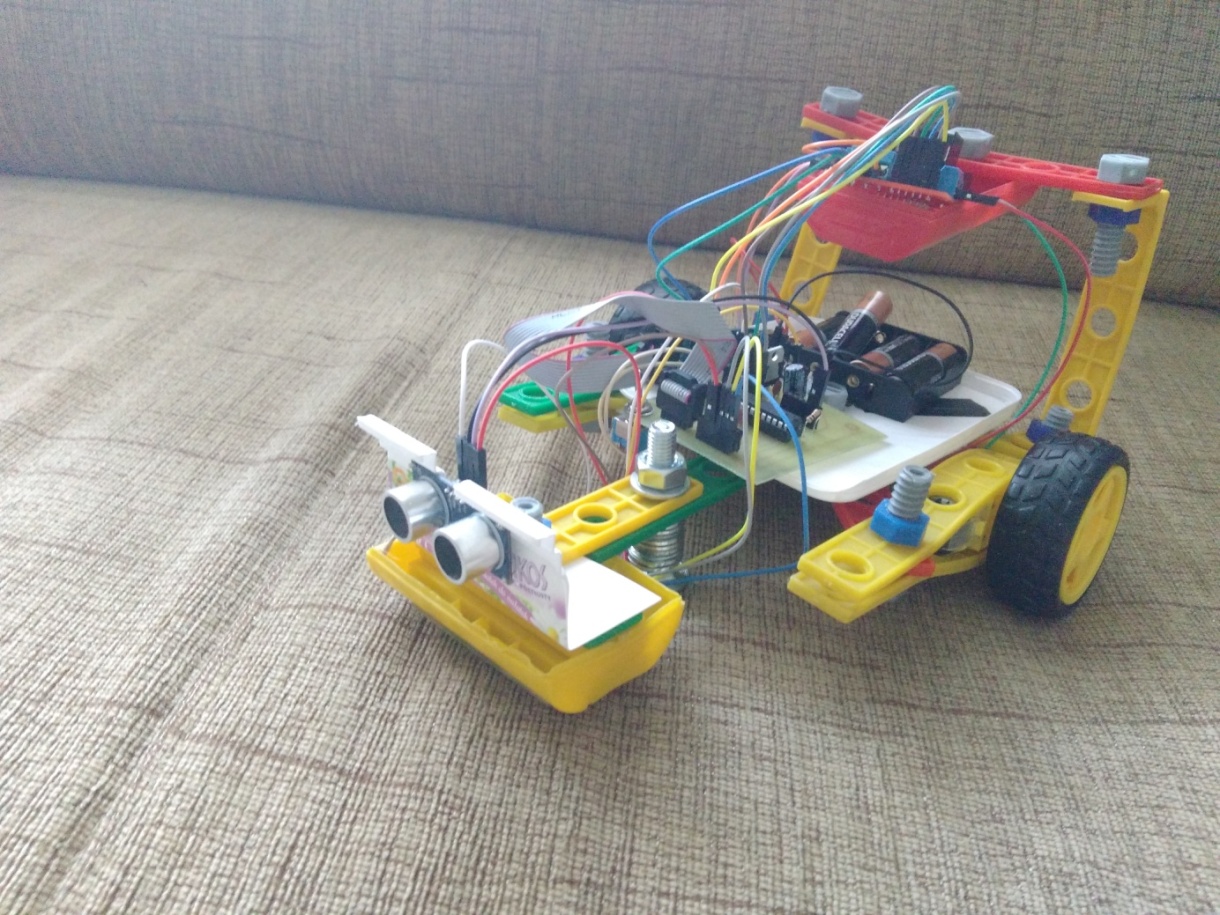
**Dokumentacja LF (line follower’a)**

**Nazwa robota:** Line Eater

**Rok:** 2017

**Zdjęcie:**



**Wykonawcy:**

Piotr Furmankiewicz:

- pisanie programu,

- testowanie,

- wytrawianie płytek,

- wiercenie,

- lutowanie elementów,

- projekty płytek,

- wybór części oraz kontrola wydatków.

Marcin Staszak:

- pisanie programu,

- testowanie,

- wytrawianie płytek,

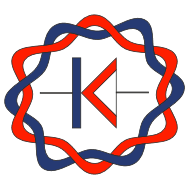
- wiercenie,

- lutowanie elementów,

- projekty płytek,

- stelaż robota.

**Logo koła naukowego:**

****

**Prowadzący:** drKlaus

**Spis treści:**

1. Ogólna idea działania (features),
2. Mechanika - konstrukcja, wymiary,
3. Elektrotechnika - zasilanie, silniki,
4. Mikrokontroler (uC),
5. Elektronika - sterowanie, płytki, czujniki, schematy, technika wykonania,
6. Program - algorytm, opis kodu z komentarzami,
7. Uruchamianie,
8. Serwisowanie,
9. Inżynieria oprogramowania (krótko i na temat),
10. Uwagi końcowe.

**1. Ogólna idea działania:**

Robota można zaklasyfikować najogólniej do rodziny LF (Line follower’ów). Uszczególniając, jest to enhanced LF, ponieważ, oprócz jazdy po linii, pokonuje tunel, obsługuje przerwę w trasie oraz omija przeszkodę. Zamysł jest taki, żeby robot bez gubienia się, przejeżdżał trasę jak najszybciej, a także jak najpłynniej, czyli bez dużych odchyleń wokół linii. Robot dostosowany jest maksymalnie do zakrętów 90 stopni. Powyżej 90 stopni robot najpewniej się pogubi (zjedzie z trasy). Podłoże musi być białe, a linia czarna. Robot dostosowany jest do szerokości linii wynoszącej 1,9 cm (szerokość taśmy izolacyjnej). Idea przyświecająca temu projektowi, to wykorzystanie jak najwięcej recyklingowego budulca, czyli stare klocki, np. LEGO, silniki od napędów DVD, wylutowane rezystory ze starych płytek itd. Budżet powinien być niewiększy niż 180 zł.

**2. Konstrukcja mechaniczna:**

* Wymiary robota:

- długość : 25,5 cm,

- maksymalna szerokość (tylna oś): 21 cm,

- maksymalna wysokość: 17,5 cm.

* Stelaż i umiejscowienie elementów:

- silniki umieszczone są z tyłu po lewej i prawej stronie wraz z kołami. Jest to najszersza część robota. Są one przytrzymywane starymi plastikowymi klockami dla dzieci, które skręcane są plastikowymi śrubkami. Konstrukcje dla obydwu silników są symetryczne i złączone na krzyż plastikowymi klockami w celu ustabilizowania całej konstrukcji oraz stworzenia po środku zaczepu do czujników linii oraz zaczepu do podstawki na baterie oraz płytki.

- mostek sterujący silnikami znajduje się w najwyższej części robota, czyli w niecce na podwyższeniu zrobionym także z wyżej wymienionych klocków,

- cztery baterie w koszyczku oraz płytka z mikrokontrolerem znajduje się pomiędzy silnikami na podstawce,

- przy podstawce umocowana jest śliska śruba zamkowa , która jest w centrum konstrukcji w celu uzyskania jak największej skrętności pojazdu.

- w najdalej wysuniętej części robota znajduje się na górze ultradźwiękowy czujnik odległości ustawiony w kierunku ruchu oraz na spodniej części przymocowana jest płytka z czujnikami linii.

- jest też parę usprawnień. Koszyczek na baterie jest przyklejony do podstawki, żeby nie spadał podczas jazdy. Śruba zamkowa jest odizolowana, by nie zewrzeć pinów płytki. Z tyłu pojazdu przez całą jego szerokość poprowadzona jest metalowa listewka, aby wyprostować całą konstrukcję oraz zapewnić zbieżność kół.

**3. Elektrotechnika:**

Jako zasilanie robota zostały zastosowane 4 baterie AA(R6)

o sumarycznym napięciu 6V. Maksymalne natężenie prądu płynącego w całym układzie wynosi około 0,6A. Czas pracy bez przerwy na pełnej mocy wynosi w przybliżeniu 15min. Do stabilizacji zasilania całego układu został użyty stabilizator LDO, czyli taki o niskim spadku napięcia, według producenta wynoszącym 0,45V przy natężeniu 1A. Jest to model L4940V5 dający na wyjściu 5V. W celu redukcji szumów zasilania na wejściu stabilizatora znajduje się jeden kondensator ceramiczny o pojemności 100nF oraz jeden elektrolityczny 470uF i taki sam zestaw kondensatorów na jego wyjściu.

Początkowo jako zasilanie w planach było 6 baterii AA o łącznym napięciu 9V, oraz tańszy stabilizator L7805V, jednak mocno się on nagrzewał przy spadku napięcia równym 4V oraz stosunkowo dużym przepływającym przez niego natężeniu. Mogło to powodować uszkodzenie układu z powodu przegrzania, oraz było mało efektywne.

Jako napęd służą silniki kątowe o wymiarach 65x26mm,

z dołączonymi do nich kołami.

Napięcie zasilania to 5V, jednak z powodzeniem można dać wyższe.

Silniki mają zintegrowaną przekładnię 48:1.

Prędkość obrotowa wynosi 80 obr/min, a moment obrotowy

0,5 kg\*cm (0,049 Nm). Silniki mogą się kręcić w obie strony co pozwala na lepsze sterowanie, w tym przypadku jest to wykonane poprzez mostek H ( dokładniej opisane w rozdziale elektronika).

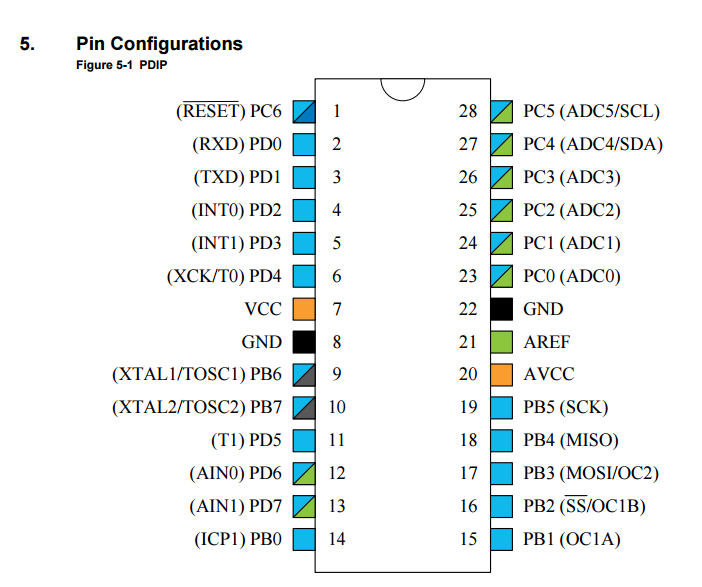
Podczas korzystania z silników nie napotkano na większe problemy.

**4. Mikrokontroler:**

Mózgiem robota jest ATmega8A. O wyborze akurat tego modelu zadecydowała cena, łatwa dostępność, wcześniejsze doświadczenia z tym modelem oraz ilość informacji dostępnych na forach internetowych, innymi słowy popularność.

Jest to mikrokontroler z domyślnym taktowaniem 1 Mhz z możliwością ustawienia 2Mhz, 4Mhz, 8Mhz oraz podłączenia zewnętrznego kwarcu, np. 16Mhz. Posiada ona 28 wyprowadzeń, z których większość może pełnić różne role takie jak: wyjścia/wejścia cyfrowe (PD[0-7] lub PB[0-7]), wejścia sygnałów przerwań, lub wejścia przetwornika ADC(PC[0-5]) oraz piny do zasilania przetwornika ADC oraz pin do standardowego zasilania +5V oraz pin do masy (GND). Ponadto pin PC6 jest odpowiedzialny za resetowanie ATmegi.

* Schemat:



Rysunek

Źródło ATmega8A Datasheet.

Żeby zabezpieczyć projekt mikrokontroler został umieszczony w podstawce przylutowanej do płytki. W razie spalenia mikrokontrolera, nie trzeba go wylutowywać, lecz wystarczy wyjąć go z podstawki i włożyć nowy.

* Otoczenie mikrokontrolera:

Do programowania mikrokontrolera użyty został programator USBasp, do którego jest wymagany tylko tak naprawdę sterownik USB, który jest instalowany po podłączeniu go do komputera. Działa on multiplatformowo (Linux Mac OS X, Windows). Do wgrywania programów użyto paczki AVRDude (głównie do testów - szczegóły w sekcji Program), w której jest aplikacja okienkowa i dodatkowe sterowniki. Zdjęcie programatora poniżej:



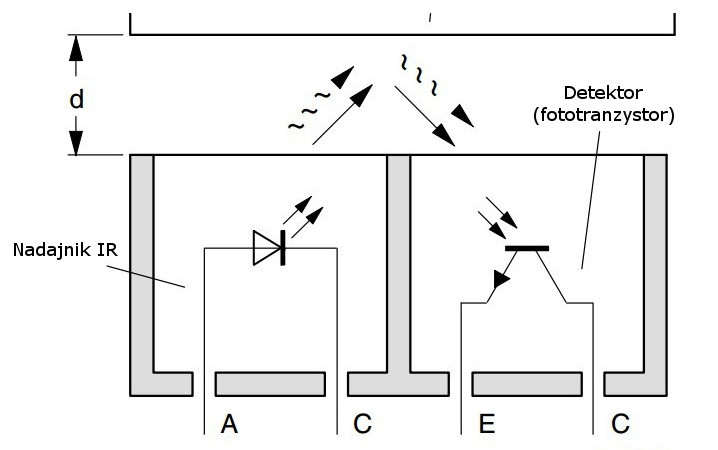
Rysunek

Źródło - https://protostack.com.au.

**5. Elektronika:**

Jako elementy służące do wykrywania linii wykorzystane zostało pięć czujników odbiciowych CNY70. Wybór padł właśnie na nie, ponieważ są łatwo dostępne, tanie, mają małe rozmiary oraz są łatwe w użyciu.

Czujnik ten wysyła wiązkę promieniowania poprzez nadajnik podczerwieni, a następnie za pomocą fototranzystora mierzy natężenie światła odbitego. Wyjściem jest sygnał napięciowy, zależny od natężenia światła padającego na ten detektor. Im więcej światła się odbije i dotrze do fotodetektora, tym napięcie na wyjściu będzie miało wyższą wartość. Jako, że promieniowanie świetlne lepiej odbija powierzchnia jasna ( a ciemna pochłania), dlatego napięcie będzie wyższe na białym materiale.



Rysunek

Czujnik ten wykrywa czarną linię na białym tle z odległości około jednego centymetra. Odległość tę można regulować za pomocą rezystora, który należy podłączyć do nóżki, która odpowiada zasilaniu nadajnika IR. W tym projekcie zastosowane zostały rezystory 100Ω, co przy napięciu 5V daje natężenie prądu płynące przez czujnik 50mA, jednak jeśli czujniki są blisko podłoża, lepiej użyć ponad 200Ω, ponieważ zmniejszy to zużycie baterii, oraz nie wpłynie negatywnie na odczyty otrzymywane z czujników.

Do wykrywania przeszkód na trasie wykorzystany został ultradźwiękowy czujnik odległości HC-SR04 wykrywający przeszkody maksymalnie z odległości 2m.



Rysunek

Pomiaru dokonuje się podając na pin TRIG stan wysoki przez 10us i mierzy się jak długo czujnik daje sygnał na pinie ECHO, a następnie otrzymany czas w us dzieli się przez 58 i otrzymuje się odległość przeszkody od czujnika wyrażoną w cm. W tym przypadku wykrywana jest przeszkoda oddalona o 15cm lub mniej.

Jeśli czujnik nie wykryje żadnej przeszkody przez czas ponad 100ms daje sygnał na ECHO, co jest nie akceptowalne w tym układzie sterującym, ponieważ znacznie spowalnia jego pracę, dlatego jeśli po 1160us czujnik nie przestanie podawać sygnału na ECHO, to jest on ignorowany. Podane 1160us odpowiada odległości około 20cm.

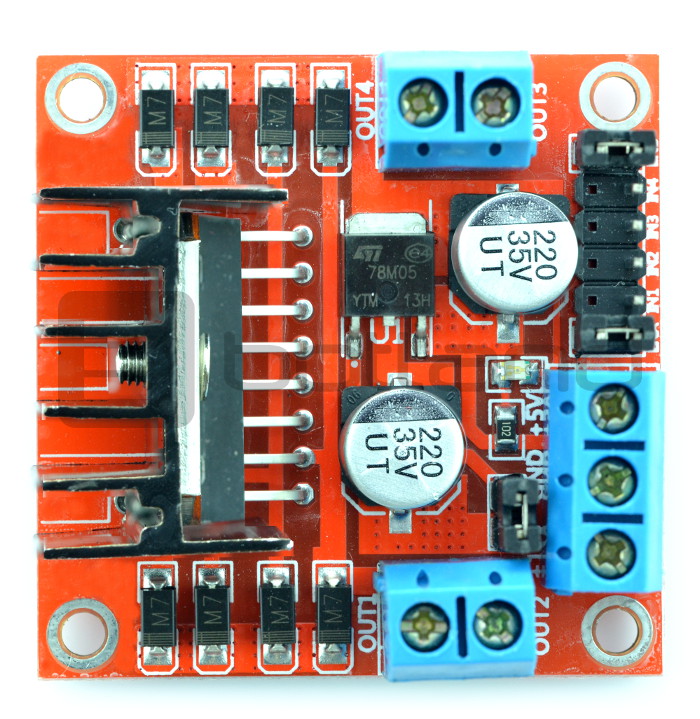
Początkowo miał być użyty cyfrowy czujnik

Sharp GP2Y0D810Z0F, wykrywający przeszkodę z odległości 10cm, jednak nie dysponowano gotową płytką do jego uruchomienia, więc próbowano wytrawić swoją, ale po dwóch nieudanych próbach, a także trudnościach w lutowaniu tak małych elementów, postanowiono zmienić czujnik. Problemem mogłaby być też mała odległość wykrywania przeszkody.

Do sterowania silnikami wykorzystany został moduł z dwukanałowym sterownikiem silników L298. Do wejścia zasilania silników podłączone zostało napięcie 5V wyprowadzone z głównej płytki z wyciągniętą zworką odpowiadającą za działanie wbudowanego stabilizatora i podłączono 5V także do wejścia zasilania części logicznej. Odpowiednie wyjścia z mikrokontrolera zostały wpięte do pinów

IN1, IN2 dla silnika prawego, oraz IN3, IN4 dla lewego, co pozwala na sterowanie kierunkiem obrotu silników.

Natomiast do pinów ENA oraz ENB po wyciągnięciu zworek podłączono wyjścia PWM z mikrokontrolera, dzięki temu uzyskano kontrolę nad szybkością obrotów każdego z silników. Same silniki zostały wpięte do wyjść OUT1,OUT2 i OUT3,OUT4.

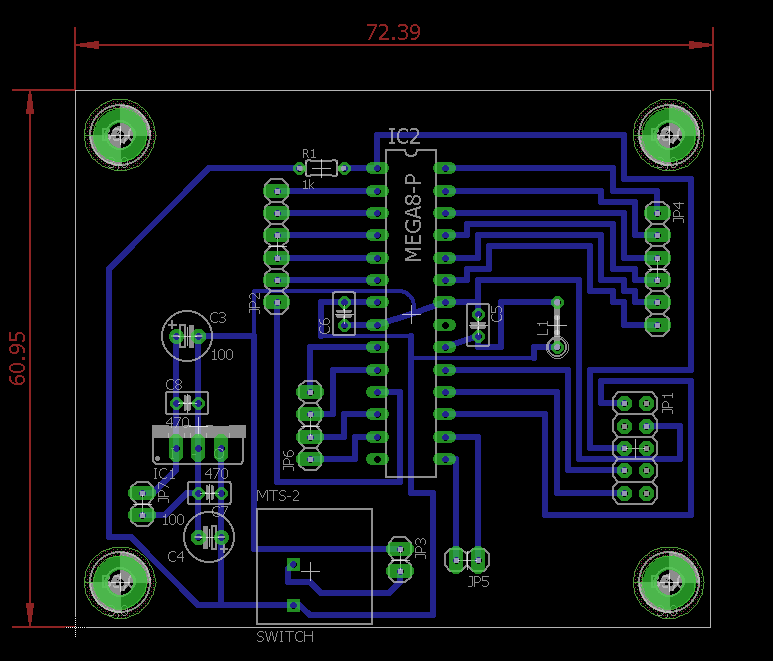


Rysunek

* Schematy:

Do projektowania płytek zastosowano narzędzia Eagle oraz kilku dodatkowych bibliotek elementów pobranych z Internetu. Główne zasady rozmieszczenia elementów to: nieprzecinające się ścieżki (akurat na to program nie pozwoli, ale to właśnie często decyduje o rozmieszczeniu elementów), kondensatory filtrujące napięcie jak najbliżej zasilania/ ATmegi, gold piny tak umiejscowione aby kable z nich wychodzące były jak najkrótsze, elementy ustawiane tak, aby nie zabierały dużo przestrzeni.

* Płytka główna z mikrokontrolerem:



Rysunek

Screen z programu Eagle- projekt własny.

Ułożenie pinów ATmegi na płytce pokrywa się ze schematem z Datasheet’u umieszczonym wyżej, więc opis płytki będzie zawierał nazwy pinów z Datasheet’u. Szerokości ścieżek zostały ustawione na 0,02mm. Tylko niektóre są węższe ze względu na konieczność poprowadzenia ich blisko pinów ATmegi.

Opis:

* PC[0-5] - scieżki wyprowadzone do gold pinów. Tam analogowo odczytywane są wartości napięcia dawane przez czujniki linii,
* PC6 - jako reset z rezystorem 10k Ohmów, by zapobiec automatycznemu resetowaniu się mikrokontrolera, jest też podłączony do programatora USBasp,
* PD[0-7] - ścieżki wyprowadzone do gold pinów:

- PD0 użyty jako Trigger czujnika odległości,

- PD1 użyty jako Echo czujnika odległości,

- PD6 użyty jako sterowanie silnikiem prawym w przód,

- PD7 użyty jako sterowanie silnikiem prawym w tył,

* PB[1-7] - ścieżki wyprowadzone do gold pinów:

- PB1 użyty jako PWM (sterowanie wypełnieniem sygnału w skali od do 255) do silnika lewego,

- PB2 użyty jako PWM (sterowanie wypełnieniem sygnału w skali od do 255) do silnika prawego,

- PB3 użyty jako MOSI do programatora USBasp,

- PB4 użyty jako MISO do programatora USBasp,

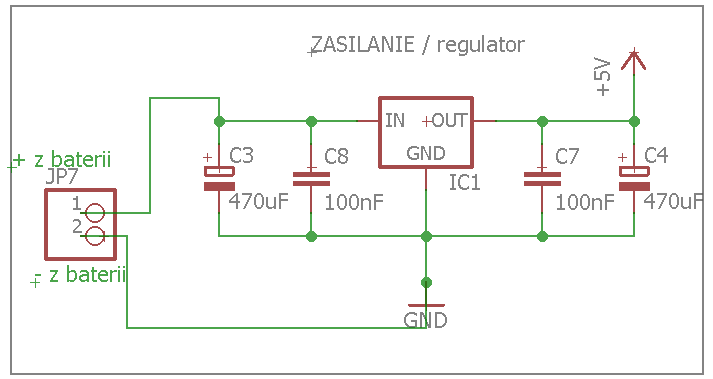
- PB5 użyty jako SCK do programatora USBasp,

- PB6 użyty jako sterowanie silnikiem lewym w przód,

- PB7 użyty jako sterowanie silnikiem lewym w tył,

PB0 nie zostało wyprowadzone.

* GND oraz VCC podłączone do zasilania +5V wyregulowane stabilizatorem napięcia. Podłączone wg schematu:

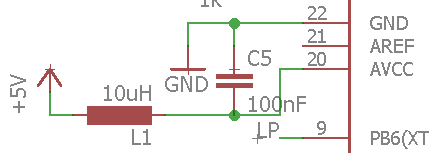


Rysunek

Screen z programu Eagle- projekt własny.

Gdzie C3 oraz C4 to kondensatory elektrolityczne, a C8 i C7 ceramiczne. Układ wykonany na podstawie noty katalogowej stabilizatora. Kondensatory redukują szumy i zakłócenia z zasilania.

* GND obok AVCC podłączone wg schematu:

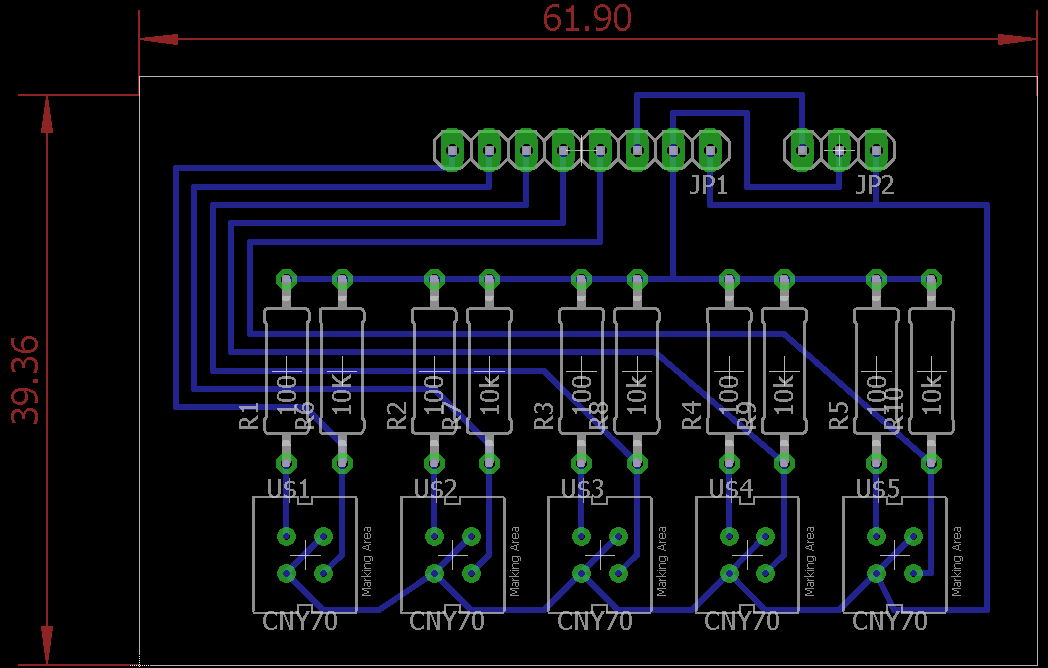


Screen z programu Eagle- projekt własny.

Gdzie L1 to cewka, a C5 kondensator ceramiczny. Napięcie ADC podłączone wg ATmega8A Datasheet.

AREF nie zostało podłączone, chociaż jest to zalecane, jednak w tym przypadku nie jest wymagana duża dokładności ADC.

* Płytka z czujnikami odbiciowymi:



Rysunek

Screen z programu Eagle - projekt własny.

Szerokości ścieżek zostały ustawione na 0,016mm.

Opis:

Czujników jest pięć, rozstawione w równej odległości od siebie oraz zostały podłączone wg ich dokumentacji. Rezystor 100 Ohmów odpowiedzialny jest za sterowanie natężeniem prądu, które płynie przez czujniki. W tym przypadku przez czujniki płynie maksymalne dopuszczalne natężenie prądu - 50mA. Dzięki temu uzyskano największy możliwy zasięg pomiaru. Czujniki emitują wtedy największe promieniowanie podczerwone. Rezystor 10k Ohmów pełni role rezystora podciągającego. Ma on za zadanie ustalenie na wyjściu ustalonego stanu logicznego i przeciwdziałanie zbieraniu ładunków z tzw. powietrza. Z tychże wyjść poprowadzone są ścieżki do gold pinów, gdzie odczytywane jest napięcie generowane przez czujniki. Gold pin numer 6 od lewej odpowiedzialny jest za odczytanie informacji z czujnika odległości. Do tego pinu numer 6 informacja jest podana z pinu nr 3 od prawej, gdzie sygnał przychodził z płytki z czujnikiem odległości. Był to zabieg, dzięki któremu było możliwe stworzenie wtyczki i pobranie informacji ze wszystkich czujników na raz. Jednak z tego zrezygnowano i stan logiczny z detektora ruchu przekazywany jest bezpośrednio do pinu PD2 (Echo) na płytce z ATmegą. Może piny do rozwiązania z wtyczką zostały niepotrzebnie zrobione, lecz przy innej konstrukcji robota taka funkcjonalność może znaleźć swoich zwolenników. Piny 7 i 8 od lewej zasilają czujniki linii, a piny 1 i 2 od prawej zasilają płytkę z czujnikiem ruchu.

* Proces tworzenia płytek:

Projekt płytki po dostosowaniu do wydruku został przeniesiony na papier kredowy. Było to konieczne ze względu na to, że po termotransferze tusz zostaje na płytce odklejając się z niego. Pierwszym etapem było wycięcie płytki o wymiarach tych co płytka na papierze kredowym. Najlepszym narzędziem okazał się nóż tapicerski, które z łatwością przecinał laminat. Następnie trzeba było płytkę dokładnie umyć tzn. odtłuścić płynem do naczyń a następnie potraktować rozpuszczalnikiem (nitro). Po tej czynności należało przyłożyć tuszem do miedzi projekt płytki i na 20 sekund 6 razy przez szmatkę przyłożyć żelazko na maksymalną temperaturę. Potem z tak przywartym papierem kredowym, płytkę trzeba było włożyć do wody o temperaturze 40 stopni Celsjusza z dodanym płynem do mycia naczyń i po minucie powoli odklejać papier, żeby tusz nie odszedł razem z nim. Następnie w odpowiednich proporcjach rozcieńczono nadsiarczan sodu w wodzie i włożono płytki do roztworu. Po kilku godzinach szczypcami wyjęto płytki i zmyto je wodą. Zmywaczem do paznokci wystarczyło zmyć tusz, który ukrył pod sobą miedź. Na koniec trzeba było wywiercić otwory oraz zabezpieczyć miedź przed utlenieniem. Służy do tego kalafonia rozrobiona w wodzie. Po schnięciu przez parę godzin można było przystąpić do lutowania przewlekanego. Na koniec odcięto wystające druciki przy lutach oraz pozbyto się resztek zalegającej cyny.

Po sprawdzeniu przejść pomiędzy pinami elementów elektronicznych można było zacząć wgrywanie programu.

Zdjęcia z tego procesu znajdują się w prezentacji.

**6. Oprogramowanie:**

W celu ułatwienia pisania kodu wykorzystano arduino ide, co wymaga wypalenia bootloadera do mikrokontrolera.

W przypadku wykorzystanej w tym projekcie ATmegi8A taktowanej wewnętrznym zegarem 8MHz jest to dość problematyczne, ponieważ żadna z obsługiwanych płytek nie wykorzystuje tego mikrokontrolera z takim taktowaniem, przez co wymagane były ręczne zmiany w pliku boards.txt.

Problemem, który napotkano podczas procesu wypalania bootloadera arduino była omyłkowa zmiana taktowania kontrolera na zewnętrzny oscylator 16MHz przez co przestał on działać. Rozwiązaniem problemu był zakup oscylatora o powyższych parametrach i po uruchomieniu ponowne przywrócenie taktowania 8MHz.

Program jest podzielony na 4 pliki,

plik main jest plikiem z główna pętlą programu,

plik config zawiera definicje wejść i wyjść,

w pliku init zadeklarowana jest funkcja inicjalizacyjna,

a w pliku linefollower są funkcje odpowiadające za działanie robota.

Plik main.ino:

#include "Ultrasonic.h" //wykorzystujemy zewnętrzną bibliotekę do obsługi czujnika odległości, jest ona dostępna pod adresem

//https://github.com/JRodrigoTech/Ultrasonic-HC-SR04

#define TRIGG 0 //pin digital 0 czyli PD0 ustawiony jako

#define ECHO 1 //pin digital 1 czyli PD1 ustawiony jako

int blad**,** pop\_blad **=** 0**,** //zmienne w których zapisujemy aktualne oraz //poprzednie odchylenie od linii

Kp **=** 2**,** Kd **=** 5**;** //zmienne służące do kalibracji członu PD

Ultrasonic odl**(**TRIGG**,**ECHO**,**1160**);** //obiekt klasy Ultrasonic wykorzystywany do pomiaru odległości, trzeci parametr oznacza timeout w us

void setup**()** **{**

initialize**();** // funkcja inicjalizująca z pliku init

**}**

void loop**()** **{**

odl**.**Timing**();** //wywołanie metody obliczającej czas w jakim ultradźwięki //docierają do przeszkody i z powrotem

**if(** odl**.**Ranging**(**1**)** **<=** 15 **)** // odl.Ranging(1) zwraca odległość przeszkody

**{** //w centymetrach

avoid**();** //jeśli przeszkoda jest w odl. mniejszej od 15cm to

**}** //wywołujemy funkcję omijającą przeszkodę

czytaj\_adc**();** // funkcja odczytująca wartości czujników podłączonych do //pinów PC5-PC1

blad **=** licz\_blad**();** // funkcja oblicza błąd ustawienia robota

int regulacja**=**PD**();** //funkcja regulująca wykorzystująca człon PD

PWM**(**225**+**regulacja**,** 225**-**regulacja**);** //funkcja odpowiadająca za //sterowanie silnikami

delay**(**10**);** //aktywne czekanie 10ms

**}**

Plik config.ino:

//EN\_11 oraz EN\_12 odpowiadają za sterowanie kierunkiem obrotu silnika pierwszego,

//analogicznie EN\_21 oraz EN\_22 dla silnika drugiego

#define EN\_11 20

#define EN\_12 21

#define EN\_21 6

#define EN\_22 7

//wyjścia PWM

#define PWM1 9

#define PWM2 10

//wejścia PC5-PC1, do których podłączone są czujniki linii

#define CZ\_1 A5

#define CZ\_2 A4

#define CZ\_3 A3

#define CZ\_4 A2

#define CZ\_5 A1

Plik init.ino

void initialize**()**

**{**

pinMode**(**EN\_11**,** OUTPUT**);** //ustalenie pinu jako wyjścia

digitalWrite**(**EN\_11**,**LOW**);** //ustawienie stanu początkowego niskiego

pinMode**(**EN\_12**,** OUTPUT**);**

digitalWrite**(**EN\_12**,**LOW**);**

pinMode**(**EN\_21**,** OUTPUT**);**

digitalWrite**(**EN\_21**,**LOW**);**

pinMode**(**EN\_22**,** OUTPUT**);**

digitalWrite**(**EN\_22**,**LOW**);**

pinMode**(**PWM1**,** OUTPUT**);**

analogWrite**(**PWM1**,**0**);** //ustawienie PWM na 0

pinMode**(**PWM2**,** OUTPUT**);**

analogWrite**(**PWM2**,**0**);**

pinMode**(**CZ\_1**,** INPUT**);** //ustawienie czujników jako wejść

pinMode**(**CZ\_2**,** INPUT**);**

pinMode**(**CZ\_3**,** INPUT**);**

pinMode**(**CZ\_4**,** INPUT**);**

pinMode**(**CZ\_5**,** INPUT**);**

//ustawienie trybu 8 bitowego nieodwracającego fast PWM i uruchomienie go

TCCR1A **=** \_BV**(**COM1A1**)** **|** \_BV**(**COM1B1**)** **|** \_BV**(**WGM10**);**

TCCR1B **=** \_BV**(**CS10**)** **|** \_BV**(**WGM12**);**

**}**

Plik linefollower.ino

//zastosowanie tych zmiennych zostanie wyjaśnione w miejscu ich użycia

int czujniki**[**5**]** **=** **{**0**};**

int wagi**[**5**]={-**30**,-**10**,**0**,**10**,**30**};**

int poprzednie**[**5**]** **=** **{**0**};**

int index**=**0**;**

//funkcje sterujące kierunkiem ruchu silników

void prawy\_silnik\_przod**()**

**{**

digitalWrite**(**EN\_22**,**HIGH**);**

digitalWrite**(**EN\_21**,**LOW**);**

**}**

void prawy\_silnik\_tyl**()**

**{**

digitalWrite**(**EN\_22**,**LOW**);**

digitalWrite**(**EN\_21**,**HIGH**);**

**}**

void lewy\_silnik\_przod**()**

**{**

digitalWrite**(**EN\_12**,**HIGH**);**

digitalWrite**(**EN\_11**,**LOW**);**

**}**

void lewy\_silnik\_tyl**()**

**{**

digitalWrite**(**EN\_12**,**LOW**);**

digitalWrite**(**EN\_11**,**HIGH**);**

**}**

//funkcja sterująca silnikami

void PWM**(**int lewy**,** int prawy**)**//jako argumenty otrzymujemy zadaną prędkość z doliczoną regulacją

**{**

**if(**lewy **>=** 0**)**//jeśli zmienna lewy >=0

to silnik powinien jechać do przodu

**{**

**if(**lewy**>**255**)**

lewy **=** 255**;**

lewy\_silnik\_przod**();**

**}**

**else** //w przeciwnym przypadku do tyłu

**{**

**if(**lewy**<-**255**)**

lewy **=** **-**255**;**

lewy\_silnik\_tyl**();**

**}**

**if(**prawy **>=** 0**)** //analogicznie dla prawego silnika

**{**

**if(**prawy**>**255**)**

prawy **=** 255**;**

prawy\_silnik\_przod**();**

**}**

**else**

**{**

**if(**prawy**<-**255**)**

prawy **=** **-**255**;**

lewy\_silnik\_tyl**();**

**}**

//ustalenie prędkości każdego z silników zależnej od regulacji

analogWrite**(**PWM1**,**abs**(**lewy**));**

analogWrite**(**PWM2**,**abs**(**prawy**));**

**}**

//funkcja omijania przeszkody

void avoid**()**

**{**

const int czas\_skretu**=**500**;**

const int czas\_po\_prostej**=**900**;**

//pełna moc!!!

digitalWrite**(**PWM1**,**255**);**

digitalWrite**(**PWM2**,**255**);**

//skręt 90 stopni w prawo

prawy\_silnik\_tyl**();**

lewy\_silnik\_przod**();**

delay**(**czas\_skretu**);**

//jazda prosto

prawy\_silnik\_przod**();**

delay**(**czas\_po\_prostej**);**

//skręt w lewo

lewy\_silnik\_tyl**();**

delay**(**czas\_skretu**);**

//jazda prosto

lewy\_silnik\_przod**();**

delay**(**1200**);**

//skręt w lewo aby powrócić na trasę

lewy\_silnik\_tyl**();**

delay**(**200**);**

//jazda prosto

lewy\_silnik\_przod**();**

delay**(**500**);**

//robot po ominięciu przeszkody jedzie prosto do momentu znalezienia //linii, gdy ją znajdzie wraca do normalnego trybu

**}**

//funkcja odczytująca wskazania czujników,

//jeśli czujnik wykrywa linię to w odpowiednie miejsce w tablicy czujniki //wpisuje 1

void czytaj\_adc**()**

**{**

**if(** analogRead**(** CZ\_1 **)** **>** 600 **)**

czujniki**[**0**]** **=** 1**;**

**else**

czujniki**[**0**]** **=** 0**;**

**if(** analogRead**(** CZ\_2 **)** **>** 600 **)**

czujniki**[**1**]** **=** 1**;**

**else**

czujniki**[**1**]** **=** 0**;**

**if(** analogRead**(** CZ\_3 **)** **>** 600 **)**

czujniki**[**2**]** **=** 1**;**

**else**

czujniki**[**2**]** **=** 0**;**

**if(** analogRead**(** CZ\_4 **)** **>** 600 **)**

czujniki**[**3**]** **=** 1**;**

**else**

czujniki**[**3**]** **=** 0**;**

**if(** analogRead**(** CZ\_5 **)** **>** 600 **)**

czujniki**[**4**]** **=** 1**;**

**else**

czujniki**[**4**]** **=** 0**;**

**}**

//regulator PD

int PD**()**

**{**

//zmienna blad zawiera aktualny wynik funkcji licz\_blad()

int rozniczka **=** blad **-** pop\_blad**;**

pop\_blad **=** blad**;**

**return** Kp**\***blad **+** Kd**\***rozniczka**;**

**}**

//funkcja wykorzystywana w funkcji licz\_blad,

//sprawdza czy wśród ostatnich 5 pomiarów były 3 takie przy których robot //powinien mocno skręcić

int sprawdz\_pomiary**()**

**{**

int a**=**0**;**

int b**=**0**;**

**for(**int i**=**0**;** i**<**5**;**i**++)**

**{**

**if(**poprzednie**[**i**]<-**5**)**

**{**

a**++;**

**}**

**else** **if(**poprzednie**[**i**]>**5**)**

**{**

b**++;**

**}**

**}**

**if(**a**>=**3**)**

**return** **-**5**;**

**else** **if(**b**>=**3**)**

**return** 5**;**

**else**

**return** 0**;**

**}**

//funkcja obliczająca błąd ustawienia robota względem linii

int licz\_blad**()**

**{**

int err **=** 0**;**

int ilosc **=** 0**;**

**for(**int i**=**0**;** i**<**5**;** i**++)**

**{**

err **+=** czujniki**[**i**]\***wagi**[**i**];** //do zmiennej err dodajemy wagę czujnika, //który wykrył linię

ilosc **+=** czujniki**[**i**];** //obliczamy ile czujników wykryło linię w tym //momencie

**}**

**if(** ilosc **!=** 0 **)** //ilość czujników wykrywających linię !=0, czyli

**{** //robot jest na linii

err **/=** ilosc**;** //dzielimy błąd przez ilość czujników, które //wykryły linię

**if(**index**>=**5**)**

index**=**0**;** //jeśli index się przepełni to go zerujemy

poprzednie**[**index**++]=**err**;** //do tablicy poprzednie zapisujemy 5 //ostatnich //pomiarów

**}**

**else** //żaden czujnik nie jest na linii, czyli robot wypadł z trasy

**{**

//poniższy kod ma za zadanie ocenić, czy robot powinien ostro //skręcić aby wrócić na trasę,

//czy jechać prosto, ponieważ najechał na przerwę w linii

**if(**sprawdz\_pomiary**()==-**5**)**

**{**

err**=-**80**;**

**}**

**else** **if(**sprawdz\_pomiary**()==**5**)**

**{**

err**=**80**;**

**}**

**else**

err**=**0**;**

**}**

**return** err**;**

**}**

**7. Uruchamianie:**

Proces uruchomienia LF jest prosty, pod warunkiem, że żadna z części płytki się nie wylutowała i robot nie otrzymał uszkodzeń mechanicznych. Jeśli tak się stało to należy przejść do sekcji nr 8 - serwisowanie. Aby robot wystartował wystarczy włożyć baterie (najlepiej nowe). Po dokonaniu tej czynności napięcie zostanie doprowadzone do mostka oraz do mikrokontrolera. Zasilanie zostało poprowadzone w taki, a nie inny sposób, ponieważ umożliwiło to programowanie robota bez wprawiania w ruch silników. Żeby odpalić silniki należy przesunąć wajchę przełącznika (switcha) w stronę pinów do programatora (dwurzędowe gold piny). Wtedy obydwa silniki ruszą w przód gdy nie są na czarnej linii. Gdy wysunięta część z czujnikami natrafi na linie robot zacznie po niej jechać. Gdy natrafi na coś czarnego co nie jest linia, to także zacznie reagować. Gdy wykryje przeszkodę, to spróbuje ją ominąć. Wszystkie funkcje robota są opisane w pierwszym rozdziale - ogólna idea działania. Żaby robota zatrzymać wystarczy przełączyć switch w drugą stronę, wyjąć baterie lub po prostu poczekać aż baterie się wyczerpią.

Dobrym usprawnieniem konstrukcji byłoby zastąpienie zwykłego koszyczka na baterie, koszyczkiem z przełącznikiem. Rozwiązałoby to problem wyjmowania baterii.

Rozwiązaniem bardziej wyrafinowanym byłoby włączanie i wyłączanie robota za pomocą pilota, lecz trzeba by było nastawić się na większe koszty, a tego chciano uniknąć.

**8. Serwisowanie i konserwacja**

Możliwe awarie:

W przypadku gdy silniki działają, jednak robot ma problemy z jazdą po linii, należy sprawdzić czy wszystkie czujniki działają poprzez wpięcie woltomierza do wyjścia każdego z czujników i do masy, a następnie sprawdzenie czy odczyty zmieniają się w zależności od koloru powierzchni, na której znajduje się czujnik. Przy białej powierzchni powinien dawać niewiele więcej niż 0V, a po ustawieniu na linii niecałe 5V lub trochę mniej. Jeśli czujniki działają prawidłowo, to problem najprawdopodobniej leży w nieprawidłowym podłączeniu czujników do mikrokontrolera, mogą one być np. podłączone w odwrotnej kolejności.

Jeśli natomiast koła się nie kręcą, należy sprawdzić czy zasilanie dochodzi do modułu sterującego silnikami, sygnalizuje to świecąca dioda. Jeśli jest zapalona to należy sprawdzić podłączenie pinów sterujących kierunkiem jazdy silników, a następnie tych z PWM.

Należy też sprawdzić podłączenie silników do modułu sterującego.

Jeśli zasilanie nie dociera do sterownika silników, to problem może leżeć w głównym zasilaniu płytki, wtedy testujemy zasilanie z baterii.

Jeśli dają one zbyt małe napięcie stabilizator może przestać działać, aby to sprawdzić mierzymy napięcie jakie dają baterie, a także napięcie między masą a wyjściem stabilizatora, gdy jest ono prawidłowe, problem może leżeć w złym podłączeniu zasilania z płytki do modułu sterującego silnikami lub wylutowanie się któregoś z pinów.

W przypadku problemów z programowaniem mikrokontrolera, także sprawdzamy zasilanie główne w podany wyżej sposób, a także czy odpowiednio wpięto programator.

**9. Inżynieria oprogramowania:**

Projekt został wykonany przez dwie osoby. Nie bawiono się w żadne analizy SWOT i tym podobne. Zadania zostały podzielone na dwie osoby wg opisu na pierwszej stronie dokumentacji. Robiono wszystko razem. Wyjątkiem było kupno części. Piotr Furmankiewicz zajął się zbieraniem paragonów, co na końcu pomogło w ogólnym rozliczeniu się. Wyjątkiem także było szukanie budulca na stelaż. Z racji tego, że Marcin Staszak mieszka blisko warsztatu samochodowego ojca, właśnie Marcin zebrał potrzebne części, bo miał ich pod dostatkiem. Czas pracy rozłożył się mniej więcej po równo. Żaden z rzekomych wykonawców nie był dla drugiego ciężarem czy pasożytem. Z reguły nie było tak, że jeden coś robił, a drugi tracił czas. Przykładem jest sytuacja: jeden robił stelaż, a drugi odbierał przesyłkę kurierską oraz konserwował płytkę kalafonią, by przygotować ją do montażu. Projekt nie był wybitnie skomplikowany, dlatego też nie napotkano na problemy z pracą zespołową. Efektem pracy jest działający robot, lecz na pewno można było zrobić to lepiej. Dobrym pomysłem było by pooglądać roboty innych wykonawców, by zainspirować się przy wykonywaniu stelażu. Jest on zdecydowanie za ciężki i obciąża silniki. Warto było także stworzyć płytkę szkoleniową, by wyćwiczyć się w lutowaniu, wierceniu i trawieniu zanim doszłoby do wykonywania płytki ostatecznej. Wadą tego przedsięwzięcia było to, że robiły je 2 a nie np. 3 lub 4 osoby. Podział obowiązków nie stałby się problemem, to raz, a dwa, praca nad robotem nie obciążałaby dwóch osób, a aż 3 lub 4. Byłoby po prostu łatwiej.

**10. Uwagi końcowe:**

W konstrukcji robota wiele rzeczy można poprawić, np.

- konstrukcja robota na jednej dużej płytce, zamiast na kilku mniejszych, wtedy można usunąć ciężki stelaż i oprzeć konstrukcję na tej właśnie płytce,

- zmniejszenie robota powinno ułatwić jego poruszanie się po trasie i zwiększyć jego skrętność,

- można także użyć 7 lub więcej czujników zamiast 5, co ułatwi płynne poruszanie się robota i pozwoli na jego przyspieszenie,

- dobrym pomysłem jest też użycie koszyczka na baterie z przełącznikiem, co jest wygodne przy uruchamianiu robota,

- aby przyspieszyć robota można także dostarczyć wyższe napięcie do silników, czyli w tym przypadku podłączyć je bezpośrednio do baterii,

- warto też dodać większe rezystory do płytki z czujnikami odbiciowymi, co pozwoli na zmniejszenie zużycia prądu, a odległość z jakiej czujniki będą wykrywały linię nadal powinna być wystarczająca.

Do dokumentacji dołączona jest **płyta DVD**, na której obrazowo opisane są kolejne czynności dotyczące wykonania płytki oraz przydatne informacje oraz linki do źródeł.