

PROJEKT

ROBOTY MOBILNE

---

## Dokumentacja

# Robot mobilny typu MicroMouse „Squick”

---

*Skład grupy:*

Piotr GORZELNIK, 248947

Patryk SZYDLIK, 248949

*Termin:* środa 18:00

*Prowadzący:*

mgr inż. Arkadiusz MIELCZAREK

1 lipca 2021

# Spis treści

<b>1</b>	<b>Opis projektu</b>	<b>2</b>
<b>2</b>	<b>Założenia Projektowe</b>	<b>2</b>
2.1	Elementy do stworzenia konstrukcji robota . . . . .	2
2.2	Możliwości rozwoju konstrukcji w przyszłości . . . . .	2
<b>3</b>	<b>Konfiguracja mikrokontrolera</b>	<b>3</b>
3.1	Konfiguracja pinów . . . . .	5
3.2	USART1 . . . . .	6
3.3	USART6 . . . . .	6
3.4	I2C1 . . . . .	6
3.5	SPI1 . . . . .	7
3.6	TIM1 . . . . .	7
3.7	TIM2 . . . . .	7
<b>4</b>	<b>Urządzenia zewnętrzne</b>	<b>8</b>
4.1	IMU LSM9DS1 . . . . .	8
4.2	VL53L0X . . . . .	8
4.3	Moduł BT HC-06 . . . . .	9
4.4	Enkodery magnetyczne Pololu . . . . .	9
<b>5</b>	<b>Projekt elektroniki</b>	<b>10</b>
<b>6</b>	<b>Konstrukcja mechaniczna</b>	<b>11</b>
6.1	Model 3D . . . . .	11
6.2	Rzeczywista konstrukcja . . . . .	12
<b>7</b>	<b>Opis działania programu</b>	<b>14</b>
7.1	Rdzeń programu . . . . .	14
7.2	Obsługa czujników i sterowników . . . . .	14
7.2.1	Obsługa enkoderów . . . . .	14
7.2.2	Kalibracja PID kontrolującego silniki . . . . .	16
7.2.3	Obsługa pomiarów ADC . . . . .	18
7.2.4	Multiplekser i czujnik IR . . . . .	19
7.3	Opis algorytmu eksploracji labiryntu (w trakcie implementacji) . . . . .	19
<b>8</b>	<b>Napotkane problemy</b>	<b>20</b>
8.1	enkodery . . . . .	20
8.2	czujniki odległości . . . . .	20
<b>9</b>	<b>Zadania niezrealizowane</b>	<b>20</b>
9.1	Autonomia . . . . .	21
9.2	Kompletna obsługa czujników odległości . . . . .	21
<b>10</b>	<b>Podsumowanie</b>	<b>21</b>
<b>11</b>	<b>Odnosińniki do repozytorium GIT Bitbucket</b>	<b>21</b>

# 1 Opis projektu

Tworzonym przez nas projektem jest robot mobilny klasy micromouse. Będzie on wyposażony w dwukołowy napęd różnicowy (robot mobilny klasy 2.0). Robot będzie korzystał z fuzji czujników odbiciowych time-of-flight oraz czujników stworzonych z nadajników IR oraz fototranzystorów w celu optymalnego rozpoznawania przeszkód (ścian labiryntu). Do orientacji w labiryncie robot będzie wykorzystywał enkodery oraz IMU. Oprócz trybu autonomicznego, robotem będzie także można sterować zdalnie przy użyciu prostej aplikacji mobilnej. Projekt planujemy w przyszłości wykorzystać do wystartowania w zawodach robotycznych. Założenia konstrukcyjne oraz funkcjonalność robota zostały stworzone w oparciu o regulamin konkurencji Micromouse 16x16 odbywającej się na międzynarodowych zawodach robotycznych Robotic Arena.

## 2 Założenia Projektowe

- Użycie mikrokontrolera z rodziny STM32F4
- Stworzenie fuzji dalmierzy laserowych VL oraz ręcznie robionych czujników
- Wykorzystanie enkoderów oraz regulatora PID do sterowania prędkościami silników
- Wykorzystanie IMU oraz modułu Bluetooth
- Pozycjonowanie się w centrum korytarza labiryntu przez regulator PID z czujników odległości
- Obsługa interfejsów komunikacyjnych I2C, SPI, UART
- Konstrukcja mieszcząca się w kwadracie o boku 100mm
- Nisko położony środek ciężkości

### 2.1 Elementy do stworzenia konstrukcji robota

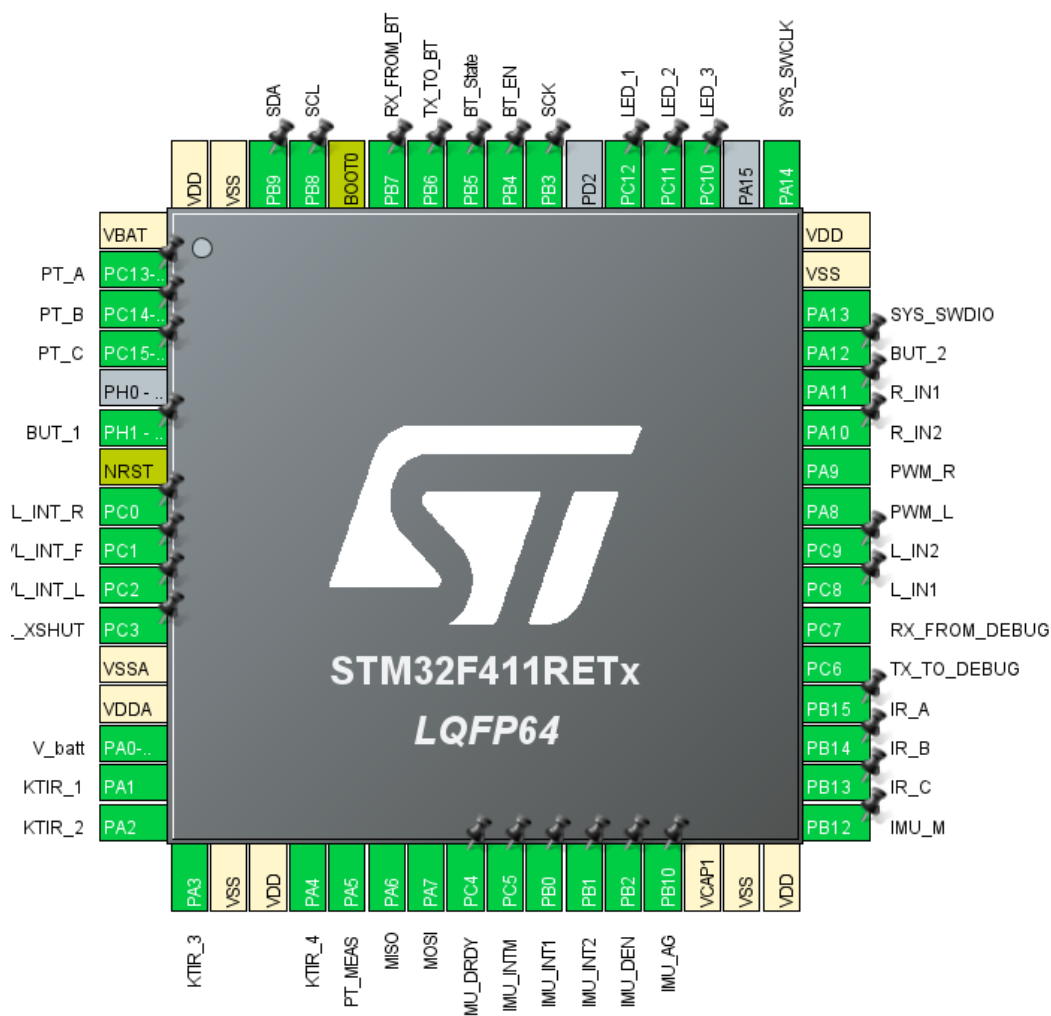
- Mikrokontroler STM32F411RET6 w obudowie LQFP 64
- Czujnik odległości time of flight VL53L0X
- Enkodery magnetyczne do silników pololu micro
- Silniki Pololi HPCB 50:1 z obustronnym wałem
- Moduł komunikacji BT
- LSM9DS1 - 9DoF IMU - 3-osiowy akcelerometr, magnetometr i żyroskop
- Nadajniki IR oraz Fototranzystory
- Dodatkowe przyciski i przełączniki dla użytkownika
- Diody LED do sygnalizacji
- Multiplexer

### 2.2 Możliwości rozwoju konstrukcji w przyszłości

- Wykorzystanie systemu FreeRTOS
- Opracowanie filtr Kalmana z IMU
- Wykorzystanie MicroPython-a na mikrokontrolerze
- Wykorzystanie w pełni możliwości FPU na STM32F4
- Rozwinięcie algorytmów eksploracji labiryntu
- Rozwinięcie algorytmów odnajdywania najkrótszej ścieżki

### 3 Konfiguracja mikrokontrolera

Do projektu wybrano mikrokontroler STM32F411RET6 w obudowie LQFP64. Wybór podyktowany został tym, że posiada on wystarczającą moc obliczeniową, ilość potrzebnych peryferiów oraz pamięć flash. Dodatkowo ważnym aspektem była kompatybilność mikrokontrolera z systemami FreeRTOS oraz MicroPython. Na zdjęciu poniżej umieszczono wstępną konfigurację pinów kontrolera, ostateczna struktura może ulec zmianie.



Rysunek 1: Konfiguracja wyjść mikrokontrolera w programie STM32CubeMX



### 3.1 Konfiguracja pinów

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
2	PC13	GPIO_Output	PT_A
3	PC14	GPIO_Output	PT_B
4	PC15	GPIO_Output	PT_C
6	PH1	GPIO_Input	BUT_1
8	PC0	GPIO_EXTI0	VL_INT_R
9	PC1	GPIO_EXTI1	VL_INT_F
10	PC2	GPIO_EXTI2	VL_INT_L
11	PC3	GPIO_Output	VL_XSHUT
14	PA0	ADC1_IN0	V_batt
15	PA1	ADC1_IN1	KTIR_1
16	PA2	ADC1_IN2	KTIR_2
17	PA3	ADC1_IN3	KTIR_3
20	PA4	ADC1_IN4	KTIR_4
21	PA5	ADC1_IN5	PT_MEAS
22	PA6	SPI1_MISO	MISO
23	PA7	SPI1_MOSI	MOSI
24	PC4	GPIO_Input	IMU_DDRDY
25	PC5	GPIO_Input	IMU_INTM
26	PB0	GPIO_Input	IMU_INT1
27	PB1	GPIO_Input	IMU_INT2
28	PB2	GPIO_Output	IMU_DEN
29	PB10	GPIO_Output	IMU_AG
33	PB12	GPIO_Output	IMU_M
34	PB13	GPIO_Output	IR_C
35	PB14	GPIO_Output	IR_B
36	PB15	GPIO_Output	IR_A
37	PC6	USART6_TX	TX_TO_DEBUG
38	PC7	USART6_RX	RX_FROM_DEBUG
39	PC8	GPIO_Output	L_IN1
40	PC9	GPIO_Output	L_IN2
41	PA8	TIM1_CH1 (PWM)	PWM_L
42	PA9	TIM1_CH2 (PWM)	PWM_R
43	PA10	GPIO_Output	R_IN2
44	PA11	GPIO_Output	R_IN1
45	PA12	GPIO_Input	BUT_2
46	PA13	SYS_SWDIO	SYS_SWDIO
49	PA14	SYS_SWCLK	SYS_SWCLK
51	PC10	GPIO_Output	LED_3
52	PC11	GPIO_Output	LED_2
53	PC12	GPIO_Output	LED_1
55	PB3	SPI1_SCK	SCK
56	PB4	GPIO_Output	BT_EN
57	PB5	GPIO_Input	BT_State
58	PB6	USART1_TX	TX_TO_BT
59	PB7	USART1_RX	RX_FROM_BT
61	PB8	I2C1_SCL	SCL
62	PB8	I2C1_SDA	SDA

Tabela 1: Konfiguracja pinów mikrokontrolera

### 3.2 USART1

Wykorzystany do komunikacji z modulem Bluetooth.

Parametr	Wartość
Baud Rate	11520
Word Length	8 Bits (including parity)
Parity	None
Stop Bits	1
Data Direction	Receive and Transmit
Over sampling	16 Samples

Tabela 2: Konfiguracja peryferium USART1

### 3.3 USART6

Dodatkowy interfejs UART do komunikacji z osobnym debugerem.

Parametr	Wartość
Baud Rate	11520
Word Length	8 Bits (including parity)
Parity	None
Stop Bits	1
Data Direction	Receive and Transmit
Over sampling	16 Samples

Tabela 3: Konfiguracja peryferium USART6

### 3.4 I2C1

Interfejs I2C do komunikacji z czujnikami VL

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 4: Konfiguracja peryferium I2C1

### 3.5 SPI1

Interfejs SPI do komunikacji z czujnikiem IMU (akcelerometr, żyroskop, magnetometr)

Parametr	Wartość
Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First
Prescaler	16
Baud Rate	3 MBits/s
Clock Polarity	Low
Clock Phase	1 Edge
CRC Calculation	Disabled
NSS Signal Type	Software

Tabela 5: Konfiguracja peryferium SPI1

### 3.6 TIM1

To peryferium jest wykorzystywane do zadawania sygnału PWM na silniki

Parametr	Wartość
I2C Speed Mode	Standard Mode
Prescaler	4
Counter Mode	Up
Counter Period	999
Internal Clock Division	No Division
Master/Slave Mode	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
PWM Generation Channel 1:	
Mode	PWM mode1
Pulse	0
Fast Mode	Disable
CH Polarity	High
PWM Generation Channel 2:	
Mode	PWM mode1
Pulse	0
Fast Mode	Disable
CH Polarity	High

Tabela 6: Konfiguracja peryferium TIM1

### 3.7 TIM2

Ten Timer służy do wywoływania przerwań z częstotliwością 50Hz

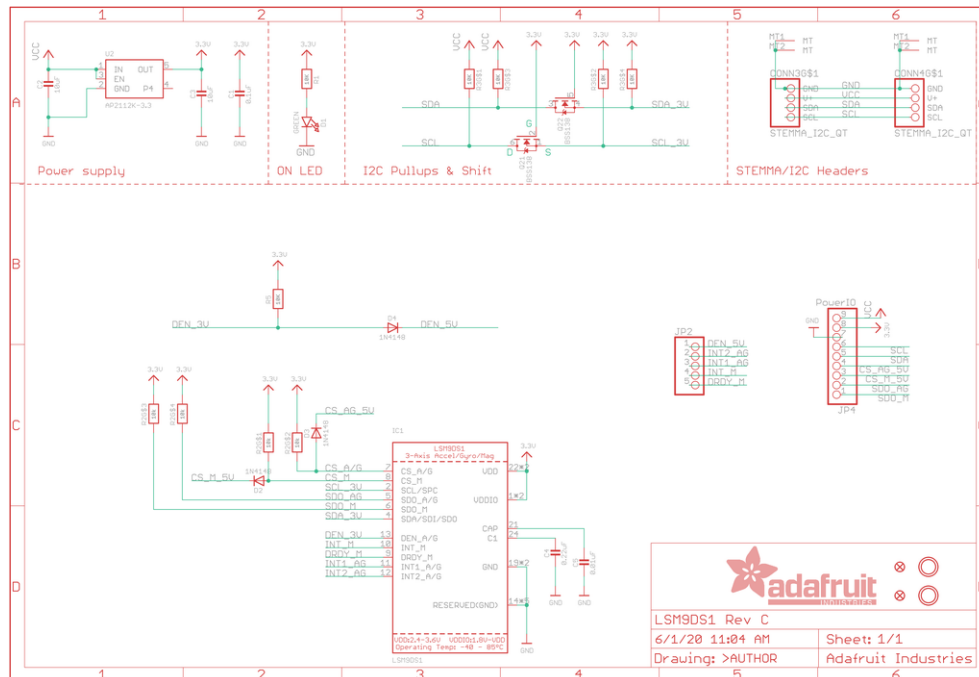
Parametr	Wartość
I2C Speed Mode	Standard Mode
Prescaler	96
Counter Mode	Up
Counter Period	9999
Internal Clock Division	No Division
Auto-reload Preload	No Disable
Master/Slave Mode	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)

Tabela 7: Konfiguracja peryferium TIM2



## 4 Urządzenia zewnętrzne

### 4.1 IMU LSM9DS1

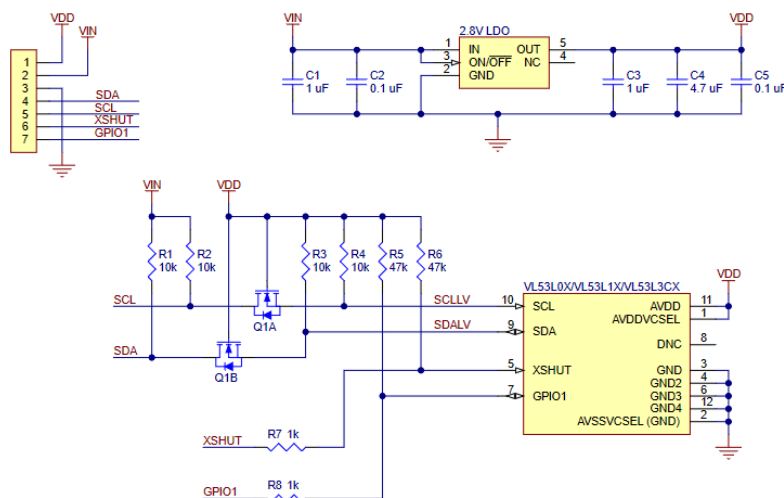


Rysunek 3: Schemat modułu IMU

źródło: [alldatasheet.com](http://alldatasheet.com)

Układ ten zawiera w sobie akcelerometr i żyroskop, które posłużą do prawidłowej analizy pozycji robota. Komunikacja z modułem odbywać się będzie po magistrali SPI.

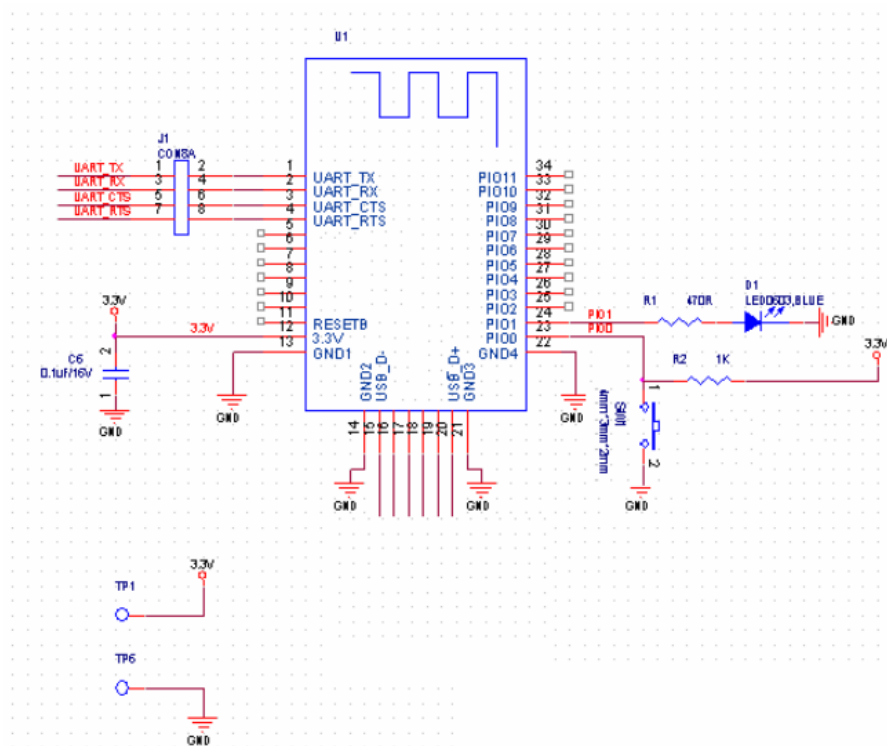
### 4.2 VL53L0X



Rysunek 4: Schemat modułu czujników odległości

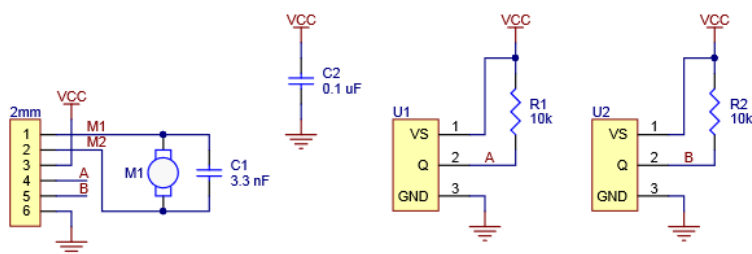
źródło: [alldatasheet.com](http://alldatasheet.com)

### 4.3 Moduł BT HC-06



Moduł ten posłuży do komunikacji bezprzewodowej robota z komputerem i umożliwi łatwe debugowanie podczas pisania i testowania kodu programu. Komunikacja z modułem następuje poprzez interfejs UART.

#### 4.4 Enkodery magnetyczne Pololu

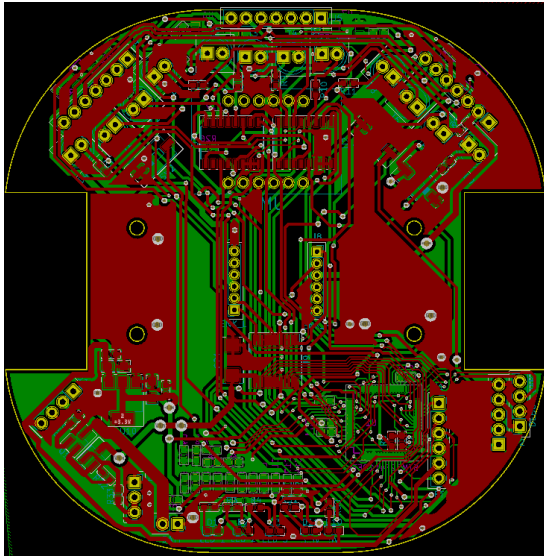


Rysunek 6: Schemat modułu enkoderów magnetycznych  
źródło: *pololu.com*

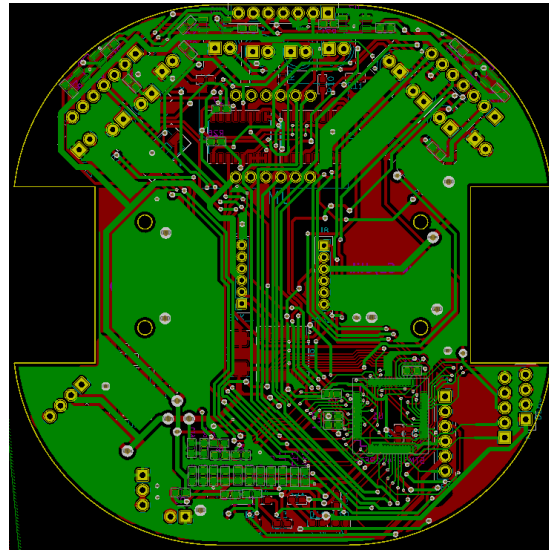
Enkodery magnetyczne wykorzystane zostaną do pomiaru prędkości oraz pozycji obrotowej kół. Zamocowane są one do przedłużonego wału silnika Pololu HCPCB.

## 5 Projekt elektroniki

Schemat całej elektroniki robota znajduje się w załączniku do tego raportu (str. 21). Zarówno schemat elektroniki jak i projekt PCB wykonaliśmy w programie KiCad. Ponieważ płytką drukowaną będzie jednocześnie pełnić rolę podwozia, staraliśmy się w miarę możliwości zrobić ją jak najmniejszą, aby zmaksymalizować manewrowalność robota w labiryncie. Z tego też powodu płytką jest dwuwarstwowa.

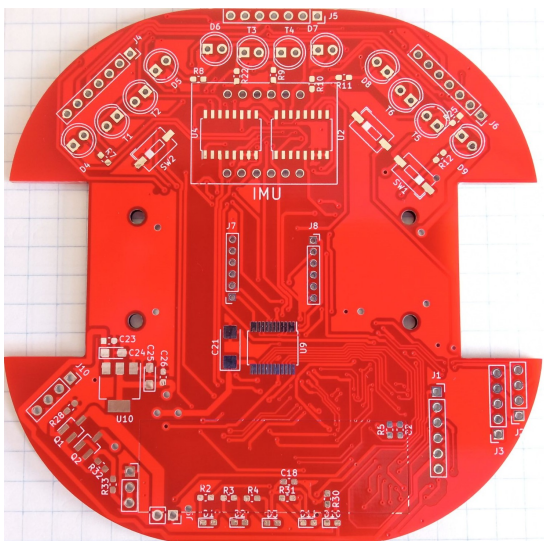


((a)) Górna warstwa płytki

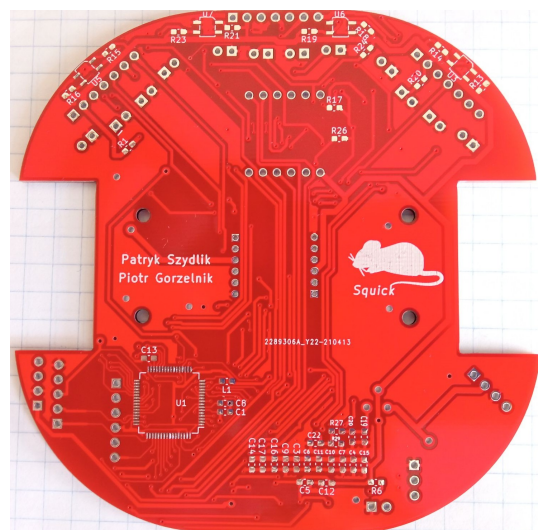


((b)) Dolna warstwa płytki

Zdecydowaliśmy się zlecić produkcję płytki w zewnętrznej firmie, aby uniknąć niedoskonałości jakie powstają przy ręcznym wytrawianiu a budżet pozwalał nam na taką wygodę. Zamówienie już do nas dotarło i przewidujemy, że do końca kwietnia ukończymy konstrukcję robota. Poniżej znajdują się zdjęcia gotowych, niepolutowanych płytek.



((a)) Górna warstwa płytki

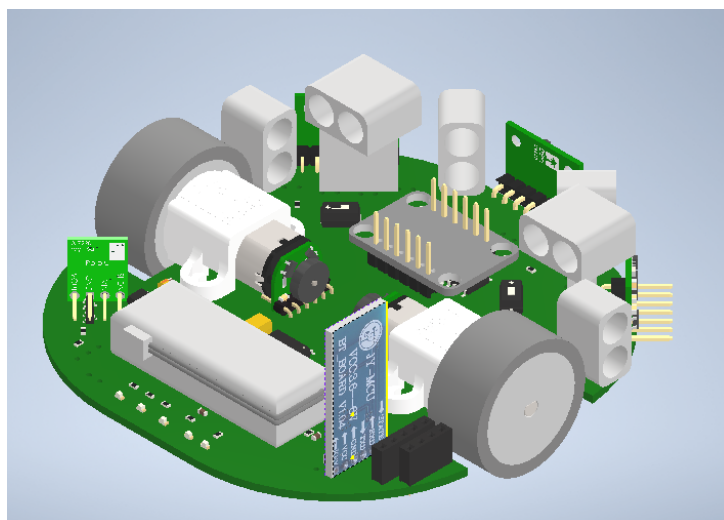


((b)) Dolna warstwa płytki

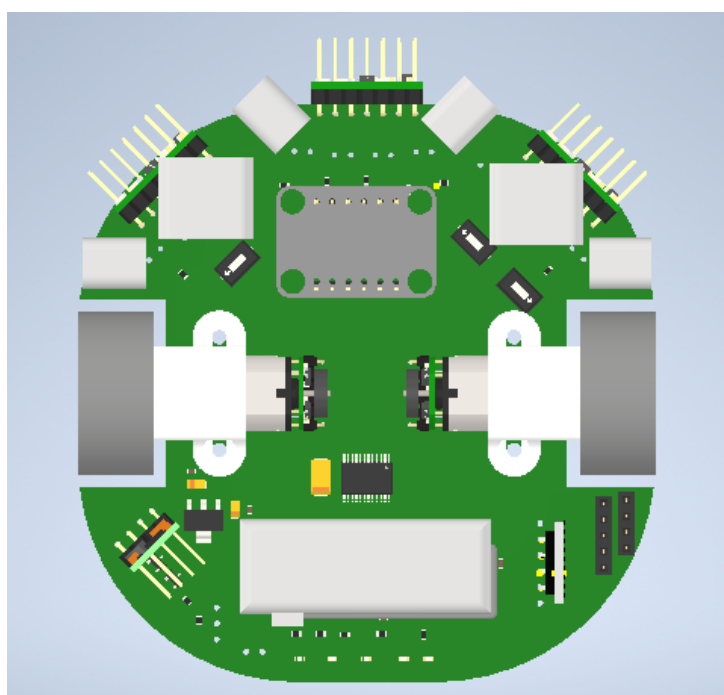
## 6 Konstrukcja mechaniczna

### 6.1 Model 3D

Trójwymiarowy model robota powstał w programie AutoCAD Inventor.

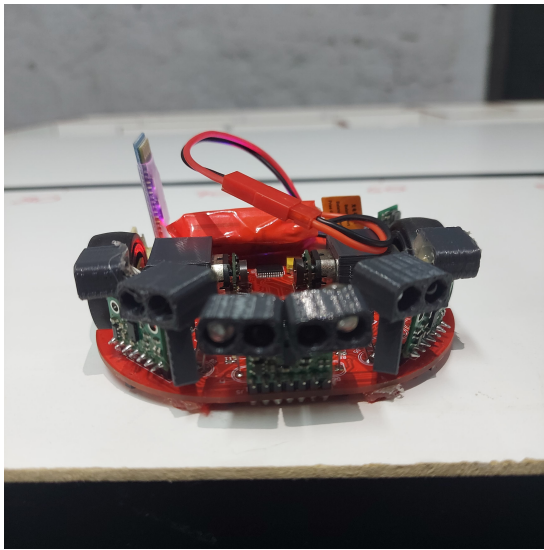


Rysunek 7: Model konstrukcji robota

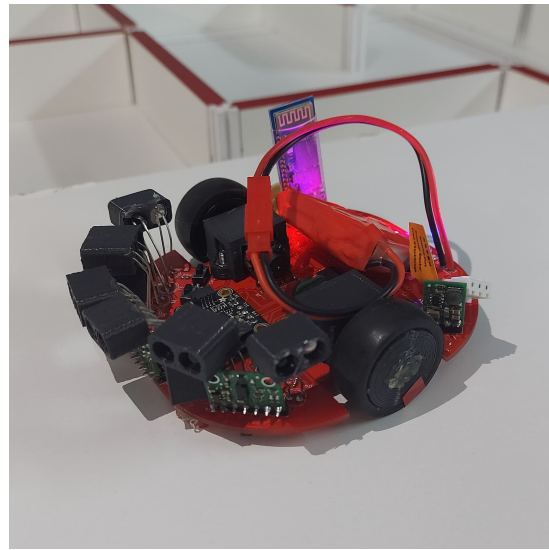


Rysunek 8: Rzut z góry modelu robota

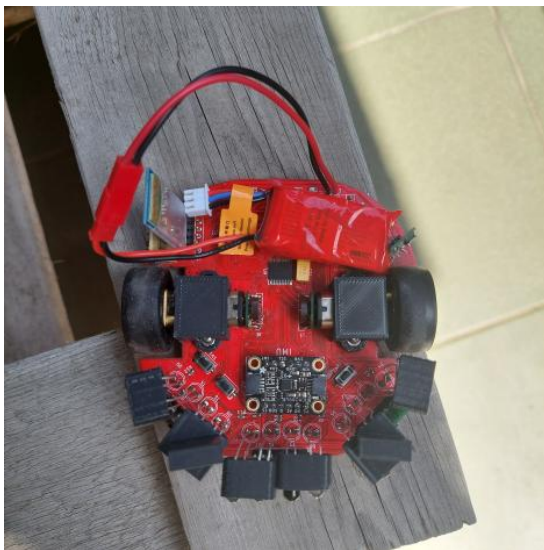
## 6.2 Rzeczywista konstrukcja



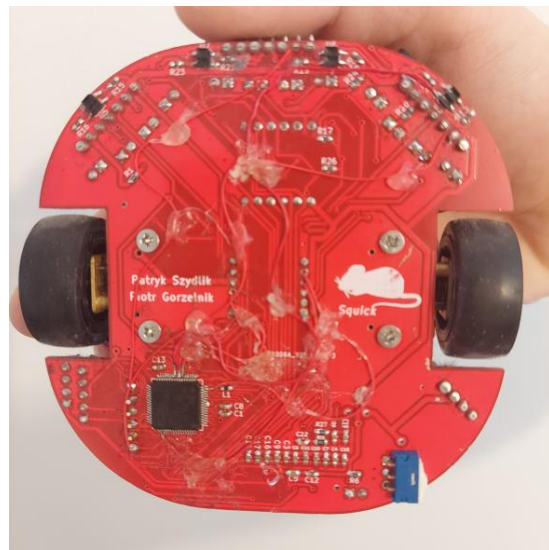
((a)) Widok z przodu



((b)) widok z boku

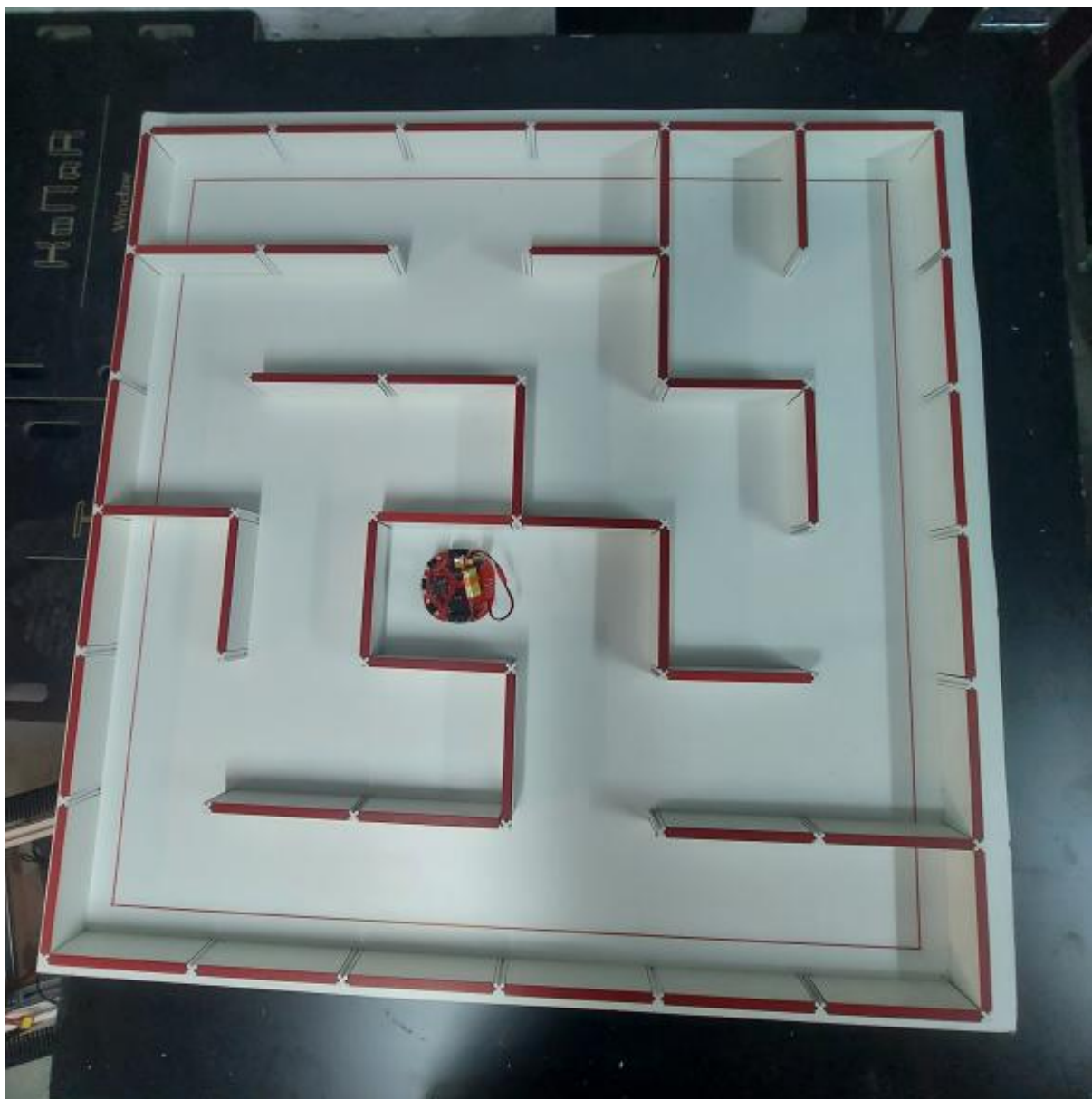


((a)) Widok z góry



((b)) Widok od spodu



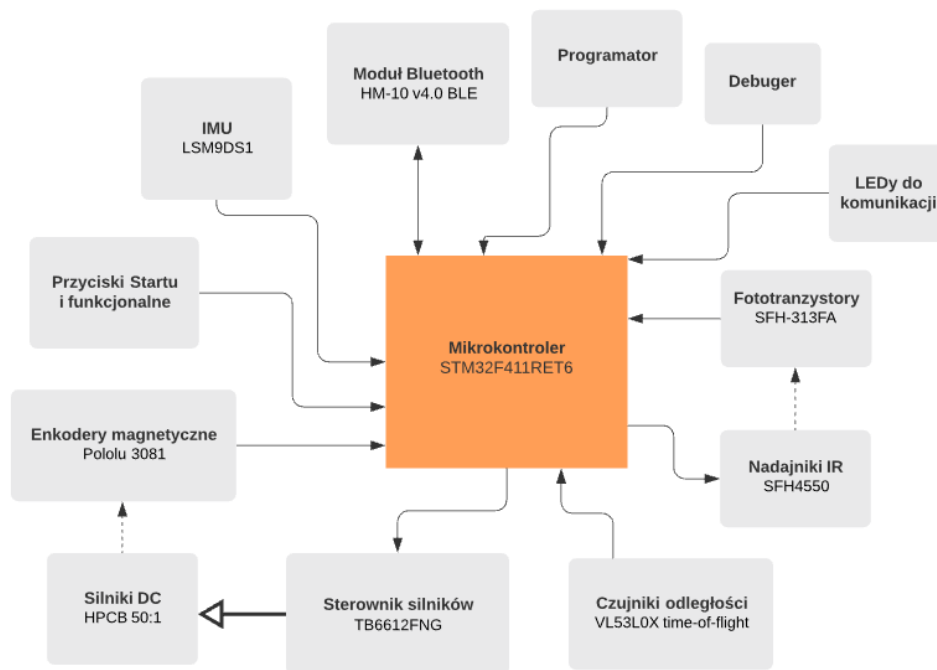


Rysunek 9: Robot w labiryncie

Jak widać na powyższych zdjęciach, zgodnie z początkowymi założeniami udało nam się stworzyć konstrukcję robota o rozmiarach i kształcie pozwalających na swobodne poruszanie się wewnątrz labiryntu.

## 7 Opis działania programu

### 7.1 Rdzeń programu



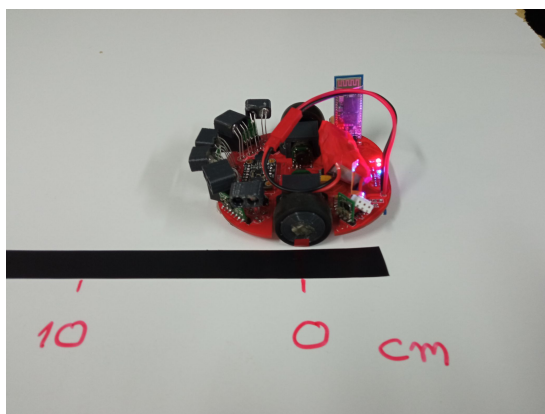
Rysunek 10: Diagram blokowy rdzenia programu

### 7.2 Obsługa czujników i sterowników

#### 7.2.1 Obsługa enkoderów

Po podłączeniu prawidłowym enkoderów Pololu oraz dodatkowym skonfigurowaniu wyjść TIM3 CH1 CH2 oraz TIM2 CH1 CH2 jako encoder mode możliwy był odczyt impulsów odbieranych przez enkodery, a następnie prawidłowa ich konwersja na jednostki odległości mierzone w mm.

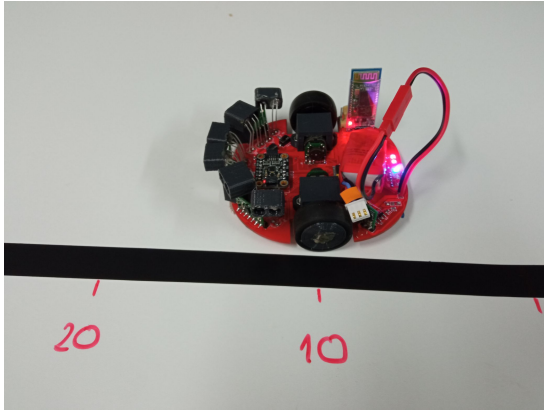
Pierwszym ważnym etapem konwersji było skalibrowanie pomiarów z enkoderów oraz ich prawidłowa konwersja:



((a)) Pozycja startowa

```
Stopping
LEFT : POS[mm] : 0    VEL[mm/s] : 0
RIGHT : POS[mm] : 0    VEL[mm/s] : 0
```

((b)) Wartości odczytów z enkoderów

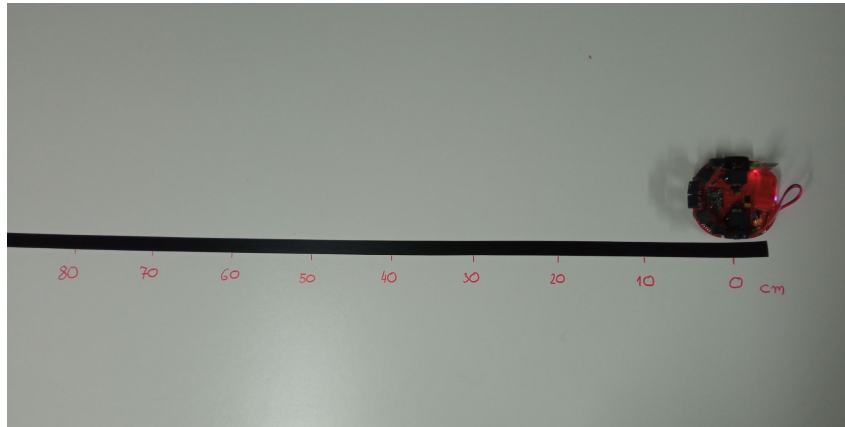


((a)) Pozycja po przesunięciu o 10cm

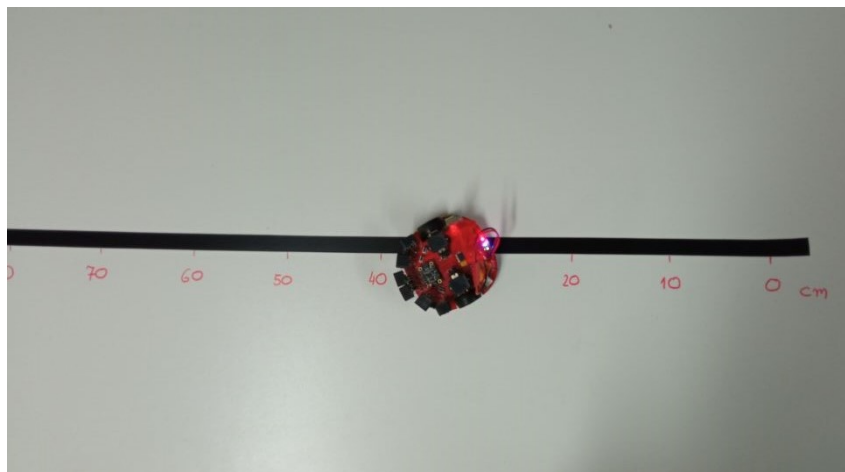
```
Stopping
LEFT : POS[mm] : 0      VEL[mm/s] : 0
RIGHT : POS[mm] : 0     VEL[mm/s] : 0
LEFT : POS[mm] : 106    VEL[mm/s] : 0
RIGHT : POS[mm] : 109   VEL[mm/s] : 0
```

((b)) Odczyty po przesunięciu

Kolejną istotną kwestią po skalibrowaniu odczytów pozycji robota ważne było aby prawidłowo przekształcać i wykorzystywać odczyty z prędkości. W tym celu wykonany był kolejny odpowiedni pomiar, w którym robot przejeżdżał przez sekundę po torze z zadany wypełnieniem PWM oraz co 100ms wyświetlał pozycję i prędkość.



Rysunek 11: Pozycja startowa na torze



Rysunek 12: Pozycja końcowa na torze



Stopping					
LEFT	:	POS [mm]	:	0	VEL [mm/s] : 0
RIGHT	:	POS [mm]	:	0	VEL [mm/s] : 0
LEFT	:	POS [mm]	:	6	VEL [mm/s] : 245
RIGHT	:	POS [mm]	:	14	VEL [mm/s] : 351
LEFT	:	POS [mm]	:	45	VEL [mm/s] : 370
RIGHT	:	POS [mm]	:	59	VEL [mm/s] : 430
LEFT	:	POS [mm]	:	85	VEL [mm/s] : 370
RIGHT	:	POS [mm]	:	104	VEL [mm/s] : 416
LEFT	:	POS [mm]	:	117	VEL [mm/s] : 370
RIGHT	:	POS [mm]	:	150	VEL [mm/s] : 430
LEFT	:	POS [mm]	:	157	VEL [mm/s] : 370
RIGHT	:	POS [mm]	:	186	VEL [mm/s] : 416
LEFT	:	POS [mm]	:	197	VEL [mm/s] : 370
RIGHT	:	POS [mm]	:	232	VEL [mm/s] : 416
LEFT	:	POS [mm]	:	237	VEL [mm/s] : 370
RIGHT	:	POS [mm]	:	278	VEL [mm/s] : 416
LEFT	:	POS [mm]	:	269	VEL [mm/s] : 370
RIGHT	:	POS [mm]	:	314	VEL [mm/s] : 416
LEFT	:	POS [mm]	:	309	VEL [mm/s] : 370
RIGHT	:	POS [mm]	:	360	VEL [mm/s] : 430
LEFT	:	POS [mm]	:	349	VEL [mm/s] : 370
RIGHT	:	POS [mm]	:	406	VEL [mm/s] : 430

Rysunek 13: Odczyt z kalibracji

### 7.2.2 Kalibracja PID kontrolującego silniki

Po prawidłowym obsłużeniu odczytów z enkoderów możliwe było napisanie regulatora PID i ręczne jego nastrojenie.

Na poniższych screenach SET oznacza wartość oczekiwaną, MEAS jest wartością zmierzoną na enkoderach, RES to poprawka regulacji PWM wysłanego na silniki.

Kp	4	Ki	10	Kd	3
Kp	4	Ki	9	Kd	3
Kp	4	Ki	8	Kd	3
Kp	4	Ki	7	Kd	3
Kp	4	Ki	6	Kd	3
Kp	4	Ki	5	Kd	3
Kp	4	Ki	4	Kd	3
Kp	4	Ki	3	Kd	3
Kp	4	Ki	4	Kd	3
Kp	4	Ki	4	Kd	2
Kp	4	Ki	4	Kd	2

Rysunek 14: Strojenie PID

SET: 1000	MEAS: 0	RES: 375
SET: 1000	MEAS: 0	RES: 375
SET: 1000	MEAS: 740	RES: 97
SET: 1000	MEAS: 1389	RES: -146
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 740	RES: 97
SET: 1000	MEAS: 740	RES: 97
SET: 1000	MEAS: 740	RES: 97
SET: 1000	MEAS: 1064	RES: -24
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 740	RES: 97
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 740	RES: 97
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 740	RES: 97
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 740	RES: 97
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 926	RES: 27

Rysunek 15: Start regulacji

SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 926	RES: 27
SET: 1000	MEAS: 926	RES: 27
SET: 2000	MEAS: 926	RES: 402
SET: 2000	MEAS: 1389	RES: 229
SET: 2000	MEAS: 2916	RES: -344
SET: 2000	MEAS: 3704	RES: -639
SET: 2000	MEAS: 3518	RES: -570
SET: 2000	MEAS: 2592	RES: -222
SET: 2000	MEAS: 1852	RES: 55
SET: 2000	MEAS: 1852	RES: 55
SET: 2000	MEAS: 2315	RES: -119
SET: 2000	MEAS: 2916	RES: -344
SET: 2000	MEAS: 2916	RES: -344
SET: 2000	MEAS: 2592	RES: -222
SET: 2000	MEAS: 2453	RES: -170
SET: 2000	MEAS: 2453	RES: -170
SET: 2000	MEAS: 2453	RES: -170
SET: 2000	MEAS: 2453	RES: -170
SET: 2000	MEAS: 2453	RES: -170
SET: 2000	MEAS: 2453	RES: -170
SET: 2000	MEAS: 2453	RES: -170
SET: 2000	MEAS: 2453	RES: -170

Rysunek 16: Zmiana set point

```

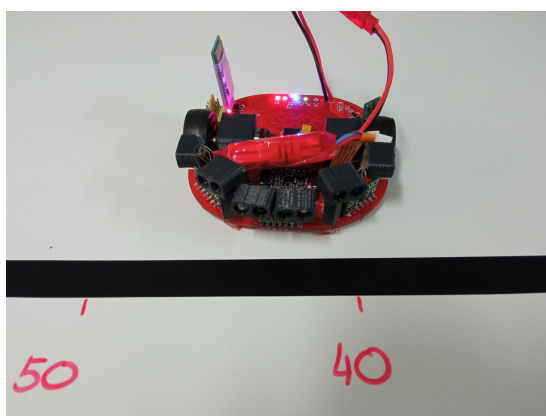
SET: 2000      MEAS: 2453      RES: -170
SET: 2000      MEAS: 2453      RES: -170
SET: 0         MEAS: 2453      RES: -920
SET: 0         MEAS: 1064      RES: -399
SET: 0         MEAS: -138      RES: 51
SET: 0         MEAS: -138      RES: 51
SET: 0         MEAS: 0         RES: 0
SET: 0         MEAS: 0         RES: 0

```

Rysunek 17: Wyzerowanie set point

### 7.2.3 Obsługa pomiarów ADC

Odczyty ADC posłużyły m.in do pomiaru i pilnowania stanu napięcia na baterii jak również do odczytów z czujników KTIR, które służą do wykrywania linii na torze. Niestety jeden z czujników KTIR nadal nie działał. Najprawdopodobniej uszkodzeniu mogły ulec ścieżki prowadzące do tegoż czujnika i nie odbieraliśmy z niego żadnych sygnałów.



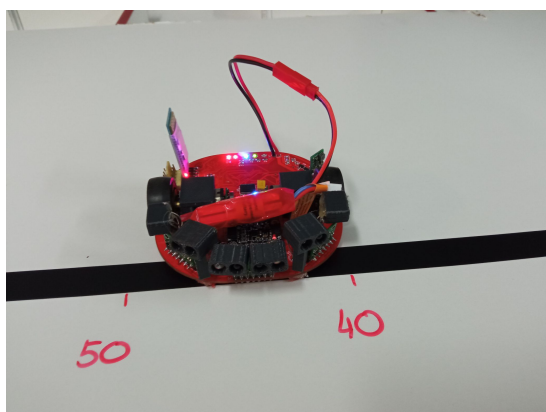
((a)) Położenie robota

```

ADC MEASUREMENTS
Batt : 3032
KTIR_1 : 692
KTIR_2 : 0
KTIR_3 : 230
KTIR_4 : 212

```

((b)) Odczyty białej powierzchni



((a)) Położenie robota nad linią

```

ADC MEASUREMENTS
Batt : 3050
KTIR_1 : 3922
KTIR_2 : 1
KTIR_3 : 3646
KTIR_4 : 3624

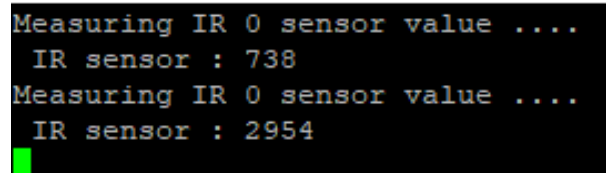
```

((b)) Odczyty czarnej powierzchni

### 7.2.4 Multiplekser i czujnik IR

Niestety popełniliśmy błąd w składaniu i lutowaniu konstrukcji, który spowodował, że przez mieszane opisy nóżek kolektora i emitera w fototranzystorach wszystkie zostały polutowane odwrotnie i odczyty są nieprawidłowe.

Jednakże udało nam się jeszcze przelutować jedną parę czujników aby zademonstrować odczyty z sensora IR.



```
Measuring IR 0 sensor value ....
IR sensor : 738
Measuring IR 0 sensor value ....
IR sensor : 2954
```

Rysunek 18: Odczyty z czujnika przy zasłoniętym i odsłoniętym czujniku

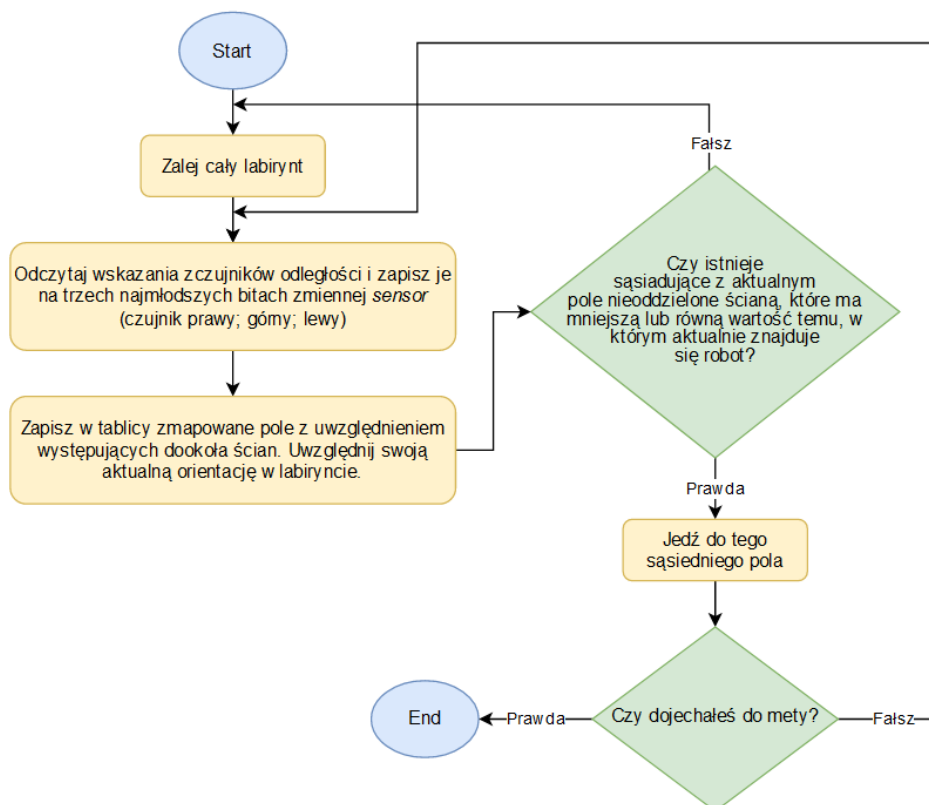
```
switch(option){
case 'k':
    sensor = choice;
    printf("Measuring IR %d sensor value .... \r\n", sensor);
    PT_change_sensor(sensor);
    IR_set(sensor);
    HAL_Delay(250);
    printf(" IR sensor : %d \r\n", adc_measurements[5]);
    option=' ';
    break;
```

Rysunek 19: Kod wywołujący pomiar

## 7.3 Opis algorytmu eksploracji labiryntu (w trakcie implementacji)

Robot będzie miał za zadanie dotrzeć w jak najkrótszym czasie z pozycji startowej do docelowej, wykorzystując dwa przejazdy po labiryncie. W pierwszym przejeździe robot mapuje labirynt do momentu odnalezienia celu (środka labiryntu). W drugim przejeździe wybiera najkrótszą drogę z pośród zmapowanych w pierwszym przejeździe komórek. Do odnajdywania najkrótszej drogi od startu (jednego z narożników) do mety (środka) w labiryncie zaimplementujemy w robocie **algorytm zalewania** (*ang. floodfill*). Zalewanie rozpoczyna się od miejsca docelowego, czyli środka labiryntu złożonego z czterech kwadratowych komórek. Przed rozpoczęciem pierwszego przejazdu robot nie ma żadnej wiedzy o strukturze labiryntu, dlatego przy pierwszym zalewaniu traktujemy labirynt jako pusty kwadrat o 256 polach - indeksy pól rosną równomiernie we wszystkich kierunkach. Wykorzystanie algorytmu zalewania podczas mapowania i poszukiwania celu powinno spowodować, że robot zacznie szybko zmierzać w kierunku środka labiryntu po wyruszeniu z pola startu.

W czasie eksploracji (pierwszego przejazdu) robot wykrywa ściany komórki, w której aktualnie się znajduje a następnie sprawdza czy istnieje sąsiedni blok o mniejszym lub równym indeksie, do którego może się przemieścić. Zalewanie labiryntu następuje dopiero wtedy, gdy wszystkie dostępne sąsiednie bloki mają większe wartości od wartości bloku, w którym aktualnie znajduje się robot. Ta operacja powoduje zmianę dotychczasowych wartości bloków labiryntu i wyznaczenie nowej trasy prowadzącej do celu (w kierunku malejących wartości). Po osiągnięciu celu następuje ostatnie zalewanie i wyznaczana jest najkrótsza droga łącząca punkt startowy z celem spośród zmapowanej części labiryntu.



Rysunek 20: Diagram algorytmu zalewania

## 8 Napotkane problemy

### 8.1 enkodery

Okazało się, że na etapie projektowania elektroniki i płytki zapomnieliśmy o poprowadzeniu czterech ścieżek z enkoderów do mikrokontrolera. Na szczęście mieliśmy kilka wolnych pinów na mikrokontrolerze, dlatego udało nam się naprawić błąd przy pomocy kynaru i kleju na gorąco (efekt widoczny na zdjęciu dolnej warstwy płytki na str. 12).

### 8.2 czujniki odległości

Wykorzystane przez nas moduły VL53L0X okazały się być dość skomplikowane w obsłudze pomimo dostarczonych przez producenta bibliotek API. Aby ustawić adres jednego takiego czujnika przez I<sup>2</sup>C, dwa pozostałe muszą być wyłączone. Nie wzięliśmy pod uwagę konieczności przemiennego wyłączania tych czujników w czasie projektowania elektroniki, dlatego wszystkie były połączone do tej samej linii XSHUT. Planowaliśmy odlutować moduły, osobno pustać ich adresy i przylutować spowrotem na miejsce. Niestety, adresy restartują się do ustawienia fabrycznego każdorazowo po wyłączeniu zasilania.

Problem udało nam się rozwiązać przerywając połączenie z linią XSHUT dwóch modułów i łącząc je za pomocą kynaru do dwóch wolnych pinów pozostałych po wcześniejszym module Bluetooth (wymieniliśmy model BT HM-10 na HC-06).

## 9 Zadania niezrealizowane

Udało nam się zrealizować prawie wszystkie założenia projektowe. Wynikło kilka opóźnień z powodu problemów technicznych z konstrukcją robota, na których naprawę musieliśmy postawić priorytet.

## 9.1 Autonomia

Z powodu niedoszacowania czasu potrzebnego na realizację wcześniejszych etapów projektu oraz napotkanych po drodze problemów technicznych, nie udało nam się dokończyć algorytmu eksploracji labiryntu na czas.

## 9.2 Kompletna obsługa czujników odległości

Z obsługą czujników odległości mieliśmy najwięcej problemów. Po wielu próbach zmiany kodu, zwracane przez nie dane nadal nie są na tyle zadowalające, aby możliwe było omijanie przeszkód.

## 10 Podsumowanie

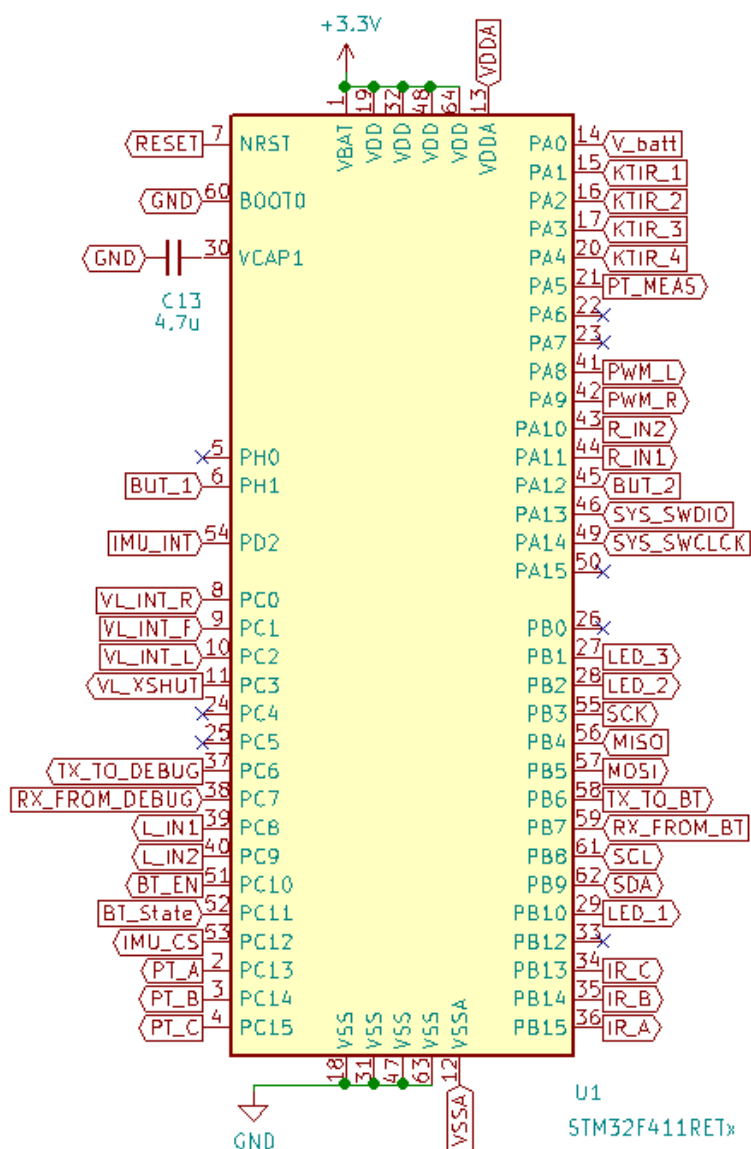
Pomimo napotkania pewnych problemów, znaczną większość z nich rozwiązaliśmy z zadowalającym efektem. Udało nam się skonstruować robota mobilnego klasy 2.0 o satysfakcjonującej konstrukcji mechanicznej i elektronicznej. Robot jest mały, zwrotny i szybki. Po dopracowaniu oprogramowania *Squick* będzie mógł rywalizować na zawodach robotycznych w kategorii Micromouse z innymi robotami.

## 11 Odnosińniki do repozytorium GIT Bitbucket

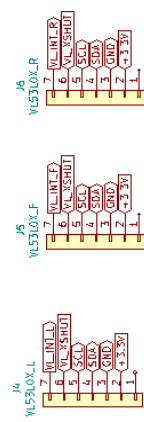
- Bitbucket Squick CubeIDE
- Bitbucket Squick KiCAD
- Bitbucket Squick Documentation
- Bitbucket Squick Inventor

## Schematy elektroniczne robota klasy Micromouse „Squick”

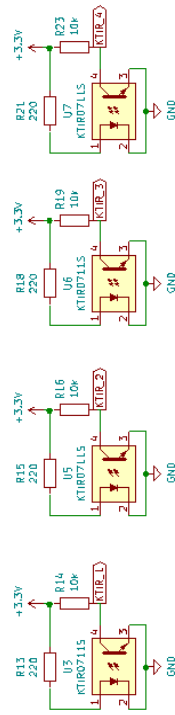
### Microcontroller STM32



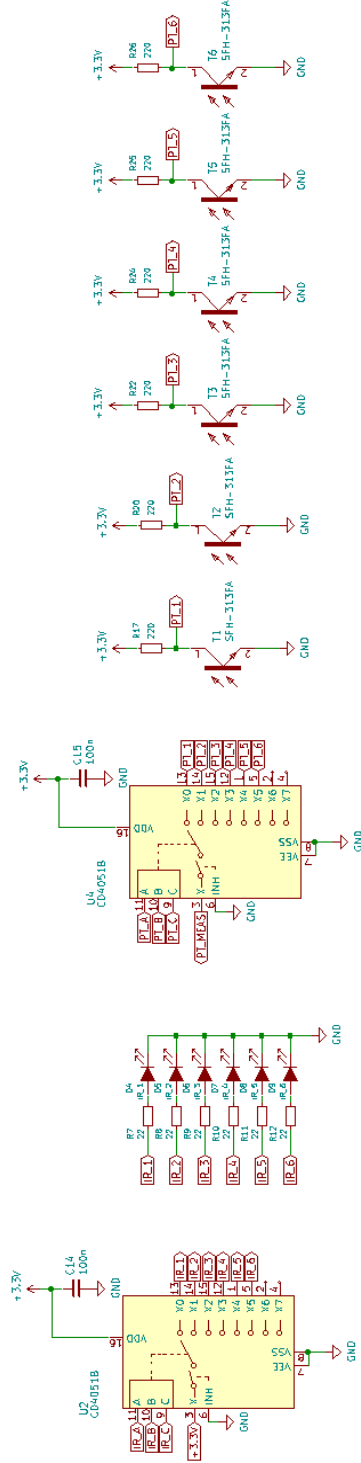
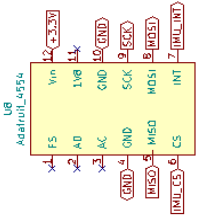
## ToF Distance Sensors



## Line sensors



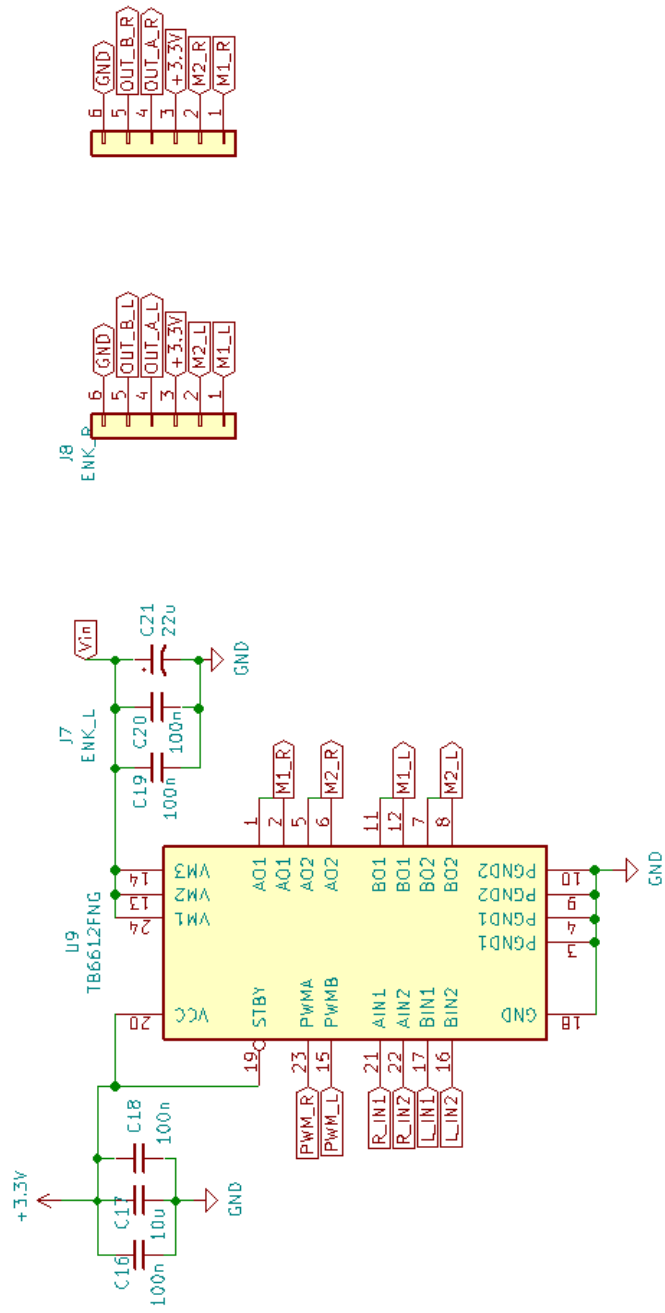
## IMU



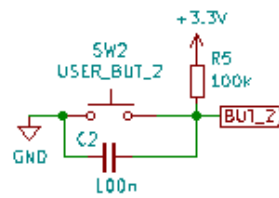
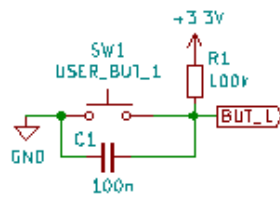


## Motor Driver

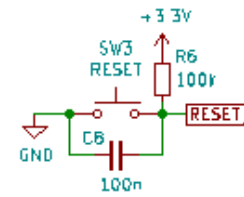
## Encoders



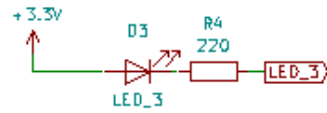
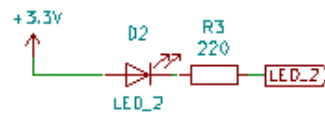
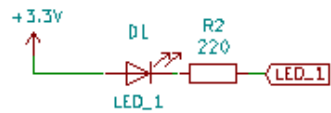
## User buttons



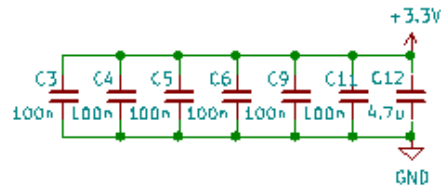
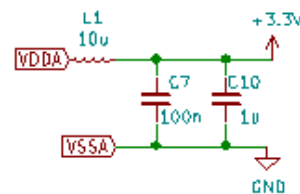
## Manual reset



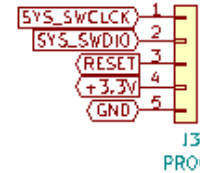
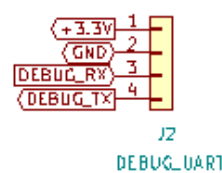
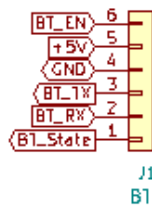
## User Diodes



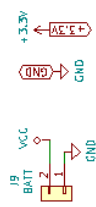
## Power noise filtering



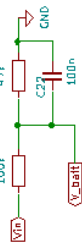
## Communication devices



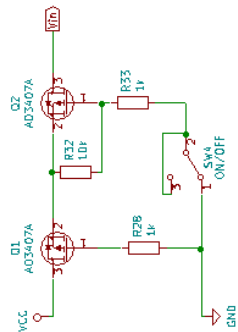
### Battery Connection



### Battery Level Probe



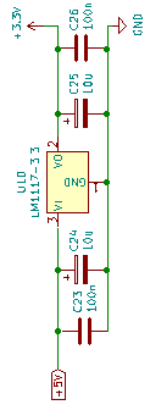
### Inverted polarisation safeguard



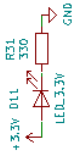
### 5V Power Converter 5V Signal Diode



### 3.3V Power Balancer



### 3.3V Signal Diode



## Literatura

- [1] Marek Galewski. *Aplikacje i ćwiczenia w języku C z biblioteką HAL*. Wydawnictwo BTC, Legionowo, Lipiec 2019.
- [2] Jan Kędzierski. *Filtr Kalmana - zastosowania w prostych układach sensorycznych*. Wrocław, Październik 2007.
- [3] Dawid Perdek. *Badanie własności algorytmów poszukiwania ścieżki w labiryncie dla robota klasy micromouse*. Wrocław, Grudzień 2015.
- [4] STMicroelectronics. *STM32F411RET6 datasheet*. 2014
- [5] STMicroelectronics. *VL53L0X datasheet*. 2016
- [6] STMicroelectronics. *LSM9DS1 datasheet*. 2015