Politechnika
Wrocławska

Sterowniki Robotów

Robot Minisumo ze sterowaniem prędkością i momentem obrotowym

Supernova

Etap III - Dokumentacja techniczna

Środa TN 11:15
Aleksander Sil
218576
Grupa 9

Spis treści

1 Cel projektu

Celem projektu jest implementacja dwóch typów sterowania robotem typu Minisumo: sterowania prędkością oraz sterowania momentem obrotowym.

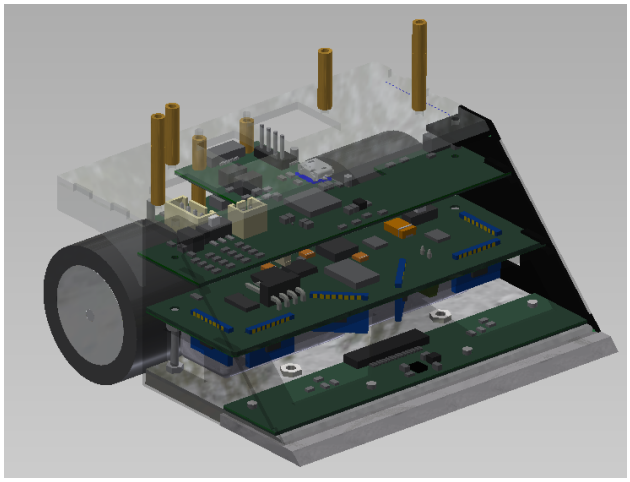
2 Założenia projektowe

2.1 Funkcjonalności

Funkcjonalności, które zostaną zaimplementowane na robocie to:

- Obsługa enkoderów kwadraturowych
- Pomiar napięcia baterii zewnętrznym przetwornikiem ADC - komunikacja z układem po magistrali I2C
- Pomiar prądu na silnikach z użyciem układu lustra prądowego wbudowanego w mostek H
- Algorytm sterowania prędkością
- Określenie stanu i orientacji robota z użyciem układu MEMS
- Algorytm sterowania momentem obrotowym
- Komunikacja poprzez Bluetooth (UART)
- Zapisywanie ustawień w pamięci EEPROM - komunikacja z układem po magistrali I2C
- Obsługa czujników odległości - komunikacja z czujnikami po magistrali I2C

Będą one implementowane na robocie, który został już fizycznie wykonany. Poniżej prezentuję jego model:



Rysunek 1: Model robota zrobiony w programie Autodesk Inventor

2.2 Wspomaganie pracy zespołowej

Projekt ten jest wykonywany przez jedną osobę przez co narzędzia takie nie będą konieczne. Wykorzystywany jednak będzie system kontroli wersji ***Git***, aby w razie niepowodzeń móc łatwo wrócić do prawidłowo działającej wersji projektu.

3 Opis zrealizowanych prac

Na obecnym etapie projektu zrealizowane zostały:

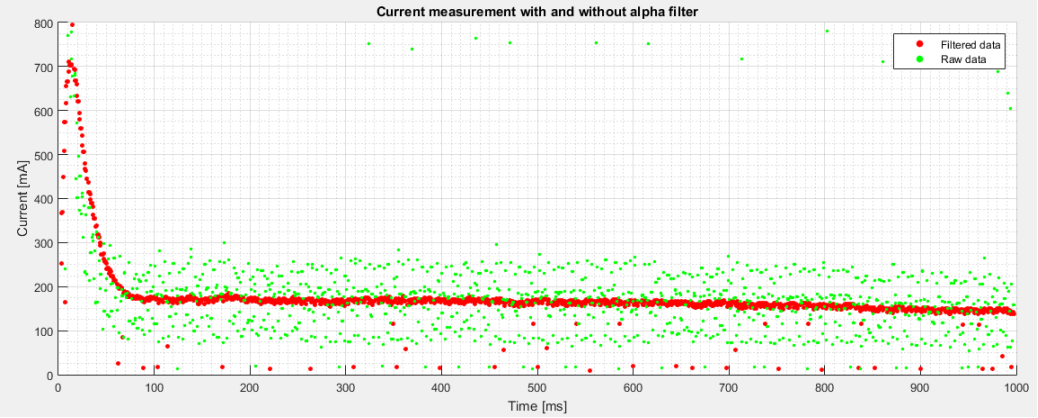
- Obsługa enkoderów kwadraturowych

  Enkodery zostały obsłużone korzystając z *Encoder Mode* dwóch timerów mikrokontrolera (TIM2 i TIM3). Tryb pracy timerów został ustawiony na "*TI1 and TI2*", aby odczyt impulsów enkodera miał większa rozdzielczość.

- Pomiar prądu na silnikach z użyciem układu lustra prądowego wbudowanego w mostek H

  Pomiar prądu został zrealizowany w 2 iteracjach elektroniki robota. W wersji z przetwornicą LM2596S mierzony sygnał był na tyle zakłócony, że mocno utrudniało próbę implementacji sterowania prądowego.

Druga przetwornica, MC34063AC, sprawdziła się zdecydowanie lepiej. Pomiar prądu w obecności nowej wersji płytki zasilania był dużo dokładniejszy. Dla poprawy jakości sygnału zastosowano filtr Alfa. Efekt filtracji widać na poniższym wykresie.



Rysunek 2: Wykres porównujący dane pomiaru prądu przed i po filtracji

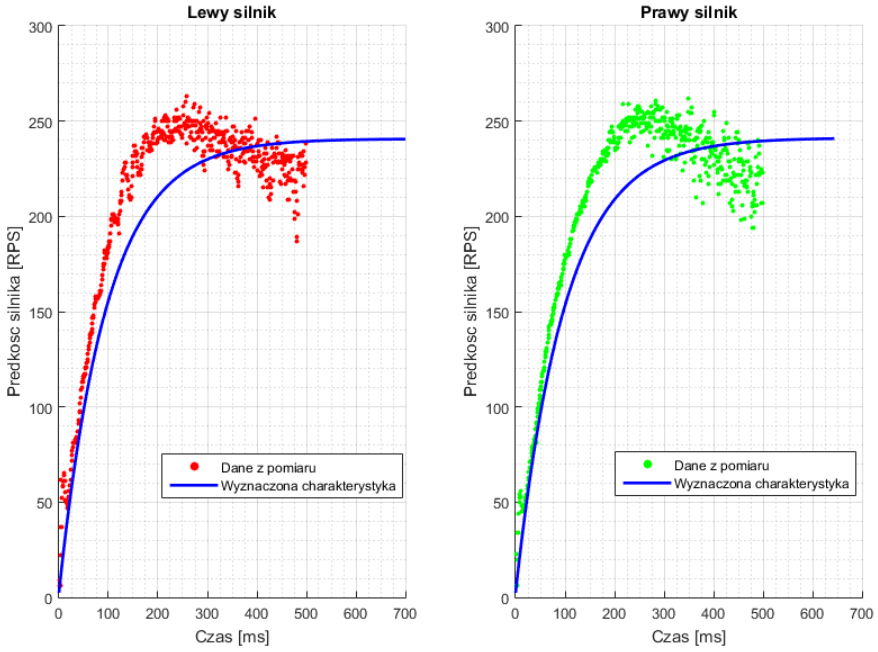
Na czerwono zaznaczone zostały odfiltrowane dane, a na zielono surowe. Skrajne wartości bliskie 0 o 800 mA wynikają z błędów w przesyłaniu danych przez UART, których nie wychwyciłem w tym przypadku. Takowy filtr jest jednym z najprostszych, które można zaimplementować w robocie, a jego wzór dla tego pomiaru wygląda następująco:

$$I_f(t) = (1 - \alpha) \cdot I_m(t) + \alpha \cdot I_f(t - 1) \tag{1}$$

Gdzie: α - współczynnik filtru, I_f - wartość wyliczona przez filtr, I_m - wartość zmierzona, t - czas

- Algorytm sterowania prędkością

  Algorytm ten jest zaimplementowany i działa bez problemów. Podczas procesu wdrażania zostały zebrane dane pomiarowe służące, z których później wyznaczony został pewien matematyczny model silnika. Dane zostały zebrane podczas testu polegającego na ustawieniu 70% wypełnienia PWM na obu silnikach, gdy robot stał na płaskiej powierzchni i były one zbierane przez 0.5 sekundy z rozdzielczością $1ms$. Dane były zapisywane w pamięci mikrokontrolera i dopiero po teście cały ich zestaw był wysyłany przez Bluetooth do komputera. Na poniższym rysunku znajduje się wykres danych pomiarowych wraz z wyliczoną charakterystyką.



Rysunek 3: Wykres zebranych danych pomiarowych wraz z wyznaczoną charakterystyką

Nie jest to najlepiej wyznaczona charakterystyka, ale nie był dostępny odpowiednio duży ring, na którym można by przeprowadzić kolejny test. Zaplanowane są dalsze testy po spełnieniu dalszych założeń projektu. Cały pomiar i analiza zostały tak zaprogramowane, aby możliwe było zmienianie jego parametrów np. czas pomiaru, a więc nie powinno to sprawić większych trudności.

- Komunikacja poprzez Bluetooth (UART)

  Komunikacja ta odbywa się na dwa sposoby: pierwszy polega na nadaniu ramki wiadomości przez robota dopiero po otrzymaniu specjalnego znaku, a druga na ciągłym wysyłaniu ramki co określony interwał czasu. Modułem Bluetooth zamontowanym na robocie jest HC-06.

- Obsługa czujników odległości - komunikacja z czujnikami po magistrali I2C

  Podstawowa wersja pomiaru czujnikami tj. w trybie blokującym, została zrealizowana. Czujniki jednak nie nadają się zbytnio do zastosowania w robotach klasy Minisumo, gdyż mają zbyt szeroki kąt wiązki

światła ($\pm 30°$) co powoduje, że pojawiają się zakłócenia spowodowane wykryciem podłoża.

- Określenie stanu i orientacji robota z użyciem układu MEMS

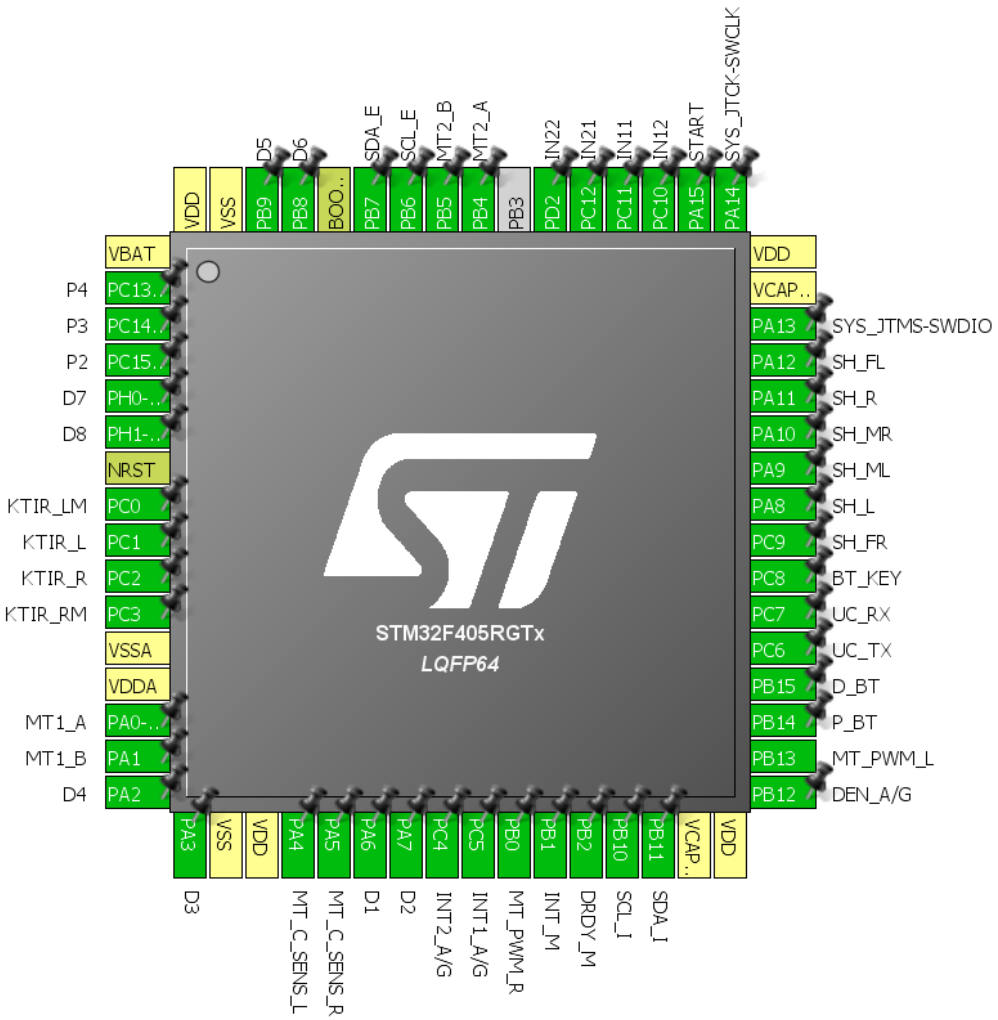
  W robocie została zaimplementowana obsługa układu LSM9DS1 w taki sposób, aby dawał on informację o przyspieszeniach liniowych oraz prędkości kątowej robota. Algorytm walki nie reaguje jeszcze na te odczyty, gdyż najpierw chciałbym zebrać dane pomiarowe z kilku walk, aby móc przeanalizować dane i zweryfikować możliwe opcje wykorzystania tych czujników jak np. wykrycie zderzenia, podbicia oraz tego, że robot jest spychany przez innego robota od boku.

- Zapisywanie ustawień w pamięci EEPROM

  Zaimplementowano cztery funkcje obsługujące tą pamięć: zapis i odczyt bajtu oraz zapis i odczyt strony o rozmiarze do 128 bajtów. Możliwe jest przez to zapisanie wielu ustawień podczas walki minisumo oraz zapamiętanie różnych taktyk. Dodatkowo pamięć może służyć zbieraniu danych o walce, aby móc później odtworzyć je na komputerze. Pozwoli to lepiej dostroić regulatory oraz łatwiej implementować nowe funkcjonalności m.in. z wykorzystaniem układu LSM9DS1.

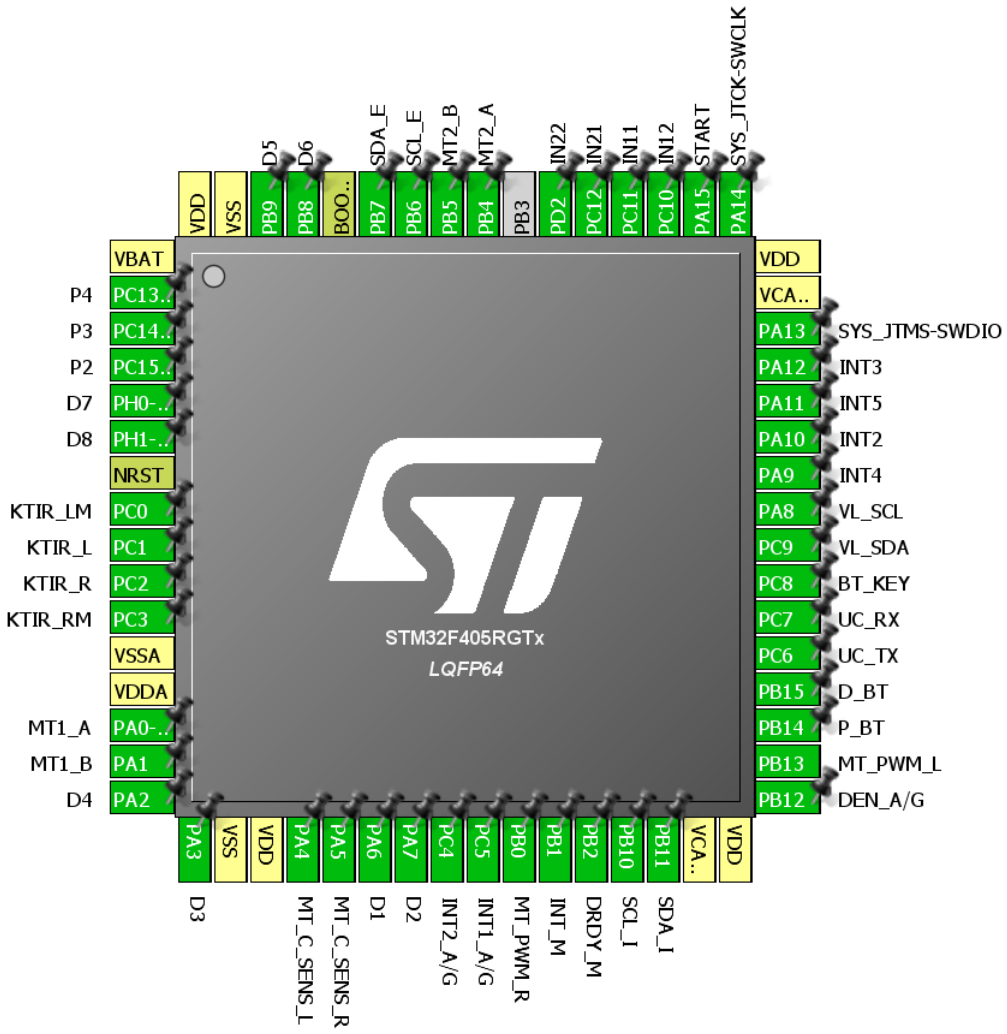
4 Konfiguracja mikrokontrolera

4.1 Dla czujników GP2Y0D340K



Rysunek 4: Konfiguracja pinów mikrokontrolera

4.2 Dla czujników VL53L0X



Rysunek 5: Konfiguracja pinów mikrokontrolera

5 Konfiguracja peryferiów

ADC2 - Pomiar prądu

Clock Prescaler: PCLK2 divided by 8
Resolution: 12 bits
Scan Conversion Mode: Enabled
Continous Conversion mode: Enabled
Number of Conversion: 2
Sampling time: 480 Cycles (oba kanały)

ADC3 - Pomiar napięcia na wyjściu czujników odbiciowych

Clock Prescaler: PCLK2 divided by 8
Resolution: 12 bits
Scan Conversion Mode: Enabled
Continous Conversion mode: Enabled
Number of Conversion: 4
Sampling time: 480 Cycles (wszystkie kanały)

I2C1 - Komunikacja z pamięcią EEPROM

I2C Speed Mode: Normal Mode
I2C Clock Speed (HZ): 100 000

I2C2 - Komunikacja z układem MEMS

I2C Speed Mode: Normal Mode
I2C Clock Speed (HZ): 100 000

I2C3 - Komunikacja z zewnętrznym ADC, czujnikami VL53L0X oraz expanderem

I2C Speed Mode: Normal Mode
I2C Clock Speed (HZ): 100 000

TIM1 - Generowanie sygnału PWM dla mostka H

Clock Source: Internal Clock
Channel 1: PWM Generation CH1N

Channel 2: PWM Generation CH2N
Prescaler: 21
Counter Period: 999

TIM2 - Obsługa enkoderów lewego silnika

Combined Channels: Encoder Mode
Prescaler: 0
Counter Period: 0xFFFE
Encoder Mode: Encode Mode TI1 and TI2
Input Filter: 15
Polarity: Rising Edge

TIM2 - Obsługa enkoderów prawego silnika

Combined Channels: Encoder Mode
Prescaler: 0
Counter Period: 0xFFFE
Encoder Mode: Encode Mode TI1 and TI2
Input Filter: 15
Polarity: Rising Edge

TIM6 - Taktowanie zaimplementowanych funkcji

Prescaler: 83
Counter Period: 999

TIM7 - Taktowanie cyklu pracy robota

Prescaler: 83
Counter Period: 999

USART6 - Komunikacja Bluetooth

Baud Rate: 9600

Konfiguracja pinów

Tabela 1: Konfiguracja pinów dla wersji z czujnikami VL53L0X

Pin Nb	PINs	FUNCTIONs	LABELs
2	PC13-ANTI_TAMP	GPIO_Input	P4
3	PC14-OSC32_IN	GPIO_Input	P3
4	PC15-OSC32_OUT	GPIO_Input	P2
5	PH0-OSC_IN	GPIO_Output	D7
6	PH1-OSC_OUT	GPIO_Output	D8
8	PC0	ADC3_IN10	KTIR_LM
9	PC1	ADC3_IN11	KTIR_L
10	PC2	ADC3_IN12	KTIR_R
11	PC3	ADC3_IN13	KTIR_RM
14	PA0-WKUP	TIM2_CH1	MT1_A
15	PA1	TIM2_CH2	MT1_B
16	PA2	GPIO_Output	D4
17	PA3	GPIO_Output	D3
20	PA4	ADC2_IN4	MT_C_SENS_L
21	PA5	ADC2_IN5	MT_C_SENS_R
22	PA6	GPIO_Output	D1
23	PA7	GPIO_Output	D2
24	PC4	GPIO_Input	INT2_A/G
25	PC5	GPIO_Input	INT1_A/G
26	PB0	TIM1_CH2N	MT_PWM_R
27	PB1	GPIO_Input	INT_M
28	PB2	GPIO_Input	DRDY_M
29	PB10	I2C2_SCL	SCL_I
30	PB11	I2C2_SDA	SDA_I
33	PB12	GPIO_Input	DEN_A/G
34	PB13	TIM1_CH1N	MT_PWM_L
35	PB14	GPIO_EXTI14	P_BT
36	PB15	GPIO_Output	D_BT
37	PC6	USART6_TX	UC_TX
38	PC7	USART6_RX	UC_RX
39	PC8	GPIO_Output	BT_KEY
40	PC9	I2C3_SDA	VL_SDA
41	PA8	I2C3_SCL	VL_SCL
42	PA9	GPIO_Input	INT4
43	PA10	GPIO_Input	INT2

Pin Nb	PINs	FUNCTIONs	LABELs
44	PA11	GPIO_Input	INT5
45	PA12	GPIO_Input	INT3
46	PA13	SYS_JTMS-SWDIO	
49	PA14	SYS_JTCK-SWCLK	
50	PA15	GPIO_Input	START
51	PC10	GPIO_Output	IN12
52	PC11	GPIO_Output	IN11
53	PC12	GPIO_Output	IN21
54	PD2	GPIO_Output	IN22
56	PB4	TIM3_CH1	MT2_A
57	PB5	TIM3_CH2	MT2_B
58	PB6	I2C1_SCL	SCL_E
59	PB7	I2C1_SDA	SDA_E
61	PB8	GPIO_Output	D6
62	PB9	GPIO_Output	D5

6 Układy zewnętrzne

LSM9DS1 - 9 osiowe IMU

Układ ten zawiera akcelerometr, magnetometr i żyroskop w pojedynczej obudowie LGA-24L. Komunikacja z nim odbywa się z użyciem interfejsu I2C lub SPI - w przypadku tego projektu wykorzystywany jest ten pierwszy.

Każde z peryferiów układu zawiera różne konfigurowalne ustawienia zakresu pracy:

- $\pm 2/\pm 4/\pm 8/\pm 16$ g - zakresy pomiaru przyspieszeń
- $\pm 4/\pm 8/\pm 12/\pm 16$ gauss - zakres pomiaru pola magnetycznego
- $\pm 245/\pm 500/\pm 2000$ deg/s - zakres pomiaru prędkości kątowej

HC-06 - moduł Bluetooth

Prosty i łatwo dostępny modul Bluetooth, który został wlutowany w płytkę główną robota. Korzystając z komend AT została zmieniona domyślna nazwa, pod którą zgłaszał się moduł oraz PIN do parowania z innymi urządzeniami. Komunikacja z modułem odbywa się z użyciem interfejsu szeregowego USART pracującego w trybie asynchronicznym. Moduł pracuje z Baud ratem $9600 Bits/s$.

M24512 - pamięć EEPROM

Komunikacja z pamięcią odbywa się za pomocą interfejsu I2C, ale jego obsługa nie została jeszcze zaimplementowana.

MC33932 - Mostek H

Układ ten jest 2 kanałowym mostkiem H o wydajności $5A$ na kanał oraz maksymalnym napięciu pracy $28V$. Został on dobrany ze znacznym zapasem, gdyż silniki pracują z maksymalnym napięciem $16.8V$, przy którym prąd zwarciowy wynosi ok. $1.7A$.

Mostek ten zawiera wyprowadzenia lustra prądowego, dzięki któremu możemy zmierzyć prąd płynący na dane silniki.

Sygnałami wejściowymi kanałów są po 2 sygnały dwuwartościowe (binarne) oraz 1 sygnał PWM.

SHARP GP2Y0D340K - cyfrowe czujniki odległości

Na pierwszej wersji płytki z czujnikami znajdowały się te czujniki. Są one najczęściej wykorzystywanymi czujnikami tego typu w robotach MicroSumo/MiniSumo. Cechują się $8ms$ czasem pojedynczego pomiaru, zasięgiem 40 cm, małym wpływem oświetlenia zewnętrznego na ich pomiar oraz wąską wiązką światła. Czujniki te zwracają stan niski, jeśli w zasięgu ich wiązki pojawi się jakiś obiekt, a w przeciwnym wypadku utrzymują stan wysoki.

KTIR0711S - czujniki odbiciowe

Układ ten służy jako czujnik białej linii w wielu robotach Minisumo. Jest to transoptor odbiciowy zamknięty w małej obudowie o wymiarach 2.7 x 3.4 mm. Doskonale nadaje się do zamontowania w pługu robota. Służy on do odróżnienia krawędzi ringu od jego środka (koloru białego od czarnego). Czujnik ten na wyjściu daje sygnał analogowy.

MC3221A5T - przetwornik A/D

Przetwornik ten jest wykorzystywany w robocie do pomiaru napięcia zasilania. Komunikacja z nim odbywa się przy użyciu interfejsu I2C. Nie zawiera on rejestrów wewnętrznych. Cechuje się on rozdzielczością 12 bitową.

MCP23008 - ekspander portów IO

Układ ekspandera portów. W robocie służy on włączaniu czujników VL53L0X w momencie ich inicjalizacji. W momencie początku inicjalizacji czujników poszczególne rejestry ekspandera ustawione są następująco:

[**IODIR**] 0b11000000 (Rejestr odpowiadający za kierunek portów IO. Pierwsze 2 są ustawione jako wejścia i są podłączone do wyjść przerwań dwóch czujników VL53L0X, a pozostałe 6 służy podłączone są do pinów Enable tych czujników i są ustawione jako wyjścia)

[**OLAT**] 0b00000000 (Ustawia wartość na pinach ustawionych jako wyjścia. Wartość 0 oznacza stan niski)

VL53L0X - czujnik odległości typu Time-of-Flight

Czujnik ten cechuje się 2 metrowym zasięgiem i stosunkowo krótkim pomiarem. Komunikacja z nim odbywa się przy użyciu interfejsu I2C. Zastosowanie go w Minisumo jest próbą zastąpienia czujników GP2Y0D340K, których sprowadzenie, nawet z Chin, powoli przestaje być możliwe. Czujniki te mają niestety ponad 2-krotnie dłuższy czas pomiaru niż SHARPy oraz cechują się dużo szerszą wiązką, co wymusza zmiany w konstrukcji omawianego w tym raporcie robota.

Firma STMicroelectronics udostępnia API do komunikacji z tymi czujnikami, dzięki któremu można wygodnie z nich korzystać.

Po uruchomieniu zasilania czujnikom tym trzeba zmieniać adresy, gdyż resetują się do wartości domyślnej. W tym celu zastosowany został ekspander, który umożliwił aplikację ich poprzez zmianę tylko jednej płytki w robocie. Ustawianie adresów odbywa się przez kolejne włączanie czujników i nadawanie im następnych adresów.

7 Opis działania programu

Program jest podzielony na trzy główne bloki:

- Inicjalizacja - inicjalizacja i włączenie peryferiów, inizjalizacja struktur itd.
- Pętla przed startem - znajduje się tu wybór różnych opcji m.in. w jaki sposób robot wystartuje walkę po otrzymaniu sygnału Start oraz wybór trybu wyświetlania stanu robota na LEDach.

- Pętla po starcie - program startuje algorytm walki i wykonuje go, aż do otrzymaniu sygnału STOP tj. stanu 0 na pinie START. Po otrzymaniu sygnału START program wraca do pętli pierwszej i nie może z niej wyjść bez restartu zasilania modułu startowego - wymóg techniczny konkurencji xSumo.

Niezależnie od pętli, w której z dwóch pętli znajduje się aktualnie robot jest wykonywana jedna funkcja, która kontroluje wszystkie działania. Na jej początku jest napisany kawałek kodu odpowiadający za to, aby każdy cykl pracy robota trwał 1 ms (można zwiększyć długość tego cyklu) - jest to potrzebne m.in. do implementacji regulatora. Poniżej znajduje się kod tej funkcji:

```
1  void State_Update(uint8_t Mode, uint8_t LED_Mode)
2  {
3  /*  Cycle synchronizing */
4  if(HAL_TIM_Base_GetState(&htim7) ==
       HAL_TIM_STATE_READY)
5  HAL_TIM_Base_Start_IT(&htim7);
6
7  Cycle_finished = 1;
8
9  HAL_GPIO_WritePin(D8_GPIO_Port,D8_Pin,GPIO_PIN_RESET
       );
10 while(!Cycle_time_elapsed)  {}
11 Cycle_time_elapsed = 0;
12
13 if(Runtime_Error == TIMEOUT) {HAL_GPIO_WritePin(
       D2_GPIO_Port, D2_Pin, GPIO_PIN_SET);}
14 else {HAL_GPIO_WritePin(D2_GPIO_Port, D2_Pin,
       GPIO_PIN_RESET);}
15 /**********************/
16
17 /*
18  * Odczytaj dane pomiaru predkosci i pradu na
       silnikach
19  */
20 f_Encoders_GetValue();
21
22 MOTOR_CONTROL.Velocity_Measured[0] = MOTORS.
       MT_Enc_data[0]*ENC_DATA2RPS;
```

```
23 MOTOR_CONTROL.Velocity_Measured[1] = MOTORS.
     MT_Enc_data[1]*ENC_DATA2RPS;
24
25 MOTOR_CONTROL.Current_Measured[0] = MOTORS.MT_C_data
     [0]*C_DATA2CURRENT;
26 MOTOR_CONTROL.Current_Measured[1] = MOTORS.MT_C_data
     [1]*C_DATA2CURRENT;
27
28 f_SH_Check();
29 f_KTIR_Update();
30 State_LEDs(LED_Mode);
31
32 if(Bluetooth_Enable){
33   GenerateMessage_FullState();
34   if(huart6.RxState == HAL_UART_STATE_READY){
35     HAL_UART_Receive_IT(&huart6, BT_ReceivedMessage,
         1);
36   }
37   if((++BT_Msg_PublishCounter >= 500) &&
       BT_ContinousTransmit){
38     BT_Msg_PublishCounter = 0;
39     State_Send();
40     HAL_GPIO_TogglePin(D1_GPIO_Port,D1_Pin);
41   }
42 }
43 switch(Mode)
44 {
45   case 0:
46   f_MT_SetSpeed(0, 0);
47   break;
48
49   case 1:
50   break;
51
52   case 2:
53   Velocity_Control();
54   break;
55
56   case 10:
57   f_MT_Update2(700,700);
58   break;
```

```
59
60    default:
61    break;
62  }
```

Jak widać znajdują się tutaj również funkcje odpowiadające za sprawdzenie aktualnego stanu czujników i przepisanie tych danych do odpowiednich struktur. To tutaj odbywa się również wywołanie funkcji wygenerowania nowej wiadomości, która następnie jest wysyłana przez moduł Bluetooth, o ile komunikacja została włączona - do tego służy przycisk **P_BT**.

Na końcu funkcji znajduje się wybór kilku opcji, które wybierane są na podstawie argumentu **Mode**. Jedną z nich jest włączenie regulacji prędkości tj. **Velocity_Control()**

Kolejną ważną funkcją jest algorytm walki, który wygląda w skrócie następująco:

```
1  void a_Find_and_kill_raw()
2  {
3  float ScaleR = 1;
4  float ScaleL = 1;
5  uint8_t FLAG_RunFromLine = 0;
6
7  int16_t Counter = 0;
8
9
10 while(HAL_GPIO_ReadPin(START_GPIO_Port, START_Pin))
11 {
12   State_Update(2,2);
13   WhiteLineReaction(&ScaleL, &ScaleR, &
       FLAG_RunFromLine);
14   if(!FLAG_RunFromLine){
15     TargetEnemy(ScaleL,ScaleR);
16   }
17   else
18   {
19     RunFromLine(&Counter, &FLAG_RunFromLine);
20   }
21 }
22 }
```

Póki na pinie START jest stan wysoki to walka trwa. State_Update z podanym argumentem aktualizuje odczyty z czujników oraz reguluje prędkość na podstawie pewnej wartości ustawionej w strukturze MOTOR_CONTROL.

Wartość ta jest zmieniana w funkcjac TargetEnemy oraz RunFromLine. WhiteLineReaction decyduje jak ma zachować się robot mając aktualne odczyty z czujników KTIR - przyjmijmy w skrócie oznaczenie [lewy czujnik , prawy czujnik] i wartości 0 albo 1 (nie widzi albo widzi białą linię):

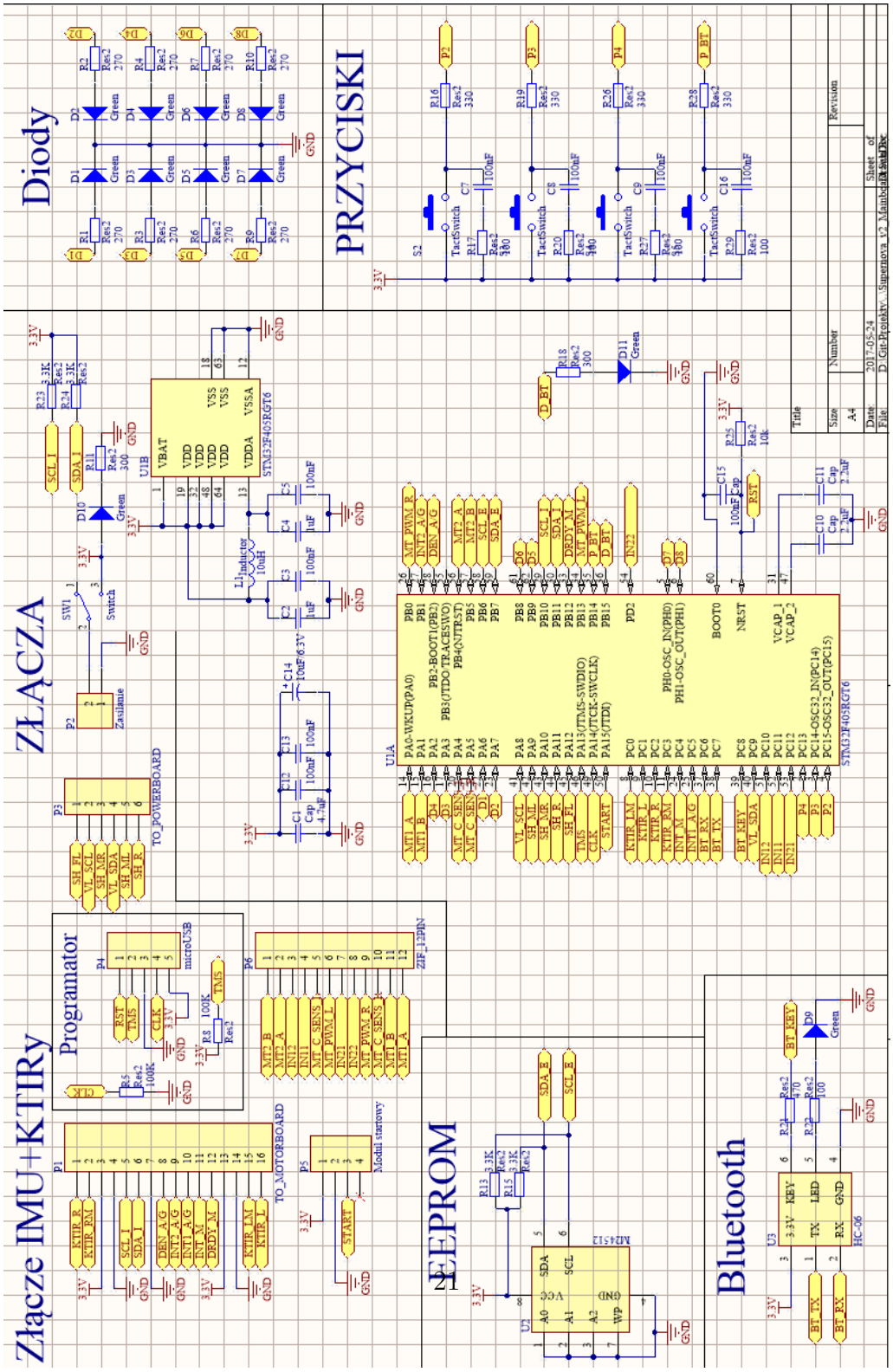
- $[0, 0]$ - nie rób nic
- $[1, 0]$ - osłab lewy silnik (zmniejsz ScaleL) - ma to służyć obróceniu robotów w zwarciu tak, aby przeciwnik ześliznął się w kierunku białej linii. Działa to lepiej niż zwiększenie prędkości na przeciwnym silniku.
- $[0, 1]$ - osłab prawy silnik
- $[1, 1]$ - ustaw flagę RunFromLine

Jeśli zostanie ustawiona flaga RunFromLine to robot wykonuje pewną sekwencję czynności służącą odjechaniu od niej - cofa się, obraca o około 180 stopni i jedzie do przodu z małą prędkością. Od momentu kiedy robot ma jechać do przodu reaguje on już znowu na odczyty z czujników odległości i szuka przeciwnika.

Funkcja TargetEnemy zawiera *"regulator IF"* tj. dużą ilość instrukcji warunkowych służących nadaniu odpowiednich nastaw prędkości dla każdej z możliwych sytuacji. Jest to metoda bardzo żmudna w kalibracji, lecz póki co okazuje się najskuteczniejsza i najmniej narażona na błędy.

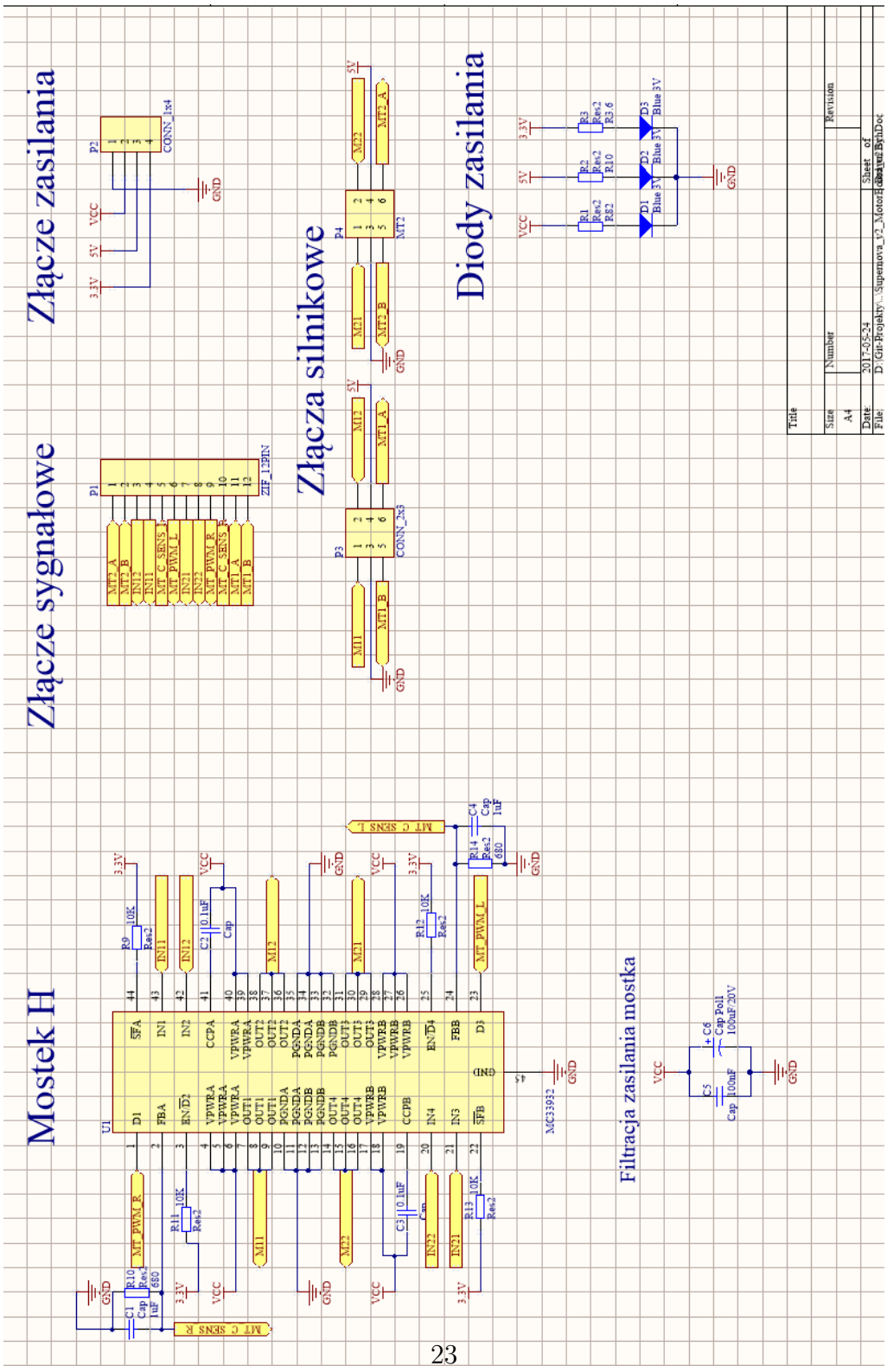
8 Elektronika

8.1 Płytka główna



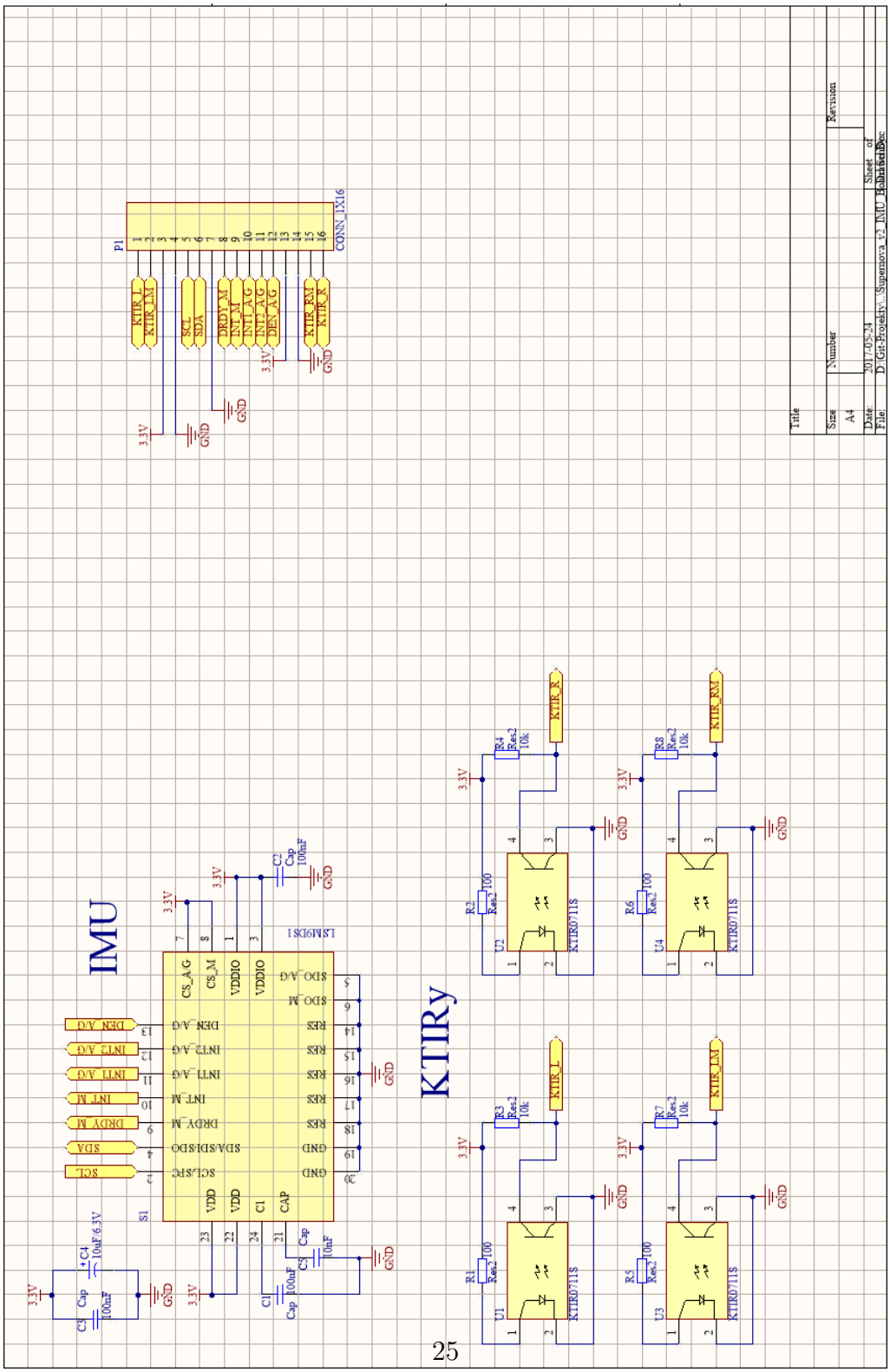
Rysunek 6: Płytka główna robota

8.2 Sterownik silników



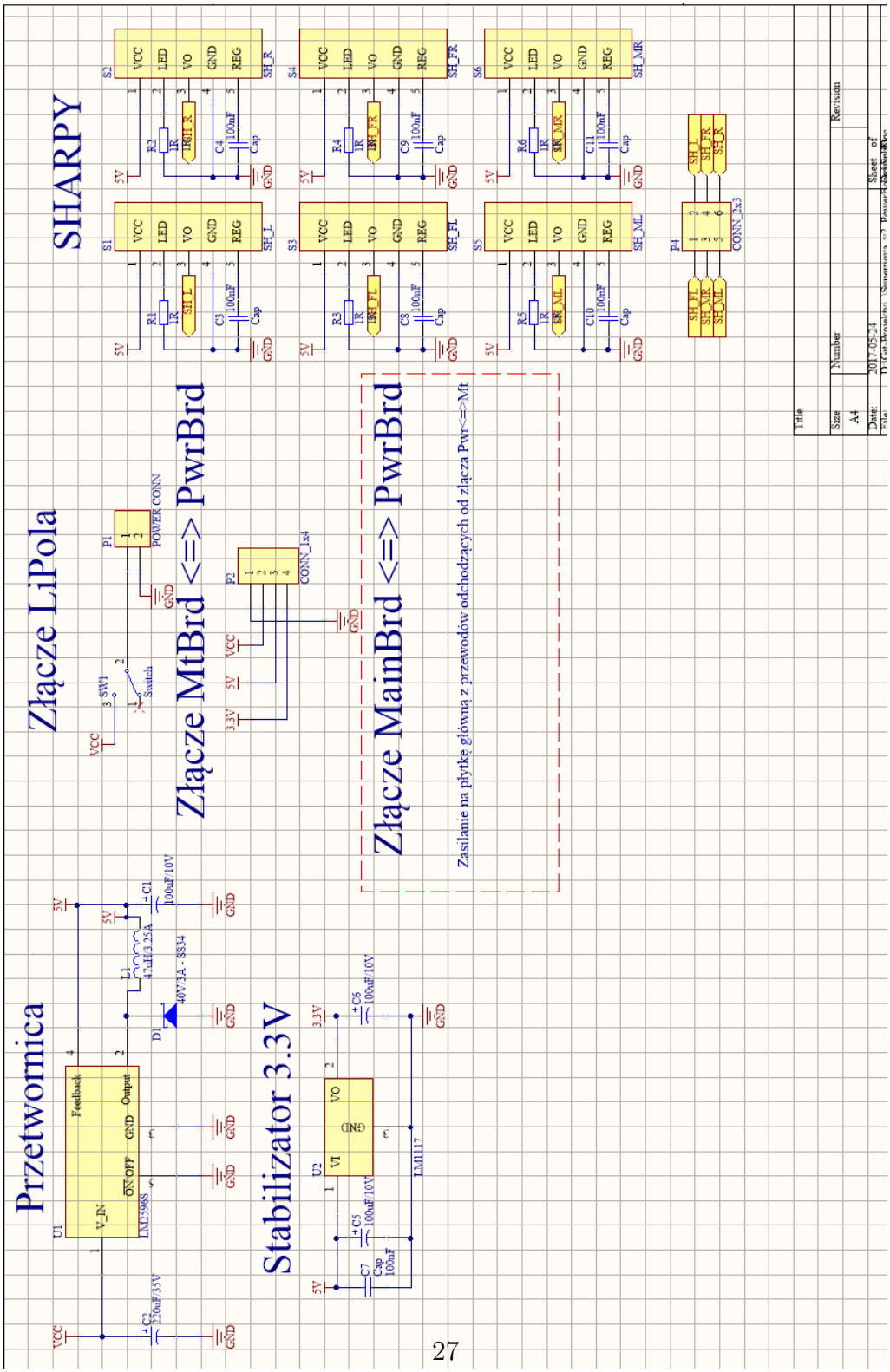
Rysunek 7: Płytka sterownika silników

8.3 Płytka czujników KTIR oraz IMU

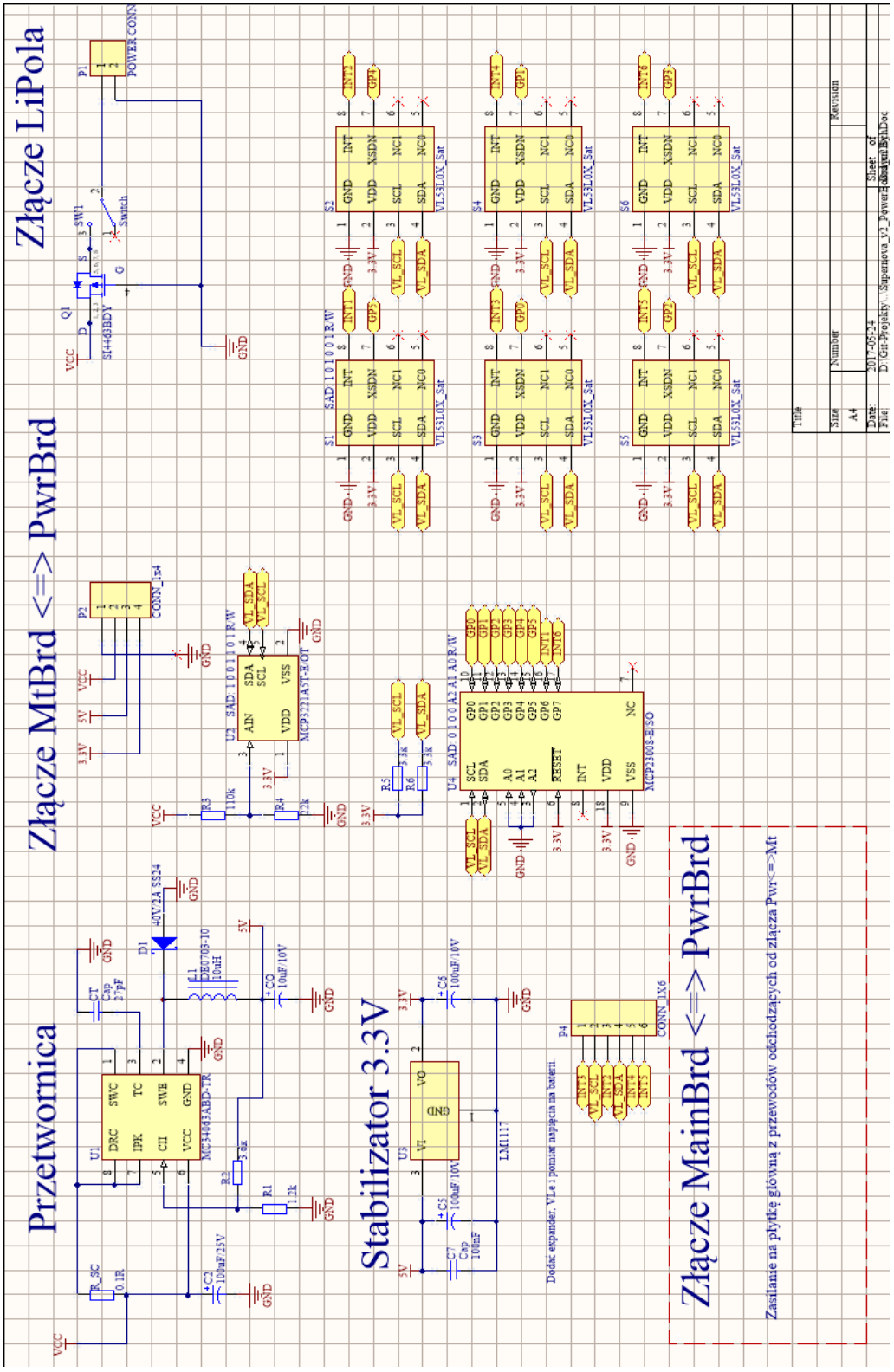


Rysunek 8: Płytka czujników KTIR oraz IMU

8.4 Płytki modułu zasilania oraz czujników odległości



Rysunek 9: Pierwsza wersja modułu zasilania, na której znajdują się czujniki SHARP GP2Y0D340K



Rysunek 10: Druga wersja modułu zasilania. Obecne czujniki VL53L0X

8.5 Opis

W trakcie eksploatacji robota oraz prób implementacji nowych funkcjonalności nasunęło mi się parę uwag, które warto tutaj zamieścić.

I2C

Ryzykownym z mojej strony posunięciem było umieszczenie 6 czujników, ekspandera oraz przetwornika A/D na jednej linii I2C bez wcześniejszego upewnienia się, że pojemność linii nie będzie zbyt duża. Na szczęście okazało się, że nie była i przy dobranych rezystorach podciągających linia działała bez zarzutu. Warto jednak zwrócić na to uwagę projektując układ wykorzystujący tą magistralę.

Taśmy FFC

O ile pomysł wykorzystania tych taśm w robocie wydawał się dosyć dobry, o tyle w trakcie eksploatacji robota okazały się one najbardziej problematyczne. Zdarza się, że upakowana w robocie taśma za bardzo się wygnie, przez co ścieżki w niej pękną lub przetrze się o jakiś element konstrukcji robota. Złącza FPC również potrafią się wyrobić przez co taśma może momentami nie stykać i potrzebne jest przelutowanie gniazda.

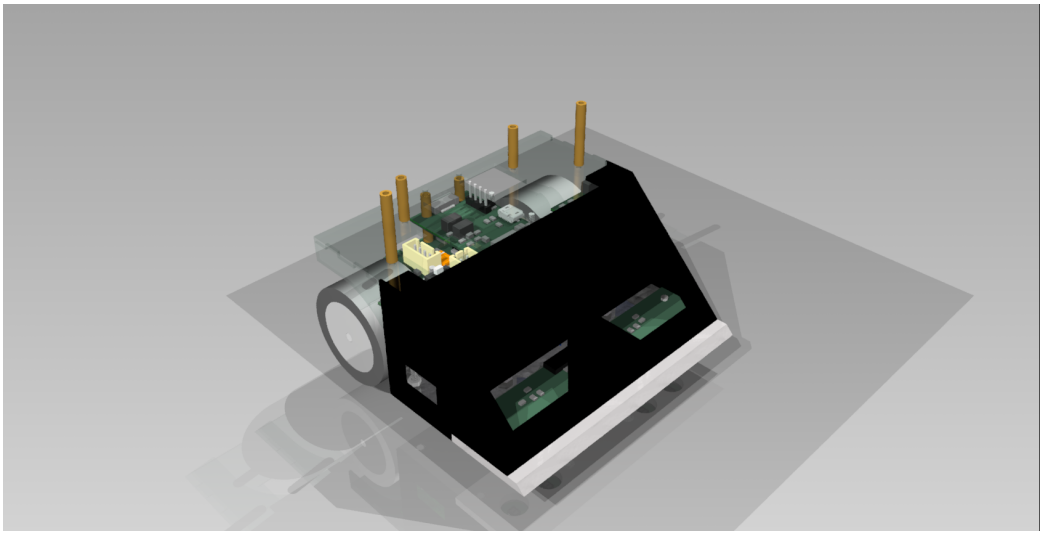
Programator

Dobrym rozwiązaniem jest wyprowadzenie programatora na gniazdo microUSB lub miniUSB. Mają one dosyć niewielkie rozmiary, łatwo się do nich podpiąć i zapewniają ochronę przed odwrotnym wsadzeniem wtyku.

Dobór mostka H

Warto przyłożyć się do wyboru mostka H, jeśli zależy nam na pomiarze prądu. W wypadku tego projektu nie najlepszym pomysłem było dobranie mostka z maksymalnym prądem 5A na kanał, gdyż przy dobieraniu rezystora wpływającego na maksymalne napięcie na pinie pomiaru prądu musiałem dokonać wyboru między rozdzielczością ADC, a liniowością pomiaru. Nota katalogowa obiecuje liniową pracę pomiaru prądu dla rezystora w zakresie $100\Omega < R_{FB} < 300\Omega$, lecz przy wartości 300Ω i maksymalnym prądzie silnika robota ($1.7A$ przy napięciu $16.8V$) napięcie na tym pinie wynosiłoby ok. $1.25V$, co daje jedynie 38% wykorzystanie zakresu ADC.

9 Mechanika



Rysunek 11: Model robota

W skład robota wchodzi:

- Podstawa stalowa składająca się z blach ze stali nierdzewnej o grubości 3mm oraz 1mm
- Nóż do strugarki o wymiarach 100x30x3 mm ze wstawką z węglika spiekanego (ostrze). Pod koniec projektu zastąpiony nożem o tych samych wymiarach wykonanym ze stali szybkotnącej (HSS).
- Obudowa z laminatu
- Dach z laminatu
- Silikonowe opony
- Aluminiowe felgi
- Silniki Faulhaber 1524T009SR z przekładnią Faulhaber 15A 19:1 oraz enkoderem Faulhaber IE512
- 6 czujników SHARP GP2Y0D340K lub 6 czujników VL53L0X (w zależności od wersji płytki)
- Drukowane w 3D mocowania na silniki

- Szeregowo połączone dwa akumulatory litowo-polimerowe Dualsky 2s 220 mAh

Dodatkowo warto zaznaczyć, że w podstawie robota zostało wywiercone i wypiłowane miejsce na złącze do ładowania akumulatorów, które schowane są we wnętrzu robota.

10 Zadania niezrealizowane

Do tej pory nie zostały jeszcze zrealizowane:

- Pomiar napiecia baterii zewnętrznym przetwornikiem ADC - komunikacja z układem po magistrali I2C
- Algorytm sterowania momentem obrotowym

11 Podsumowanie

Niestety w projekcie nie udało się dokończyć jednego z założonych zadań tj. sterowania momentem obrotowym. Jest to jednak wykonalne w krótkim czasie, gdyż wszystkie potrzebne struktury są już zaimplementowane, włącznie z filtracją pomiaru. Projekt będzie dalej rozwijany przez okres wakacyjny, aby stanąć w jeszcze lepszej formie na zawodach w nadchodzącym sezonie jesienno-zimowym.

12 Bibliografia

- Kurs STM32 F4 - Bartłomiej Kurosz
- Stack Overflow
- Materiały na stronie ST Microelectronics
- Dokumentacja silnika