncoderValue(int encoderValueFR ,int encoderValueFL ,int encoderValueRR ,int encoderValueRL)`
Funkcja `Model::tick()` jest wywoływana 24 razy na sekundę. Spowodowane jest to częstotliwością odświeżania ekranu LCD. W funkcji `Model::tick()` znajdują się instrukcje warunkowe if, które są wykonywane kiedy nastąpi poprawne odebranie informacji poprzez funkcje `xQueueReceive` z wątku głównego. `DisplayConnectedStatus(status)` jest funkcją która jest odpowiedzialna za sprawdzanie czy panel HMI jest połączony z robotem. test wykonywany jest co 2 sekundy. W zależności od statusu połączenia funkcja wywołuje funkcję `updateConnectedStatus(uint8_t status)` z argumentem 0=false lub 1=true jako status połączenia. `DisplayEncoderValue()` jest funkcją która po otrzymaniu informacji zwrotnej z wątku głównego na temat aktualnej prędkości odczytanej z enkoderów wywołuje funkcje `MainView::updateEncoderValue()` z czterema argumentami zawierającymi wartości prędkości kół. Następnie wywołana funkcja wyświetla na ekranie aktualną prędkość.		

2. Po odpowiedniej interakcji użytkownika z ekranem LCD. Dane są przesyłane od klasy View do klasy Presenter po czym do klasy Model, gdzie są odpowiednio przetwarzane i wysyłane do głównego wątku poprzez funkcję xQueueSend (np. komendy do sterowania ruchem robota). Tabela 5.2 przedstawia funkcje zaimplementowane do tego przypadku.

Tab.5.2.Funkcje przypadku drugiego

View →	→ Presenter →	→ Model
`MainView::buttonUpVirtualFunction()`	`MainPresenter::SendUpPresenter()`	`Model::buttonUpVirtualFunction()`
`MainbuttonDownVirtualFunction()`	`MainPresenter::SendDownPresenter()`	`buttonDownVirtualFunction`

`MainView::buttonLeftVirtualFunction()`	`MainPresenter::SendLeftPresenter()`	`Model::buttonLeftVirtualFunction()`
`MainView::buttonRightVirtualFunction()`	`MainPresenter::SendRightPresenter()`	`Model::buttonRightVirtualFunction()`
`MainView::buttonLeftRotationVirtualFunction()`	`MainPresenter::SendLeftRotationPresenter()`	`Model::buttonLeftRotationVirtualFunction()`
`MainView::buttonRightRotationVirtualFunction()`	`MainPresenter::SendRightRotationPresenter()`	`Model::buttonRightRotationVirtualFunction()`
`MainView::buttonStopVirtualFunction()`	`MainPresenter::SendStopPresenter()`	`Model::buttonStopVirtualFunction()`
`MainView::sliderVirtualFunction(int value)`	`MainPresenter::SendSliderCurrentValue(value)`	`Model::sliderVirtualFunction(int value)`
Każdy z "łańcuchów" funkcji wywołujących się nawzajem działa w ten sam sposób. Dlatego ograniczę się do szczegółowego opisania jednego przypadku. Klikając przycisk nr. 3 (Rys.4.7) wywołujemy funkcję `MainView::buttonUpVirtualFunction()`, która wywołuje funkcje `MainPresenter::SendUpPresenter()`, która następnie wywołuje funkcje `Model::buttonUpVirtualFunction()`. Ta funkcja poprzez `xQueueSend(QueueSend,&sterowanie,0)` wysyła do wątku głównego liczbę całkowitą wynoszącą 1, która jest interpretowana w wątku głównym jako wysłanie komendy jazdy do przodu do robota mobilnego.		

W listningu 5.3 funkcja `GRAPHICS_MainTask()` aktywuje panel dotykowy i powoduje przejście do realizacji kodu w klasie **Model**

```
void StartDefaultTask(void const * argument)
{
  MX_FATFS_Init();//inicjalizacja FATFS
  MX_USB_HOST_Init();//inicjalizacja USB_HOST
  GRAPHICS_MainTask();//aktywacja aplikacji graficznej i przejście do Modelu
  for(;;)// pętla, która nigdy nie zaistnieje
  {
    osDelay(1);
  }
}
```

listning.5.3

- **Model**

  W listningu 5.4 szczegółowo opisuje kod w klasie Model.

```
uint8_t  sterowanie = 0;//zmienna przechowująca rodzaj ruchu robota
uint8_t  status = 0;//zmienna pomocnicza, do określania połączenia z robotem
//tablica przechowująca informacje zwrotną z enkoderów
int encoderValueResponse[]={0,0,0,0};
```

```
//zmienna pomocnicza, pozwalająca sprawdzać status połączenia co określony czas
int timer = 0;
bool connected = false;

extern "C"
{
//definiowanie wskaźników dla ‘kolejek’. Są to struktury którymi wysyła się dane między wątkami
xQueueHandle QueueSend;//kolejka przechowująca komendy ruchu robota
//kolejka przechowująca ustawioną wartość prędkości ustawiane suwakiem
xQueueHandle QueueSendSliderValue;
//kolejka pomagająca sprawdzić status połączenia z robotem
xQueueHandle QueueReceive;
////kolejka przechowująca wartości enkoderów
xQueueHandle QueueReceiveEncoderValue;
}

Model::Model() : modelListener(0)
{
//definiowanie rodzaju danych, które będą przesyłały kolejki
	QueueSend = xQueueCreate(1,sizeof(uint8_t));
	QueueSendSliderValue = xQueueCreate(1,sizeof(int));
	QueueReceive = xQueueCreate(1,sizeof(uint8_t));
	QueueReceiveEncoderValue = xQueueCreate(1,sizeof(int)*4);
}

void Model::tick()//funkcja aktywowana 24 razy na sekundę
{
//Ten fragment kodu odbiera informacje od głównego wątku jeśli jest połączenie z robotem. W zależności czy jest połączenie czy nie, wysyła odpowiednią wiadomość do klasy Presenter.Połączenie jest sprawdzane co 2 sekundy
		if(xQueueReceive(QueueReceive, &status, 0) == pdTRUE)
		{
			connected = true;
		}
		if(timer % (24*2) == 0 && connected == true)
		{
			modelListener->DisplayConnectedStatus(status);
			status = 0;
			connected = false;
		}
		else if(timer % (24*2) == 0 && connected == false)
		{
			modelListener->DisplayConnectedStatus(status);
			connected = false;
		}

//Ten fragment kody wysłuchuje danych na temat wartości enkoderów przesyłanych z głównego wątku. Kiedy je dostanie, przesyła je do klasy Presenter, czyli wyświetla na ekranie
```

```
if(xQueueReceive(QueueReceiveEncoderValue, encoderValueResponse, 0) == pdTRUE)
	{
modelListener->DisplayEncoderValue(encoderValueResponse[0],encoderValueResponse[
1],encoderValueResponse[2],encoderValueResponse[3]);
	}
timer++;
}

//poniższe 7 funkcji jest aktywowane jednym z 7 przycisków na ekranie
odpowiedzialnych za ruch robota.Funkcje są wywoływane w klasie Presenter. Każda
z funkcji wysyła do głównego wątku odpowiednia komende.
void Model::buttonUpVirtualFunction()//komenda jazdy do przodu
{
	sterowanie = 1;
	xQueueSend(QueueSend,&sterowanie,0);
}
void Model::buttonDownVirtualFunction()//komenda jazdy do tyłu
{
	sterowanie = 2;
	xQueueSend(QueueSend,&sterowanie,0);
}
void Model::buttonRightVirtualFunction()//komenda jazdy w prawo
{
	sterowanie = 3;
	xQueueSend(QueueSend,&sterowanie,0);
}
void Model::buttonLeftVirtualFunction()//komenda jazdy w lewo
{
	sterowanie = 4;
	xQueueSend(QueueSend,&sterowanie,0);
}
void Model::buttonRightRotationVirtualFunction()//komenda obrotu w prawo
{
	sterowanie = 5;
	xQueueSend(QueueSend,&sterowanie,0);
}
void Model::buttonLeftRotationVirtualFunction()//komenda obrotu w lewo
{
	sterowanie = 6;
	xQueueSend(QueueSend,&sterowanie,0);
}
void Model::buttonStopVirtualFunction()//komenda zatrzymania robota (stop)
{
	sterowanie = 7;
	xQueueSend(QueueSend,&sterowanie,0);
}
//Funkcja aktywowana przesunięciem suwaka od prędkości. Funkcje aktywuje klasa
Presenter. Dane od razu są wysyłane do głównego wątku
void Model::sliderVirtualFunction(int value)//zmiana wartości prędkości
{
```

```
        xQueueSend(QueueSendSliderValue,&value,0);
}
```

listning.5.4

- **Presenter**

  W tej klasie są tylko funkcje pośredniczące. Jest to spowodowane tym, że w moim projekcie jest tylko jeden ekran. Przy większej ilości ekranów ich logikę umieszcza się w klasie Presenter, a nie w klasie Model jak ja to zrobiłem, jednak przy jednym ekranie producent pokazuje w materiałach szkoleniowych, żeby jak napisałem użyć klasy Presenter jako ‘pośrednika’.W listningu 5.5 szczegółowo opisuje kod w klasie Presenter.

```
//Poniższe funkcje są wywoływane z klasy View i wywołują funkcje z klasy Model
void MainPresenter::SendUpPresenter()
{
        model->buttonUpVirtualFunction();
}
void MainPresenter::SendDownPresenter()
{
        model->buttonDownVirtualFunction();
}
void MainPresenter::SendRightPresenter()
{
        model->buttonRightVirtualFunction();
}
void MainPresenter::SendLeftPresenter()
{
        model->buttonLeftVirtualFunction();
}
void MainPresenter::SendRightRotationPresenter()
{
        model->buttonRightRotationVirtualFunction();
}
void MainPresenter::SendLeftRotationPresenter()
{
        model->buttonLeftRotationVirtualFunction();
}
void MainPresenter::SendStopPresenter()
{
        model->buttonStopVirtualFunction();
}
void MainPresenter::SendSliderCurrentValue(int value)
{
        model->sliderVirtualFunction(value);
}
```

```
//Poniższe funkcje są wywoływane z klasy Model i wywołują funkcje z klasy View
void MainPresenter::DisplayConnectedStatus(uint8_t status)
{
	view.updateConnectedStatus(status);
}
void MainPresenter::DisplayEncoderValue(int encoderValueFR ,int encoderValueFL
,int encoderValueRR ,int encoderValueRL)
{

view.updateEncoderValue(encoderValueFR,encoderValueFL,encoderValueRR,encoderValu
eRL);
}
```

listning.5.5

- **View**

  W listningu 5.6 szczegółowo opisuje kod w klasie View.

```
uint16_t tabON[]={'O','N','\0'};
uint16_t tabOFF[]={'O','F','F','\0'};
void MainView::setupScreen()//funkcja aktywowana przy włączeniu ekranu
{
    MainViewBase::setupScreen();
}
void MainView::tearDownScreen()//funkcja aktywowana przy wyłączeniu ekranu
{
    MainViewBase::tearDownScreen();
}

void MainView::buttonUpVirtualFunction()//komenda jazdy do przodu
{
	presenter->SendUpPresenter();
}
void MainView::buttonDownVirtualFunction()//komenda jazdy do tyłu
{
	presenter->SendDownPresenter();
}
void MainView::buttonLeftVirtualFunction()//komenda jazdy w lewo
{
	presenter->SendLeftPresenter();
}
void MainView::buttonRightVirtualFunction()//komenda jazdy w prawo
{
	presenter->SendRightPresenter();
}
void MainView::buttonLeftRotationVirtualFunction()//komenda obrotu w lewo
{
	presenter->SendLeftRotationPresenter();
}
void MainView::buttonRightRotationVirtualFunction()//komenda obrotu w prawo
```

```
{
	presenter->SendRightRotationPresenter();
}
void MainView::buttonStopVirtualFunction()//komenda zatrzymania robota (stop)
{
	presenter->SendStopPresenter();
}
//funkcja wywoływana przesyłająca ustawioną wartość prędkości przez suwak.
void MainView::sliderVirtualFunction(int value)
{
	presenter->SendSliderCurrentValue(value);
//poniższa część funkcji aktualizuje wartość ustawioną na ekranie w postaci
liczby nad suwakiem
	Unicode::snprintf(textAreaSliderBuffer,TEXTAREASLIDER_SIZE,"%d",value);
	textAreaSlider.invalidate();
	value = 0x0000;
}

//Funkcja wyświetlająca status połączenia z robotem na ekranie (ON lub OFF)
void MainView::updateConnectedStatus(uint8_t status)
{
	if(status == 1)
	{
		Unicode::snprintf(textAreaBuffer,TEXTAREA_SIZE,"%s",tabON);
		textArea.resizeToCurrentText();
		textArea.invalidate();
	}
	if(status == 0)
	{
		Unicode::snprintf(textAreaBuffer,TEXTAREA_SIZE,"%s",tabOFF);
		textArea.resizeToCurrentText();
		textArea.invalidate();
	}

}
//funkcja wyświetlająca wartości odczytane z enkoderów na ekranie
void MainView::updateEncoderValue(int encoderValueFR ,int encoderValueFL ,int
encoderValueRR ,int encoderValueRL)
{
Unicode::snprintf(textAreaEncoderFRBuffer,TEXTAREAENCODERFR_SIZE,"%d",encoderVal
ueFR);
Unicode::snprintf(textAreaEncoderFLBuffer,TEXTAREAENCODERFL_SIZE,"%d",encoderVal
ueFL);
Unicode::snprintf(textAreaEncoderRRBuffer,TEXTAREAENCODERRR_SIZE,"%d",encoderVal
ueRR);
Unicode::snprintf(textAreaEncoderRLBuffer,TEXTAREAENCODERRL_SIZE,"%d",encoderVal
ueRL);
textAreaEncoderFR.invalidate();
textAreaEncoderFL.invalidate();
textAreaEncoderRR.invalidate();
```

```
textAreaEncoderRL.invalidate();
}
```

listning.5.6

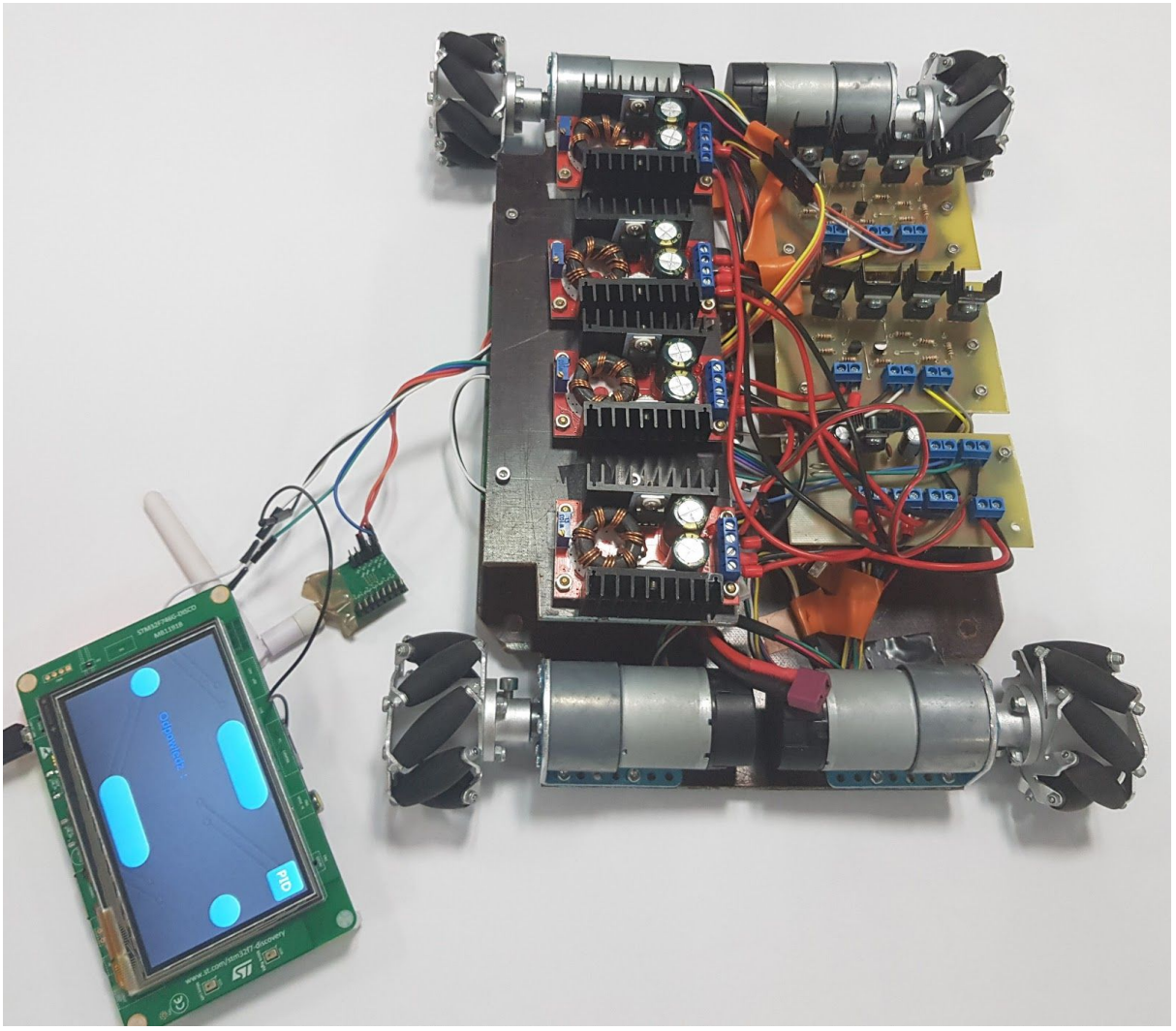
6.Testy funkcjonalne

W tym rozdziale przedstawiam trzy przeprowadzone testy zaimplementowanego na zestawie uruchomieniowym STM746G-DISCOVERY. panelu HMI. Celem podstawowym było połączenie się z robotem interfejsem UART i poprzez przesyłane komendy, nadanie wybranego kierunku ruchu o ściśle określonej prędkości. Następnie dodałem do panelu możliwość 7 rodzajów ruchu i nastawiania prędkości suwakiem. W drugim teście sprawdzałem odbieranie połączenia bezprzewodowego z użyciem dwóch modułów ZigBee. Trzeci test dotyczył odbioru informacji zwrotnych od robota. Robot cyklicznie przesyła prędkość którą odczytuje z enkoderów w postaci tablicy znakowej np.”Speed=1000=1000=1000=1000\n”. Moim zadaniem było napisać funkcję, która odbiera te dane, przetwarza i wyświetla na ekranie.

Na wstępie należy zaznaczyć, że z powodu uszkodzenia prawego przedniego koła nie mogłem przeprowadzić satysfakcjonujących mnie testów. Badania przeprowadzałem umieszczając robota na kartonowej podstawce Rys.6.3. Problemem była awaria enkodera, co udowodniłem w trzecim teście.

6.1.Test połączenia przewodowego panelu HMI z robotem

Do tego testu przeszedłem po wnikliwej analizie kodu wgranego do robota. Kod jest opisany w pracy dyplomowej [23]. Podczas analizy dowiedziałem się jakie komendy trzeba wysłać, żeby wywołać określony efekt. Po napisaniu oprogramowania na panel HMI i odpowiednim połączeniu pinów na mikrokontrolerach zacząłem pierwszy test. Rysunek 6.1 przedstawia przewodowe połączenie panelu HMI z robotem.

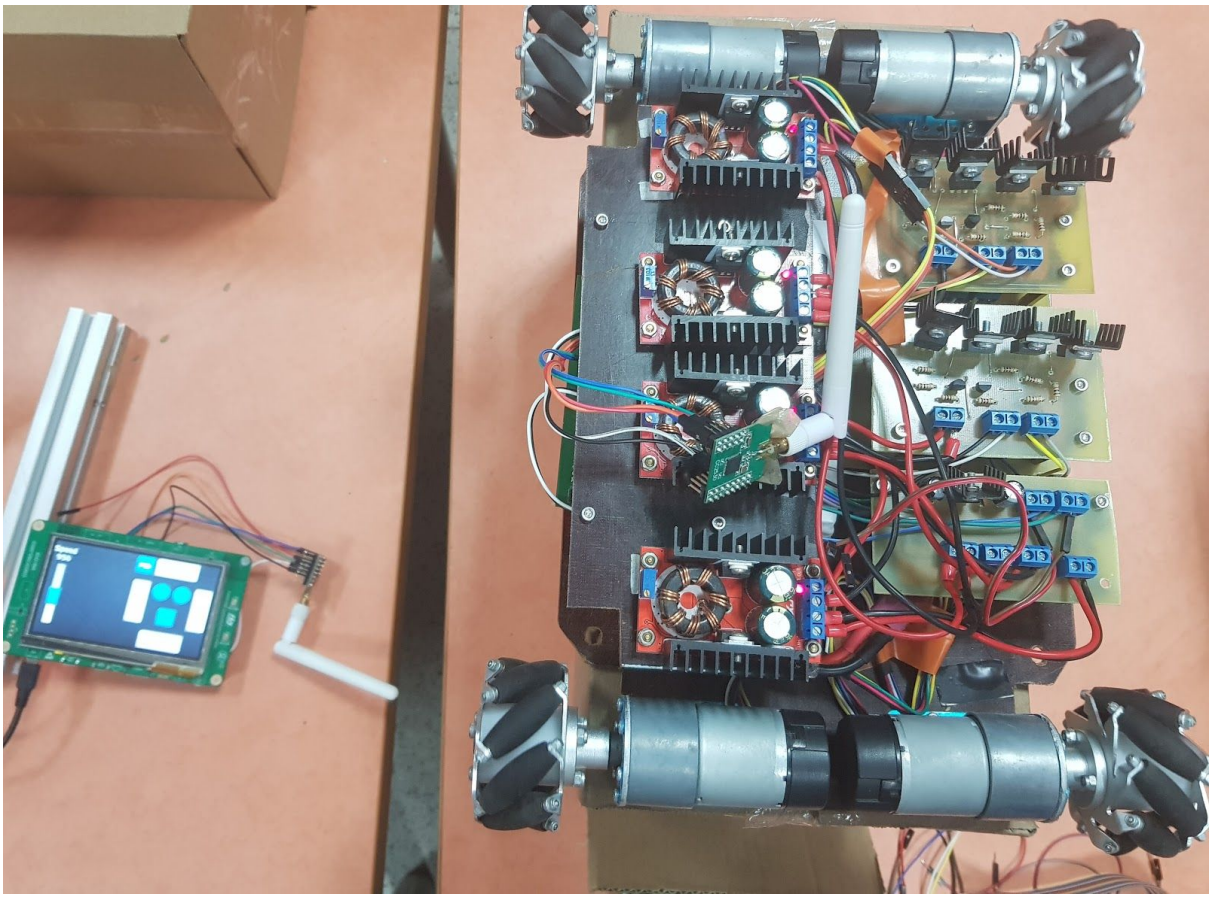


Rys.6.1.Zdjęcie Panelu HMI połączonego z robotem mobilnym przewodowo
Źródło: Materiał własny

Test nie przeszedł w 100% pomyślnie. Trzy koła robota zaczęły się kręcić w zadanym tempie po wysłaniu komendy, lecz prawe przednie koło kręciło się z maksymalną prędkością. Co uniemożliwiło mi testy prowadzenia robota na płaskiej powierzchni. Co warto zaznaczyć wszystkie koła kręciły się w dobrym kierunku.

6.2.Test połączenia bezprzewodowego panelu HMI z robotem

Przed przystąpieniem do testu zaimplementowałem do panelu HMI możliwość 6 rodzajów ruchu i przycisk stop (ustawienie prędkości kół na zero). Ponadto dodałem możliwość ustawienia prędkości suwakiem po lewej stronie ekranu HMI, oraz wyświetlenie tej wartości nad suwakiem. Użyłem dwóch wcześniej skonfigurowanych przez twórcę robota [22] modułów ZigBee, które podpiąłem do odpowiednich pinów na mikrokontrolerach.

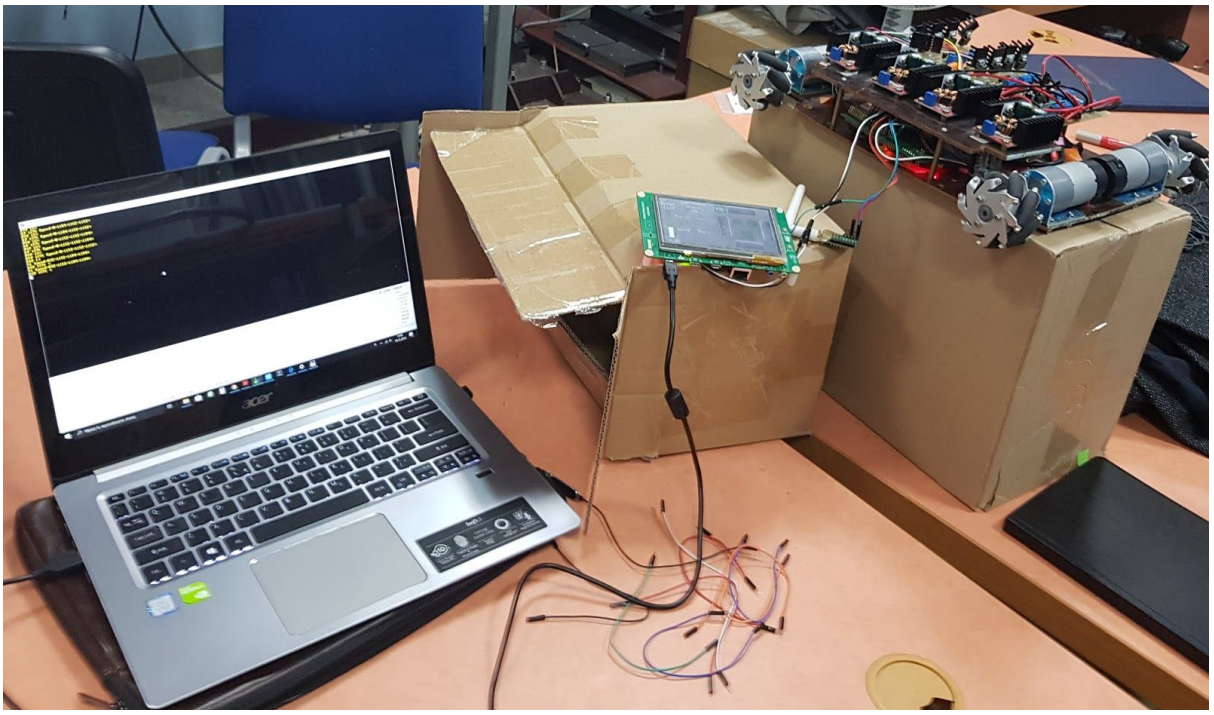


Rys.6.2.Zdjęcie Panelu HMI połączonego z robotem mobilnym bezprzewodowo
Źródło: Materiał własny

Jak w pierwszym teście, przez uszkodzone jedno koło przeprowadziłem tylko połowiczne testy. Zaimplementowane przeze mnie funkcjonalności działały. Komunikacja bezprzewodowa działała i robot odbierał komendy na odległość (Rys.6.2).

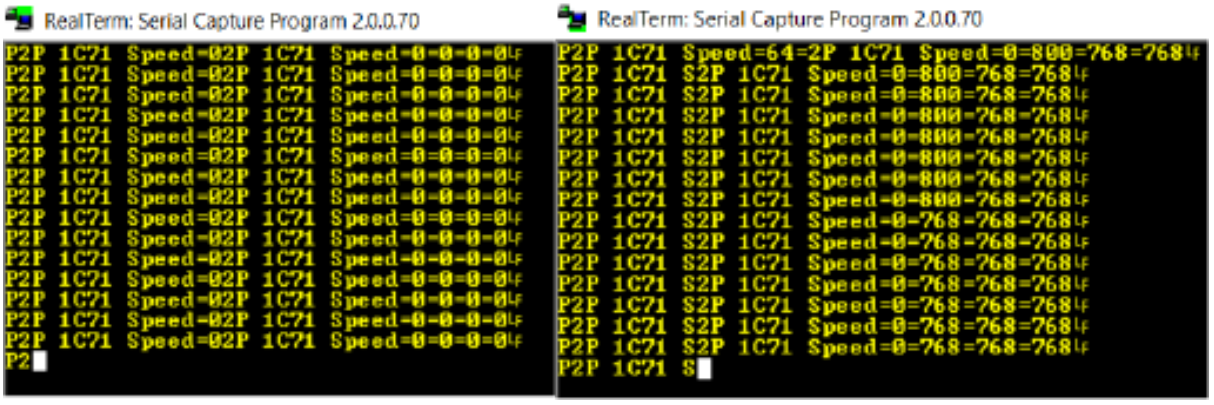
6.3.Test odbierania informacji zwrotnych od robota przez panel HMI

Przed trzecim testem zaimplementowałem funkcje, która odczytuje informację zwrotną dotyczącą prędkości kół od robota i przetwarza ją do postaci czterech liczb całkowitych (znak przed liczbą oznacza kierunek obrotu koła). Dodałem ponadto na ekranie wyświetlanie stanu połączenia z robotem i wyświetlanie aktualnych prędkości kół. Rysunek 6.3 przedstawia zdjęcie zrobione podczas trzeciego testu.



Rys.6.3.Zdjęcie Panelu HMI połączonego z robotem. Test odbierania informacji zwrotnych od robota
Źródło: Materiał własny

Funkcja wyświetlania statusu połączenia działała. Odczytywanie informacji zwrotnej od robota i wyświetlanie jej na początku na ekranie komputera przebiegło pomyślnie, dzięki czemu dowiedziałem się co jest przyczyną wadliwego działania jednego z kół. Ostatnia funkcja wyświetlania aktualnego stanu prędkości kół na ekranie nie została zaimplementowana pomyślnie, powodem jest błędny kod napisany przeze mnie. Podczas konwersji tablicy znaków na cztery liczby całkowite występuje błąd, który uniemożliwia konwersje.



Rys.6.4.Zdjęcie ekranu przedstawiające informacje zwrotne od robota
Źródło: Materiał własny

Rysunek 6.4 przedstawia informacje zwrotną dotycząca prędkości poszczególnych kół. W tym teście po poprawnej komunikacji z robotem nadaje prędkość równą 768

impulsów/sekundę. Trzy koła kręcą się prawidłowo, lecz prawe przednie koło kręci się z maksymalną prędkością do przodu. Wartość, które dostajemy są wartościami z enkoderów przy kołach. Z tego można wywnioskować że błąd leży po stronie enkodera, który nie przesyła wartości do mikrokontrolera sterującego robotem, przez co regulator PID chcąc dojść do zadanej wartości rozpędza koło do maksymalnej wartości, lecz nie dostając informacji zwrotnej błędnie ustawia prędkość.

7.Podsumowanie

W ramach pracy inżynierskiej został zaimplementowany na mikrokontrolerze serii STM32 panel HMI, który steruje robotem mobilnym. W trakcie realizacji projektu zapoznałem się z budową, funkcjami oraz zastosowaniami aktualnie dostępnych na rynku paneli HMI. Ponadto nabrałem doświadczenia w programowaniu nowoczesnych 32-bitowych mikrokontrolerów z rdzeniem ARM w języku C i C++. Sam projekt panelu HMI pozwolił mi w praktyczny sposób wykorzystać zdobyte w toku studiów umiejętności. Był to mój pierwszy tak skomplikowany programistyczny projekt. W trakcie pisania oprogramowania musiałem narzucić sobie pewien podział pracy. Najpierw powstały małe fragmenty kodu odpowiedzialne za pojedyncze funkcje, następnie zostały one przetestowane, po czym zintegrowane z całością kodu. Podczas testowania kolejnych wersji oprogramowania ujawniły się wady robota mobilnego, który jest na wyposażeniu laboratorium “Adaptroniki” na wydziale Inżynierii Mechanicznej i Robotyki. Robot został zbudowany trzy lata temu w ramach pracy inżynierskiej. Od tego czasu wielu studentów dodawało kolejne funkcjonalności w ramach swoich prac dyplomowych. Jest zrozumiałym i logicznym, że po trzech latach eksploatacji robot doświadcza drobnych awarii, których nie byłem w stanie naprawić. Jednak cała ta sytuacja ma też plusy. Dzięki temu musiałem szukać różnych rozwiązań aby zidentyfikować przyczyny problemu. Co mi się udało. W samej pracy zaadaptowałem się do zastanych rozwiązań, wykorzystanych w robocie. Warto tu wymienić komunikację radiową poprzez moduł Zigbee. Rozwiązanie, to w szybki sposób pozwoliło mi bezprzewodowo połączyć panel HMI z robotem mobilnym. Podczas pracy nad projektem szybko zdałem sobie sprawę, że korzystanie z zastanych rozwiązań jest bardziej opłacalne niż projektowanie wszystkiego samemu od podstaw. Szczególnie, że jest to normalna praktyka inżynierska, pozwalająca przyspieszyć prace. Powyższe problemy nie pozwoliły mi

przeprowadzenia całości zaplanowanych testów sterowania robotem przy ruchu po płaskiej powierzchni.

Do stworzenia panelu HMI został wykorzystany zestaw uruchomieniowy STM32F746G-Discovery, który łączy mikrokontroler z ekranem dotykowym. Do stworzenia aplikacji na mikrokontroler wykorzystano środowisko programistyczne Athollic TRUEStudio. Wstępna inicjalizacja peryferiów mikrokontrolera, oraz konfiguracja taktowaniem zegarów została przeprowadzona w środowisku STM32CubeMX. Następnie stworzono interfejs interfejs graficzny w środowisku TouchGFX. Stworzony przy użyciu tych dwóch programów kod źródłowy został przeniesiony do środowiska Athollic TRUEStudio, a następnie dopisane fragmenty kodu źródłowych obsługujące wybrane funkcjionalności takie jak: obsługę przycisków, obsługę suwaka zmieniającego prędkość.

Kierunkami dalszego rozwoju pracy mogą dotyczyć:

- wprowadzenia zaawansowanego systemu sterowania
- zaimplementowania ekranu, posiadającego możliwość wielokrotnego dotyku
- zaprojektowania platformy, która w wygodny i ergonomiczny sposób pozwalała by trzymać w rękach panel dotykowy
- zaimplementowania większej ilości interfejsów komunikacyjnych
- odczyt na ekranie wszystkich parametrów, które można pozyskać z sensorów zainstalowanych na robocie

Literatura

[1]Podstawowe informacje na temat paneli HMI. Dostępny: (20.12.2019)
https://www.anaheimautomation.com/manuals/forms/hmi-guide.php
[2]Artykuł o panelach HMI. Dostępny: (20.12.2019)
https://automatykab2b.pl/raporty/49286-panele-operatorskie-hmi-raport-rynek
[3]Artykuł o najwazniejszych parametrach panelu HMI Dostępny: (20.12.2019)
https://automatykab2b.pl/temat-miesiaca/46259-idealny-hmi/strona/1
[4]Specyfikacja panelu KTP700 BASIC. Dostępny: (20.12.2019)
https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6AV2123-2GA03-0AX0
[5]Specyfikacja tabletu SIMATIC ITP1000. Dostpępny: (20.12.2019)
https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/?mlfb=6AV7880-.....-...2
[6]Specyfikacja zestawu uruchomieniowego STM32F746G-DISCOVERY. Dostępny: (20.12.2019)
https://www.st.com/en/evaluation-tools/32f746gdiscovery.html
[7]Opis oprogramowania Wonderware InTouch. Dostępny: (20.12.2019)
https://www.astor.com.pl/produkty/oprogramowanie-przemyslowe/systemy-scada/wonderware-intouch.html
[8]Opis oprogramowania SIMATIC WinCC. Dostępny: (20.12.2019)
https://pl.wikipedia.org/wiki/WinCC
[9]Opis oprogramowania TouchGFX. Dostępny: (20.12.2019)
https://www.st.com/en/development-tools/touchgfxdesigner.html
[10]Definicja procesora. Dostępny: (20.12.2019) https://pl.wikipedia.org/wiki/Procesor
[11]Definicja mikrokontrolera. Dostępny: (20.12.2019) https://pl.wikipedia.org/wiki/Mikrokontrole
[12]Definicja mikroprocesora. Dostępny: (20.12.2019) https://pl.wikipedia.org/wiki/Mikroprocesor
[13]Opis firmy ARM Holdings. Dostępny(20.12.2019) https://en.wikipedia.org/wiki/Arm_Holdings
[14]Opis firmy STMicroelectronics. Dostępny(20.12.2019) https://en.wikipedia.org/wiki/STMicroelectronics
[15]Informacje na temat serii mikrokontrolerów STM8 MCUs. Dostępny: (20.12.2019)
https://www.st.com/en/microcontrollers-microprocessors/stm8-8-bit-mcus.html#overview
[16]Informacje na temat serii mikroprocesorów STM32 MPUs. Dostępny: (20.12.2019)
https://www.st.com/en/microcontrollers-microprocessors/stm32-arm-cortex-mpus.html#overview
[17]Informacje na temat serii mikrokontrolerów STM32 MCUs. Dostępny: (20.12.2019)
https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html
[18]Informacje na temat serii mikrokontrolerów SPC5. Dostępny: (20.12.2019)
https://www.st.com/en/automotive-microcontrollers/spc5-32-bit-automotive-mcus.html
[19]Specyfikacja zestawu uruchomieniowego **STM32MP157C-DK2**. Dostępny: (20.12.2019)
https://www.st.com/en/evaluation-tools/stm32mp157c-dk2.html
[20]Opis narzędzi programistycznych firmy STMicroelectronic. Dostępny: (20.12.2019)
https://www.st.com/en/development-tools/stm32-software-development-tools.html

[21]Opis biblitek programistycznych oferowanych lub wspieranych przez STMicroelectronic Dostępny: (20.12.2019) https://www.st.com/en/embedded-software/stm32cube-mcu-mpu-packages.html
[22]Praca dyplomowa inżynierska. Bartłomiej Piwowarczyk. *Projekt, wykonanie i sterowanie mobilnym robotem z niezależnym napędem czterech kół.*AGH, Kraków, rok 2017
[23]Praca dyplomowa magisterska. Nikodem Piotr Kastelik. *Implementacja algorytmów sterowania robotem mobilnym z niezależnym napędem czterech kół.*AGH, Kraków, rok 2019
[24] Brian W. Kenrighan, Dennis M. Ritchie, *The C Programming Language*, published by Prientice-Hall Inc., Englewood Cliffs, New Jersey, 1988
[25] Aleksander Kurczyk, *Mikrokontrolery STM32 dla początkujących*, Wydawnictwo btc, Legionowo 2019