

PROJEKT

ROBOTY MOBILNE

Założenia projektowe

Robot gąsienicowy ReTank

Przygotował:
Marcin Bober, 249426
Termin: srTN18

Prowadzący:
mgr inż. Arkadiusz Mielczarek

Spis treści

1	Opis projektu	2
2	Bank energii	2
2.1	Ogniska	2
2.2	Balanser	2
3	Płytką drukowaną	3
3.1	Mikrokontroler	3
3.2	Sterowniki silników	4
3.3	Akcelereometr	4
3.4	Inne	5
4	Mechanika	5
4.1	Podwozie	5
4.2	Silniki	5
5	Harmonogram	6
6	Podwozie	6
7	Płytką drukowaną	7
8	Program	8
9	Komunikacja z robotem	8
9.1	Ping	9
9.2	Dystans przeszkody	9
9.3	Bateria	9
9.4	Prędkość	9
9.5	Moc silników	9
9.6	Akcelometr	9
9.7	Żyroskop	10
10	Aplikacja do komunikacji	10
11	Rezultat	11

1 Opis projektu

Projekt dotyczy budowy robota mobilnego, sterowanego zdalnie. Zadanie obejmuje zaprojektowanie części elektronicznej, oprogramowania na mikrokontroler i oprogramowania na urządzenie sterujące. W skład części elektronicznej wchodzi projekt płytka drukowana i banku energii.

2 Bank energii

2.1 Ogniwka

Zasilania robotowi zapewniać będą trzy ogniwka 16850. Aby minimalizować koszty, zostały one pozy-skane ze starej baterii z laptopa. Nie są one dobrze opisane co sprawia że nie jestem w stanie odnaleźć ich specyfikacji. Ich żywotność również jest nie znana, co uniemożliwia oszacowanie ich łącznej pojemności.



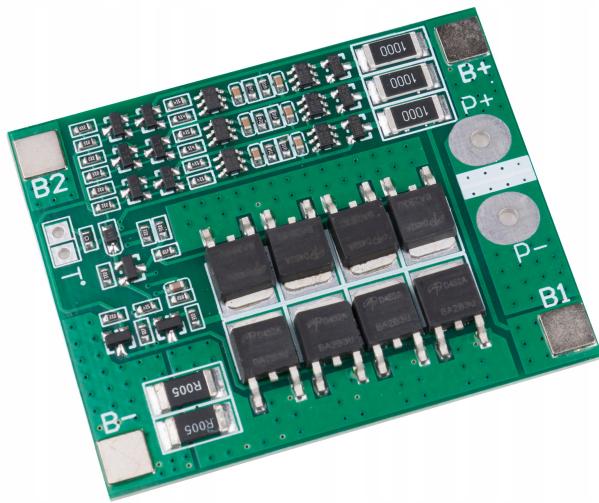
Rysunek 1: Ogniwka 16850

2.2 Balanser

Do zarządzania ogniwami użyty zostanie gotowy układ BMS przystosowany do współpracy z trzema ogniwami 16850. Zapewnia on podstawowe zabezpieczenia w tym:

- zabezpieczenie przed nadmiernym rozładowaniem poniżej 3.0V,
- zabezpieczenie przed przeładowaniem powyżej 4.25V,
- zabezpieczenie przed przeciążeniem prądowym,
- zabezpieczenie przed zwarciem.

Napięcie generowane przez baterię ogniw powinno oscylować w zakresie 9.0V - 12.7V



Rysunek 2: Balanser baterii

3 Płytki drukowane

Jednym z podstawowych elementów projektu jest stworzenie płytki drukowanej integrującej wszystkie wymagane elementy.

3.1 Mikrokontroler

Sercem układu zostanie układ ESP32-WROVER. Najważniejszym kryterium decydującym o wyborze tego konkretnego układu jest wbudowany moduł WiFi i jego niska cena oscylująca w okolicach 2.50\$. Jest to zdecydowanie najtańszy sposób na połączenia robota do Internetu.



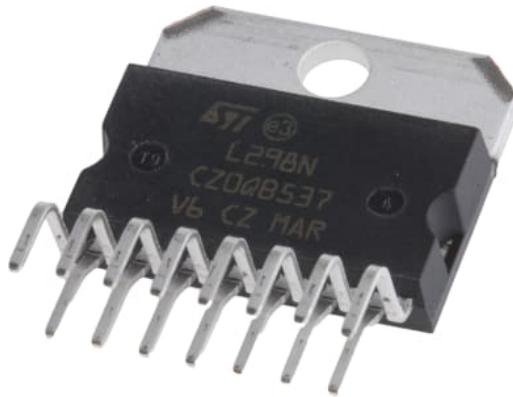
Rysunek 3: Mikrokontroler ESP32

Najważniejsze cechy:

- Wbudowany MCU o dwóch rdzeniach z taktowaniem do 240 MHz
- Protokoły Wi-Fi: 802.11 b/g/n/d/e/i/k/r
- Protokoły BT: Bluetooth v4.2 BR/EDR oraz Bluetooth BLE
- Napięcie zasilania: 2.2 V - 3.6 V
- Średni pobór prądu: 80 mA

3.2 Sterowniki silników

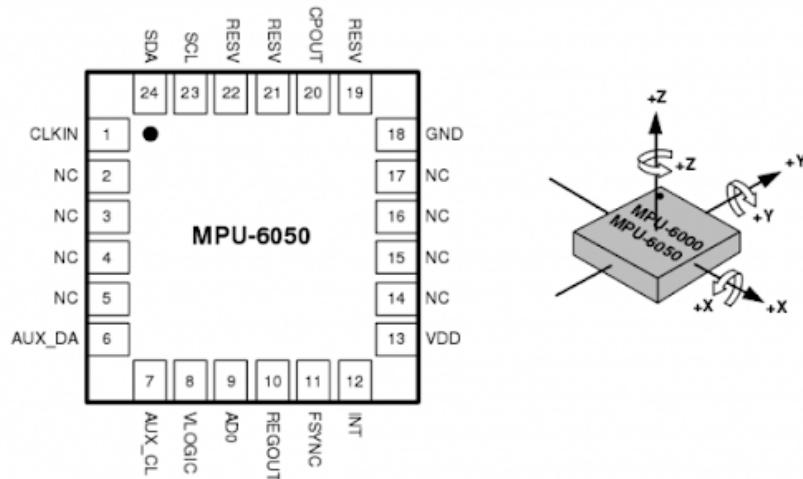
Układ będzie wspierał sterowanie dwoma silnikami prądu stałego. Każdy z nich będzie mógł być wyposażony w osobny enkoder liczący obroty silnika. Do tego celu posłuży układa L298N w obudowie Multiwatt 15 pin.



Rysunek 4: Układ scalony L298N

3.3 Akcelerometr

Robot będzie wyposażony w akcelerometr i żyroskop, aby móc mierzyć przyspieszenia oraz prędkości kątowe. Będzie to możliwe dzięki użyciu układu MPU-6050.



Rysunek 5: Układ scalony MPU6050

3.4 Inne

Aby móc komfortowo użytkować i prototypować robota niezbędne będą dodatkowe elementy takie jak:

- wyprowadzenie JATG,
- wyprowadzenie interfejsu I2C,
- wyprowadzenie kamery ArduCam OV2640,
- wyprowadzenie UART,
- wejście zasilania,
- złącze czujnika ultradźwiękowego,
- dwa wejścia enkoderów,
- dwa wyjścia na silniki.

4 Mechanika

4.1 Podwozie

Aby zredukować koszty i lepiej zintegrować wszystkie elementy ze sobą, postanowiłem samemu zaprojektować podwozie i wydrukować je przy użyciu drukarki 3D. Materiał który do tego użyję to ABS.

4.2 Silniki

Silniki których użyję są produkcji DFRobot. Mają one doczepione enkodery, które pozwolą mi bardzo dokładnie kontrolować prędkości silników.



Rysunek 6: Silnik z enkoderem

Ich skrócona specyfikacja:

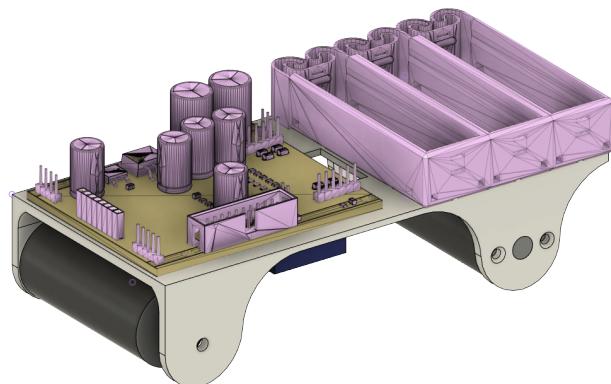
- Napięcie zasilania: 6 V
- Napięcie enkodera: od 3,3 V do 5 V
- Współczynnik redukcji: 1:20
- Prędkość bez obciążenia: 300 RPM - 0,1 A

5 Harmonogram

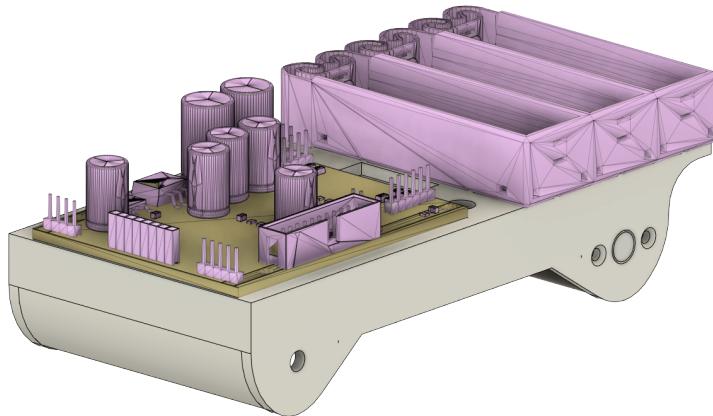
1. Sporządzenie listy i zdobycie wymaganych elementów. (17.03)
2. Zaprojektowanie i wykonanie podwozia. (24.03)
3. Zaprojektowanie i wykonanie pakietu baterii. (31.03)
4. Zaprojektowanie płytki drukowanej. (7.04)
5. Napisanie i przetestowanie bazowej wersji programu na MCU. (14.04)
6. Zbudowanie komunikacji z kontrolerem. (21.04)
7. Dodanie obsługi enkoderów. (28.04)
8. Dodanie obsługi żyroskopu. (5.05)
9. Testowanie i dopieszczanie. (12.05)

6 Podwozie

Podwozie zostało zaprojektowane w programie Fusion360. Pierwsza wersja cierpiąła na problemy związane z małą sztywnością konstrukcji, co znacząco wpływało na jakość sterowania. Z tego też powodu została zaprojektowana druga, znacznie wytrzymalsza wersja podwozia.



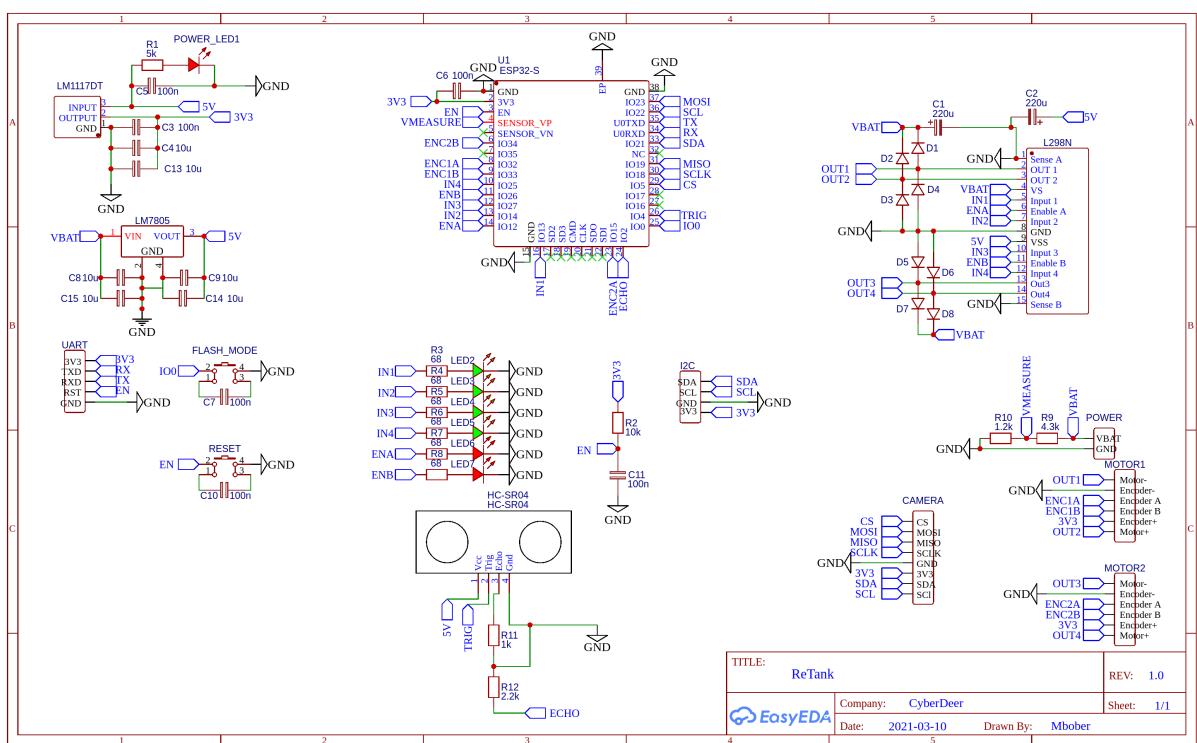
Rysunek 7: Pierwsza wersja podwozia



Rysunek 8: Druga wersja podwozia

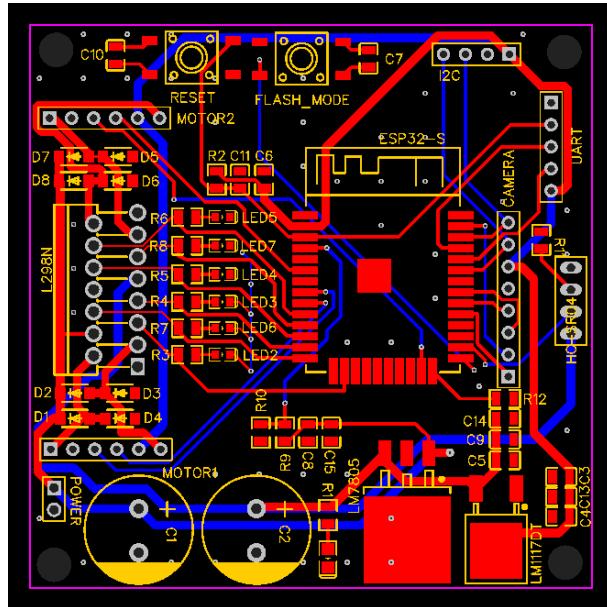
7 Płytki drukowane

W celu zwiększenia jakości i poprawy niezawodności projektu, została zaprojektowana płytka drukowana, która ma za zadanie integrować wszystkie niezbędne peryferia i minimalizować zajmowane miejsce.



Rysunek 9: Schemat połączeń

2021-04-16 zostało złożone zamówienie na 5 sztuk płyt. Zamówienie jest jeszcze w trakcie transportu. Ostateczną wersję można zobaczyć poniżej. Z powodu ograniczonej przestrzeni i niewystarczającej liczby pinów mikrokontrolera, zrezygnowałem z portu JTAG.



Rysunek 10: PCB

8 Program

Program na mikrokontroler został napisany z użyciem warstwy abstrakcji ESP-IDF, udostępnionej przez producenta układu. Dla zwiększenia możliwości sprzętu, zdecydowałem się na użycie systemu czasu rzeczywistego FreeRTOS. Dzięki temu kod został podzielony na oddzielne moduły, a każdy z nich pracuje na swoim odrębnym wątku. Pozwoliło to również ustawać wyższy priorytet funkcjom, które mają krytyczny wpływ na dzianie robota. Komunikacja międzyprocesowa została zrealizowana poprzez kolejki priorytetowe, co znacząco usprawnia cały proces.

Całość kodu dostępna jest w moim repozytorium na GitHub.

9 Komunikacja z robotem

Połączenie z robotem odbywa się poprzez sieć WiFi. W pierwszej kolejności nawiązywane jest połączenie przy użyciu protokołu TCP. Jeśli się ono powiedzie to uruchamiana jest dodatkowa transmisja z wykorzystaniem UDP. Dzięki takiej koncepcji mamy dwa niezależne kanały komunikacji. Pierwszy służy do przesyłania danych które mają niski priorytet czasowy, potrzebując potwierdzenia odebrania i ewentualnej retransmisji danych. Druga droga komunikacji powstała, aby przesyłać ciągły strumień nowych danych. Zależy nam na jaknajniższym opóźnieniu, a ewentualny błąd transmisji nie jest krytyczny, bo informacje te szybko się przedawniają i są zastępowane przez nowe, świeższe.

Każda ramka zaczyna się od wybranej dużej litery alfabetu, określającej rodzaj przesyłanych danych. Dla protokołu TCP są to:

- P - ping,
- D - dystans przeszkody,
- B - bateria,
- S - realna predkość.

Natomiast dla protokołu UDP wyróżniamy:

- E - moc silników,
- A - akcelometr,
- G - żyroskop.

Wszystkie paczki danych zakończone są średnikiem, przed którym znajduje się osmibitowy cykliczny kod nadmiarowy. Niestety ze względu na różnorodność transmitowanych informacji, w tym miejscu kończą się cechy wspólne poszczególnych ramek.

9.1 Ping

Jest to najprostrza z obecnych tu ramek. Nie przenosi żadnych informacji. Oznacza jedynie konieczność odesłania do nadawcy identycznej wiadomości, aby można było wyznaczyć chwilę czasowe, niezbędne do obliczenia opóźnienia transmisji.

Forma ramki to: $P\#;$
Gdzie $\#$ oznacza CRC.

9.2 Dystans przeszkodej

Przesyła informacje z robota o odległości odczytanej z czujnika ultradzwiekowego.

Forma ramki to: $D < \text{uint8_t} > \#;$

Gdzie $\#$ oznacza CRC. Przed nim znajduje się wartość odległości wyrażonej w centymetrach, w zakresie 0-100cm.

9.3 Bateria

Przesyła informacje z robota o poziomie baterii.

Forma ramki to: $B < \text{uint8_t} > \#;$

Gdzie $\#$ oznacza CRC. Przed nim znajduje się poziom baterii wyrażonej w procentach, w zakresie 0-100%.

9.4 Prędkość

Przesyła informacje z robota o prędkości na kołach.

Forma ramki to: $S < \text{uint8_t} > < \text{uint8_t} > \#;$

Gdzie $\#$ oznacza CRC. Przed nim znajdują się dwie wartości oddzielone spacją odnoszące się do prędkości poszczególnych kół wyrażonej w metrach na sekunde.

9.5 Moc silników

Przesyła informacje z aplikacji do robota o zadanej mocy silników.

Forma ramki to: $E < \text{uint8_t} > < \text{uint8_t} > \#;$

Gdzie $\#$ oznacza CRC. Przed nim znajdują się dwie wartości oddzielone spacją odnoszące się do zadanej mocy poszczególnych kół wyrażonej w procentach. Zakresy tych wartości muszą mieścić się od 0 do 100%

9.6 Akcelometr

Przesyła informacje z robota o aktualnych wskazaniach akcelometru.

Forma ramki to: $A < \text{uint8_t} > < \text{uint8_t} > < \text{uint8_t} > \#;$

Gdzie $\#$ oznacza CRC. Przed nim znajdują się trzy wartości oddzielone spacją odnoszące się do aktualnych wskazań akcelometru.

9.7 Żyroskop

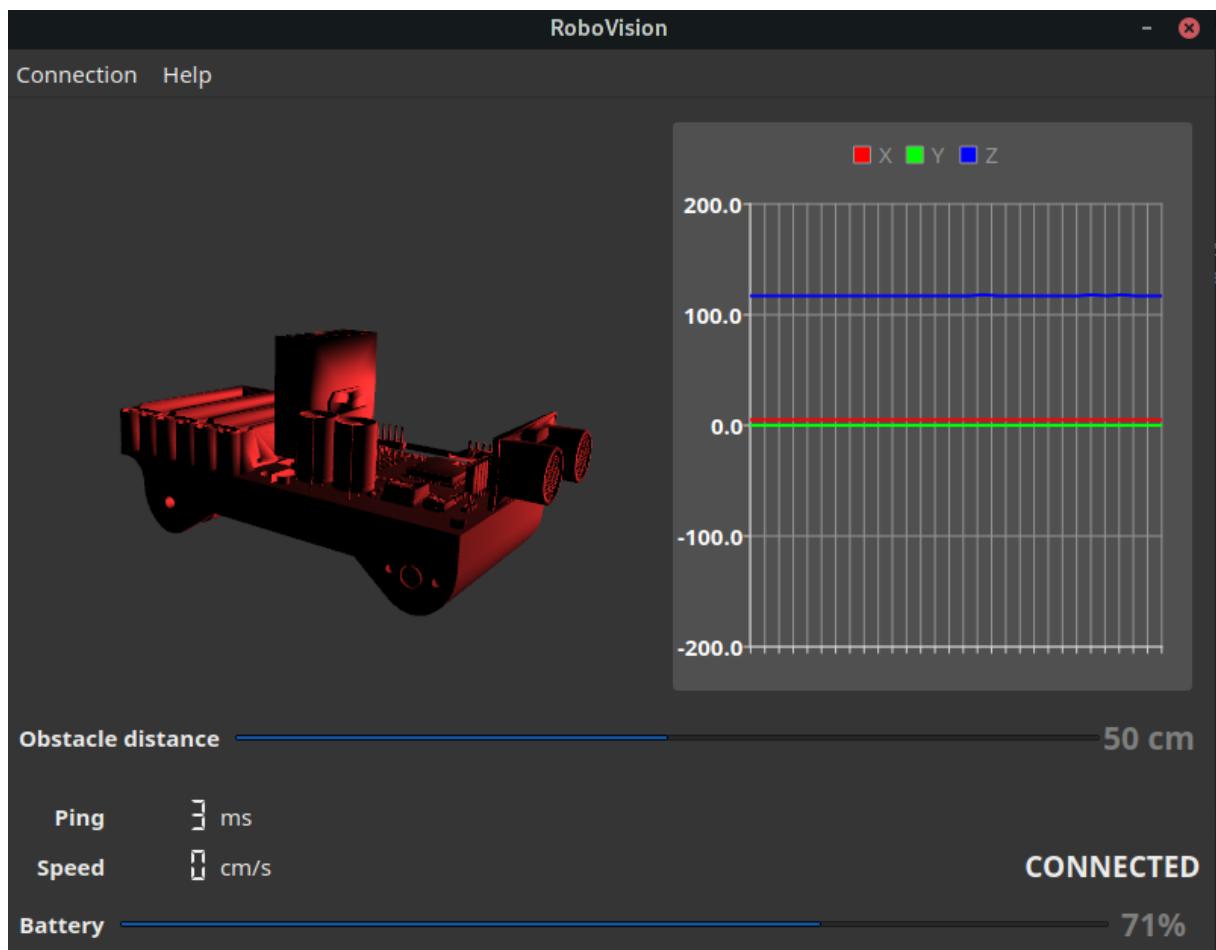
Przesyła informacje z robota o aktualnych wskazaniach żyroskopu.

Forma ramki to: $G < \text{uint8_t} >< \text{uint8_t} >< \text{uint8_t} > \#;$

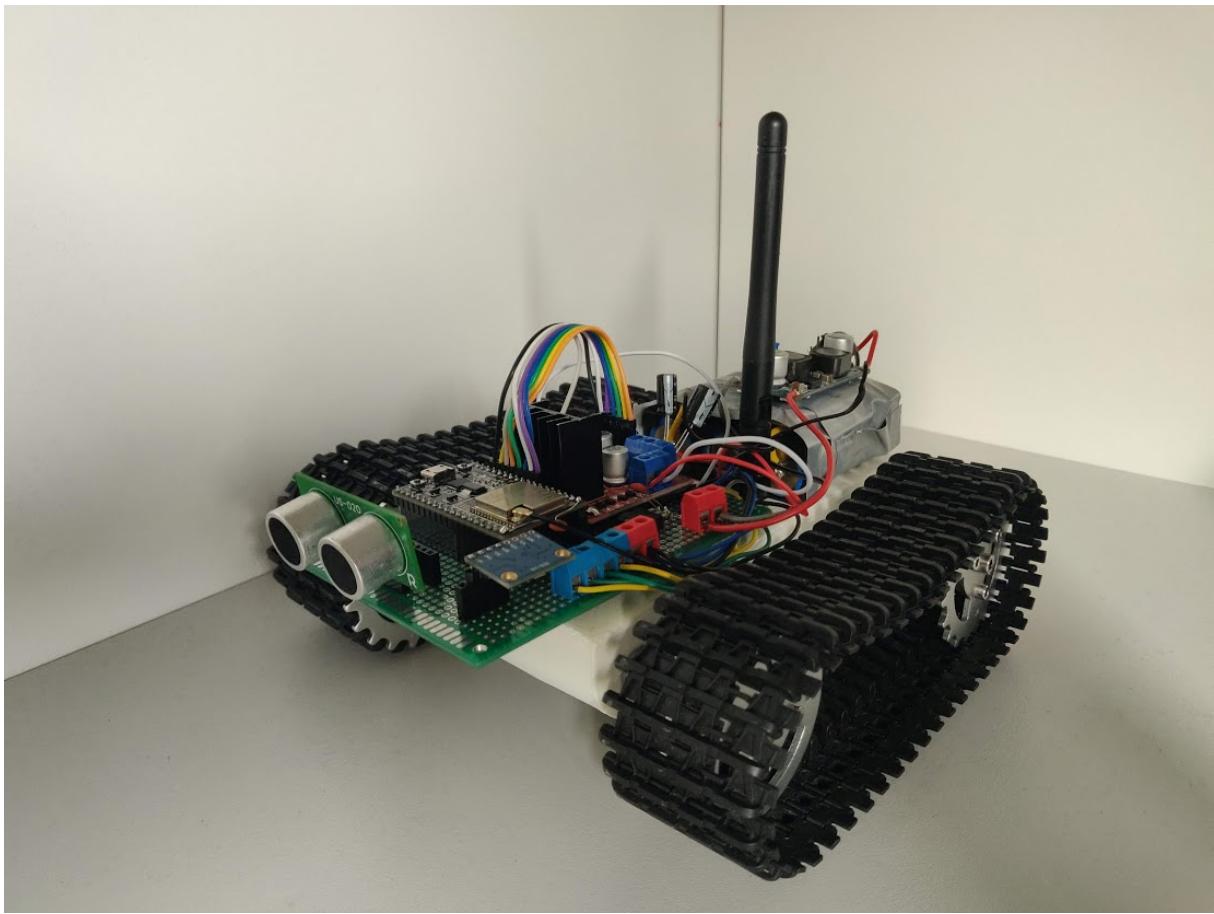
Gdzie $\#$ oznacza CRC. Przed nim znajdują się trzy wartości oddzielone spacją odnoszące się do aktualnych wskazań żyroskopu.

10 Aplikacja do komunikacji

Rezultaty komunikacji z komputerem i działania całego projektu możemy z łatwością podziwiać z pomocą mojej specialnie przygotowanej aplikacji w QT dostępnej w moim repozytorium na GitHub.



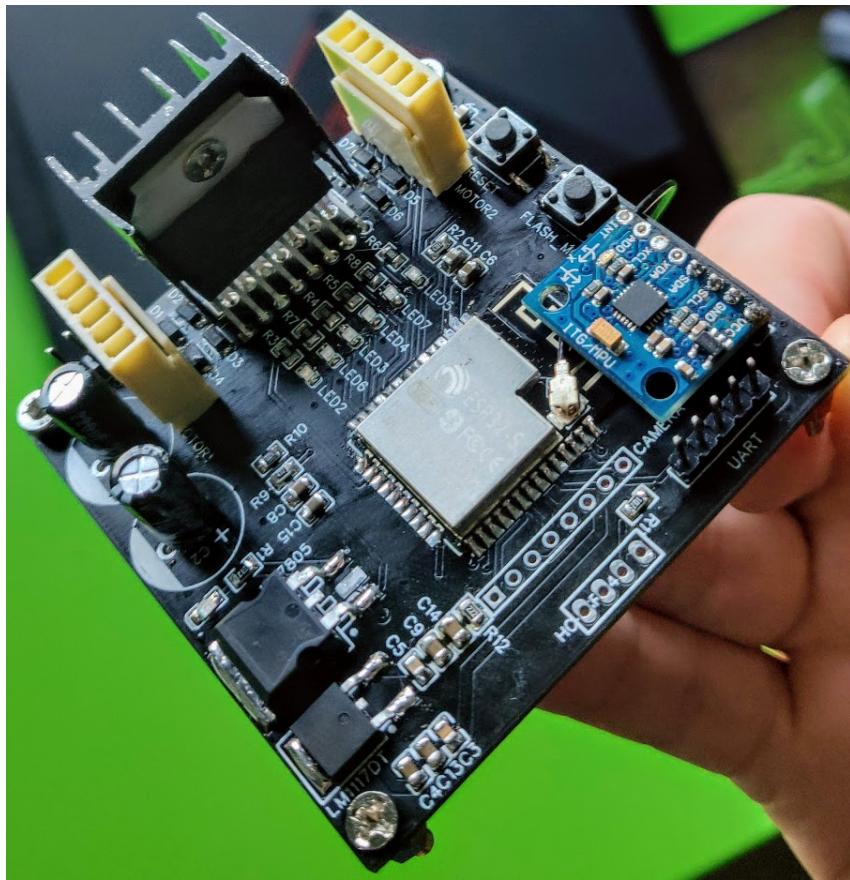
Rysunek 11: Aplikacja RoboVision

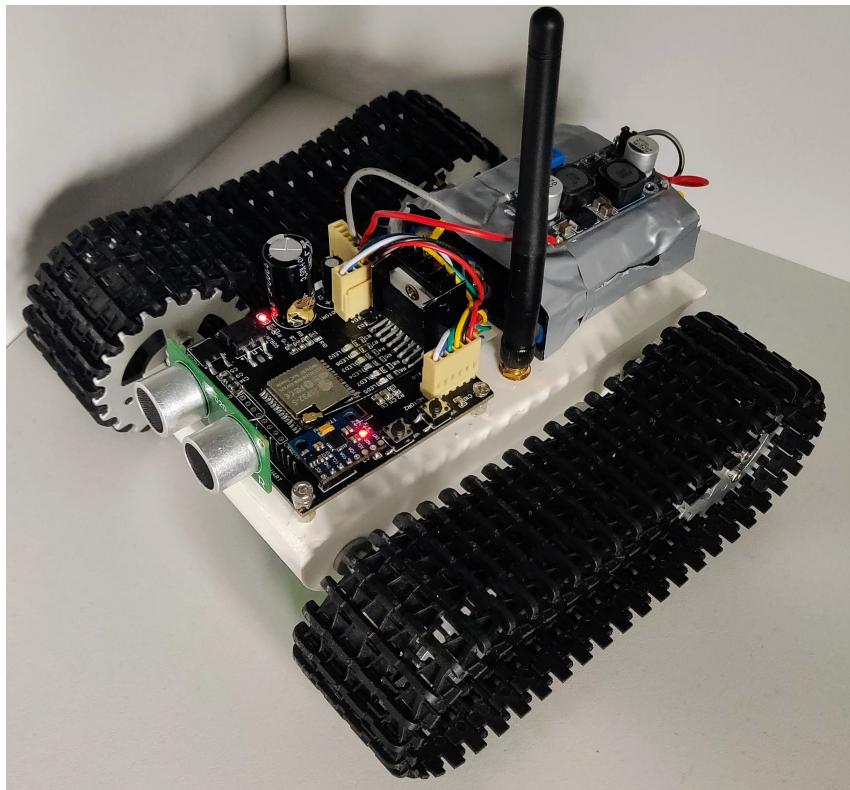


Rysunek 12: Prototyp robota

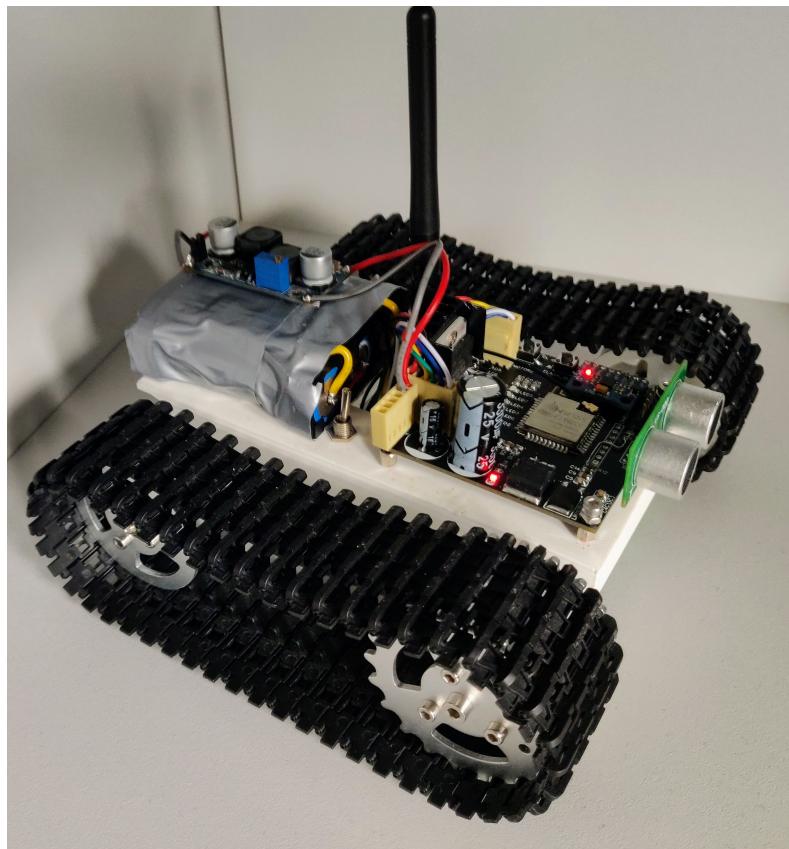
11 Rezultat

Projekt robota został zakończony sukcesem. Wszystkie problemy z jakimi borykały się prototypy zostały wyeliminowane. Można więc uznać projekt za zamknięty.

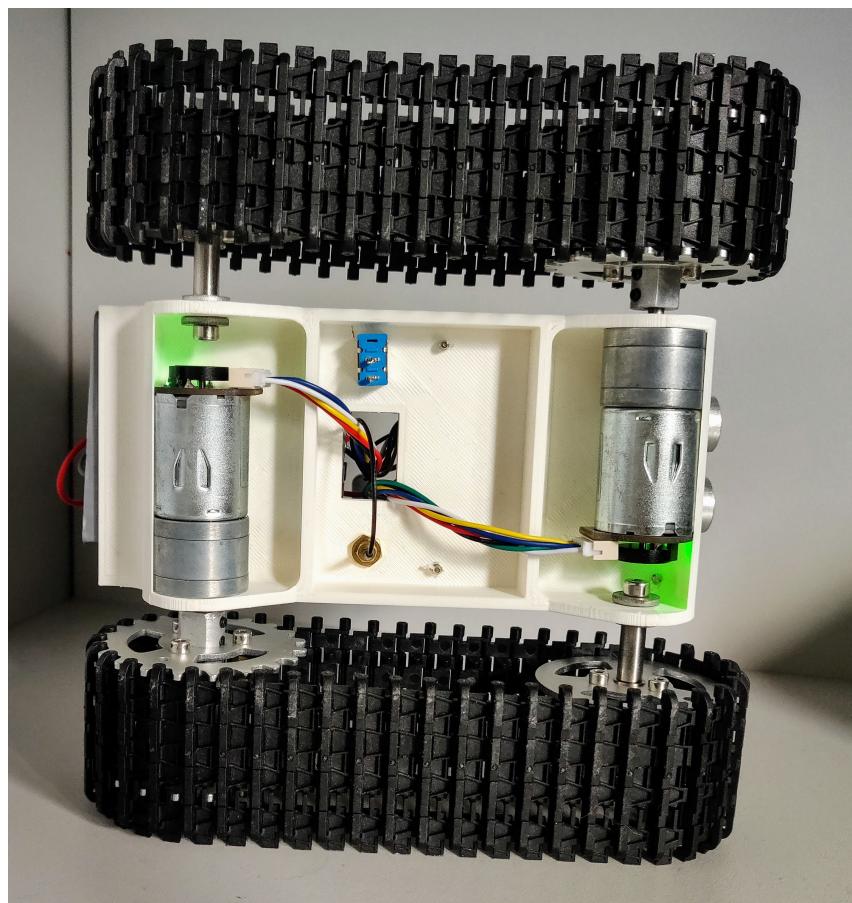




Rysunek 15: Ostateczna wersja robota



Rysunek 16: Ostateczna wersja robota



Rysunek 17: Ostateczna wersja robota