Projekt

Sterowniki robotów

Raport końcowy

Robot mobilny micromouse Miki

Skład grupy (1):
Dominik Kędzierski, 218241
Piotr Matuszak, 218582

Termin: srTN13

Prowadzący:
mgr inż. Wojciech Domski

19 czerwca 2017

Spis treści

Spis tablic

Spis rysunków

Lista listingów

1 Wstęp

Dokument zawiera raport końcowy z konstrukcji robota kategorii micromouse poruszającego się w labiryncie. Osiągnięto drugi kamień milowy - robot porusza się w labiryncie.

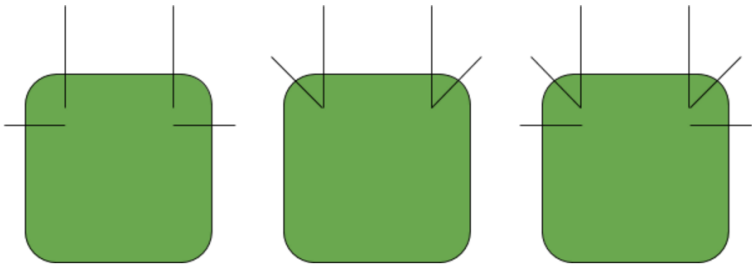
2 Założenia konstrukcyjne

Konstrukcję ograniczają w dużej mierze regulamin konkurencji micromouse[1] oraz sam labirynt. Głównym parametrem labiryntu jest wewnętrzna wielkość pojedynczej kratki wynosząca około 168 mm. Aby móc poruszać się w labiryncie pokonując zakręty robot musi być oczywiście mniejszy. Ponadto jeśli chcemy przemieszczać się również po skosie, maksymalna szerokość robota może wynosić 110 mm. [1, s. 14]. Poza tym należy przeprowadzić optymalizację wagi robota aby poruszał się on szybko.

Wziąć pod uwagę należy również umiejscowienie i kierunek patrzenia czujników odległości. Istnieją 3 podstawowe typy układów takich czujników (Rys. 1):

- 4 czujniki, 2 czujniki patrzące do przodu, po jednym patrzącym na boki,
- 4 czujniki, 2 czujniki patrzące do przodu, po jednym patrzącym pod kątem na boki,
- 6 czujników, 2 patrzące do przodu, po jednym patrzącym na boki z każdej strony i po jednym patrzącym pod kątem na boki.

W tej konstrukcji zdecydowano się na wykorzystanie drugiego układu. Pozwala on na zebranie informacji o obecności ścian bocznych w kratce znajdującej się przed tą, w której znajduje się robot i wcześniejsze planowanie ruchu. Konstrukcja robota jest platformą klasy (2,0). Klasa ta wyróżnia się prostotą sterowania. Jej nieodporność na nierówności terenu nie stanowi problemu, ponieważ konkurencja micromouse odbywa się w „laboratoryjnych" warunkach - podstawę labiryntu stanowi zawsze płaska czarna płyta, pozbawiona nierówności.



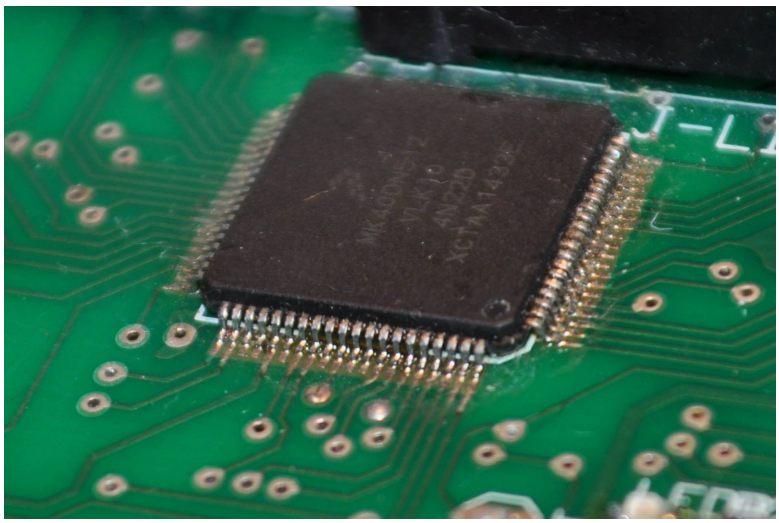
Rysunek 1: Sposoby rozmieszczenia czujników w robotach typu micromouse

3 Wybór elementów elektronicznych

3.1 Mikrokontroler

Zgodnie z założeniami projektowymi w robocie został wykorzystany 80-pinowy mikrokontroler NXP K40 (Rys. 2), taktowany zewnętrznym oscylatorem o częstotliwości 100 MHz. Dzięki posiadanemu przez niego dekoderowi kwadraturowemu można w łatwy sposób skorzystać z wykorzystanych w projekcie enkoderów AS5040. Pozwala również między innymi na komunikację z wykorzystaniem interfejsu UART (wykorzystane przy komunikacji Bluetooth), I2C (żyroskop, akcelerometr) i JTAG (programowanie).

[1] `http://roboticarena.pl/static/Main/regulaminy/Micromouse.pdf`



Rysunek 2: Mikrokontroler na płytce

3.2 Sterowanie

3.2.1 Sterownik silników

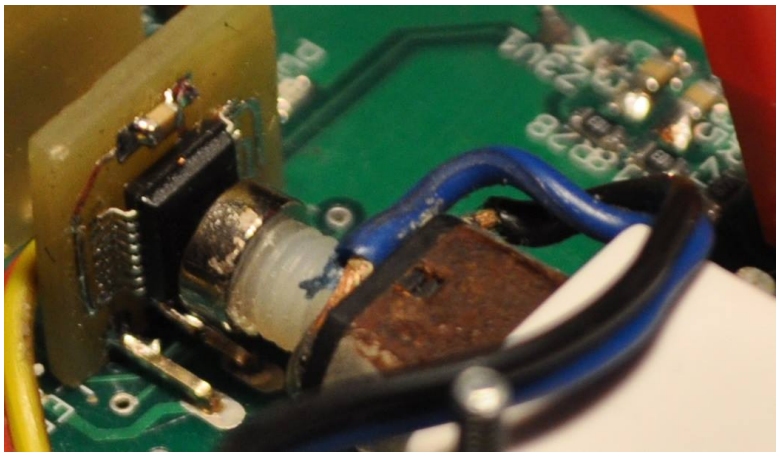
Do sterowania silnikami wykorzystano popularny podwójny mostek H TB6612. Do sterowania silnikami wykorzystuje się w nim 2 PWM (po jednym na każdy silnik) do sterowania prędkością i 4 GPIO (po dwa na silnik) do sterowania kierunkiem obrotu.

3.2.2 Silniki

W robocie wykorzystano silniki Pololu z przekładnią 10:1 i maksymalnym RPM 1000 z wysuniętą osią w celu zamocowania magnesów do enkoderów. Głównym ich atutem jest duża dostępność.

3.3 Sensory wewnętrzne

3.3.1 Enkodery



Rysunek 3: Enkoder magnetyczny

Do pomiaru kąta obrotu kół wykorzystano 10-bitowe enkodery magnetyczne AS5040 firmy AMS o rozdzielczości 1024 impulsów na obrót (Rys. 3).

3.3.2 Układ pomiaru napięcia baterii

Do pomiaru napięcia baterii zbudowano odpowiednio zabezpieczony dzielnik napięcia Pomiar dokonywany jest przez ADC mikrokontrolera. Stan napięcia na baterii jest wyświetlany na czterech różnokolorowych diodach umieszczonych na płytce.

3.4 Sensory zewnętrzne

3.4.1 Żyroskop i akcelerometr

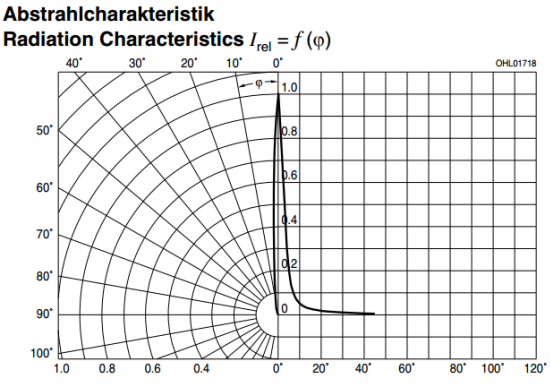
Został wykorzystany moduł MPU-6050 z 3 osiowym akcelerometrem i 3-osiowym żyroskopem. Komunikacja z mikrokontrolerem odbywa się za pomocą interfejsu I2C. Zostanie on użyty do pomiaru poślizgu i kąta obrotu robota. Podłączany jest poprzez złącze goldpin.

3.4.2 Czujniki odległości od ścian labiryntu

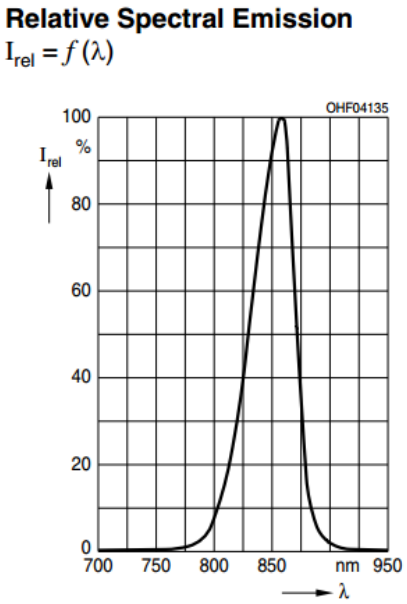
Wykonano własne czujniki odległości robota od ścian labiryntu. Nie skorzystano z gotowych rozwiązań, ponieważ dostępne układy czujników mierzących odległość: nie spełniają wymagań dotyczących zasięgu, są zbyt duże na potrzeby robota (rozmiar robota musi mieścić się w określonych granicach, ponadto masa robota jest istotna) jak analogowe czujniki Sharp lub są drogie (TOF).

Czujniki składają się z diody podczerwonej oraz fototranzystora. Światło rzucane przez diody odbija się od zawsze białych ścian labiryntu i trafia do fototranzystora. Na podstawie ilości odebranego światła wnioskować się będzie o odległości robota od ścianek labiryntu.

Do budowy czujników wykorzystano diody podczerwone SFH4550 [2] o długości fali w piku emisji 860 nm (Rys. 4b) i wąskim kącie nadawania wynoszącym 6 stopni (Rys. 4a). Mały kąt nadawania związany jest z wymaganą dużą precyzją czujników. Niewielki kąt pozwala na wyeliminowanie odbić z sąsiednich diod, które mogłyby wpłynąć na pomiar.



(a) Kąt nadawania



(b) Długość fali

Rysunek 4: Właściwości diody IR

Niewielki kąt nadawania wymaga jednak dużej jasności diod. Przez diody przepływa dość spory prąd (nawet do 1A). Stąd też w celu ochrony mikrokontrolera w sterowaniu diodami

pośredniczy układ ULN2003AD, w którego wnętrzu znajduje się 7 układów Darlingtona. Mikrokontroler steruje bazą układu tranzystorów. Każda nóżka podłączona do układu steruje oddzielną diodą. Diody podłączone są do kolektorów, emiter jest wspólny dla wszystkich układów Darlingtona wewnątrz układu.

Do odbioru światła odbitego od ścian labiryntu użyto fototranzystorów VISHAY BPW96C [3] o długości fali w punkcie maksymalnej czułości 850 nm (Rys. 5b) i kącie widzenia 20 stopni (Rys. 5a). Ważne jest, aby długość fali nadawanej i długość fali odbieranej była możliwie blisko siebie. Kąt odbierania musi być szerszy niż przy diodzie nadawczej, aby pozwalał on na zebranie odbitego sygnału.

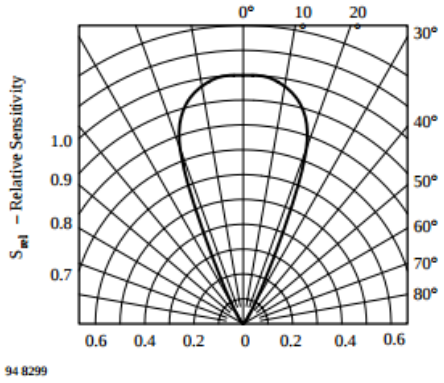


Figure 9. Relative Radiant Sensitivity vs. Angular Displacement

(a) Kąt odbierania

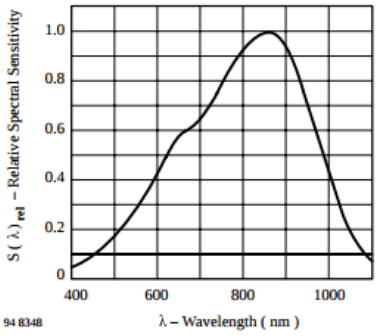
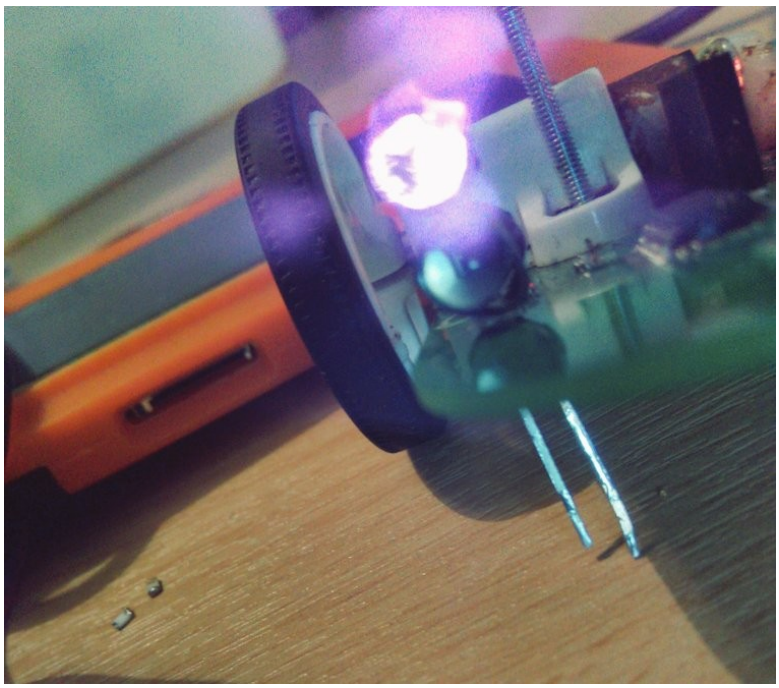


Figure 8. Relative Spectral Sensitivity vs. Wavelength

(b) Długość fali odbieranej

Rysunek 5: Właściwości fototranzystora

Pomiaru odległości dokonuje się poprzez pomiar napięcia na fototranzystorze przy wykorzystaniu ADC mikrokontrolera. Wymagającym zadaniem okazał się taki dobór rezystorów na fototranzystorze, aby największa zmiana mierzonego napięcia odbywała się w pożądanej odległości tj. w zakresie 2-8 cm od czujnika. Prowadzony będzie pomiar różnicowy w celu wyeliminowania składowej stałej (różnica między napięciami zmierzonymi przy zapalonej i zgaszonej diodzie). Nie stanowić będzie również problemu różnica w odczycie przy różniących się kolorach i fakturach powierzchni, ponieważ ściany labiryntów są zawsze białe i jednakowe.



Rysunek 6: Działający układ pomiarowy

3.5 Komunikacja

3.5.1 Złącze programatora

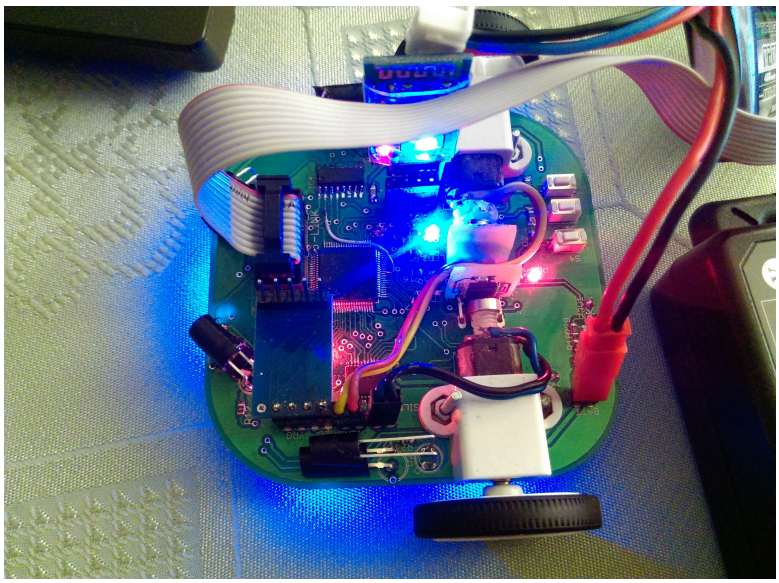
Procesor programowany jest przy wykorzystaniu interfejsu JTAG. Do programowania wykorzystywana jest płytka ewaluacyjna. Interfejs pozwala na przeprowadzenie debugowania programu na mikrokontrolerze podczas jego działania oraz podgląd i edycję zmiennych w środowisku CodeWarrior.

3.5.2 Moduł Bluetooth

Wykorzystano moduł Bluetooth HC-06, pracującym w trybie Slave. Pozwoli on na transmisję danych z robota do komputera, sterowanie robotem i debugowanie programu. Moduł jest wpięty w płytkę z wykorzystaniem czteropinowego złącza goldpin.

3.5.3 Diody i przyciski

Dla potrzeb debugowania programu oraz sygnalizacji stanu robota uwzględniono w projekcie 4 diody LED sygnalizujące odbiór danych z czujników lub inne parametry, 4 diody sygnalizujące stan napięcia baterii, 6 diod służących jako dekoracja oraz dioda informująca o podłączonym zasilaniu. Ponadto na robocie znajdują się 3 przyciski.



Rysunek 7: Zapalone diody w robocie

4 Program robota

Program robota napisany został w języku C w środowisku CodeWarrior udostępnianym przez producenta. Program ten:

- wykonuje pomiar napięcia baterii,
- oblicza PID dla kontroli trakcji,
- obsługuje komunikację poprzez interfejs UART,
- przeprowadza pomiar z czujników odległości od ścian,
- uruchamia algorytm floodfill,
- podejmuje decyzję o ruchu,
- wykonuje decyzję o ruchu,
- aktualizuje mapę labiryntu.

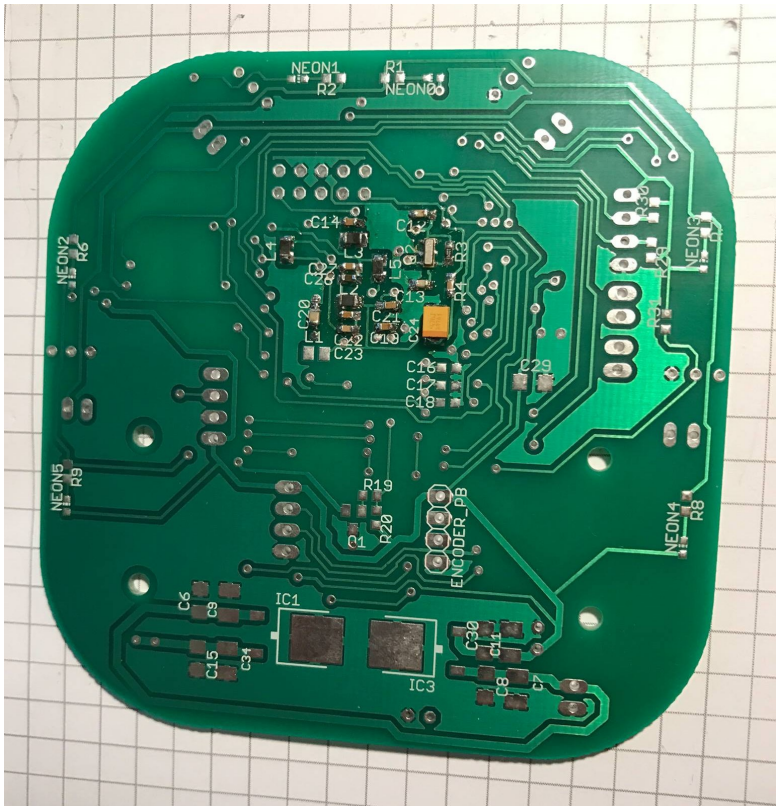
Listing 1: Główna pętla programu

```
1     for(;;) {
2     if (ADC0_end) {
3       MyADC0_Measure(FALSE);
4       ADC0_end = FALSE;
5     }
6     if (ADC1_end) {
7       MyADC1_Measure(FALSE);
8       ADC1_end = FALSE;
9     }
10
11    if (time) {
12      if(button) {
13        NEON_ClrVal();
14      }
15      else {
```

```
16            NEON_SetVal();
17            Calc_Motor_PID();
18          }
19
20          communication();
21          rangeMeasuring();
22          //-------------------------------------------------
23
24          if(decision==TRUE && allow_altmove==FALSE){
25              for(i=0;i<1;i++) {
26                  floodFillStack();
27              }
28              for(i=0;i<16;i++) {
29                  floodFillBruteforce();
30              }
31          makedecision();
32          }
33          test_altmove_dir();
34          addWalls();
35          time=0;
36      }
37      CheckPower();
38  }
```

5 Schemat elektryczny i projekt PCB

Zarówno schemat elektryczny jak i projekt PCB zostały wykonane w programie Eagle. Płytka PCB musi spełniać konkretne wymagania dotyczące wielkości robota. Udało się osiągnąć niewielką szerokość robota wraz z kółkami, jednak jest ona niewiele mniejsza niż 110 mm, stąd robot będzie miał kłopoty z poruszaniem się na ukos. Problem w utrzymaniu małej szerokości stanowiły użyte silniki, które wraz z układem enkoderów magnetycznych powodowały trudności ze zmieszczeniem się w pożądanej szerokości. W przyszłych konstrukcjach planowane jest wykorzystanie mechanizmu przekładni i umieszczenie silników obok siebie.

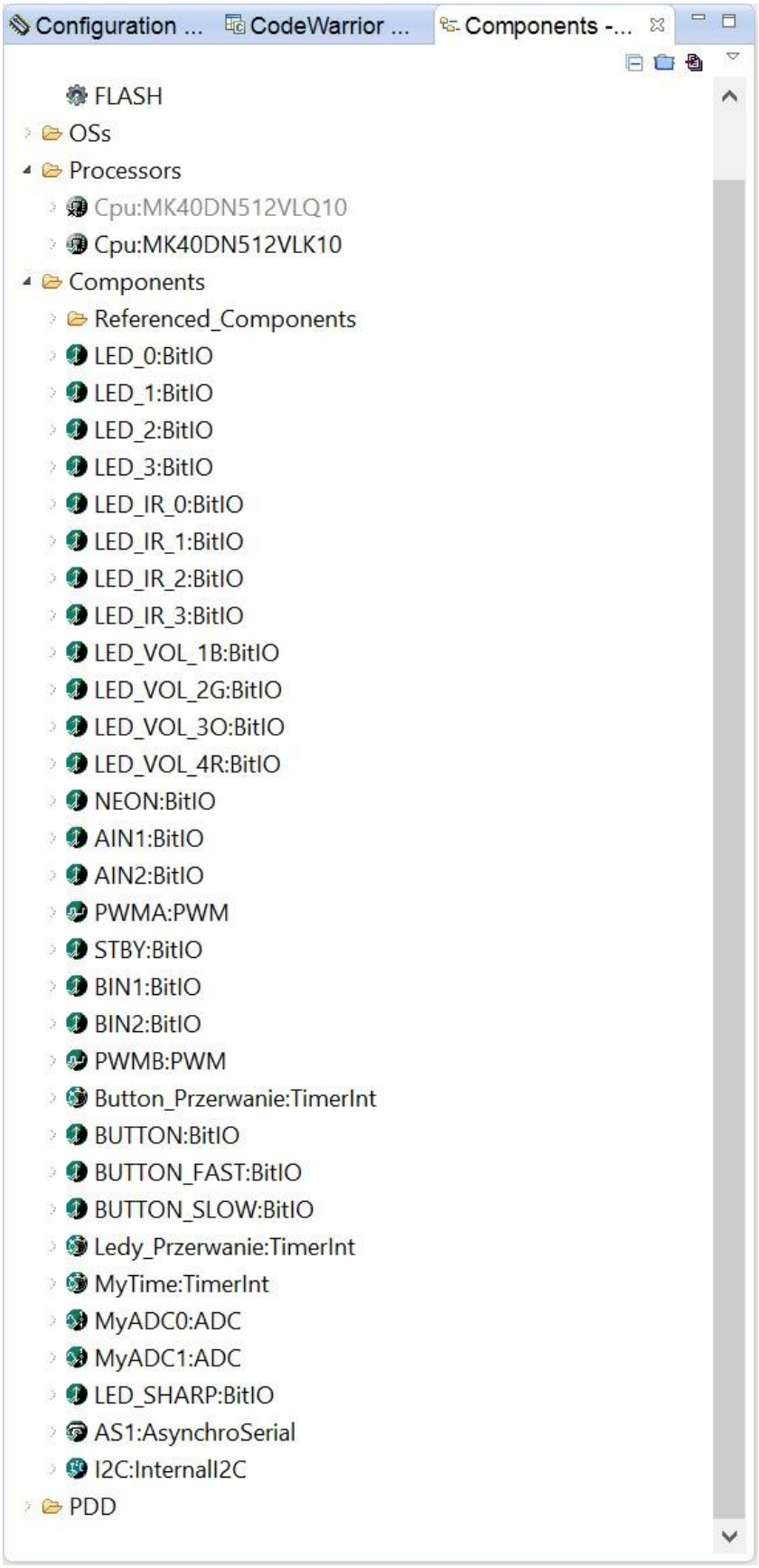


Rysunek 8: PCB robota w trakcie lutowania

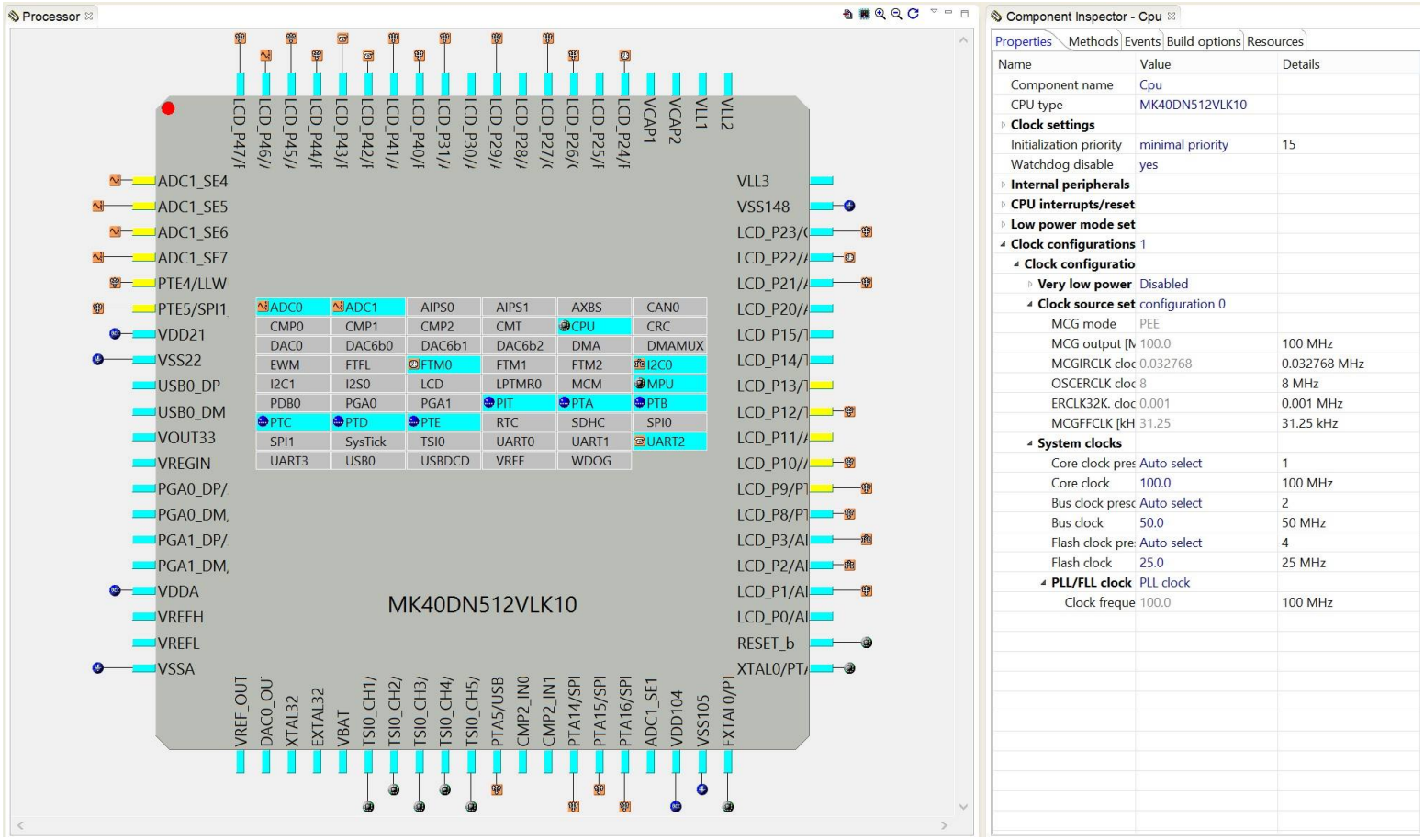
6 Obsługa sensorów

Rysunek 9 przedstawia użyte komponenty w mikrokontrolerze w postaci fasolek (beans). Używanych jest 22 wejść/wyjść liniowych do obsługi mostka H, diod LED, diod podczerwonych oraz przycisków. Dodatkowo do kontroli prędkości silników wykorzystuje się 2 wyjścia PWM. Okres sygnału PWM wynosi 100μs (zegar FTM1). Wykorzystuje się 3 przerwania czasowe - do debouncingu przycisków, kontroli czasu włączenia czujników odległości oraz wykonywania głównej pętli programu co określony czas. Wykorzystywane są 2 kanały ADC do fototranzystorów i do pomiaru napięcia baterii. Włączony został też interfejs UART do komunikacji z robotem. Konfigurację pinów mikrokontrolera przedstawia rysunek 10 i tablica 1.

Główna pętla programu jest wywoływana co 5ms dzięki wkorzystaniu przerwania pochodzacego od zegara PIT.



Rysunek 9: Konfiguracja wejść i wyjść



Rysunek 10: Konfiguracja pinów, zegara i peryferiów mikrokontrolera

Tablica 1: Konfiguracja pinów

Pin	Nazwa pinu	Funkcja	Komentarz	Pin	Nazwa pinu	Funkcja	Komentarz
1	PTE0	ADC1	Fototranzystor	41	XTAL	Kwarc	Zewnętrzny oscylator
2	PTE1	ADC1	Fototranzystor	42	RESET		
3	PTE2	ADC1	Fototranzystor	44	PTB1	I/O	Darlington
4	PTE3	ADC1	Fototranzystor	45	PTB2	I2C	SCL
5	PTE4	I/O	Sterowanie mostka H	46	PTB3	I2C	SDA
6	PTE5	I/O	LED	47	PTB8	I/O	Sterowanie mostka H
7	VDD	Power	Chip Power Supply	48	PTB9	I/O	Sterowanie mostka H
8	VSS	Power	Chip Ground	49	PTB10	I/O	Darlington
17	VDDA	Power	Chip Power Supply	51	PTB16	I/O	Sterowanie mostka H
20	VSSA	Power	Chip Ground	53	PTB18	Dekoder kwadraturowy	Enkoder
25	VBAT		3V3	54	PTB19	Dekoder kwadraturowy	Enkoder
26	PTA0	JTAG	TCK	56	PTC1	I/O	Sterowanie mostka H
27	PTA1	JTAG	TDI	57	PTC2	PWM	Sterowanie mostka H
28	PTA2	JTAG	TDO	58	PTC3	I/O	Sterowanie mostka H
29	PTA3	JTAG	TMS	59	VSS	Power	Chip Ground
31	PTA5	I/O	Darlington	65	PTC4	PWM	Sterowanie mostka H
32	PTA12	Dekoder kwadraturowy	Enkoder prawy	67	PTC6	I/O	Przycisk
33	PTA13	Dekoder kwadraturowy	Enkoder prawy	68	PTC7	I/O	LED czujnik
35	PTA15	I/O	LED	70	PTC9	I/O	LED Neon
36	PTA16	I/O	LED	72	PTC11	I/O	Darlington
38	VDD	Power	Chip Power Supply	73	PTD0	I/O	Przycisk
39	VSS	Power	Chip Ground	74	PTD1	I/O	Przycisk
40	EXTAL	Kwarc	Zewnętrzny oscylator	75	PTD2	UART	RX
				76	PTD3	UART	TX
				77	PTD4	I/O	LED (dot. czujników)
				78	PTD5	I/O	LED (dot. czujników)
				79	PTD6	ADC	Napięcie baterii
				80	PTD7	I/O	LED (dot. czujników)

6.1 Obsługa sensorów wewnętrznych

6.1.1 Pomiar napięcia baterii

Funkcja `CheckPower` przedstawiona na listingu 2 posiada 4 progi napięcia. Gdy napięcie jest większe niż 4V na cele, zapalone są 4 diody. Poniżej 4V gaśnie jedna dioda. Gdy napięcie spadnie poniżej 3,9V zgaszone są dwie diody, a poniżej około 3,7V gasną 3 diody. Poniżej 3,6V zgaszone są wszystkie diody.

Listing 2: Funkcja CheckPower

```
void CheckPower(){
    if (voltage>33950) LED_VOL_4R_ClrVal(); else LED_VOL_4R_SetVal();
    if (voltage>34800) LED_VOL_3O_ClrVal(); else LED_VOL_3O_SetVal();
    if (voltage>35490) LED_VOL_2G_ClrVal(); else LED_VOL_2G_SetVal();
    if (voltage>35890) LED_VOL_1B_ClrVal(); else LED_VOL_1B_SetVal();
}
```

6.1.2 Odczyt danych z enkoderów

Wykorzystywane enkodery (AS5040) posiadają wyjście kwadraturowe. Użyty mikrokontroler posiada dwa dekodery kwadraturowe. To peryferium trzeba samemu włączyć w kodzie programu. Enkodery są wykorzystywanie do utrzymania odpowiedniej pozycji oraz prędkości robota. Na podstawie ilości impulsów mamy możliwość kontrolowania przemieszczenia oraz kąta

obrotu robota. Do kontroli obrotu silników wykorzystywany jest algorytm PID. Sprawdza się on bardzo dobrze.

6.2 Obsługa sensorów zewnętrznych

6.2.1 Odczyt danych z czujników odległości

Aby poprawnie zmierzyć odległość potrzebny jest pomiar różnicowy w celu wyeliminowania składowej stałej z otoczenia. Zaimplementowano algorytm włączający diodę IR (uwzględniający opóźninienie reakcji układu Darlingtona) Po odczekaniu 20ms zbierany jest pomiar z włączonej diody następnie przez 20ms dioda jest wyłączana. Po upływie tego czasu zbierany jest ponownie pomiar. Różnica między obywdoma pomiarami jest odczytem z czujników pomiaru odległości. Funkcja odpowiadająca za zebranie danych z czujników przedstawiona jest na listingu 3. Pomimo zastosowania pomiaru różnicowego dane z czujników były zaszumione. Okazało się niezbędne zastosowanie filtra medianowego, który poprawił stabilność odczytów. Planuje się zaimplementowanie filtra dolnoprzepustowego w celu osiągnięcia lepszych efektów.

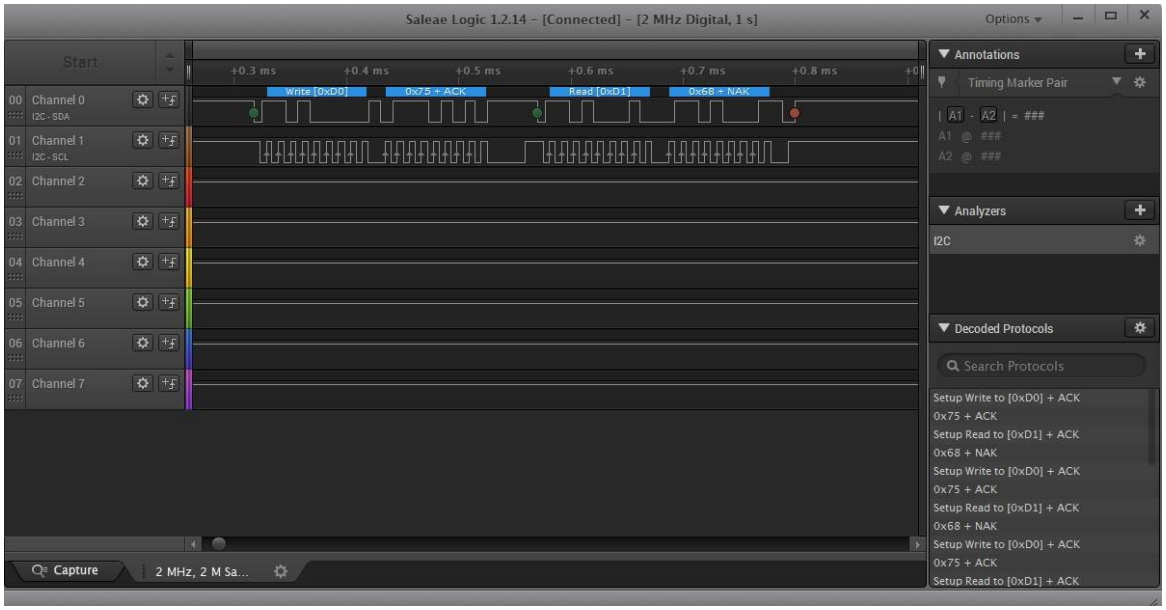
Listing 3: Funkcja rangeMeasuring

```
void rangeMeasuring() {
        extern int loop;
        extern word ledoff_val[];
        extern word ledon_val[];
        extern int led_val[];
        extern uint16 sensor_off[];
        extern uint16 sensor_on[];
        extern uint16 sensor[4];
        loop=loop+1;
        if (loop == 1) {
            LED_IR_0_SetVal();
            LED_IR_3_SetVal();
            LED_IR_1_ClrVal();
            LED_IR_2_ClrVal();
        }
        if (loop == 20) {
            ledon_val[0] = sensor[0];
            ledon_val[3] = sensor[3];
            ledoff_val[1] = sensor[1];
            ledoff_val[2] = sensor[2];
        }
        if (loop == 21) {
            LED_IR_0_ClrVal();
            LED_IR_3_ClrVal();
            LED_IR_1_SetVal();
            LED_IR_2_SetVal();
        }
        if (loop == 42) {
            ledoff_val[0] = sensor[0];
            ledoff_val[3] = sensor[3];
            ledon_val[1] = sensor[1];
            ledon_val[2] = sensor[2];
            if((int)ledoff_val[0]>(int)ledon_val[0]){
            led_val[0] = (int)ledoff_val[0] - (int)ledon_val[0];}
            else{led_val[0]=0;}
            if((int)ledoff_val[1]>(int)ledon_val[1]){
            led_val[1] = (int)ledoff_val[1] - (int)ledon_val[1];}
            else{led_val[1]=0;}
            if((int)ledoff_val[2]>(int)ledon_val[2]){
            led_val[2] = (int)ledoff_val[2] - (int)ledon_val[2];}
```

```
41                 else{led_val[2]=0;}
42                 led_val[3] = (int)ledoff_val[3] - (int)ledon_val[3];
43                 loop = 0;
44             }
45 }
```

6.2.2 Obsługa akcelerometru i żyroskopu

W projekcie wykorzystany został układ MPU-6050, na którego pokładzie znajduje się akcelerometr i żyroskop. Komunikacja z układem odbywa się za pomocą interfejsu I^{2}C.



Rysunek 11: Pogląd odpowiedzi z rejestru WHOAMI w analizatorze stanów logicznych

Adresem urządzenia jest `0x68`. W pierwszej kolejności wysyłany jest sygnał WAKEUP poprzez wpisanie do rejestru `0x6B` wartości `0x00` (Listing 4).

Listing 4: Przykład odbioru odczytów osi x akcelerometru

```
1   char data_i2c[2]={0x6B,0x00};
2
3   I2C_SelectSlave(0x68);
4   I2C_SendBlock(data_i2c,2,&tmp);
5   I2C_SendStop();
```

Następnie odczytywane są wartości wysokiego i niskiego bitu danych z osi `X` akcelerometru (przebiegającej wzdłuż robota od jego przodu do tyłu) i osi `Z` żyroskopu (normalna do powierzchni płytki). Przykładowy odczyt danych oraz ich złożenie zostało przedstawione na listingu 5.

Obecnie jednak odbierane dane nie są wykorzystywane w programie robota. Ponadto zauważono znaczne skoki wartości i brak stabilnych odczytów.

Listing 5: Przykład odbioru odczytów osi x akcelerometru

```
1   I2C_SelectSlave(0x68);
2   I2C_SendChar(0x3B);
3   I2C_RecvChar(&reciv);
4   high=reciv;
5
6   I2C_SendChar(0x3C);
```

```
7   I2C_RecvChar(&reciv);
8   I2C_SendStop();
9   low=reciv;
10  temp_rec = high<<8;
11  data_acc_x = temp_rec | low;
```

7 Komunikacja

Do debugowania, sprawdzania stanu czujników, oraz poprawności obliczania pozycji w labiryncie itp. wykorzystano interfejs UART (baud rate: 9600). Napisano aplikacje w qt, która otwiera serialport i odczytuje potrzebne dane. Transmisja danych jest realizowana poprzez moduł Bluetooth HC-06. Stworzono protokół danych składający się z 12 bajtów. Przesyłane dane zostały przedstawione w tabeli 2 (odbieranie) i 3 (wysyłanie).

Tablica 2: Protokół transmisji od robota

Bajt	Funkcja	Opis	Przykładowe wartości
0	Start	bajt rozpoczynający transmisje	0xFF
1	RX/TX	bajt decydujący czy jest potrzeba sterowania robotem, czy odebrania od niego danych	0x02/0x01
Dalej przedstawiana jest ramka odbieranych danych (czyli po wysłaniu żądania 0x02 w 1 bajcie transmisji)			
2	Lewy przedni czujnik odległości	Dane z czujnika odległości	0x00 - 0xFF
3	Lewy boczny czujnik odległości	Dane z czujnika odległości	0x00 - 0xFF
4	Prawy boczny czujnik odległości	Dane z czujnika odległości	0x00 - 0xFF
5	Prawy przedni czujnik odległości	Dane z czujnika odległości	0x00 - 0xFF
6	Pozycja	Dane o pozycji są wymagane, aby wiedzieć, w której kratce labiryntu robot się znajduje	0x00 - 0xFF
7	Kierunek	Dane o kierunku są wymagane, aby wiedzieć, które ścianki robot “widzi”	1000 - N 0100 - E 0010 - S 0001 - W
8	Lewy enkoder	Ilość impulsów z enkodera	0x00 - 0xFF
9	Prawy enkoder	Ilość impulsów z enkodera	0x00 - 0xFF
10	Koniec ruchu	Informacja o zakończeniu ruchu	0x00 - 0x01
11	Nieużywane	-	-

Tablica 3: Protokół transmisji sterującej robotem

Bajt	Funkcja	Opis	Przykładowe wartości
0	Start	bajt rozpoczynający transmisje	0xFF
1	RX/TX	bajt decydujacy czy jest potrzeba sterowania robotem, czy odebrania od niego danych	0x02/0x01
Dalej przedstawiana jest ramka wysyłanych danych (czyli po wysłaniu zadania 0x01 w 1 bajcie transmisji)			
2	LED	włączanie diody LED	0x00-0x01
3	LED	włączanie diody LED	0x00-0x01
4	LED_NEON	włączanie diod LED podświetlenia	0x00-0x01
5	LED	włączanie diody LED	0x00-0x01
6	LED	włączanie diody LED	0x00-0x01
7	Rodzaj ruchu	do przodu, do tyłu, lewo, prawo	0x01, 0x02, 0x03, 0x04, 0x05
8	Typ ruchu	do przodu pół kratki, cała kratka, kalibracja boczna, kalibracja przednia	0x01, 0x02, 0x03, 0x04
9	nieużywane	-	-
10	nieużywane	-	-
11	nieużywane	-	-

8 Rozwiązywanie labiryntu

8.1 Zapisywanie danych dotyczących labiryntu

W celu zapisu danych dotyczących labiryntu stworzono w pamięci dwie tablice. Jedna zapisuje dane dotyczące ścianek otaczających dane pole, druga - najkrótszą odległość do celu w labiryncie. W celu zaoszczędzenia miejsca w pamięci ścianki zapisywane są w sposób przedstawiony na listingu 6. Kierunek „północny" jest określony przez startowy zwrot robota - zawsze taki sam w konkurencji micromouse.

Listing 6: Zapisywanie ścianek w pamięci

```
#define WEST    1   // binarnie 00000001
#define SOUTH   2   // binarnie 00000010
#define EAST    4   // binarnie 00000100
#define NORTH   8   // binarnie 00001000
```

Ponadto w ten sam sposób zapisywana jest aktualna orientacja robota. W kolejnej zmiennej zapisywana jest pozycja robota w labiryncie.

Stworzono szereg funkcji, które wraz z ruchem robota aktualizują położenie ścianek w pamięci robota, jego pozycję i orientację (listing 7).

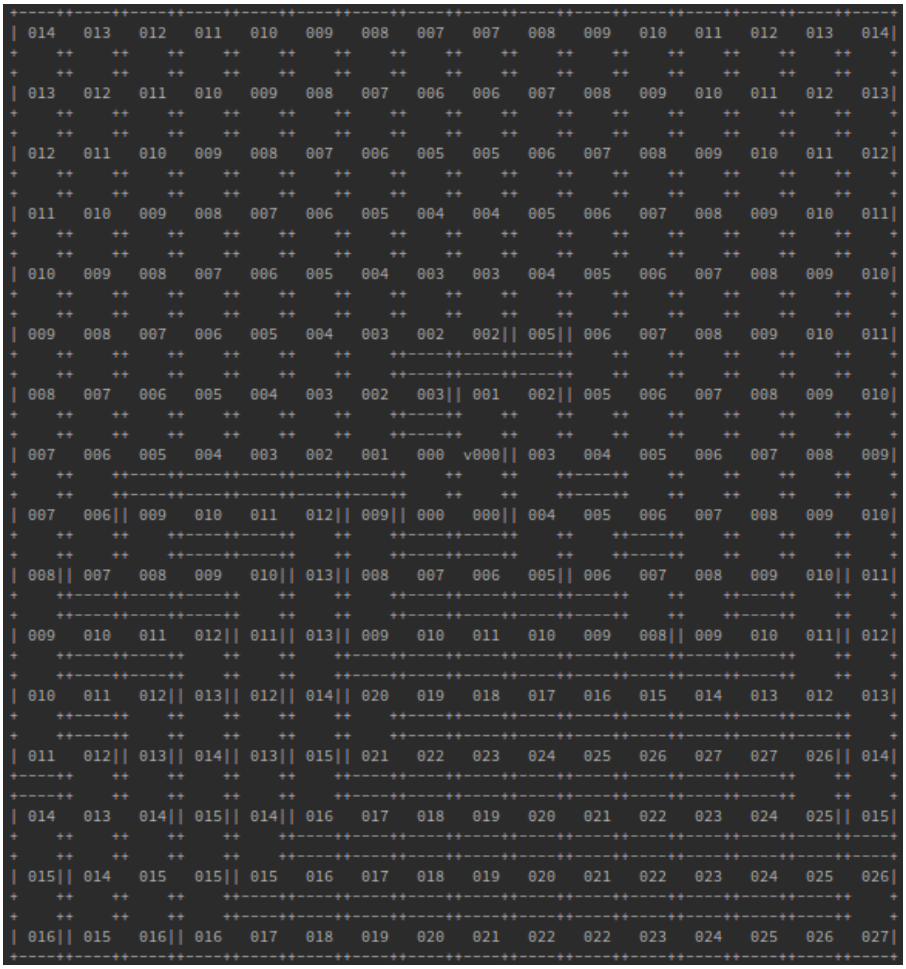
Listing 7: Dodawanie prawej ściany

```
void addRightWall() {
    extern unsigned char map[256];
    extern unsigned char poslab;
    extern unsigned char dir;
    switch (dir){
        case NORTH:
            map[poslab+16] = map[poslab+16] | EAST;
            map[poslab+16+1] = map[poslab+16+1] | WEST;
            break;
        case EAST:
```

```
11                map[poslab+1] = map[poslab+1] | SOUTH;
12                map[poslab+1-16] = map[poslab+1-16] | NORTH;
13                break;
14            case SOUTH:
15                map[poslab-16] = map[poslab-16] | WEST;
16                map[poslab-16-1] = map[poslab-16-1] | EAST;
17                break;
18            case WEST:
19                map[poslab-1] = map[poslab-1] | NORTH;
20                map[poslab-1+16] = map[poslab-1+16] | SOUTH;
21                break;
22        }
23    }
```

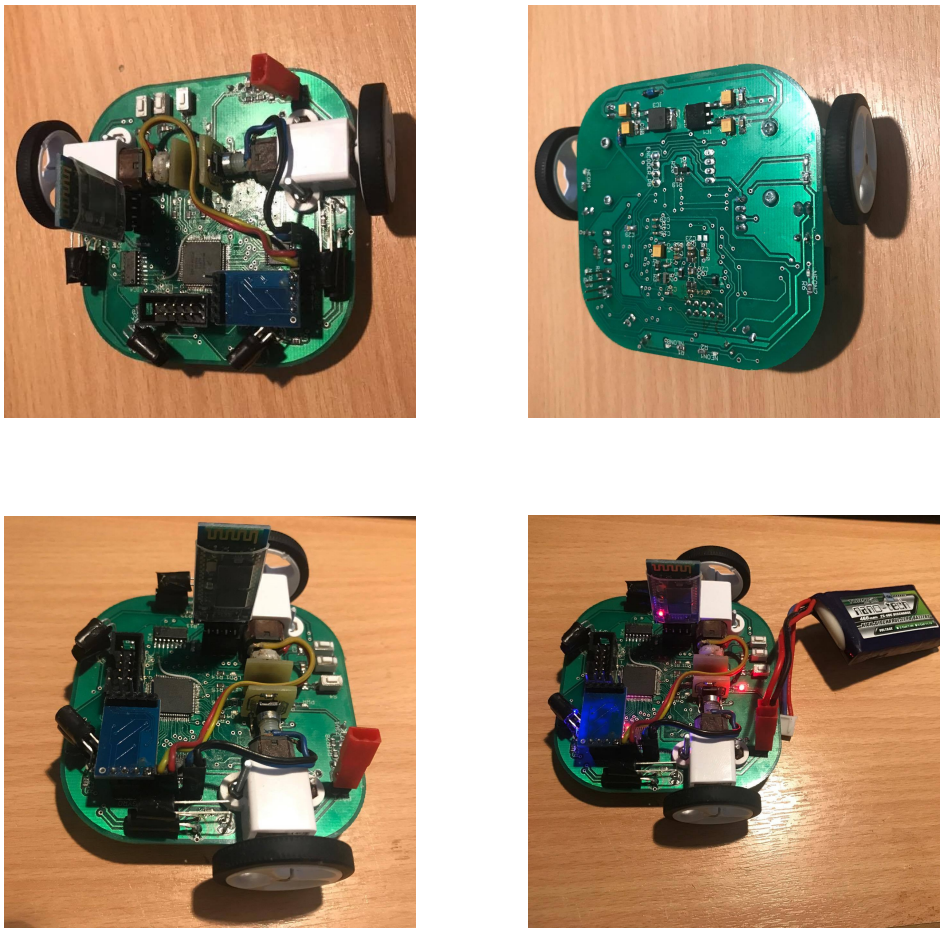
8.2 Algorytm floodfill

Zaimplementowano w robocie wstępną wersję algorytmu floodfill, aby umożliwić robotowi autonomiczne podejmowanie decyzji. Algorytm w symulacji poprawnie doporowadza robota do celu, jednak nadal wymaga poprawy. Robot potrafi dojechać do celu.



Rysunek 12: Symulacja - robot (oznaczony jako v) dotarł do celu

9 Podsumowanie



Rysunek 13: Zdjęcia robota

W części mechanicznej największy problem stanowiło wykonanie i dobranie odpowiednich parametrów czujników odległości - są one najbardziej istotnym elementem robota i wymagają możliwie największej precyzji. Innym problemem było również zmieszczenie się w preferowanej szerokości płytki a także odpowiednie rozmieszczenie elementów na płytce. Zaimplementowanie szybkiego i właściwie działającego algorytmu floodfill okazuje się największym wyzwaniem w procesie programowania robota.

Załączniki

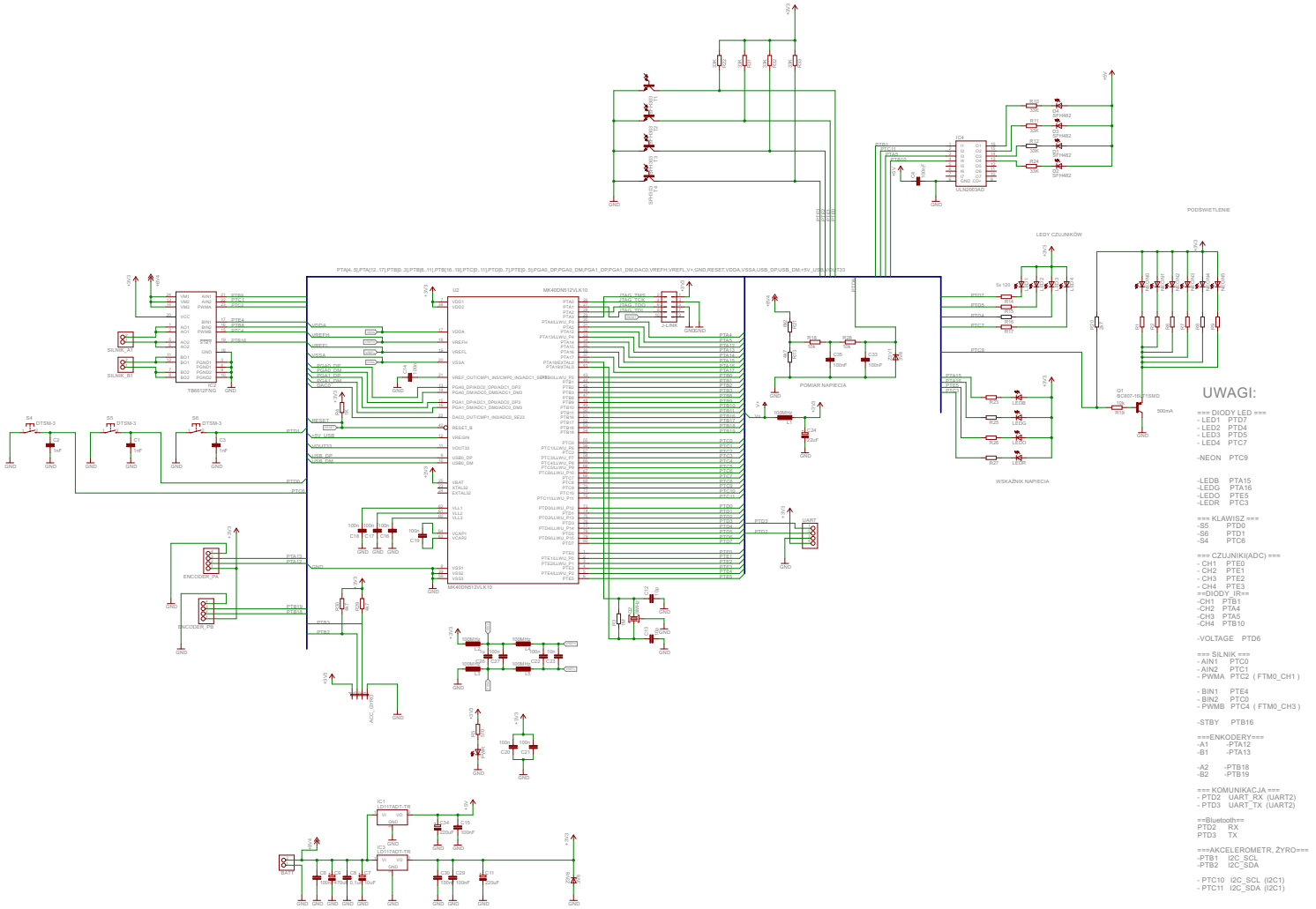
1. Schemat elektryczny robota
2. Projekt PCB

Literatura

[1] Łukasz Chojnacki. *Projekt optymalnego układu pomiarowego dla robota typu micromouse*. Politechnika Wrocławska, Wydział Mechaniczny, 2015.

[2] OSRAM Opto Semiconductors. Infrared Emitter (850 nm) SFH4550. `http://www.osram-os.com/Graphics/XPic7/00209835_0.pdf`, 2015.

[3] Vishay Semiconductors. BPW96B, BPW96C Silicon NPN Phototransistor. `http://www.vishay.com/docs/81532/bpw96.pdf`, 2011.


PODŚWIETLENIE
LEDY CZUJNIKÓW
PTA[4..5],PTA[12..17],PTB[0..3],PTB[8..11],PTB[16..19],PTC[0..11],PTD[0..7],PTE[0..5],PLPGA0_DP,PGA0_DM,PGA1_DP,PGA1_DM,DAC0,VREFH,VREFL,V+,GND,RESET,VDDA,VSSA,USB_DP,USB_DM,+5V_USB,XTAL32,EXTAL32
MK40DN512VLK10
PCMIAR NAPIĘCIA
WSKAŹNIK NAPIĘCIA
ENCODER_PA
ENCODER_PB
UART
UWAGI:
=== DIODY LED ===
- LED1 PTD7
- LED2 PTD4
- LED3 PTD5
- LED4 PTC7
-NEON PTC9
-LEDB PTA15
-LEDG PTA16
-LEDO PTE5
-LEDR PTC3
=== KLAWISZ ===
-S5 PTD0
-S6 PTD1
-S4 PTC6
=== CZUJNIKI(ADC) ===
- CH1 PTE0
- CH2 PTE1
- CH3 PTE2
- CH4 PTE3
==DIODY IR==
-CH1 PTB1
-CH2 PTA4
-CH3 PTA5
-CH4 PTB10
-VOLTAGE PTD6
=== SILNIK ===
- AIN1 PTC0
- AIN2 PTC1
- PWMA PTC2 (FTM0_CH1)
- BIN1 PTE4
- BIN2 PTC0
- PWMB PTC4 (FTM0_CH3)
-STBY PTB16
===ENKODERY===
-A1 -PTA12
-B1 -PTA13
-A2 -PTB18
-B2 -PTB19
=== KOMUNIKACJA ===
- PTD2 UART_RX (UART2)
- PTD3 UART_TX (UART2)
==Bluetooth==
PTD2 RX
PTD3 TX
===AKCELEROMETR, ŻYRO===
-PTB1 I2C_SCL
-PTB2 I2C_SDA
- PTC10 I2C_SCL (I2C1)
- PTC11 I2C_SDA (I2C1)