

PROJEKT

ROBOTY MOBILNE

Założenia projektowe

Ślepy Micromuse

ŚM

Skład grupy:
Mateusz KOBAK, 241502

Termin: ptTN15

Prowadzący:
dr inż. Wojciech DOMSKI

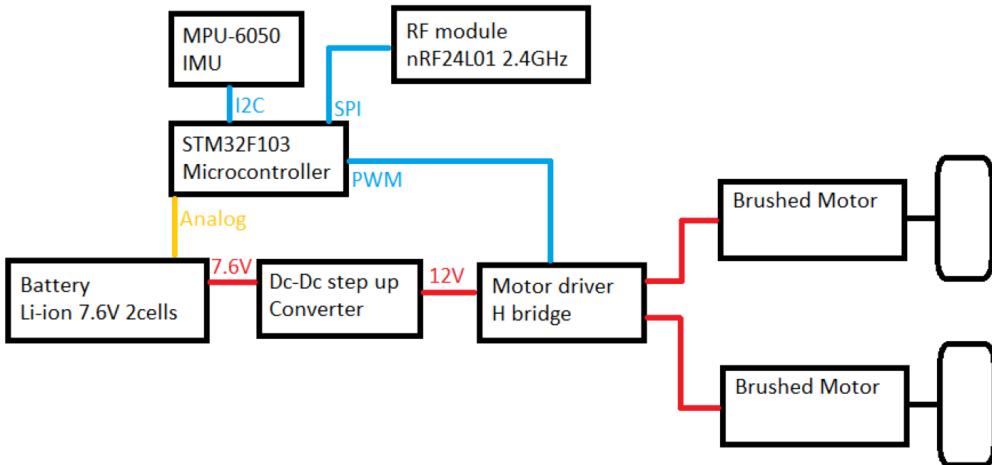
Spis treści

1	Opis projektu	2
2	Konfiguracja mikrokontrolera	3
2.1	Konfiguracja pinów	5
2.2	I2C	5
2.3	SPI	5
3	Urządzenia zewnętrzne	5
3.1	Akcelerometr z żyroskopem MPU-6050	6
3.2	Moduł radiowy NRF24L01	6
3.3	Mostek H	6
4	Opis działania programu	6
5	Konstrukcja mechaniczna	9
6	Wykonanie	10
7	Harmonogram pracy	12
8	Podsumowanie	13
	Bibliografia	14

1 Opis projektu

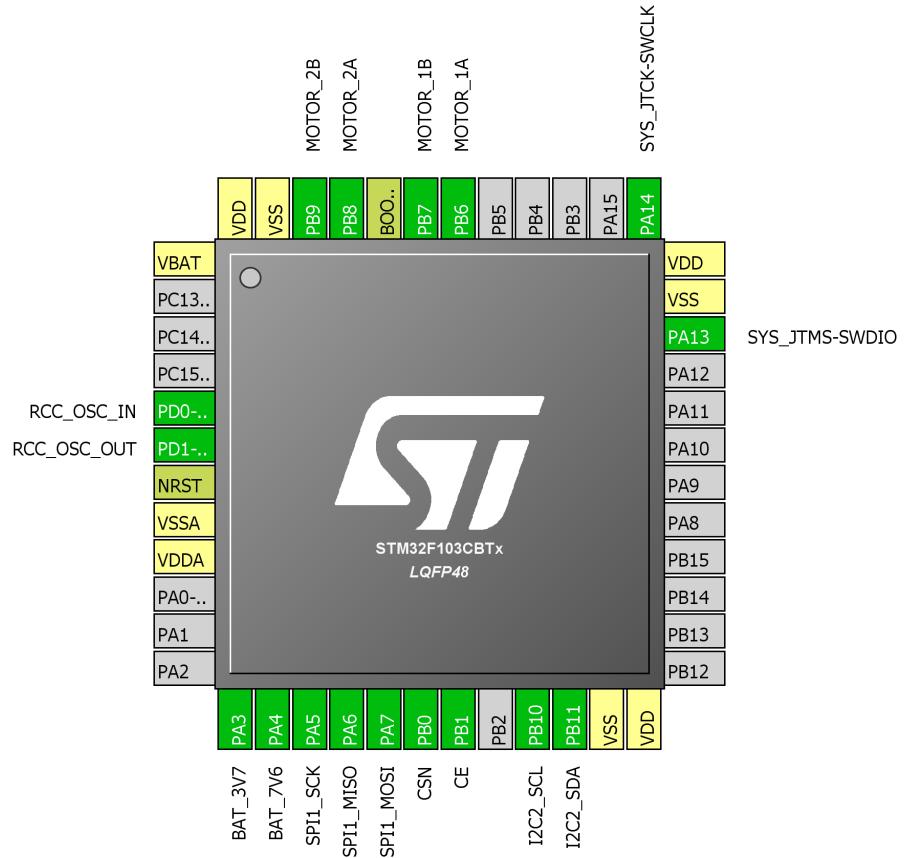
Główym celem projektu jest stworzenie Micromouse który będzie posiadał tylko akcelerometr i żyroskop do rozpoznawania położenia przeszkód i położenia samego robota. Takie rozwiązanie jest niekonwencjonalne gdyż z reguły wszystkie roboty tego typu posiadają czujniki optyczne do wykrywania przeszkód. Micromouse to kategoria robotów których zadaniem jest wydostanie się z labiryntu, ten robot będzie miał takie samo zadanie lecz strategia wyszukiwania wyjścia z labiryntu będzie inna niż standardowo. Robot aby zidentyfikować ściany które są dookoła niego musi w nie stuknąć, nie za mocno by nie uszkodzić labiryntu, a zamontowany akcelerometr wykryje kierunek uderzenia i tak będzie wiadomo gdzie znajduje się przeszkoda. Wykrycie kierunku uderzenia nie jest trywialne więc do tego posłuży specjalnie zamontowana obręcz na sprząźnie której da niewielkie możliwości ruchowe, tak aby przy uderzeniu obrącz lekko się przesunęła i akcelerometr na niej zamontowany mógł poprawnie odczytać przesunięcie. Ruchoma obręcz jest potrzebna gdyż zakładamy że koła nie mają poślizgu, więc wykrycie kierunku byłoby utrudnione. żyroskop daje możliwość odczytu obrotu całego robota, wszystko to jest potrzebne ponieważ robot nie będzie miał enkoderów ani innych sensorów które pozwolą śledzić jego przemieszczenie. Robot będzie posiadać mikrokontroler STM32F103 i sensor MPU-6050. Silniki dobrane będą tak by robot dostatecznie precyzyjnie mógł się poruszać, koła dobrane są tak aby zapewnić przyczepność i móc między innymi w algorytmie wykluczyć poślizg. Dodatkowo wykorzystany zostanie moduł nRF24L01 do komunikacji zdalnej, aby robotem sterować i pobierać od niego informacje sensoryczne, będzie to część na projekt wizualizacja danych sensorycznych. Dodatkowo pomiar napięcia baterii by zabezpieczyć baterie przed rozładowaniem. Moduł mikrokontrolera zawiera stabilizator, tak jak IMU czy moduł radiowy, więc na schemacie została on pominięty

Sensor MPU-6050 [1] Jest 3 osiowym akcelerometrem i 3 osiowym żyroskopem w jednym, komunikacja z nim przebiega przez magistralę I2C
moduł nRF24L01 [3] To moduł radiowy 2.4Ghz, komunikacja z nim odbywa się poprzez magistralę SPI
Schemat modułowy robota przedstawia rysunek 1.

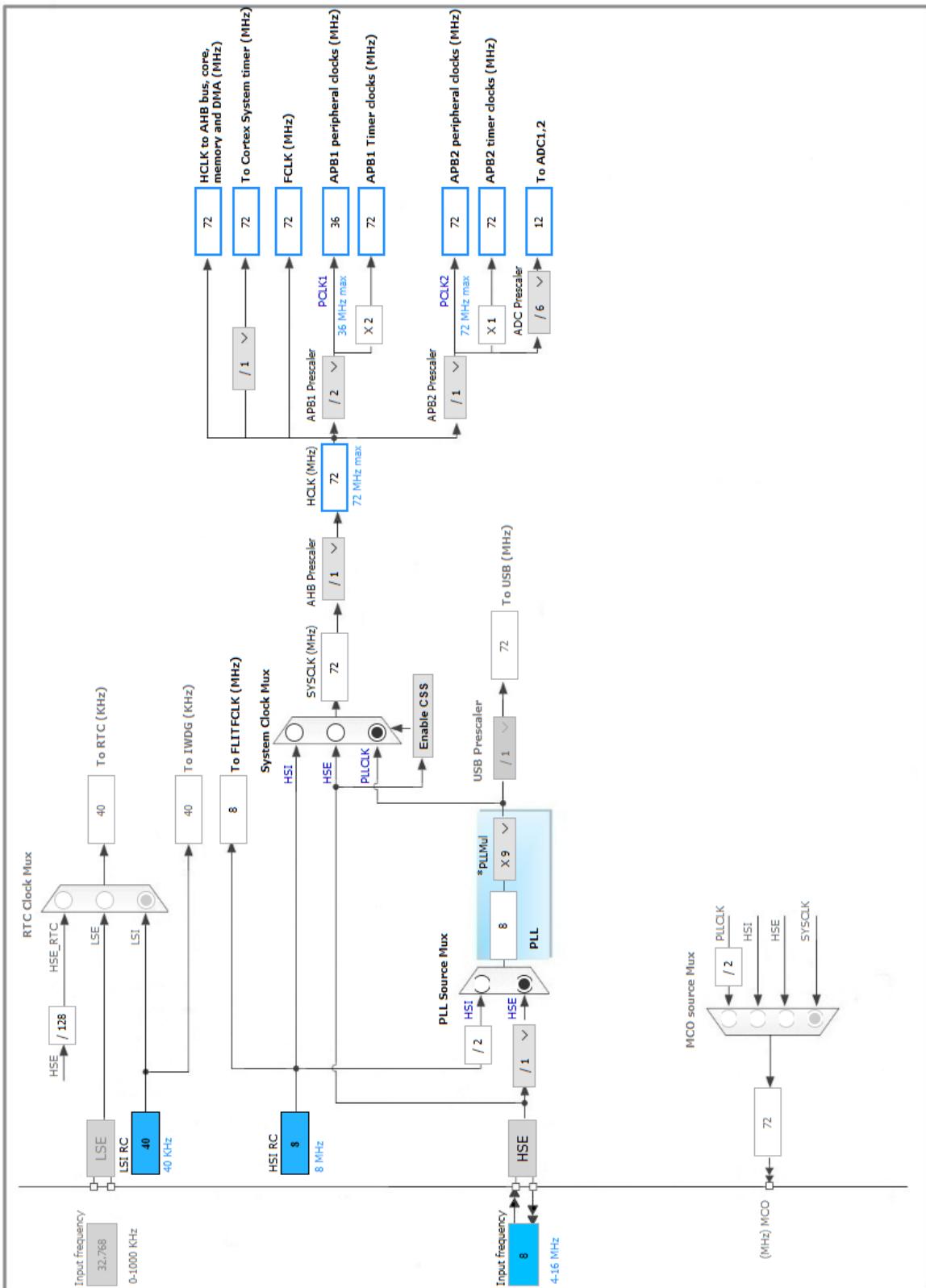


Rysunek 1: Architektura systemu

2 Konfiguracja mikrokontrolera



Rysunek 2: Konfiguracja wyjść mikrokontrolera w programie STM32CubeMX



Rysunek 3: Konfiguracja zegarów mikrokontrolera

2.1 Konfiguracja pinów

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
5	PD0-OSC_IN	RCC_OSC_IN	
6	PD1-OSC_OUT	RCC_OSC_OUT	
13	PA3	ADC1_IN3	BAT_3V7
14	PA4	ADC1_IN4	BAT_7V6
15	PA5	SPI1_SCK	
16	PA6	SPI1_MISO	
17	PA7	SPI1_MOSI	
18	PB0	GPIO_Output	CSN
19	PB1	GPIO_Output	CE
21	PB10	I2C2_SCL	
22	PB11	I2C2_SDA	
34	PA13	SYS_JTMS-SWDIO	
37	PA14	SYS_JTCK-SWCLK	
42	PB6	TIM4_CH1	MOTOR_1A
43	PB7	TIM4_CH2	MOTOR_1B
45	PB8	TIM4_CH3	MOTOR_2A
46	PB9	TIM4_CH4	MOTOR_2B

Tabela 1: Konfiguracja pinów mikrokontrolera

2.2 I2C

Magistrala I2C użyta została do odczytywania danych z IMU. Możliwe będzie ustawienie trybu fast-mode 400KHz gdyby potrzebne było szybkie odczytywanie danych. Konfiguracja I2C. Tabela 2.

Parametr	Wartość
I2C Speed Mode	Standard Mode
I2C Clock Speed (Hz)	100 000
Clock No Stretch Mode	Disabled
Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0
General Call address detection	Disabled

Tabela 2: Konfiguracja peryferium I2C

2.3 SPI

Moduł RF posiada magistralę SPI do wymiany danych, dodatkowo potrzebuje on dwóch pinów do zarządzania trybem pracy. Konfiguracja magistrali SPI. Tabela 3.

Parametr	Wartość
Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge

Tabela 3: Konfiguracja peryferium SPI

3 Urządzenia zewnętrzne

Robot posiada akcelerometr i moduł radiowy NRF które potrzebują odpowiedniej konfiguracji, do przemieszczania się wykorzystano gotowy dwukanałowy mostek H.

3.1 Akcelerometr z żyroskopem MPU-6050

Akcelerometr służy do wykrywania uderzenia z obiektem i obliczania przemieszczenia, żyroskop natomiast wspomaga zachować kierunek jazdy i służy też do wykrywa kąta nachylenia pierścienia aby odczytać kąt pod jakim robot jest ustawiony względem przeskody. Do konfiguracji i obsługi MPU-6050 posłużono się gotową biblioteką [?], jej obsługa sprowadza się do zapisywania i odczytywania odpowiednich danych z rejestrów urządzenia. Inicjalizacja czujnika odbywa się poprzez utworzenie struktury w której wpisywane są wartości, a następnie wywołanie funkcji która wpisuje odpowiednie dane do rejestrów czujnika. Konfiguracja modułu:

```
1 MPU_ConfigTypeDef MpuConfig;
2 MpuConfig.Accel_Full_Scale = AFS_SEL_8g;
3 MpuConfig.ClockSource = Internal_8MHz;
4 MpuConfig.CONFIG_DLPF = DLPF_260A_256G_Hz;
5 MpuConfig.Gyro_Full_Scale = FS_SEL_1000;
6 MpuConfig.Sleep_Mode_Bit = 0;
7
8 MPU6050_Init(&hi2c2);
9 MPU6050_Config(&MpuConfig);
```

3.2 Moduł radiowy NRF24L01

Moduł NRF jest użyty jako interfejs wymiany danych między robotem a urządzeniem sterującym. Moduł ten jest bardzo popularny i wykorzystywany w wielu amatorskich konstrukcjach takich jak drony czy roboty mobilne. Obsługa modułu jest zrealizowana za pomocą gotowej biblioteki [?]. Urządzenie pracować będzie w trybie nadajnika, aby ułatwić szybkie wysyłanie danych z czujników, tryb ten umożliwia wymianę danych w dwie strony poprzez system potwierdzania odebrania informacji przez odbiornik. Ta funkcja również jest obsługiwana przez bibliotekę. Inicjalizacja wygląda następująco:

```
1 NRF24_begin(CSN_GPIO_Port, CSN_Pin, CE_Pin, hspi1);
2
3 NRF24_stopListening();
4 NRF24_openWritingPipe(PipeAddres);
5 NRF24_setAutoAck(true);
6 NRF24_setChannel(52);
7 NRF24_setPayloadSize(16);
8 NRF24_enableDynamicPayloads();
9 NRF24_enableAckPayload();
```

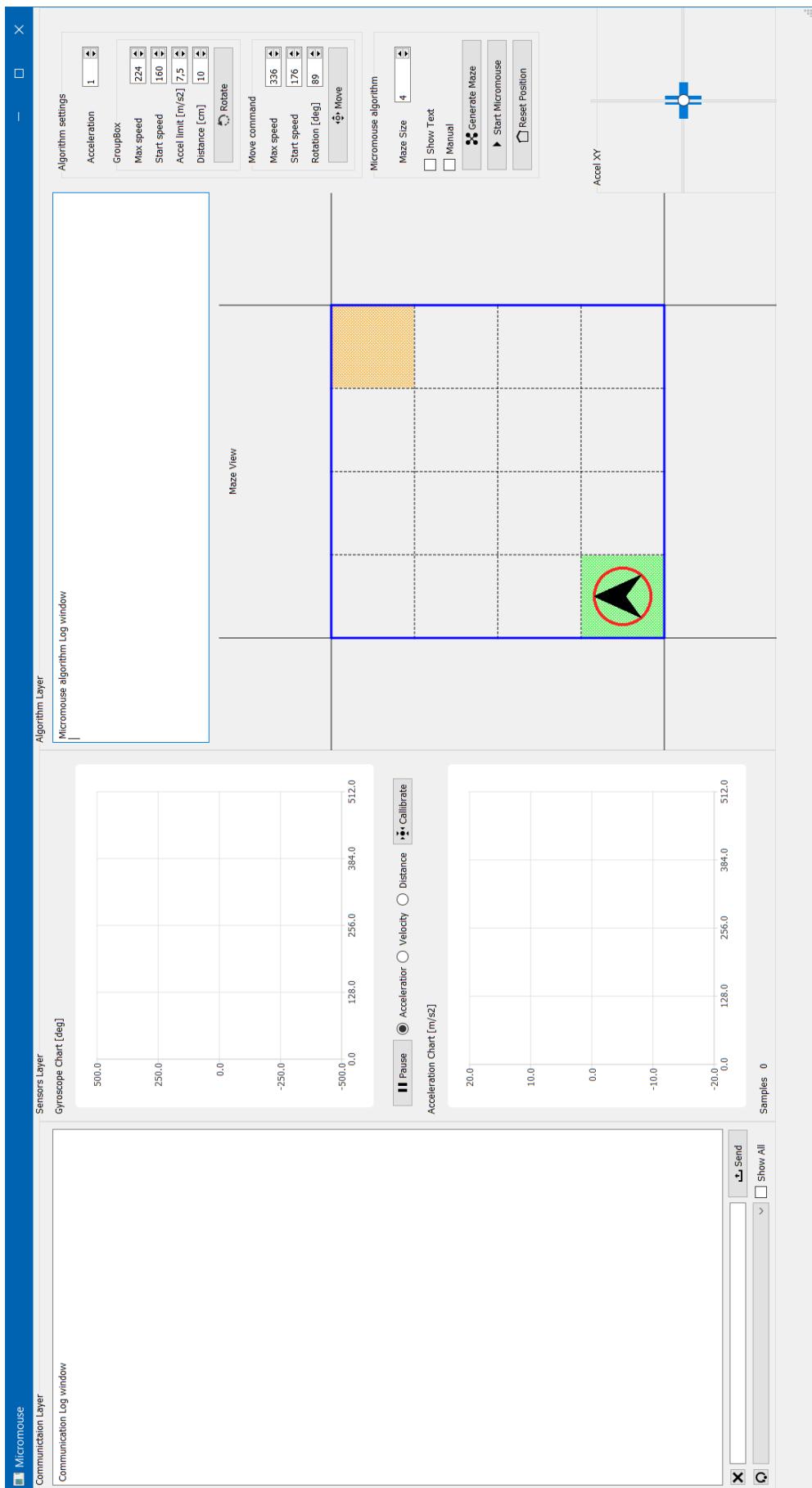
3.3 Mostek H

Do sterowania silnikami użyto gotowego dwukanałowego mostka H, każdy kanał sterowany jest poprzez użycie dwóch linii PWM. Sterowanie kierunkiem obrotów odbywa się poprzez wybór na której linii sygnał PWM jest aktywny, stan gdy obie linie są aktywne jest zabroniony, lecz nie powodowałoby to uszkodzenia sterownika, prędkością steruje się wypełnieniem sygnału PWM. Zegar systemowy mikrokontrolera jest ustawiony na 72Mhz, prescaler wynosi 72 a counter period na 1024 aby ułatwić zadawanie wartości wypełnienia, częstotliwość sygnału wynosi więc około 1kHz.

4 Opis działania programu

Algorytm ma za zadanie wykrywać ściany w labiryncie i zapamiętywać ich ustawienie w pamięci, po odkryciu całego labiryntu wyznaczana będzie ścieżka optymalna z punktu startowego do centrum labiryntu, czyli mety. Następnie robot będzie musiał przebyć wyznaczoną ścieżkę jak najszybciej [2]. Algorytm został zaimplementowany w aplikacji stworzonej na projekt wizualizacji sensorycznej. Takie rozwiązanie bardzo usprawniło tworzenie algorytmu, dodatkowo aplikacja służy też do wizualizacji i sterowania manualnego. Mostkiem łączącym robota z komputerem był joystick JODO który stworzony został na projekt sterowników robotów. Istotne jest to że wymiana danych mimo swej prędkości nie jest wstępnie zapewnić dokładności w takich operacjach jak przemieszczanie się czy obracanie, liczenie podwójnej całki z tak otrzymywanych danych byłoby bezużyteczne, robot więc posiada funkcje do przemieszczania się i obrotu samodzielnie, zapewnia to dokładność.

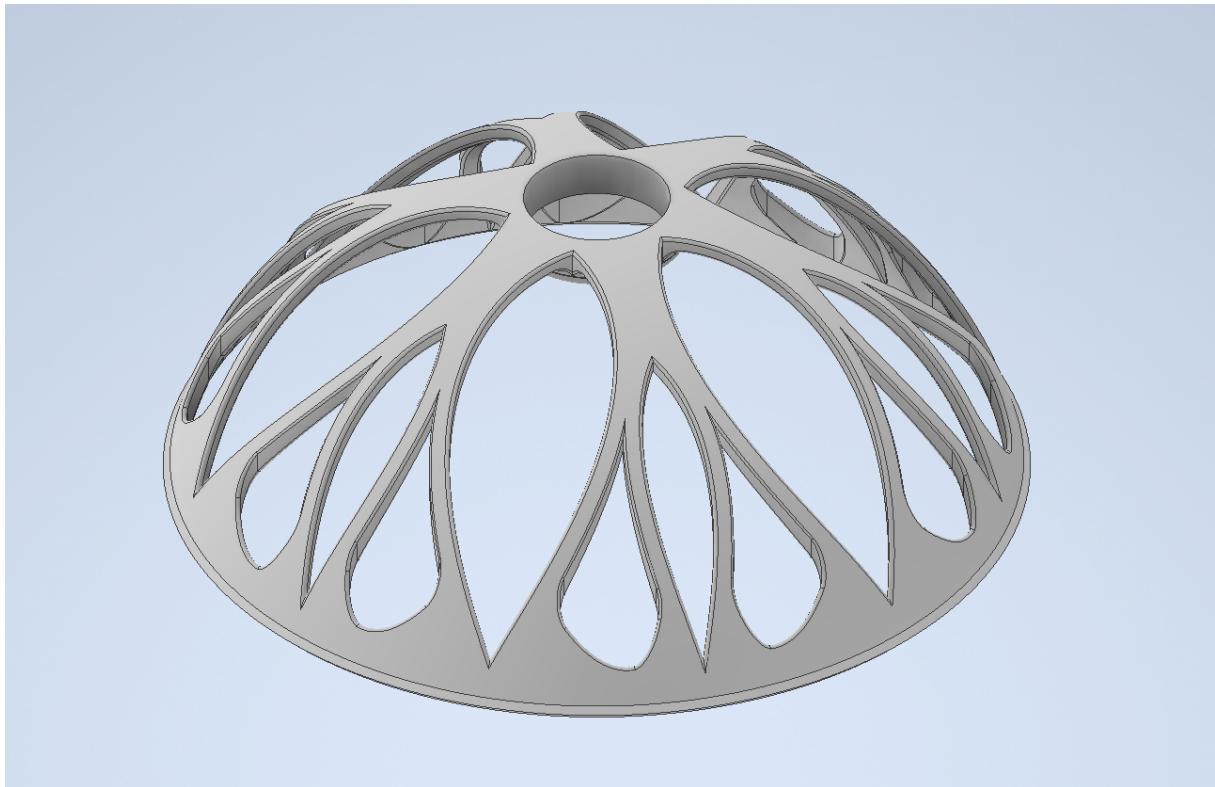
Jazda o zadane przemieszczenie odbywa się poprzez dostarczenie też informacje o wartości progowej przyspieszenia które oznaczałoby zderzenie się z przeszkodą, niezależnie od przyczyny zakończenia ruchu, robot wysyła wartość przemieszczenia jaką udało mu się wyliczyć na podstawie odczytów z akcelerometru. Podczas obrotu nie podaje się warunku stopu. Uzyskana dokładność obrotu jest bardzo zadowalająca, lecz przemieszczanie się jest obarzone takim błędem że nie możliwe jest zadanie robotowi przemieszczenia większego niż 18 cm (rozmiar jednej komórki labiryntu) bez błędu na tyle dużego który zakłóci działanie algorytmu. Problem ten rozwiązuje sposób przemieszczania się w którym robot zawsze przemieszcza się najdalej jak potrafi, algorytm dostając informację zwrotną dopasowyswa ją do labiryntu, jednak takie rozwiązanie powoduje następny problem, robot będzie mógł wpaść w pułapki z których nie będzie dało się w ten sposób wyjechać, rozwiązaniem jest połączenie tych dwóch taktyk tak by algorytm sam decydował o sposobie przemieszczania się. Okno aplikacji prezentuje rysunek 4



Rysunek 4: Aplikacja sterująca

5 Konstrukcja mechaniczna

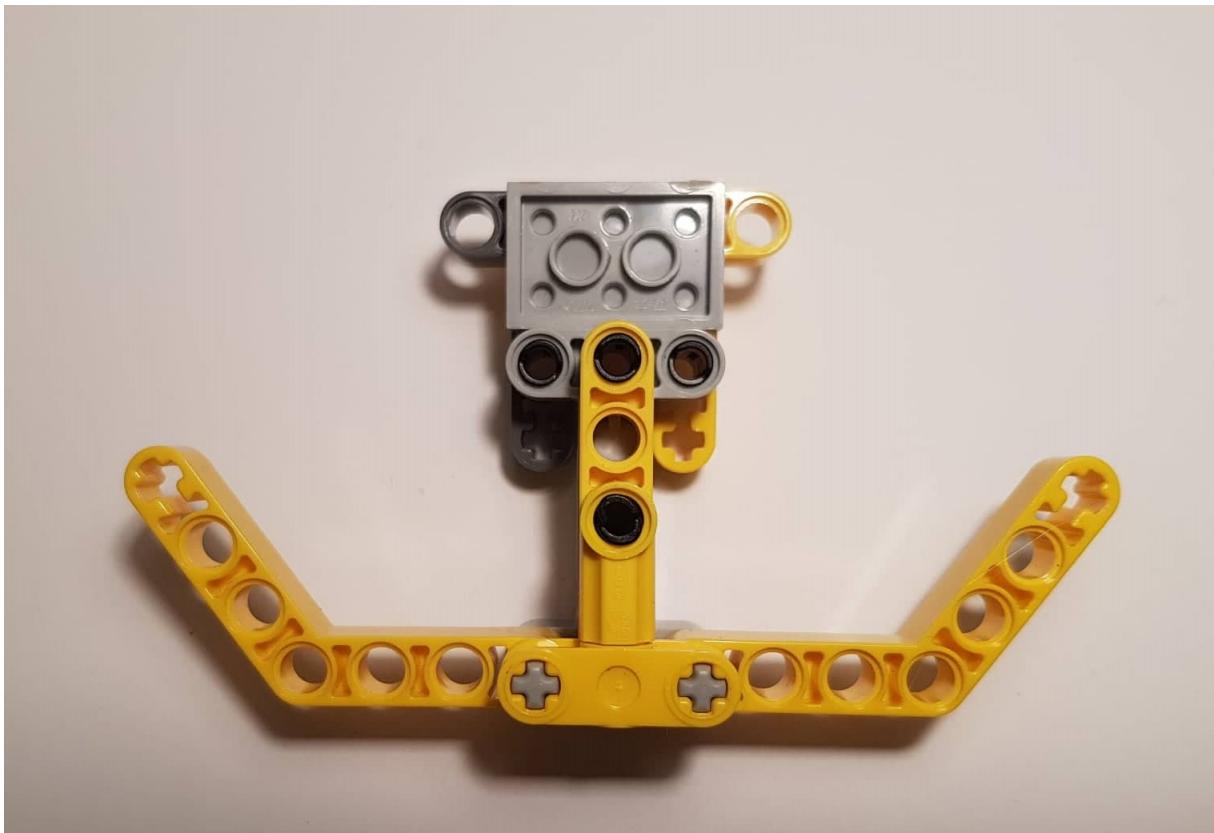
Pierścień rys.5 został zaprojektowany w programie Autodesk Inventor 2020. Górną część posiada miejsce na zamontowanie łożyska aby wykluczyć tarcie o przeszkody które powodowałoby wibracje i mogło doprowadzić do błędego odczytania nachylenie, pierścień mógłby się blokować pod innym kątem niż jest rzeczywiście robot ustawiony względem przeszkody. Cała konstrukcja została zaprojektowana tak aby oszczędzić filament i ułatwić drukowanie.



Rysunek 5: Projekt pierścienia

Jednakże takie rozwiązanie posiada poważną wadę, w momencie uderzenia o ścianę, robot siłą bezwładności mimo zatrzymania silników nie zatrzymuje się natychmiast, skutkuje to obróceniem się robota i to niezależnie od tego czy pierścień posiada łożysko czy nie. Aplikacja sterująca posiada możliwość aby wykryć nachylenie pierścienia i skorygować to odpowiednim obrotem, lecz jest to zbyt niedokładne i potrzebne było dodanie elementu który by pozycjonował robota podczas uderzenia.

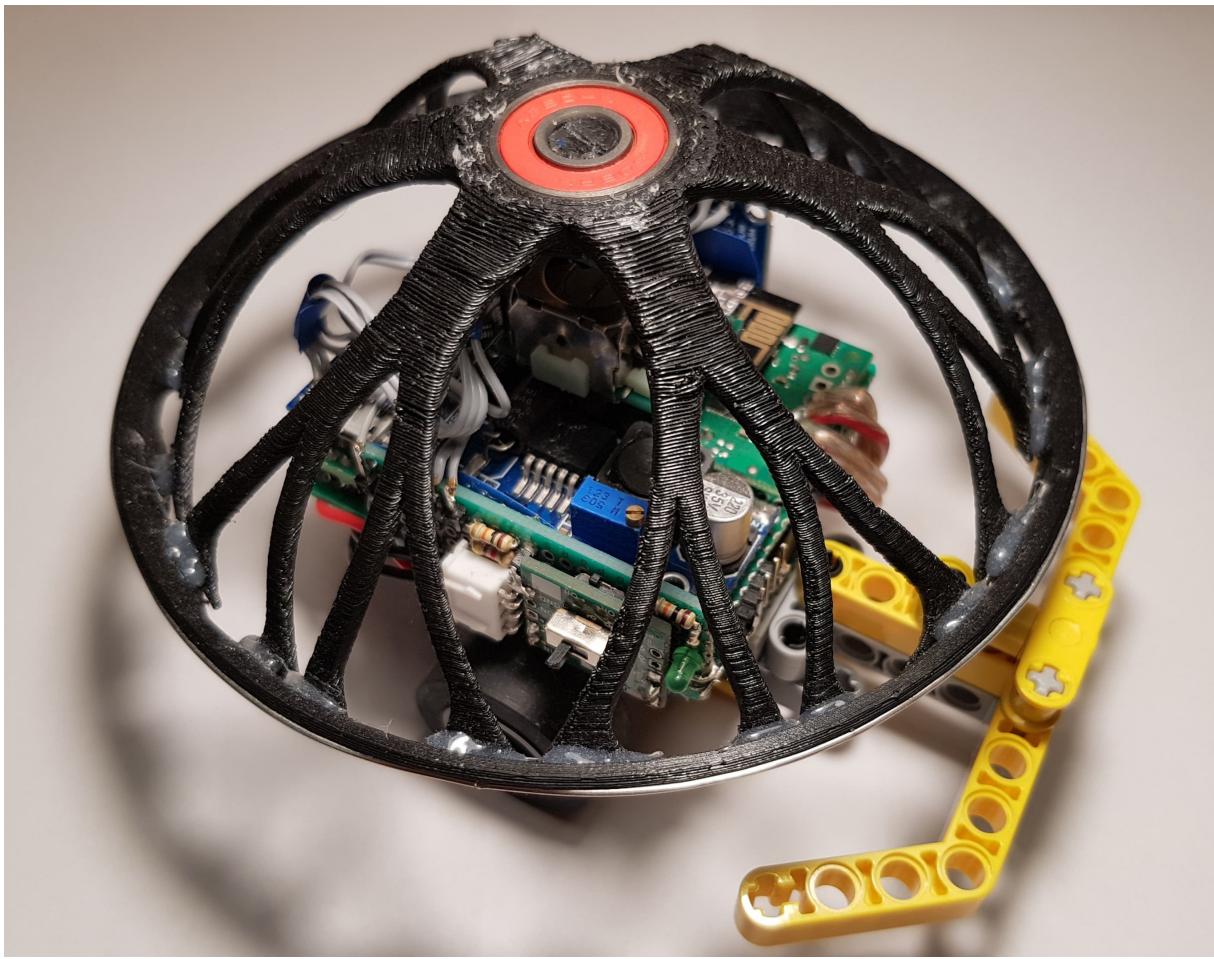
Dodano zderzak który podczas zderzenia mimowolnie robota pozycjonuje. Algorytm sterowania był jednak na tyle przystosowany do robota że usunięcie pierścienia nie wchodzi w grę, dodatkowo samo dodanie nowego elementu wymagało niewielkich korekcji ustawień algorytmu, mowa tutaj o regulatorze jazdy o zadaną odległość. Zderzak przedstawia rysunek 6



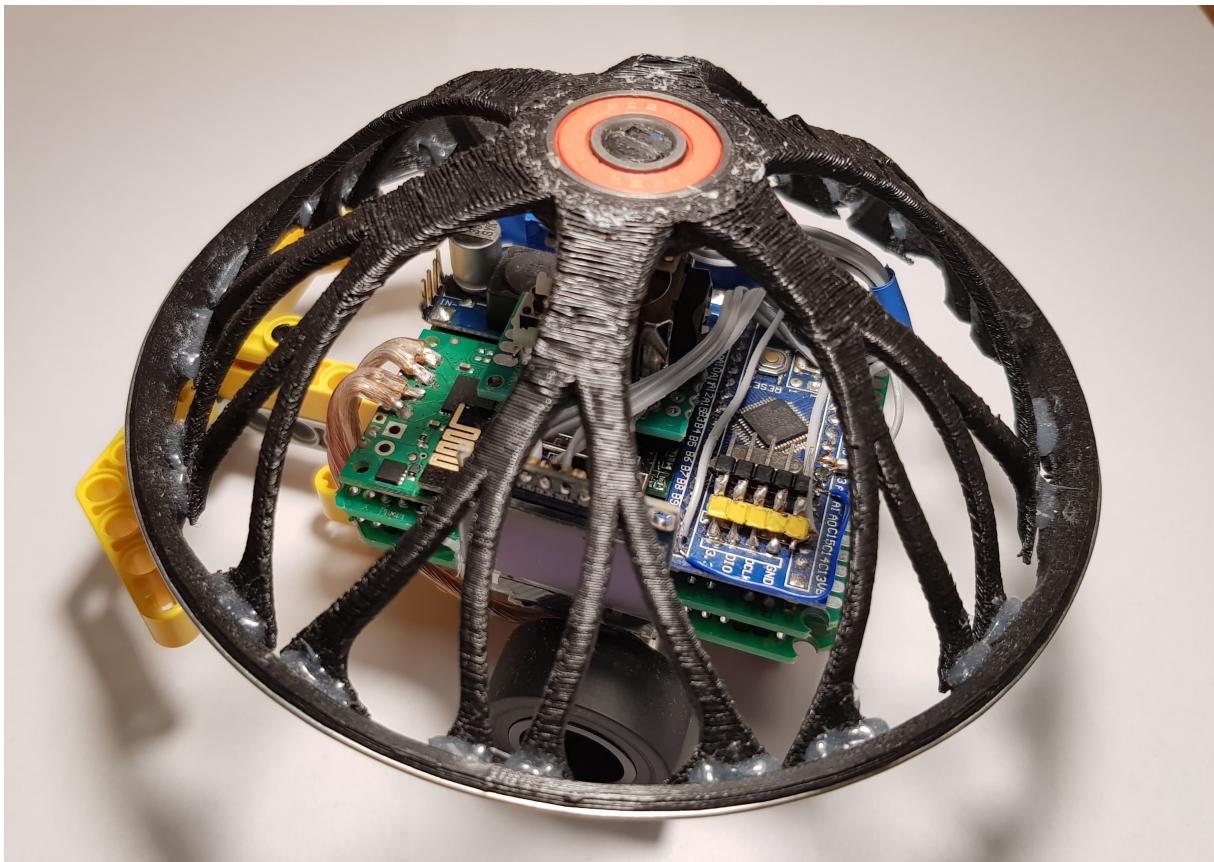
Rysunek 6: Zderzak pozycjonujący.

6 Wykonanie

Robot w większości zlutowany jest z płyt prototypowych, bateria mieści się w specjalnej kieszeni, pierścień umocowany jest na joysticku który również jest przylutowany. Elementy lego są przymocowane do płytki dolnej wraz z silnikami na klej na gorąco, cała konstrukcja jest solidna i przyjemna dla oka. Gotowego robota przedstawia zdjęcie 7 i 8



Rysunek 7: Robot



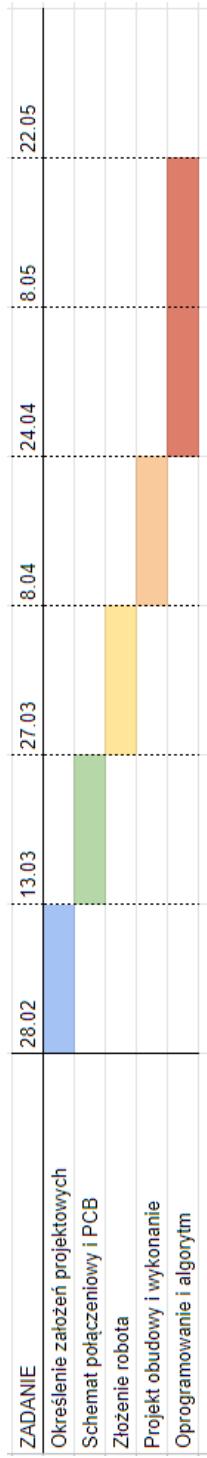
Rysunek 8: Robot

7 Harmonogram pracy

Wyróżnione zostały dwa kamienie milowe:

- Uruchomienie urządzenia
- Poprawna detekcja przeszkód

Uruchomienie urządzenia wiąże się z utworzeniem schematu i złożeniem robota w całość. Sam test peryferiów to pierwszy szczebel gdyż potem zostaje tylko kwestia programistyczna. Poprawne wykrywanie przeszkód to pierwszy krok do całego algorytmu, lecz zamyka on dział konfiguracji wszystkich modułów ze sobą.



Rysunek 9: Diagram Gantta

8 Podsumowanie

Robot będzie nowatorską konstrukcją, nie będzie on stworzony by zajmować pierwsze miejsca w zawodach Micromouse, lecz jest po to by zaspokoić ciekawość i chęć stawiania sobie wyzwań. Projekt ten będzie wymagał zaawansowanej interpretacji danych z akcelerometru i żyroskopu, specjalnego algorytmu [4] identyfikacji labiryntu i znalezienia najszybszej ścieżki aby labirynt ukończył.

Link do repozytorium GitHub: [link]

Prezentacja projektu pod adresem: [link]

Projekt JODO GitHub: [link]

Literatura

- [1] InvenSense. MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking Devices. Sier. 2013.
- [2] marcin13021988. Roboty MicroMouse – 5 metod przeszukiwania labiryntu. Sier. 2009.
- [3] Nordic Semiconductor. nRF24L01+ Single Chip 2.4Ghz Transciever. Preliminary Product Specification v1.0. Mar. 2008.
- [4] Robert PIOTROWSKI. Robot typu Micromouse – wykonanie, sterowanie i optymalizacja. Gru. 2014.