Wiedza i doświadczenie projektowe wizytówką absolwenta kierunku automatyka i robotyka na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej

POKL.04.01.02-00-020/10

Program Operacyjny Kapitał Ludzki współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego

---

Gliwice, 05.X.2013r.
Międzywydziałowe Koło Naukowe High Flyers
Wydział Automatyki Elektroniki i Informatyki
Kierunek Automatyka i Robotyka
Dr hab. inż. Marek Pawełczyk

Sprawozdanie z realizacji projektu:

MobilCam 3. System wizji oraz fonii.

Zespół projektowy:

Maksymilian Byliński (***lider projektu***)	
Jakub Kała	
Tomasz Korzyniec	
Krzysztof Jaskóła	
Artur Piernikarczyk	
Michał Besler	
Mateusz Drost	
Tobiasz Heller	

Podpis Opiekuna:

1. Opis projektu.

1.1. Cel projektu:

Obsługa kamery cyfrowej wraz z przetwarzaniem obrazu na pojeździe mobilnym, zainstalowanej w celu rozpoznawania obiektów znajdujących się w polu widzenia. Dodatkowo pojazd zostanie wyposażony w macierz mikrofonów wykorzystaną do rozpoznawania dźwięków pochodzących z otoczenia.

Zastosowanie fal radiowych do przesyłania wyżej wymienionych danych, tj. wizji oraz fonii.

Budowa obrotowego sonaru w celu zwiększenia dokładności pomiaru odległości od obiektu, wykonanego za pomocą kamery.

1.2. Założenia projektu:

1.2.1. Transport wizji oraz fonii do jednostki nadrzędnej – GCS.

1.2.2. Sterowanie pojazdem za pomocą mowy ludzkiej.

1.2.3. Dokładniejsze rozpoznawanie otoczenia za pomocą kamery oraz macierzy mikrofonów.

1.2.4. Zastosowanie mikroprocesora o większej mocy obliczeniowej.

1.2.5. Zaprojektowanie sztucznej inteligencji.

1.2.6. Dokładniejsze wyznaczanie odległości do obiektu.

1.3. Oczekiwane wyniki:

Uzyskanie sztucznej inteligencji pozwalającej na omijanie przeszkód, dokładne wyznaczanie mapy terenu, poruszanie się z punktu do punktu oraz rozpoznawanie potencjalnych obiektów.

1.4. Ocena ryzyka projektu:

Poziom trudności wykonania projektu jest wysoki. Nie istnieją problemy związane z dostępnością podzespołów do realizacji projektu. Koszty projektu są jasno określone. Harmonogram został dobrany w sposób umożliwiający stopniowy rozwój projektu oraz został wyszczególniony okres, który jest przeznaczony na nadrobienie ewentualnych zaległości w projekcie. Ponadto wszyscy członkowie zespołu wyrazili chęć podjęcia się wykonania projektu.

Mając na uwadze interdyscyplinarny kierunek kształcenia studentów na Wydziale Automatyki, Elektroniki i Informatyki, ich zainteresowania tematyką autonomicznych pojazdów mobilnych istnieje duże prawdopodobieństwo pozytywnego wykonania projektu.

2. Podział projektu na zadania.

1. Obsługa mikroprocesora ARM11:
   1.1. Określenie wymagań sprzętowych oraz programowych.
   1.2. Wybór systemu operacyjnego.
   1.3. Podłączenie systemu elektronicznego w jedną całość.
2. Obsługa Kamery cyfrowej:
   2.1. Rozpoznanie budowy kamery oraz sposób przesyłania obrazu z kamery do mikroprocesora.
   2.2. Wysłanie obrazu na jednostkę nadrzędną.
   2.3. Przetwarzanie obrazu z kamery.
3. Obsługa macierzy mikrofonów:
   3.1. Zaprojektowanie oraz wykonanie macierzy mikrofonów.
   3.2. Odczyt sygnałów pochodzących z mikrofonów.
   3.3. Przetwarzanie sygnału dźwiękowego.
4. Wykonanie obrotowego sonaru:
   4.1. Projekt sonaru.
   4.2. Podłączenie sonaru do systemu elektronicznego.
   4.3. Wykonanie algorytmu sterowania oraz dokonywania pomiarów.
   4.4. Testowanie.
5. Wykonanie raportu.

3. Harmonogram prac projektowych.

Osadzając wyszczególnione zadania na osi czasu, sporządzono harmonogram realizacji projektu.

Miesiąc		**Kwiecień**				**Maj**					**Czerwiec**				**Wrzesień**			
Projekt		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ARM11	1.1																	
	1.2																	
	1.3																	
Kamera cyfrowa	2.1																	
	2.2																	
	2.3																	
Macierz mikrofonów	3.1																	
	3.2																	
	3.3																	
Sonar	4.1																	
	4.2																	
	4.3																	
	4.4																	
Raport	5.1																	

4. Kamienie milowe.

- Dobór oraz obsługa systemu operacyjnego dostosowanego do mikroprocesora ARM11.
- Zaprojektowanie i wykonanie transmisji oraz przetwarzania toru wizji oraz fonii.
- Budowa sonaru zastosowanego w celu polepszenia dokładności wskazań przez kamerę.
- Zaprojektowanie jednostki nadrzędnej.

5. Określenie zasad odpowiedzialności członków zespołu.

- Maksymilian Byliński– Odpowiedzialny za projekt oraz obsługę systemu operacyjnego dla mikroprocesora ARM11.
- Artur Piernikarczyk– Odpowiedzialny za oprogramowanie systemu elektronicznego.
- Jakub Kała – Odpowiedzialny za komunikację pojazdu z GCS.
- Mateusz Drost –Odpowiedzialny za projekt jednostki GCS.
- Michał Besler– Odpowiedzialny za budowę sonaru.
- Tomasz Korzyniec – Odpowiedzialny za projektowanie algorytmów fonii.
- Krzysztof Jaskóła – Odpowiedzialny za projektowanie algorytmów wizji.
- Tobiasz Heller – Odpowiedzialny za projektowanie algorytmów.

6. Realizacja poszczególnych zadań projektu.

6.1. Obsługa mikroprocesora ARM11.

6.1.1. Określenie wymagań sprzętowych oraz programowych.

Jako platformę sprzętową wybrane zostało Raspberry Pi. Jest to komputer wielkości karty kredytowej, wyposażony w:
-ARM1176(częstotliwość taktowania 700MHz)
-VideCore IV GPU
-512MB pamięci RAM
-BCM2835
-slot kart SD
-dwa gniazda USB
-gniazda HDMI, kompozyt RCA
Dodatkowo wyposażony jest w:
-8xGPIO
-UART
-magistrale I2C oraz I2S
-magistrala SPI

Istnieją dwie możliwości uruchomienia urządzenia:

Zdalnie-obsługa i połączenie z układem jest możliwe przy pomocy protokołu ssh. Połączenie jest realizowane z innego komputera przy użyciu zazwyczaj połączenia lokalnego.
Wymagania:
-połączenie lokalne pomiędzy układem a komputerem użytkownika
-karta SD z systemem operacyjnym
-zasilanie układu
-komputer do obsługi protokołu

Uruchomienie przy użyciu tylko i wyłącznie Raspberry PI polega to na tym iż układ pełni rolę samodzielnego komputera.
Wymagania:
-połączenie z Internetem(zalecane)
-karta SD z systemem operacyjnym
-zasilanie układu
- monitor
-klawiatura
-mysz

6.1.2. Wybór systemu operacyjnego

Jako system operacyjny wybrana została dystrybucja Linuxa bezpośrednio dedykowana dla Raspberry PI, jest to Raspbian. System ten jest oparty na Debianie.

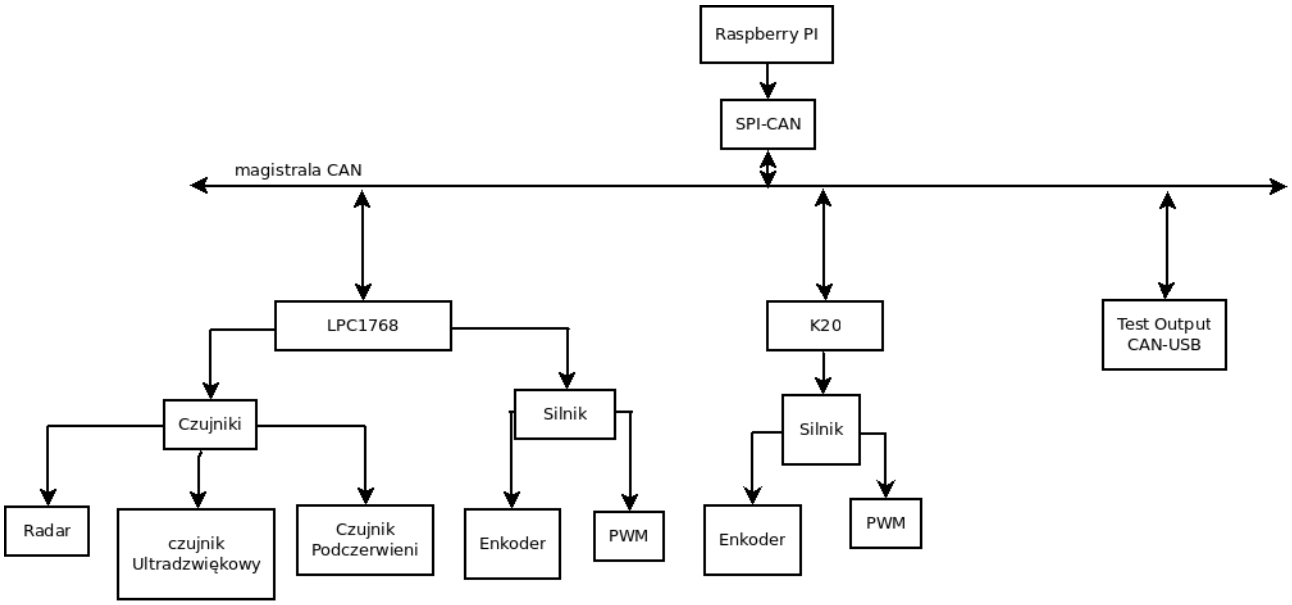
6.1.3. Podłączenie systemu elektronicznego w jedną całość.

System elektroniczny platformy mobilnej został zmodyfikowany ze względu na ograniczenia jaki stawiał ostatni. Atmegę 128 zastąpiono mikroprocesorem Raspberry Pi, mikrokontrolerem LPC 1768 oraz K20. Raspberry PI pełni rolę głównego procesora w którym docelowo ma znajdować się algorytm autonomii, przetwarzany ma być dźwięk oraz obraz. Ze względu na jego dużą moc obliczeniową oraz system operacyjny wyżej wymienione zadania mogą być oprogramowane z wykorzystaniem istniejącego kodu.

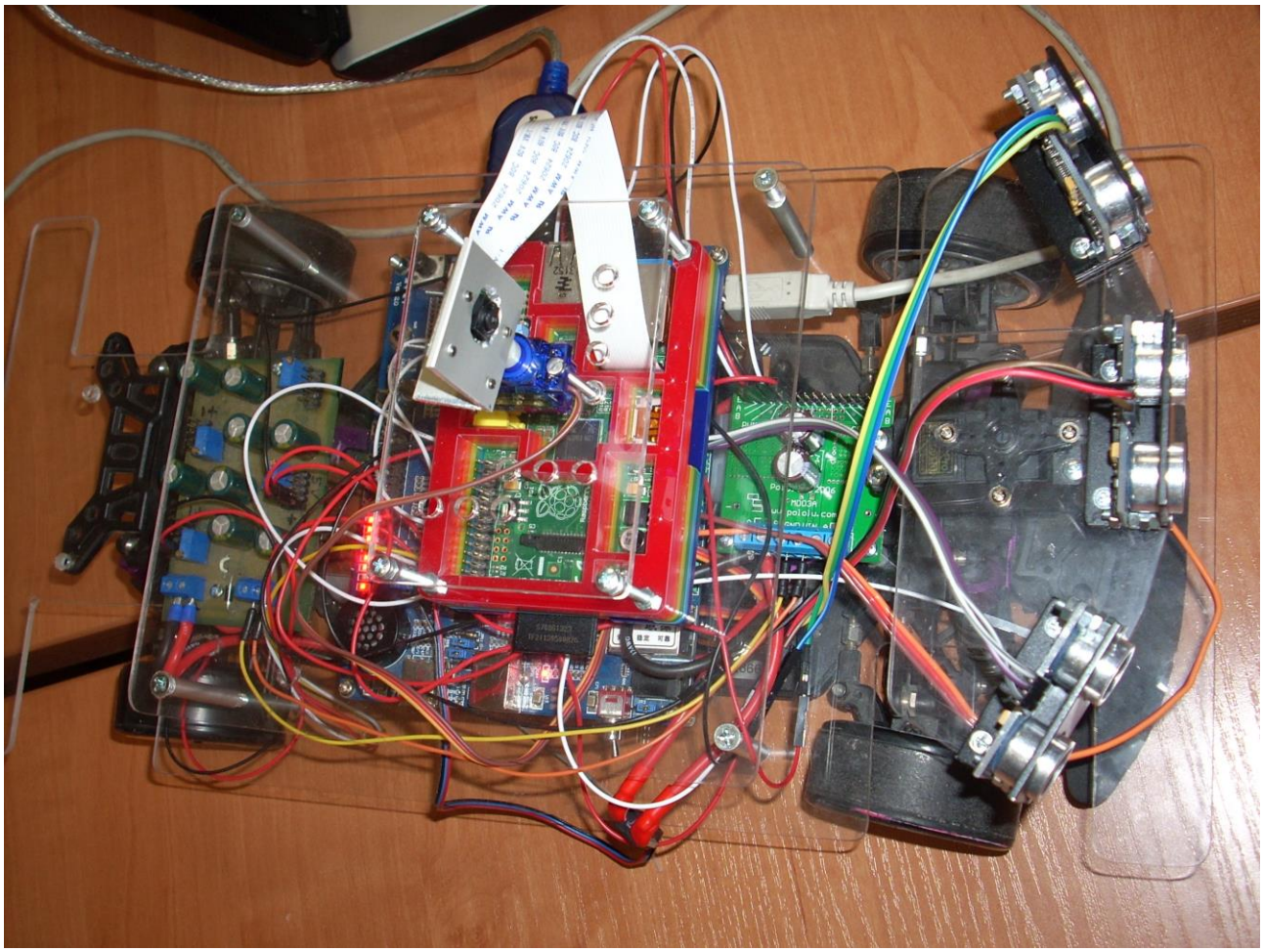
LPC 1768 jest mikrokontrolerem odpowiedzialnym za obsługę radaru opartego o 3 czujniki ultradźwiękowe Pololu 1605 oraz pozostałych czujników na platformie wraz z sterowaniem silnikiem za pomocą sterownika Pololu MD03A . Ze względów czasowych nie zastosowano systemu czasu rzeczywistego. Mogło by to spowodować znaczne opóźnienia czasowe.

Ostatni mikrokontroler odpowiedzialny będzie za sterowanie ramieniem robota. Jako jedyny posiada jednostkę zmiennoprzecinkową co ułatwi wyznaczanie macierzy odwrotnej, która to pozwoli wyznaczyć obroty poszczególnych silników. Aktualnie nie jest wykorzystywany w systemie ze względu na trwające prace nad ramieniem.
Komunikacja między urządzeniami będzie wykonywana za pomocą protokołu CAN z warstwą aplikacji opartą na ARING 825 używaną w jednostkach latających np. airbus a380. Protokół jest już oprogramowany i wykorzystywany na jednostce K20.
Ze względu na brak interfejsu CAN w Raspberry Pi postanowiono zastosować przejściówkę SPI-CAN. Przejściówka takowa została zaprojektowana i dostarczona do wykonania. Ze względu na konieczność oprogramowania interfejscu w Raspberry PI oraz LPC1768 aktualnie wykorzystuje się UART jako jednostkę umożliwiającą komunikację między urządzeniami. Jest to możliwe dlatego, iż aktualnie jeszcze nie wykorzystano K20.
Całość jest zasilana z baterii przez moduł powstały w ramach projektu MobilCAM 3.

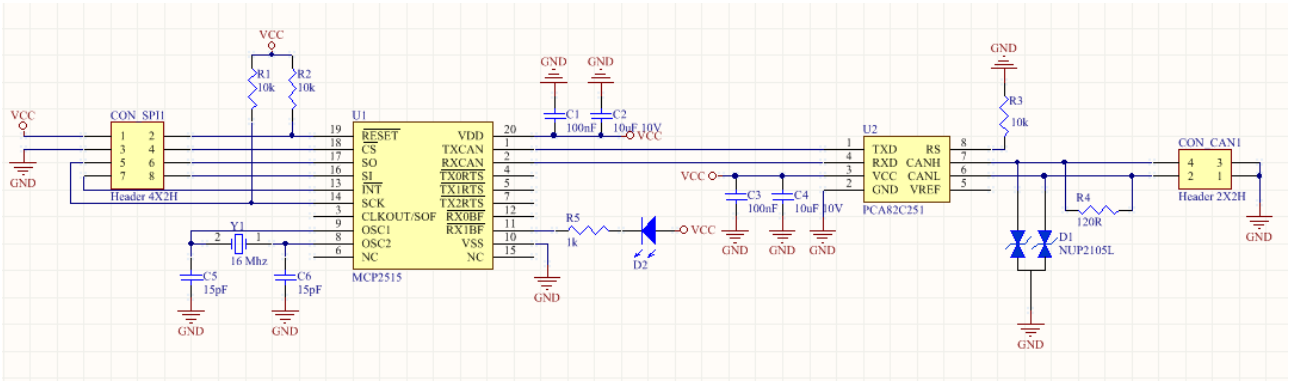
Schemat blokowy systemu



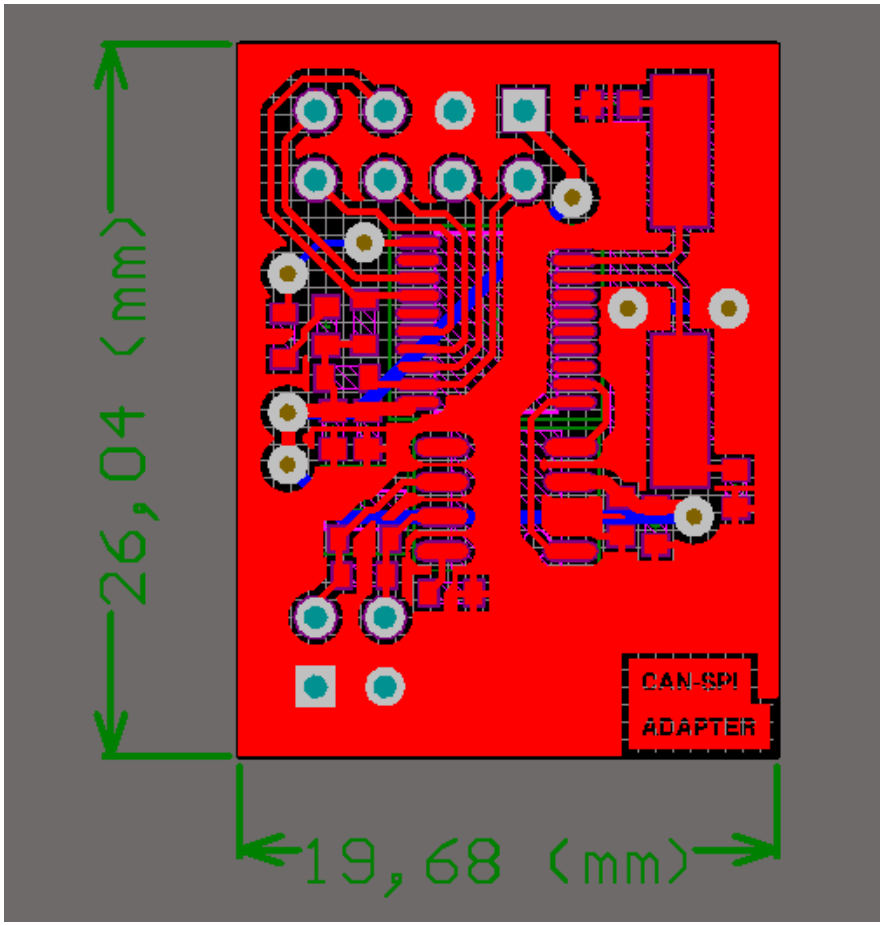
Zdjęcie systemu elektronicznego



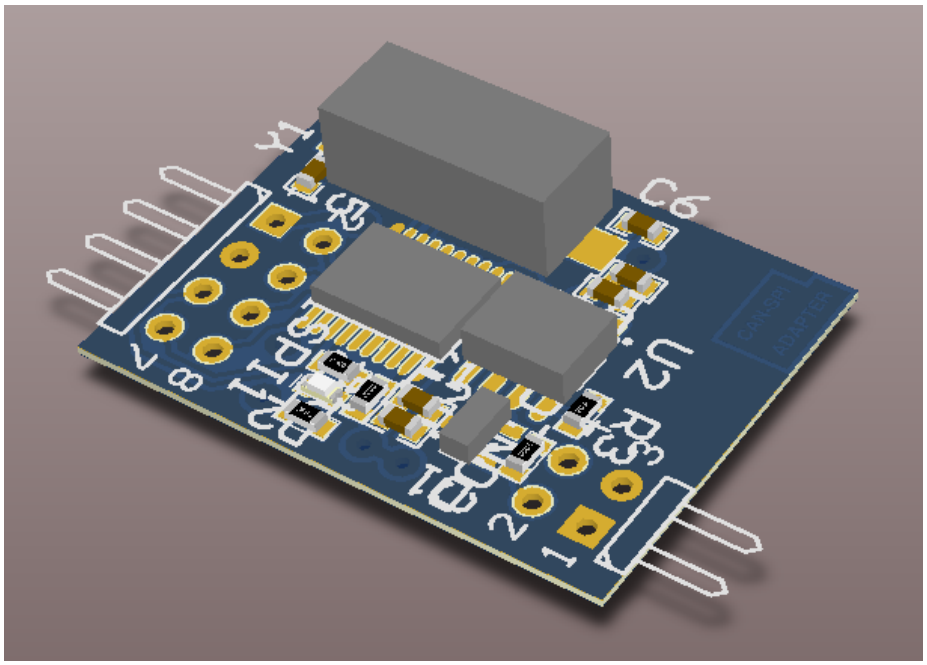
Schemat ideowy przejściówki



Schemat PCB przejściówki



Widok w 3D

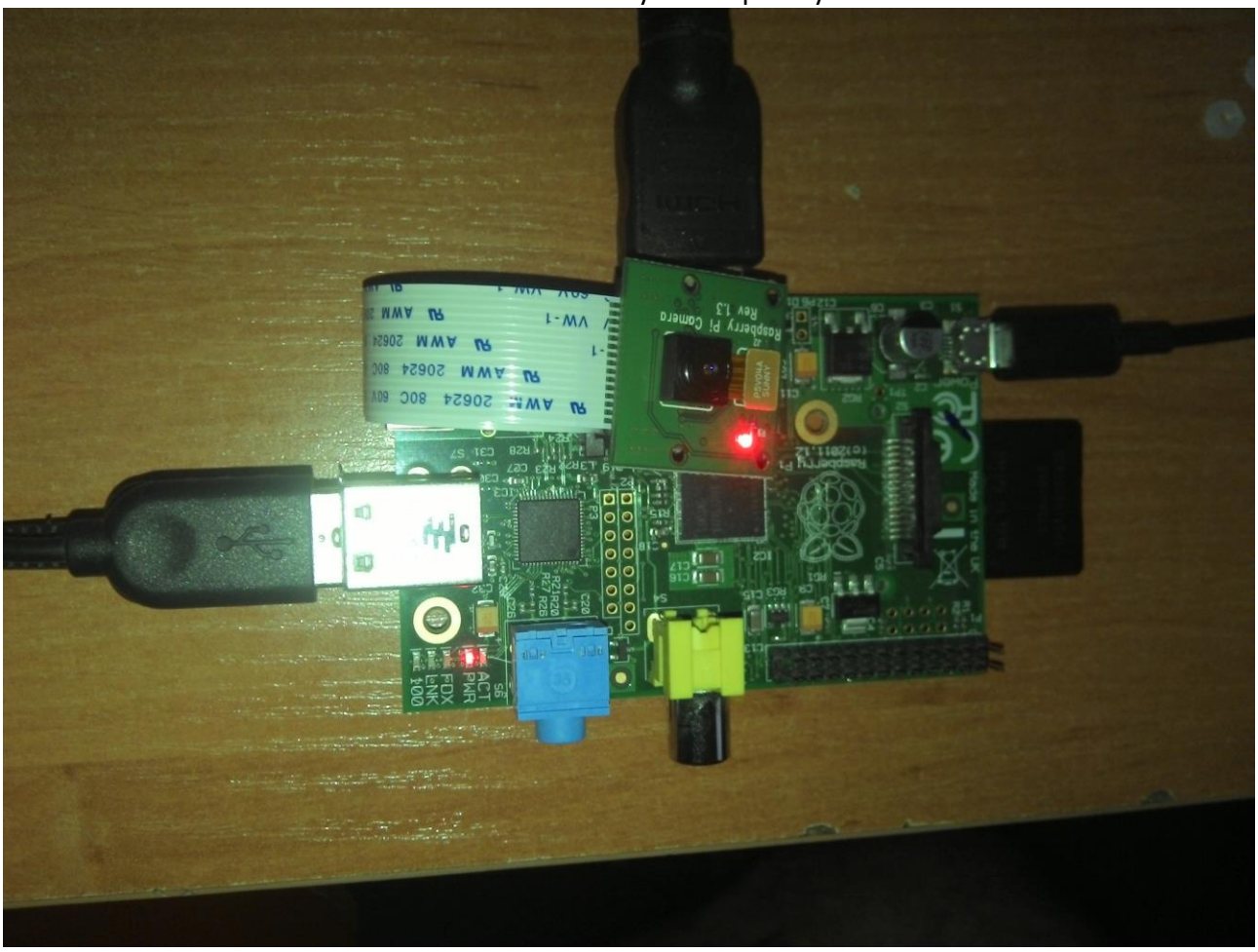


6.2. Obsługa kamery cyfrowej

6.2.1. Rozpoznanie budowy kamery oraz sposób przesyłania obrazu z kamery do mikroprocesora.

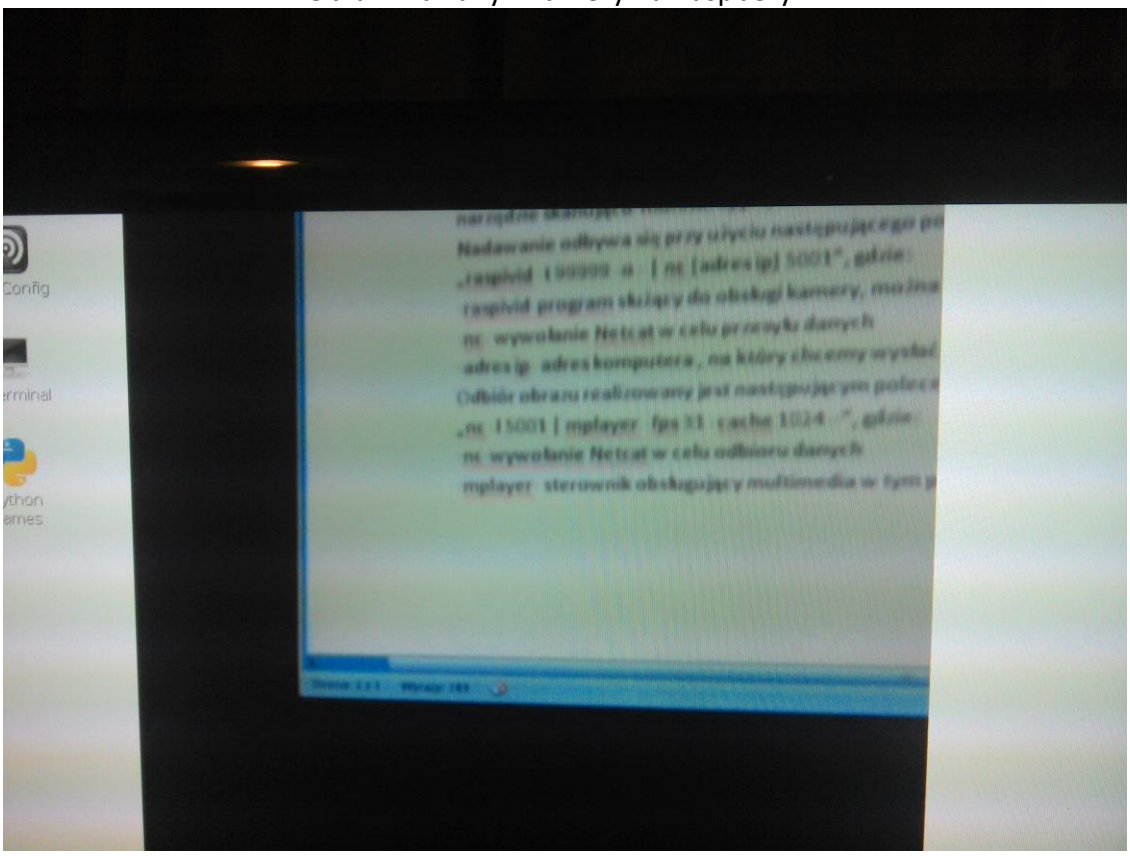
Użyta została kamera Raspberry PI Camera, która jest dedykowana do komputera Raspberry.
Dane techniczne:
-matryca 5MPx
-wspierane rozdzielczości-1080p/720p/640x480 video
-30fps dla 1080p

Podłaczenie kamery do Raspberry PI



Aby przesłać obraz z kamery do mikroprocesora początkowo należy uruchomić konfigurację, komendą ras pi-config, następnie należy włączyć obsługę kamery(Enable Camera). Kolejną czynnością jest odpowiednie wywołanie polecenia raspivid, bądź raspistill. Przykład takiego polecenia poniżej:
„ raspivid –t 10000 –w 200 –h 200 –o”, gdzie:
-t- czas w milisekundach
-o- określenie wyjścia
-w,h- rozdzielczość

Obraz widziany z kamery na Raspbery PI



6.2.2. Wysyłanie obrazu na jednostkę nadrzędną.

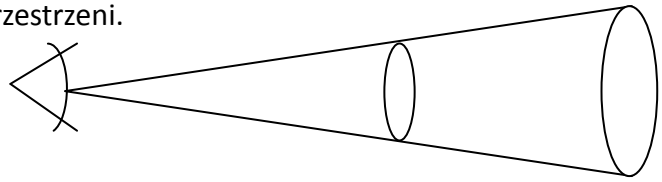
Przesył obrazu odbywa się przy wykorzystaniu kabla Ethernet bądź modułu wifi jako medium. Protokół jaki jest wykorzystywany do tego celu to Netcat, jest to tak naprawdę narzędzie skanująco-monitorujące, a także prosty serwer.
Nadawanie odbywa się przy użyciu następującego polecenia:
„raspivid –t 99999 –o - | nc [adres ip] 5001”, gdzie:
-raspivid-program służący do obsługi kamery, można go zastąpić programem raspistill
-nc- wywołanie Netcat w celu przesyłu danych
-adres ip- adres komputera , na który chcemy wysłać obraz z kamery
Odbiór obrazu realizowany jest następującym poleceniem:
„nc –l 5001 | mplayer –fps 31 –cache 1024 –”, gdzie:
-nc-wywołanie Netcat w celu odbioru danych
-mplayer- sterownik obsługujący multimedia w tym przypadku obraz/wideo.

6.2.3. Przetwarzanie obrazu z kamery

Odczytanie odległości do obiektu na podstawie obrazu z kamery oraz znanej telemetrii platformy latającej jest zadaniem dość złożonym. Głównym czynnikiem umożliwiającym wyliczenie drogi do zadanej przeszkody jest ruch kamery. Jest to spowodowane tym, iż robiąc zdjęcie z jednego miejsca nie jesteśmy w stanie określić trójwymiarowej mapy terenu. Prostym przykładem potwierdzającym słuszność tej tezy jest łapanie nadlatującej piłki z otwartym jednym lub dwoma oczami. W przypadku, gdy łapacz ma otwarte obydwoje oczu to bez problemu łapie nadlatującą piłkę. Natomiast, jeśli jedno oko zostanie zamknięte to wtedy mogą wystąpić znaczne trudności z chwyceniem nadlatującego obiektu. Człowiek patrząc widzi świat w trójwymiarowej przestrzeni, lecz gdy zamknie jedno oko i będzie patrzeć nieruchomo przed siebie to nie będzie w stanie określić dokładnej odległości do obiektu.

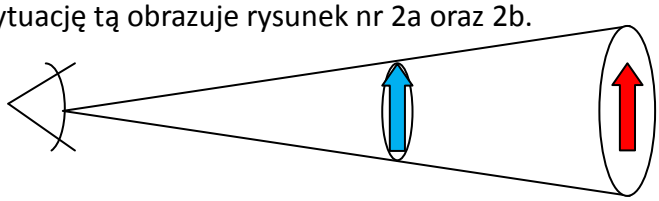
Wstęp teoretyczny

Rozróżniamy dwa główne tory ruchu – ruch prostopadły do płaszczyzny obrazu lub równoległy do płaszczyzny obrazu[1]. W praktyce oznacza to tyle, iż albo idziemy w kierunku którym patrzymy albo idziemy bokiem. W przypadku gdy zbliżamy się lub oddalamy od obiektu, obiekt ten sprawia złudne wrażenie zmiany swoich rozmiarów. Natomiast gdy poruszamy się równolegle do naszego obrazu (na przykład jedziemy autem i obserwujemy drzewa) wydaje się nam, że obiekt porusza się z jakaś prędkością, a im się dalej znajduje tym wolniej się porusza. Spowodowane to jest tym, iż patrząc widzimy tak naprawdę wycinek przestrzeni.



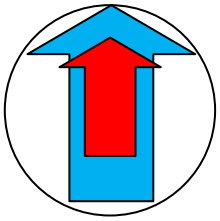
Rys.1. Wycinek przestrzeni, jaką zauważa człowiek dla bliskiej i dalekiej odległości od obiektu

Rysunek 1 ukazuje, iż dla różnych odległości od obiektu mamy różny zakres percepcji otoczenia. W przypadku gdy widzimy dwa lub więcej obiektów, które znajdują się w różnych odległościach to tak naprawdę możemy przyjąć, że otrzymujemy dwa obrazy. Ze względu na różną wielkość odebranych obrazów to obrazy te zostają znormalizowane do jednej wielkości. Wtedy również występuje złudzenie zmniejszenia się obiektów w większej odległości. Sytuację tą obrazuje rysunek nr 2a oraz 2b.



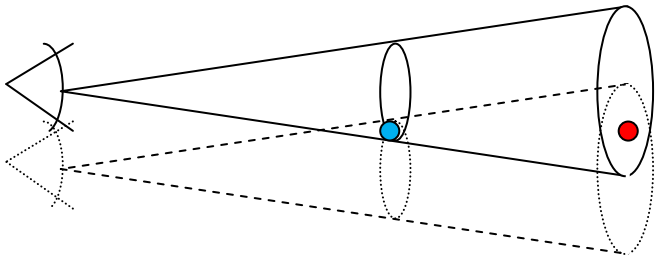
Rys.2a Interpretacja wielkości tego samego obiektu dla różnych odległości

[1] Rozróżniamy ruch prostopadły oraz ruch po danym okręgu – obiekty są od nas w tej samej odległości jeśli leżą na okręgu, a nie na prostej. Lecz ze względu, iż będziemy mieć bardzo małe przesunięcia kątowe, gdyż będziemy mieli relatywnie wysoką częstotliwość pobieranych próbek, to możemy przyjąć bez straty dla założenia, że obiekt leży na stycznej do okręgu.

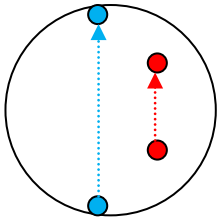


Rys.2b Interpretacja wielkości tego samego obiektu dla różnych odległości po założeniu obrazów(widok z przodu)

W podobny sposób można wyjaśnić zjawisko różnych pozornego przemieszczania obiektów znajdujących się w innych odległościach.



Rys 3.a Interpretacja pozornej drogi nieruchomego punktu dla różnych odległości



Rys 3.b. Interpretacja pozornej drogi nieruchomego punktu dla różnych odległości

t. Sposób wyliczania odległości na podstawie zestawienia dwóch obrazów

W celu wyliczenia odległości od obiektu skorzystamy z uproszczonego modelu percepcji. Ułatwienie będzie polegać na tym, iż weźmiemy pod tylko jeden wymiar uzyskanego obrazu. Udogodnienie to w żaden sposób nie wpłynie na niepoprawność algorytmu wyliczającego odległość od obiektu.

Założenia algorytmu:

-znamy dokładny kierunek oraz wartość prędkości pojazdu na którym jest kamera

-znamy orientację kamery względem pojazdu

-potrafimy odczytać dokładne wymiary obiektu z obrazu z kamery

-znamy zakres kątowy kamery

-rozdzielczość kamery

Zasada działania algorytmu:

Znając kierunek oraz wartość prędkości pojazdu, jak również orientację kamery względem pojazdu jesteśmy w stanie wyliczyć składową prędkości prostopadłą do kamery oraz składową równoległą do kamery. Jeśli prędkości składowe kamery zostaną wymnożone przez czas, który mija pomiędzy odbiorem kolejnej ramki zawierającej obraz to otrzymamy składowe drogi przebytej przez pojazd w układzie odniesienia kamery.

Równania umożliwiające wyliczenie składowych drogi przebytej przez kamerę:

$$l_{prost} = v_{pojazdu} * \cos(\alpha) * t$$

$$l_{rów} = v_{pojazdu} * \sin(\alpha) * t$$

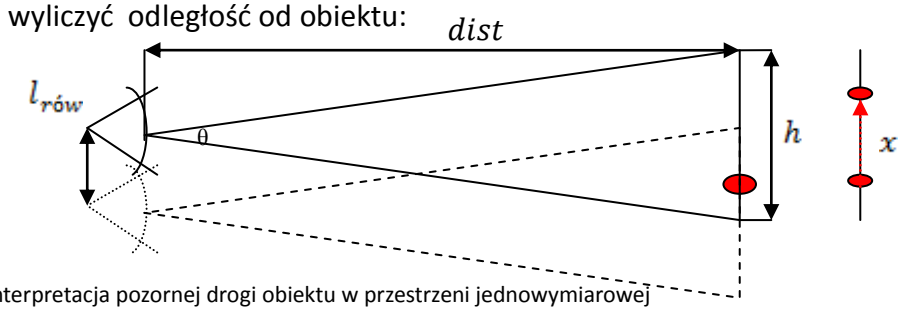
α- orientacja kamery względem kierunku ruchu pojazdu

t- czas pomiędzy odbiorem kolejnej ramki zawierającej obraz

Problem wyliczenia odległości od przeszkody na podstawie obrazu można podzielić na dwa podproblemy. Pierwszy z nich to sytuacja gdy pojazd porusza się zgodnie z orientacja kamery. Drugi natomiast to przypadek gdy poruszamy się prostopadle do orientacji kamery tj. równolegle do naszego odbieranego obrazu. Cała reszta przypadków jest połączeniem dwóch wyżej wymienionych sytuacji.

Pierwszy podproblem: poruszamy się prostopadle do orientacji kamery (równolegle do obrazu)

Wykonując ruch prostopadły do orientacji kamery wydaje się nam, iż obiekt pozornie się przesuwa. Znając drogę przebytą oraz drogę pozornie przebytą przez obiekt jesteśmy w stanie wyliczyć odległość od obiektu:

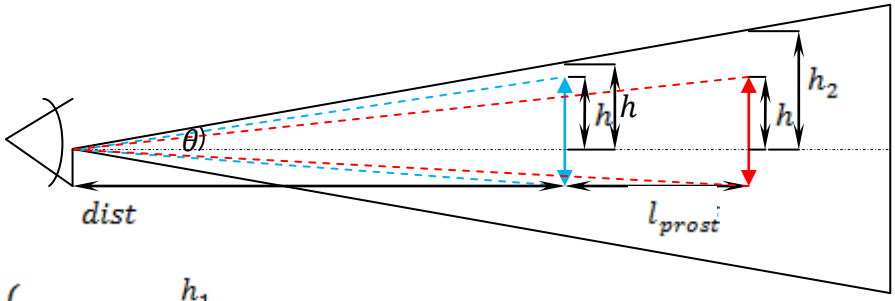


Rys.4. Interpretacja pozornej drogi obiektu w przestrzeni jednowymiarowej

W czasie ruchu prostopadłego do orientacji kamery przesunęliśmy się względem obiektu o drogę równą $l_{rów}$ natomiast według obrazu obiekt przesunął się jedynie o x pikseli znając rozdzielczość kamery jesteśmy w stanie wyliczyć ile procent całego wymiaru wynosi nasze przesunięcie x (przykładowo jeśli rozdzielczość kamery to 1920x1080 to procentowo przesuniecie wynosi $\frac{x}{1920}$ lub $\frac{x}{1080}$). Wiemy również, że przesunięcie x jest równe $l_{rów}$. Jeśli teraz $l_{rów}$ podzielimy przez ilość procent całego wymiaru obrazu to otrzymamy wartość która będzie określać jaką długość lub szerokość ma przestrzeń na obrazie (jeśli przesunęliśmy się o 100m a na obrazie wyszło ze przesunęliśmy 200 pikseli to cały obraz będzie zawierać przestrzeń $h = 100 * \frac{1920}{200}\,[m]$ lub $h = 100 * \frac{1080}{200}\,[m]$). Jeśli poznaliśmy już wymiar przestrzeni zawartej na obrazie to teraz korzystając z funkcji trygonometrycznych jesteśmy w stanie wyliczyć odległość od obiektu: $dist = \frac{\frac{h}{2}}{tg\left(\frac{\theta}{2}\right)}$

Drugi podproblem: poruszamy się prostopadle do płaszczyzny obrazu

Gdy poruszamy zbliżamy się do określonego obiektu, zauważamy, iż przeszkoda ta zwiększa swoje wymiary. Spowodowane to jest tym, że obiekt nie zmienia wielkości, lecz nam maleje obszar percepcji. Znając wartość przesunięcia w kierunku równoległym do orientacji kamery oraz odczytując wielkości zadanego obiektu na obrazie z kamery jesteśmy w stanie wyliczyć odległość kamery od obiektu.



$$\begin{cases} tg\theta = \frac{h_1}{dist} \\ tg\theta = \frac{h_2}{dist + l_{prost}} \end{cases} => dist = \frac{h_1 l_{prost}}{h_2 - h_1} = \frac{l_{prost}}{\frac{h_2}{h_1} - 1}$$

Wysokości h, h_1, h_2 są to wysokości obiektów odczytane w metrach, a więc są to wysokości rzeczywiste obiektu, zakresu percepcji. Wysokości te są niestety nieznane. Lecz na podstawie uzyskanego obrazu jesteśmy w stanie wyliczyć zależności $n_1 = \frac{h_1}{h}$ oraz $n_2 = \frac{h_2}{h}$. $h_1$oraz h_2 jest równe połowie rozdzielczości uzyskanego obrazu (960/540px dla przykładowego obrazu). Obiekt jest opisany współrzędnymi x_{11}, x_{12} dla obiektu niebieskiego oraz x_{21}, x_{22} dla obiektu czerwonego. Punkty te określają odpowiednio początek oraz koniec obiektu. W celu wyliczenia h należy od współrzędnych początku obiektu odjąć połowę rozdzielczości obrazu ($x_{11} - 960px, x_{21} - 960px$). Tak więc, gdy znamy już ile wynosi n_1, n_2 to możemy wyliczyć ile wynosi $h_1$oraz h_2 ($h_1 = n_1 h = \frac{960}{x_{11} - 960} h$),a następnie możemy podstawić to do równania $dist = \frac{l_{prost}}{\frac{h_2}{h_1} - 1}$ otrzymując następujące rozwiązanie $dist = \frac{l_{prost}}{\frac{n_2 h}{n_1 h} - 1} = \frac{l_{prost}}{\frac{n_2}{n_1} - 1}$. Jeśli przyjmiemy, że s wynosi połowę rozdzielczości to $n_1 = \frac{s}{x_{11} - s}$ oraz $n_2 = \frac{s}{x_{21} - s}$. Gdy postawimy to do zależności pozwalającej wyliczyć odległość od obiektu to otrzymamy następujące równanie $dist = \frac{l_{prost} |x_{11} - s|}{x_{11} - x_{21}}$. Równanie to umożliwia nam wyliczenie odległości od obiektu przy znajomości współrzędnych wyznaczających początek obiektu.

W przypadku gdy wektor ruchu pojazdu jest złożony z dwóch wektorów składowych: prostopadłego oraz równoległego do orientacji kamery, to w celu wyliczenia odległości od celu należy rozwiązać jeden z dwóch wyżej wymienionych podproblemów. Najlepiej skorzystać z rozwiązania podproblemu nr 2, gdyż daje on natychmiastowe wyniki. Dla poproblemu pierwszego należy wziąć pod uwagę, iż obiekt zmienia swoją odległość wobec nas, co powoduje, że należy scałkować uzyskaną odległość.

Pierwszy sposób rozwiązania[2]

W celu wyliczenia odległości do obiektów należały wpierw wyodrębnić obiekty zawarte w obrazie. Aby uzyskać interesujące nas obszary obrazu musimy wpierw wydobyć krawędzie. Zostały użyte następujące algorytmy detekcji krawędzi:
-Laplasjany
-Filtry gradientowe
-Filtr Kirscha

Laplasjany:
Wykorzystano następujące maski:

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

1	-1	1
-2	4	-2
1	-2	1

-1	-1	-1
-1	9	-1
-1	-1	-1

0	-1	0
-1	5	-1
0	-1	0

Filtry gradientowy:
Obraz wynikowy zostaje wyliczony ze wzoru:

$$G(x,y) = \sqrt{G_x^2 + G_y^2}$$

Filtr Sobela:
Maska G_x Maska G_y

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Filtr Prewitta:
Maska G_x Maska G_y

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

[2] Wszystkie filtry detekcji krawędzi, algorytmy pogrubiania i pocieniania linii oraz segmentacji zostały zaprojektowane w oparciu o „Systemy wizyjne robotów przemysłowych" Ryszarda Tadeusiewicza.

Filtr Roberta:

Maska G_x Maska G_y

-1	-0
0	-1

0	1
-1	0

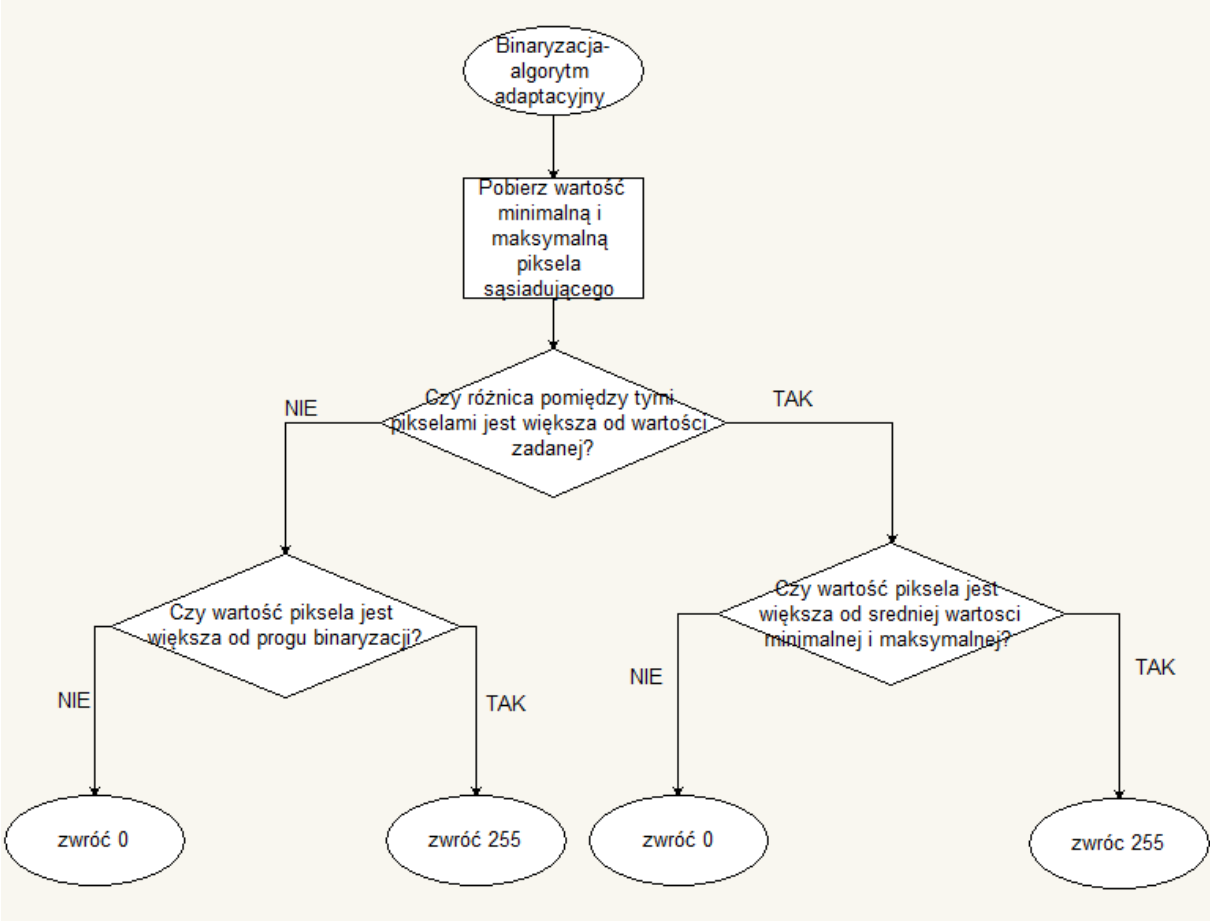
Filtr Kirscha

Filtracja Kirscha składa się z ośmiu filtrów umożliwiających detekcje linii w ośmiu różnych kierunkach. Filtry te można wyliczyć korzystając z wzoru:$\max_{0\le k\le 7}\{1, max\,[|5S_k - 3T_k|]\}$, gdzie $S_k = f_{k\ mod\ 8} + f_{k+1\ mod\ 8} + f_{k+2\ mod\ 8}$ oraz $T_k = f_{k+3\ mod\ 8} + f_{k+4\ mod\ 8} + f_{k+5\ mod\ 8} + f_{k+6\ mod\ 8} + f_{k+7\ mod\ 8}$. Zmienna k przyjmuje wartości według następującej numeracji:

0	1	2
7	i,j	3
6	5	4

Najlepsze wyniki zwracał filtr Kirscha, lecz jego wykonanie zajmowało nawet ośmiokrotnie większy czas niż w przypadku filtrów gradientowych i prawie dwudziestokrotny niż w przypadku filtrów Laplace'a.

Filtry te umożliwiają nam detekcję krawędzi lecz zwracają one obraz w odcieniach szarości. W celu jasnego określenia krawędzi należy przeprowadzić binaryzację obrazu. Najprostszą metodą binaryzacji jest określenie progu poniżej którego piksele przyjmują wartość 0, a powyżej wartość 255. Niestety, metoda ta zwraca różne wyniki w zależności od kontrastu uzyskanych krawędzi – jeśli krawędź jest zbyt jasna to algorytm binaryzacji może jej nie wychwycić. Dlatego też została stworzona adaptacyjna metoda binaryzacji, której działanie zostało przedstawione na poniższym rysunku:



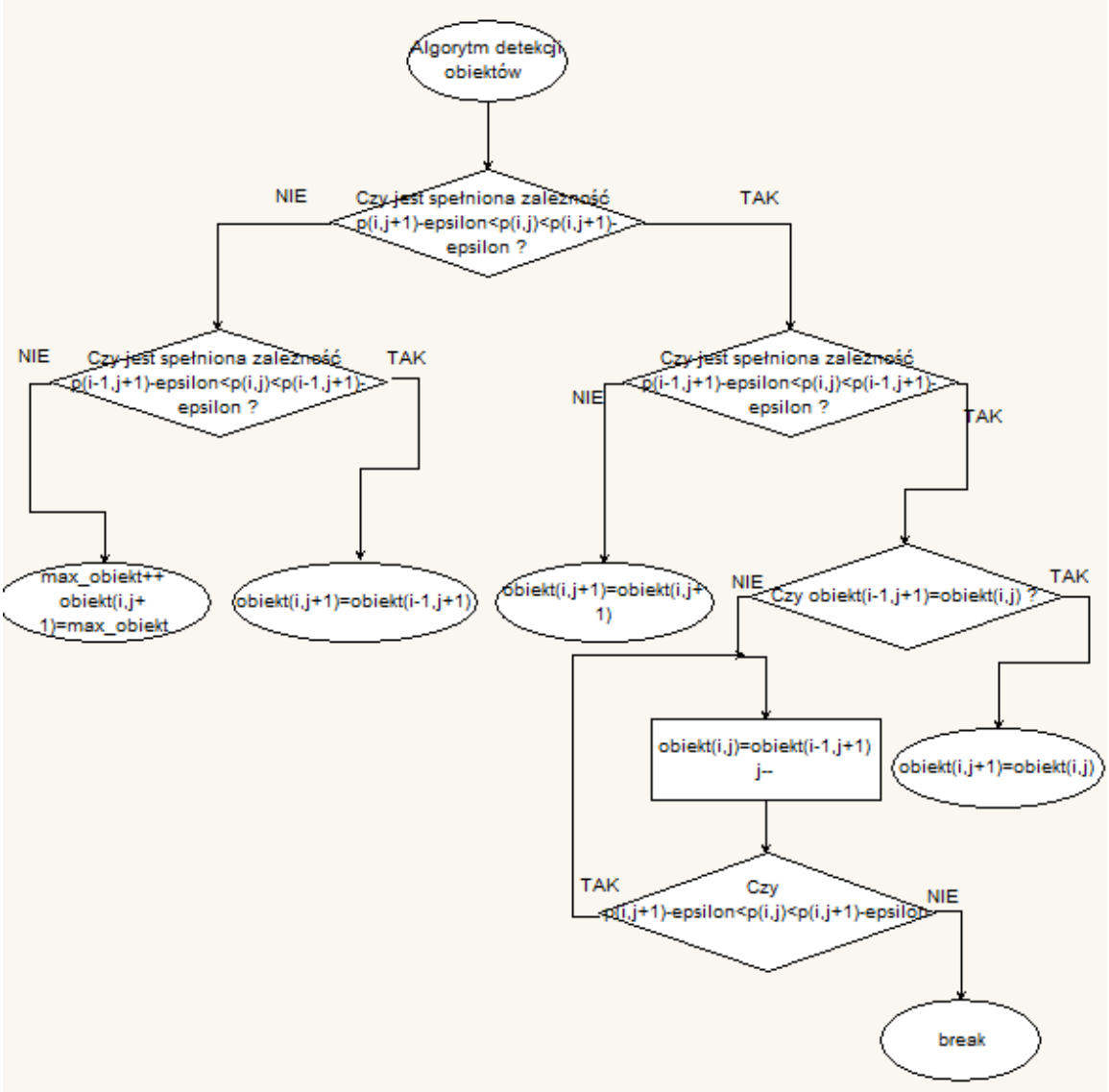
W celu zniwelowania błędów binaryzacji oraz detekcji krawędzi zastosowano pogrubianie a następnie odcienianie linii. Pogrubianie linii polega na tym, iż jeśli w bliskim sąsiedztwie danego piksela, który jest krawędzią znajduje się jakiś piksel który również jest krawędzią to wszystkie dziewięć pikseli (zadany piksel oraz jego sąsiedzi) zostaje uznanych za krawędź. Algorytm pocieniania działa przeciwnie – tzn. jeśli wszystkie dany piksel wraz z całym sąsiedztwem wskazuje na krawędź to wszyscy sąsiedzi są uznawane za tło, a dany piksel jest krawędzią. Zastosowanie takiej metody pozwoli na zniwelowanie pojedynczych pikseli które są błędami binaryzacji oraz pozwala na połączenie krawędzi w przypadku, gdy wykryta krawędź nie jest ciągła. Następnie obraz poddano segmentacji to znaczy wyodrębniono obiekty, które znajdują się na obrazie. Wyodrębniono dwa sposoby segmentacji – przez podział i tzw. 'pożar prerii'. Segmentacja liniowa polega na tym, iż sprawdzane jest czy różnica koloru pomiędzy jednym a drugim pikselem mieści się w zakresie. Jeśli tak to jest to ten sam obiekt. W przypadku segmentacji 'pożar prerii' jest wybierany piksel, następnie jest sprawdzane czy piksele sąsiednie mają ten sam kolor co piksel zadany. Jeśli jest spełniona ta zależność to dla wszystkich otrzymanych wyników jest znów sprawdzany warunek czy piksele sąsiednie mają ten sam kolor. Funkcja ta jest funkcją rekurencyjną tzn. wywołuje samą siebie co powoduje zajęcie dość dużej ilości pamięci.

Niestety, wyżej opisany algorytm wykrywania obiektów nie dawał dobrych wyników – dla dużych obrazów następowało załączanie się zabezpieczeń dla ilości wywołań funkcji rekurencyjnej. Spowodowane to było tym, że metody wykrywania krawędzi oraz binaryzacji dawały zbyt duże błędy, których dylatacja, następnie erozja linii nie była w stanie aproksymować. Następowały wtedy przerwy w liniach kontur, co powodowało 'wylewanie się' obiektów poza swoje granice. W celu poprawy wyników zaimplementowano wykrywanie linii na obrazie kolorowym.

Metoda ta polegała na podzieleniu obrazu na trzy podobrazy (na paletę czerwona, zieloną i niebieską), wykryciu krawędzi i zbinaryzowaniu każdego podobrazu a następnie zsumowaniu ich, tworząc jeden obraz wynikowy. Uzyskany wynik jednak w dalszym ciągu uniemożliwiał przeprowadzenie poprawnej segmentacji. Skwantowanie obrazu również nie poprawiło jakości algorytmu wykrywania obiektów. Dlatego też utworzono własny algorytm wykrywania obiektów.

Drugi sposób rozwiązania

Drugim podejściem rozwiązania problemu wykrywania obiektów było zaprojektowanie własnego sposobu wyodrębnienia obiektów. Program ten polegał na badaniu różnicy kolorów pomiędzy poszczególnymi pikselami. Dokładna zasada algorytmu została przedstawiona poniżej:

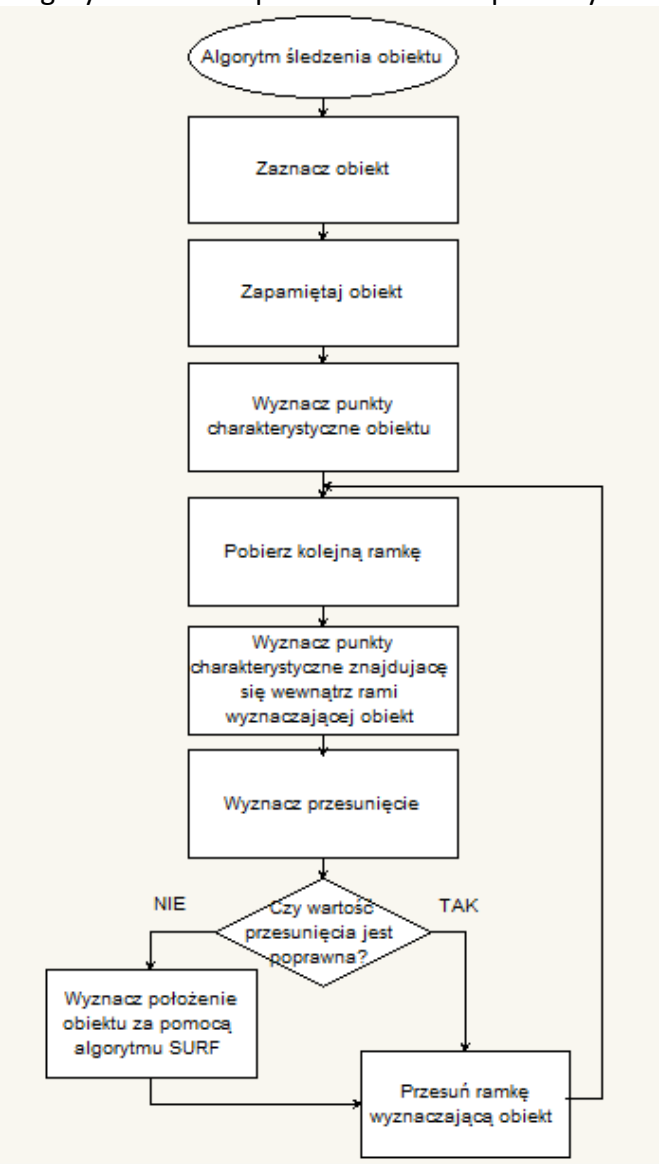


Algorytm ten umożliwiał na utworzenie optymalnej liczby obiektów, dzięki czemu nie tworzyły się pseudoobiekty mające ten sam kolor i znajdujące się koło siebie. Wadą tego rozwiązania jest to, iż jest to metoda zwracająca idealne wyniki jedynie dla obrazów utworzonych w programie Paint. Znaczy to, że algorytm ten nie działa jedynie dla obrazów bez zaszumień i błędów. Nawet kwantyzacja obrazu uzyskanego z kamery nie poprawiła

znacząco poprawności wyników tego algorytmu.
Ze względu, iż oba sposoby rozwiązań nie spełniły swoich założeń to należało znaleźć całkiem nową ścieżkę wyznaczenia obiektu.

Trzeci sposób rozwiązania

Kolejnym sposobem rozwiązania było ręczne wyznaczenie interesującego obiektu na obrazie. W tym celu użyto biblioteki OpenCV. Wadą tego sposobu było to, iż po przesunięciu obrazu nie algorytm sam nie wykrywał nowej pozycji obiektu. W tym celu należało zastosować rozwiązanie umożliwiające śledzenie obiektu. Spośród wielu istniejących skupiono się metodzie zwanej *Optical Flow* oraz *SURF*. Oba algorytmy opierają tzw. Punktach charakterystycznych tzn. na punktach obrazu, które są stosunkowo proste do odnalezienia oraz które istnieją niezależnie od sposobu jego przetworzenia. Metoda *Optical Flow* skupia się głównie na wyznaczeniu punktów charakterystycznych, natomiast algorytm *SURF* umożliwia wykrycie zadanego obiektu na obrazie. Połączenie tych dwóch rozwiązań umożliwiło utworzenie oprogramowania pozwalającego na śledzenie zadanego obiektu. Zasada działania tego algorytmu została przedstawiona na poniższym schemacie blokowym.



Utworzony algorytm początkowo czeka na wyznaczenie interesującego nas obszaru, następnie zostają wyznaczone punkty charakterystyczne zaznaczonego obszaru.

Nie zostają wyznaczone wszystkie punkty charakterystyczne, gdyż pozwala to na zmniejszenie mocy obliczeniowej. Po wyznaczeniu tych punktów następuje pobranie następnej ramki danych z kamery oraz obliczenie punktów charakterystycznych znajdujących się wewnątrz interesującego nas obszaru. W celu wyliczenia przesunięcia początkowo stosowano proste przyrównanie odległości – wybierano dwa punkty znajdujące się w centrum interesującego nas obszaru, a następnie wyliczano odległość jednego oraz drugiego punktu po kolei do wszystkich uzyskanych punktów następnej ramki i sprawdzano czy któreś z długości się pokrywają. Jeśli tak to zostało odnalezione przesunięcie. W przeciwnym wypadku stosowano algorytm *SURF* umożliwiający od nowa wykrycie danego obszaru na obrazie. Algorytm *SURF* stosowano jedynie w krytycznych wypadkach, albowiem jest to metoda dość zasobożerna w stosunku do metody *Optical Flow*. Niestety, proste przyrównanie odległości nie pozwalało wykrywać obrotów obiektów, jak również ich przybliżania oraz oddalania. Powodowało one również dość duże błędy. Dlatego też metoda ta została zastąpiona. Drugie rozwiązanie tego problemu polegało na pobraniu trzech punktów znajdujących się w centrum interesującego nas obszaru. Ze względu, iż odległości mogą się zmieniać to skupiono się na innym aspekcie – kątach jakie tworzą proste przechodzące przez te trzy punkty. Przy wszystkich możliwościach przetwarzania obrazu kąty nie zmieniają swoich wartości. Natomiast główną ich wadą było wyliczenie ich wartości. Po wyrysowaniu problemu oraz rozpisaniu wszystkich zależności otrzymano następujące wzory: $tg\alpha_{12} = \frac{p1.y - p2.y}{p1.x - p2.x}$, $tg\alpha_{13} = \frac{p1.y - p3.y}{p1.x - p3.x}$, $\alpha = |atg\frac{-tg\alpha_{12} + tg\alpha_{13}}{1 + tg\alpha_{12} * tg\alpha_{13}}|$, gdzie $p1, p2, p3$ wskazuje kolejno na punkty znajdujące się w centrum. Wzory te po lekkim zmodyfikowaniu można użyć do wyliczenia kolejnych kątów. Otrzymane kąty są z zakresu <0,90), więc w przypadku wystąpienia kąta rozwartego nasz wynik jest niepoprawny. W tym celu stosujemy poprawkę, która w przypadku gdy suma wszystkich trzech kątów jest mniejsza od 180 stopni to największy z kątów jest równy 180 pomniejszony o sumę wszystkich kątów podzielonych przez dwa. Ze względu na stosunkowo małą ilość obliczeń dalszy sposób wyliczenia przesunięcia jest podobny jak w poprzednim wypadku tzn. dla następnej ramki wybieramy każde możliwe trzy punkty i liczymy i sprawdzamy kąty pomiędzy nimi do momentu gdy otrzymany kąt jest inny kąty zadane. Gdy zostaną już wybrane zbiory punktów spełniających te zależności to spośród wszystkich zbiorów zostaje wybrany ten zbiór, który jest najbardziej podobny do zbioru zadanego – znajduje się stosunkowo blisko punktów centralnych, odległości pomiędzy punktami są podobne oraz kąt obrotu jest niewielki. Każdy z tych trzech warunków wskazuje również na trzy podstawowe cechy: odpowiednio przesunięcie, powiększenie lub pomniejszenie obiektu oraz obrót. Dzięki temu jesteśmy w stanie precyzyjnie określić położenie interesującego nas obszaru, a co za tym idzie, jesteśmy w stanie wykryć ruch naszego obiektu.

Po zaimplementowaniu algorytmów umożliwiających wyliczenie odległości od obiektu okazało się, iż uzyskane wartości posiadają dość duże błędy (rzędu nawet kilkunastu procent). Spowodowane to jest ruchem kamery, szumami i zakłóceniami oraz niedokładnym wyznaczeniem punktów charakterystycznych, jak również faktem, że punkty te nie zawsze wyznaczały te same obszary interesującego nas obiektu. W celu poprawy otrzymanych wyników należałoby użyć filtrów dolnoprzepustowych lub średniej ruchomej. Niestety, ze względu na ograniczony czas projektu nie udało mi się zaimplementować wyżej wymienionych filtrów.

6.3. Obsługa macierzy mikrofonów.

6.3.1. Zaprojektowanie oraz wykonanie macierzy mikrofonów

Jednym z najważniejszym elementów rozpoznawania mowy jest wykonanie modułu rejestrującego dźwięk. Moduł powinien spełniać następujące warunki:

- rejestrować częstotliwości z zakresu mowy ludzkiej
- tłumić niepożądane częstotliwości
- umożliwiać rozpoznanie kierunku z którego dochodzi dźwięk

Aby zrealizować powyższe warunki wykorzystaliśmy mikrofony ADMP441ACEZ oraz BESTAR BCM9745P-44 są to mikrofony o charakterystyce dookólnej. Komunikacja z pierwszym mikrofonem - cyfrowym realizowana jest przy użyciu magistrali I2S, natomiast drugi mikrofon jest urządzeniem analogowym.

Wykorzystanie macierzy mikrofonów umożliwi określenie kierunku z którego docierać będzie dźwięk. Zakres częstotliwości dla których taki układ mikrofonów ma charakterystykę kierunkową(w naszym przypadku wszechkierunkową) jest określony za pomocą wzoru:

$F<=c/2d$

Gdzie:

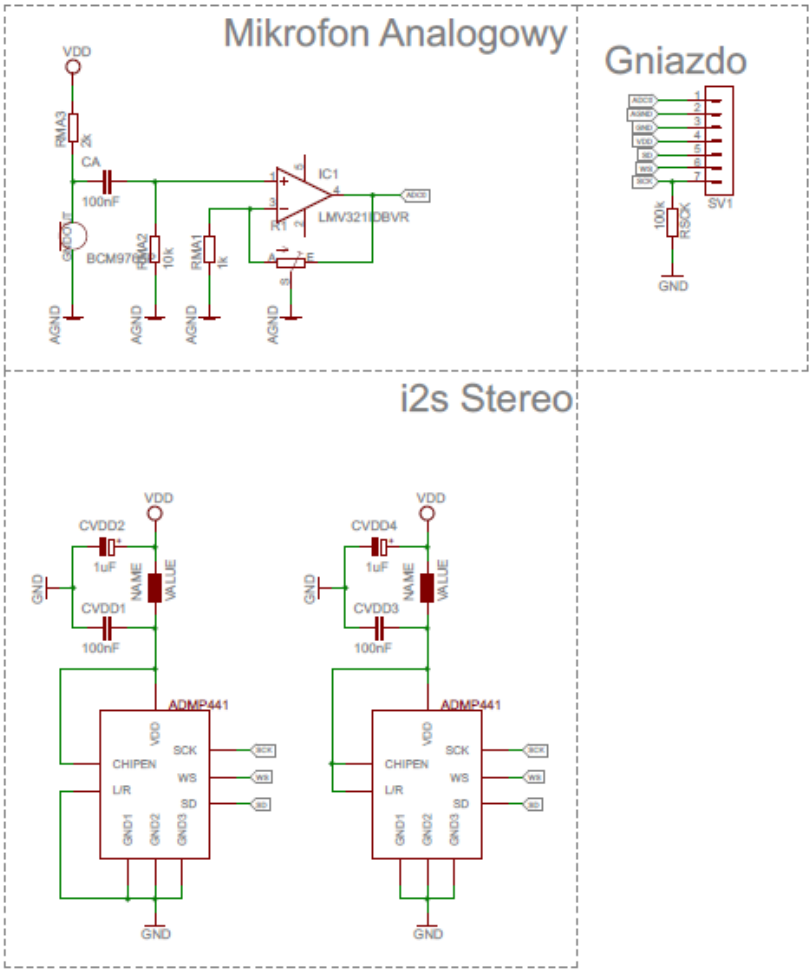
f – częstotliwość [Hz]

c – prędkość dźwięku [m/s]

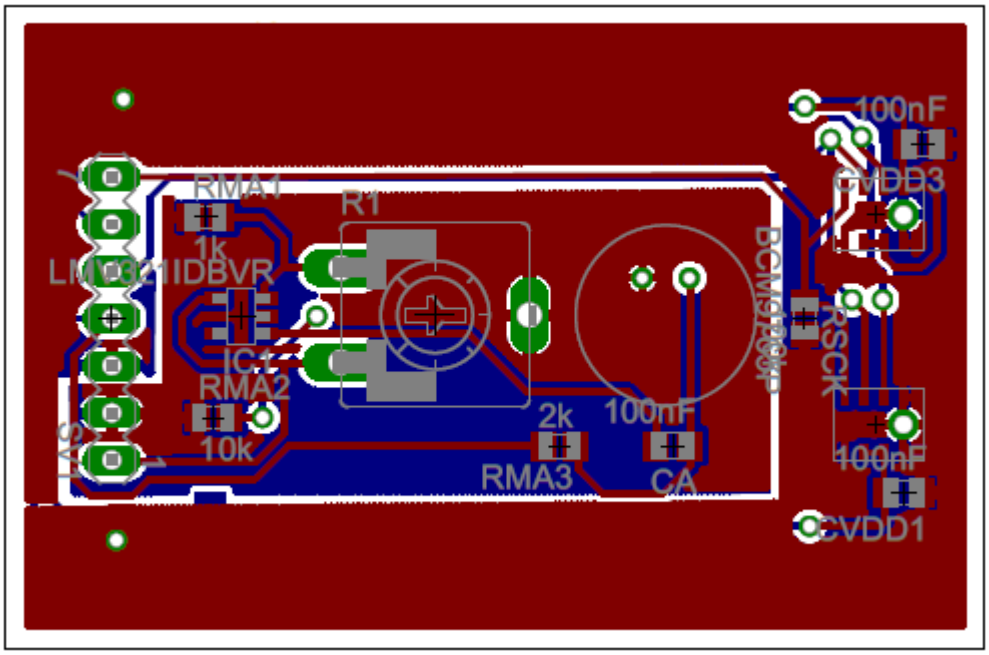
d – odległość między mikrofonami [m]

Odległość została dobrana tak, aby obliczona częstotliwość mieściła się w zakresie mowy ludzkiej.

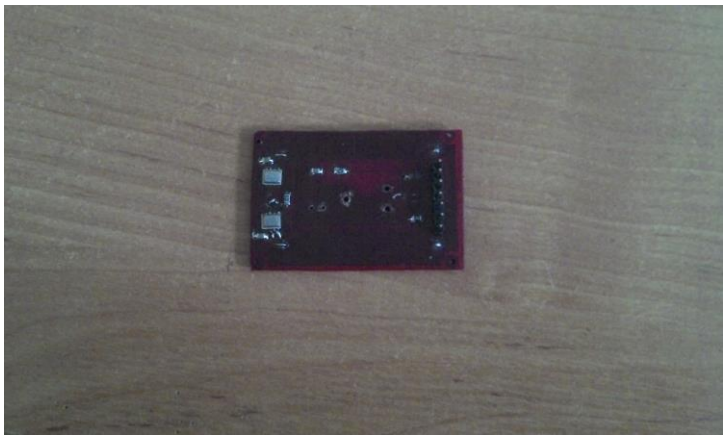
Schemat ideowy macierzy mikrofonów



Schemat montażowy macierzy mikrofonów

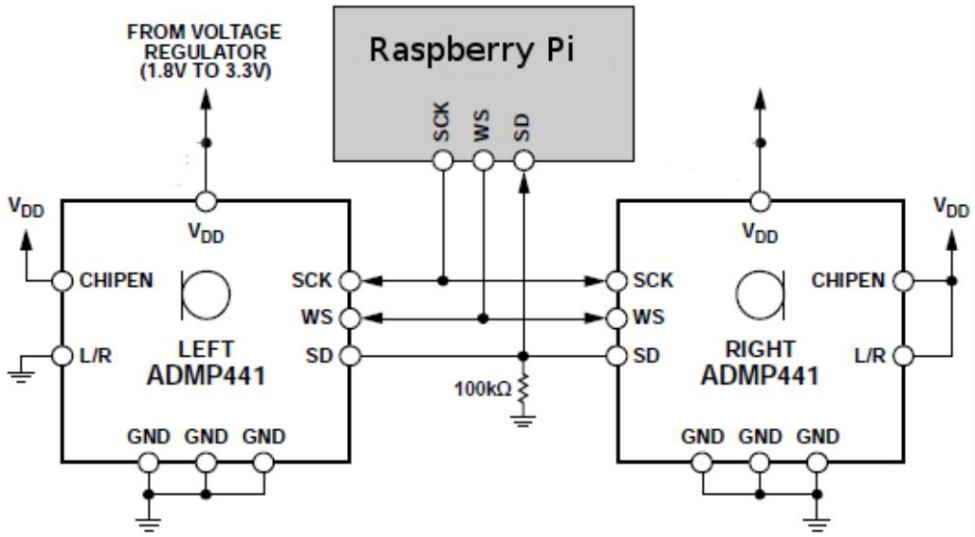


Zdjęcia wykonanego urządzenia



6.3.2. Odczyt sygnałów pochodzących z mikrofonów

Odczyt sygnałów z mikrofonów ADMP441ACEZ zrealizowany jest przy użyci magistrali I2S, schemat ideowy podłączenia mikrofonów.



Komunikacja z analogowym mikrofonem BESTAR BCM9745P-44, z powodu braku odpowiednich elementów oraz czasu nie została zrealizowana.

6.3.3. Przetwarzanie sygnału dźwiękowego oraz algorytm rozpoznawania mowy.

Głównym założeniem tej części projektu było utworzenie oprogramowania umożliwiającego wykrycie komend głosowych, a następnie umożliwić sterowanie pojazdem wedle wydanych rozkazów.

Najprostszym sposobem porównania dwóch sygnałów jest porównanie ich wartości w kolejnych chwilach czasu. Jeśli wartości te są do siebie zbliżone to występuje wysokie prawdopodobieństwo, że te sygnały są podobne. Niestety, w przypadku komend głosowych okazało się, iż metoda ta jest zbyt prosta i nie do końca spełnia swoje założenia. Głównym problemem był fakt, że jeden człowiek nigdy nie wypowie danego słowa tak samo – może powiedzieć część słowa szybciej lub głośniej. Tak więc metoda ta była błędna już w przypadku jednego mówcy, nie wspominając już o innych osobach, które chciałyby sterować

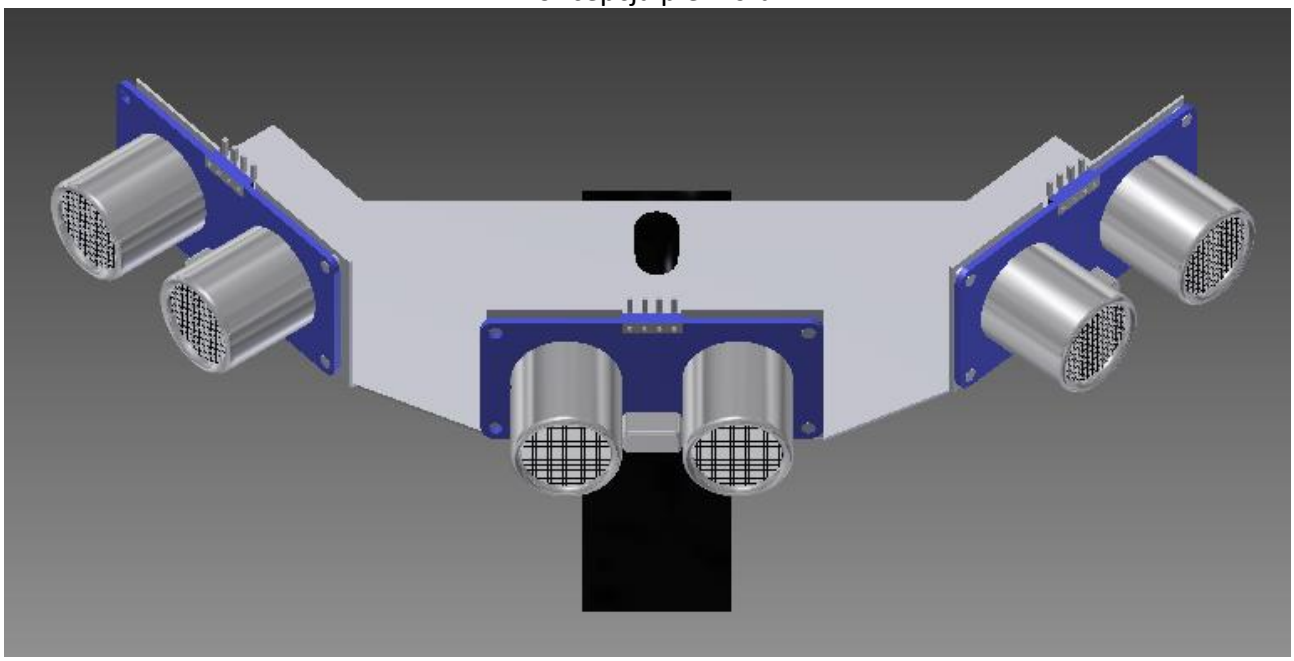
pojazdem za pomocą mowy. Kolejnym krokiem było zapoznanie się z tematyką dotyczącą widma sygnału oraz dyskretną transformatą Fouriera. Umożliwiło nam to wykrycie częstotliwości odebranych sygnałów. Dzięki czemu zostało utworzone testowe oprogramowanie, które umożliwiło wykrywanie gwizdania.
Początkowo oprogramowanie to miało wykrywać klaśnięcie, lecz w trakcie testów okazało się, że wszystkie programy, które teoretycznie reagują na klaskanie to tak naprawdę reagują na głośny dźwięk. Spowodowane to jest tym, że w przeciwieństwie do gwizdu klaśnięcie generuje sygnał oparty na dość szerokim zakresie częstotliwości. Po przetestowaniu powyższego oprogramowania zaprojektowano i zaimplementowano algorytm wykrywania słów oparty o FFT. Program ten zwrócił lepsze wyniki od poprzedniego programu, ale w przypadku zmiany osoby mówiącej do mikrofonu jak również zmiany tempa mowy program ten zwracał błędne dane. Kolejnym krokiem było dogłębne zrozumienie zasady działania aparatu mowy. Mowę ludzką można podzielić na głoski dźwięczne i bezdźwięczne.
W przypadku głosek dźwięcznych trakt głosowy jest pobudzany okresowo, składając się z szeregu delt Diraca.
Okres ten jest indywidualny dla każdego człowieka i wskazuje na ton jego głosu – dla mężczyzn częstotliwość ta waha się pomiędzy 80-480Hz, natomiast w przypadku kobiet od 160 do 960Hz. Dla głosek bezdźwięcznych jako modelowe pobudzenie traktu głosowego przyjmuje się szum biały, który po przejściu przez język, podniebienie czy wargi staje się szumem kolorowym o nierównym widmie. Zależności te zostały wykorzystane w kolejnej wersji oprogramowania.
Wersja ta rozpoznawała słowa z podanej bazy na podstawie porównania współczynników cepstralnych metodą DTW obydwu sygnałów. Program ten został utworzony w oparciu o przykładowy kod rozpoznawania izolowanych słów zamieszczony w książce ‘Cyfrowe przetwarzanie sygnałów’ Tomasza P. Zielińskiego. Głównymi zaletami tego programu jest możliwość porównania dwóch sygnałów o różnej długości, jak również wysoka skuteczność rozpoznawania słów w przypadku jednego mówcy. W przypadku innych mówców skuteczność ta spada do około 50%. Wadą natomiast jest fakt, program ten nie wyklucza w żaden sposób słów, które nie pasują do żadnego wzorca w bazie. Wszelkie próby poprawienia skuteczności jak również wykluczenia niepasujących słów nie przyniosły znaczących efektów. W przyszłości oprogramowanie to zostanie przepisane na platformę Raspberry PI, a wszelkie wydawanie komend będzie odbierane przez urządzenie dopiero po wyzwoleniu przerwania sygnałem zewnętrznym (docelowo przyciskiem).

6.4. Wykonanie obrotowego sonaru.

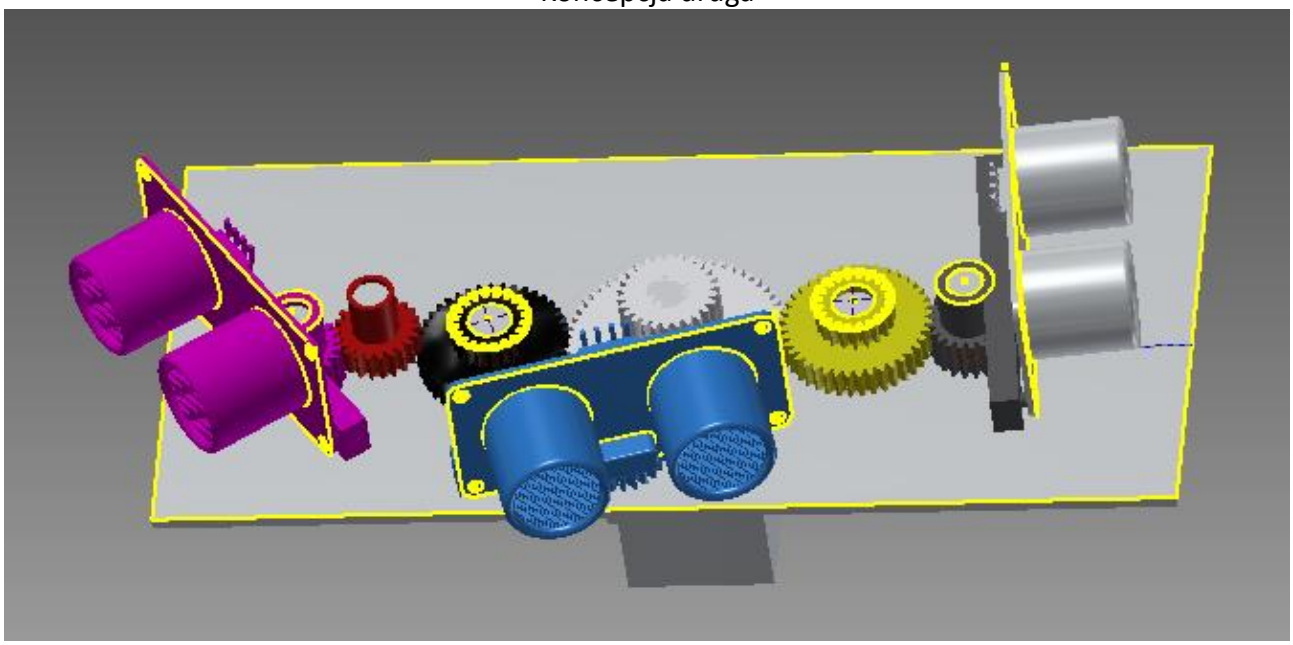
6.4.1. Projekt sonaru oraz podłączenie do systemu elektronicznego.

Sonar został zbudowany w oparciu o trzy czujniki ultradźwiękowe, serwomechanizm modelarski oraz kilka kółek zębatych. Czujniki ultradźwiękowe potrafią wykrywać obiekty w odległości od 2 do około 250cm, i orientacji <-30,30stopni> w stosunku do orientacji czujnika. Jeden czujnik jest zamontowany w dłuższej osi serwomechanizmu i w pozycji neutralnej bada teren dokładnie przed sonarem. Dwa pozostałe czujniki zamontowane są po bokach odchylone od osi o ok 35 stopni. Został wykorzystany standardowy serwomechanizm modelarski sterowany sygnałem PWM, potrafiący obracać się w zakresie 0-180 stopni. Istnieją dwa sposoby obracania się sonaru. W pierwszym sposobie wszystkie czujniki zamocowane są na jednej platformie napędzanej przez serwomechanizm. W drugim sposobie każdy z czujników porusza się osobno. W tym sposobie serwomechanizm jest umieszczony centralnie, a na jego wale znajduje się zamocowane pierwsze, największe koło zębate. Następnie przy udziale kolejnych 2 – 3 (zależnie od czujnika oraz konfiguracji) kółek zębatych obrót serwomechanizmu zostaje przeniesiony na czujnik. Każdy czujnik poruszą się w odgórnie ustalonym zakresie, zależnym od wielkości oraz ilości zębów kółek zębatych.
Każde z tych rozwiązań ma swoje dobre i złe cechy. W pierwszym przypadku konstrukcja całości ulega bardzo poważnemu uproszczeniu, co również może prowadzić do mniejszej awaryjności całego sonaru. Niestety jest to okupione trudniejszym przetwarzaniem danych, trzeba wyznaczać pozycję każdego czujnika osobno. Dodatkowo mamy mniejszy wpływ na zakres kątowy skanowanego terenu. W drugim przypadku przy zastosowaniu tych samych kół zębatych, boczne czujniki działają idealnie symetrycznie. Dodatkowo możemy zmieniać zakres ich działania, zmieniając zastosowane koła zębate. Należy tylko pamiętać by zakresy kątowe poszczególnych czujników nie nachodziły na siebie, ponieważ może to prowadzić do uzyskania błędnych danych pomiarowych.
Ostatecznie została wybrana koncepcja pierwsza. Jej realizacja jest znacznie prostsza to też pozwoliło na skrócenie czasu wykonania sonaru.

Koncepcja pierwsza



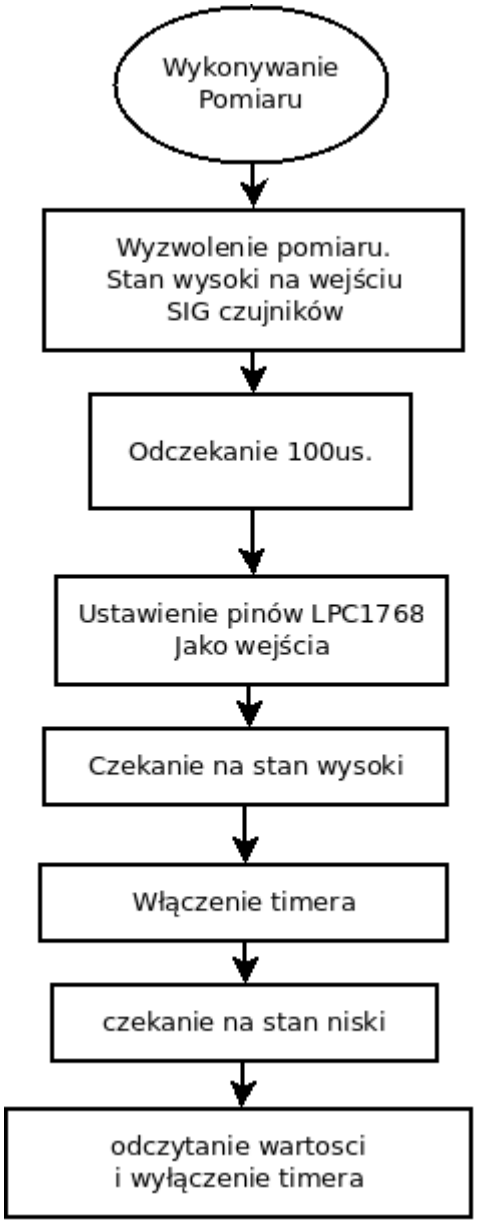
Koncepcja druga



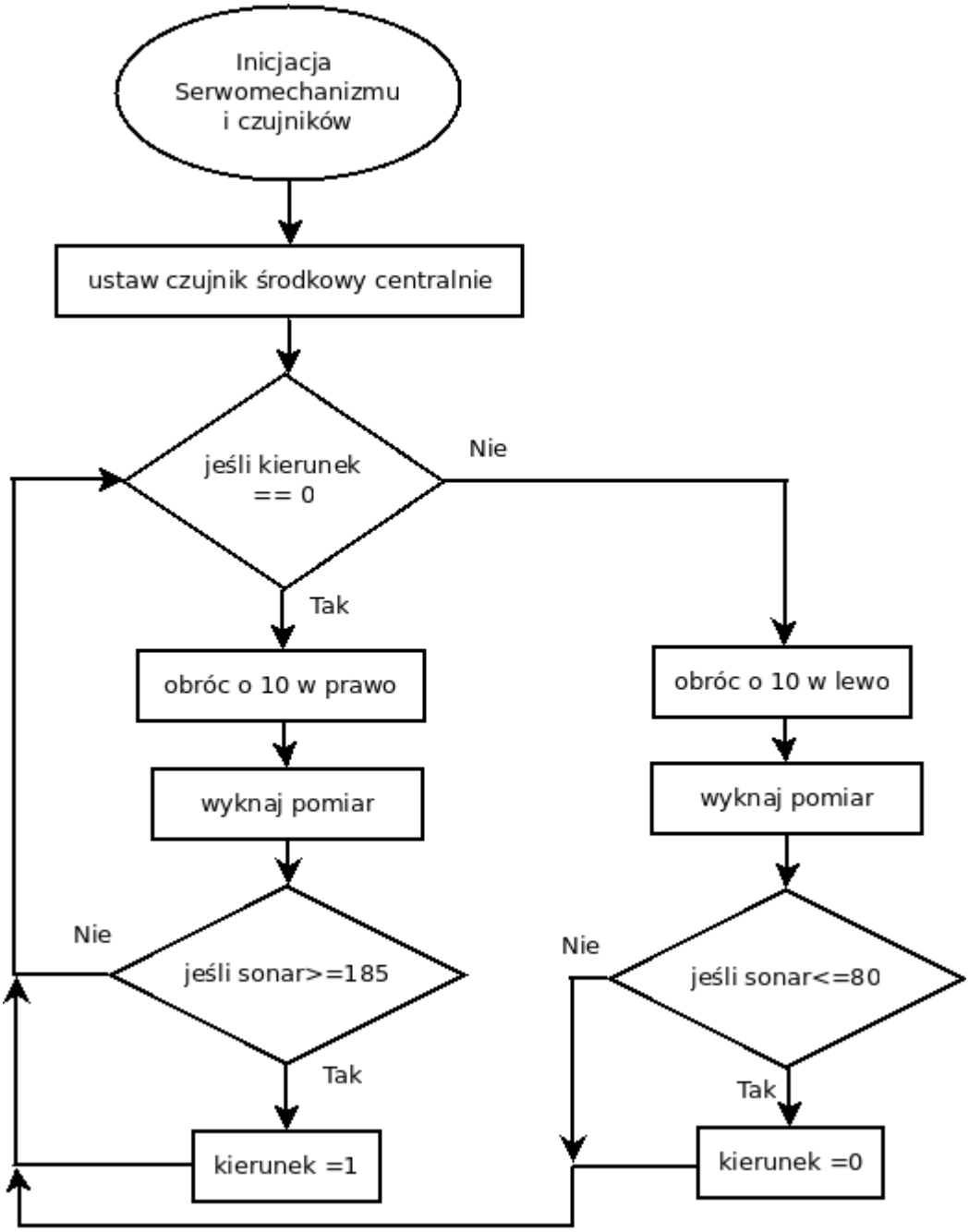
6.4.2. Wykonanie algorytmu sterowania oraz testowanie sonaru.

Algorytm sterowania radarem jest trywialny. Schemat blokowy znajduje się poniżej. Operacja pomiaru dokonywana jest co 20ms ze względu na szybkość działania czujników. Prędkość ta jest zadowalająca, gdyż daje nam to pomiar z częstotliwością 50Hz. Taka gęstość pomiaru pozwala na adaptację prostego algorytmu utrzymywania odpowiedniej odległości od ścian, który został opisany w projekcie MobilCAM 3. Również pozwala na rozwinięcie tego algorytmu oraz tworzenie mapy terenu. Jedynym problemem jaki aktualnie występuje to sam serwomechanizm. Ze względu, iż zastosowano serwomechanizm z plastikowymi zębatkami z projektu MobilCAM 3 występują pewne luzy, które to mogą wprowadzać zakłócenia przy większej częstotliwości serwomechanizmu niż 20Hz. Występują wtedy jeszcze nie stłumione drgania radaru.

Schemat blokowy pomiaru

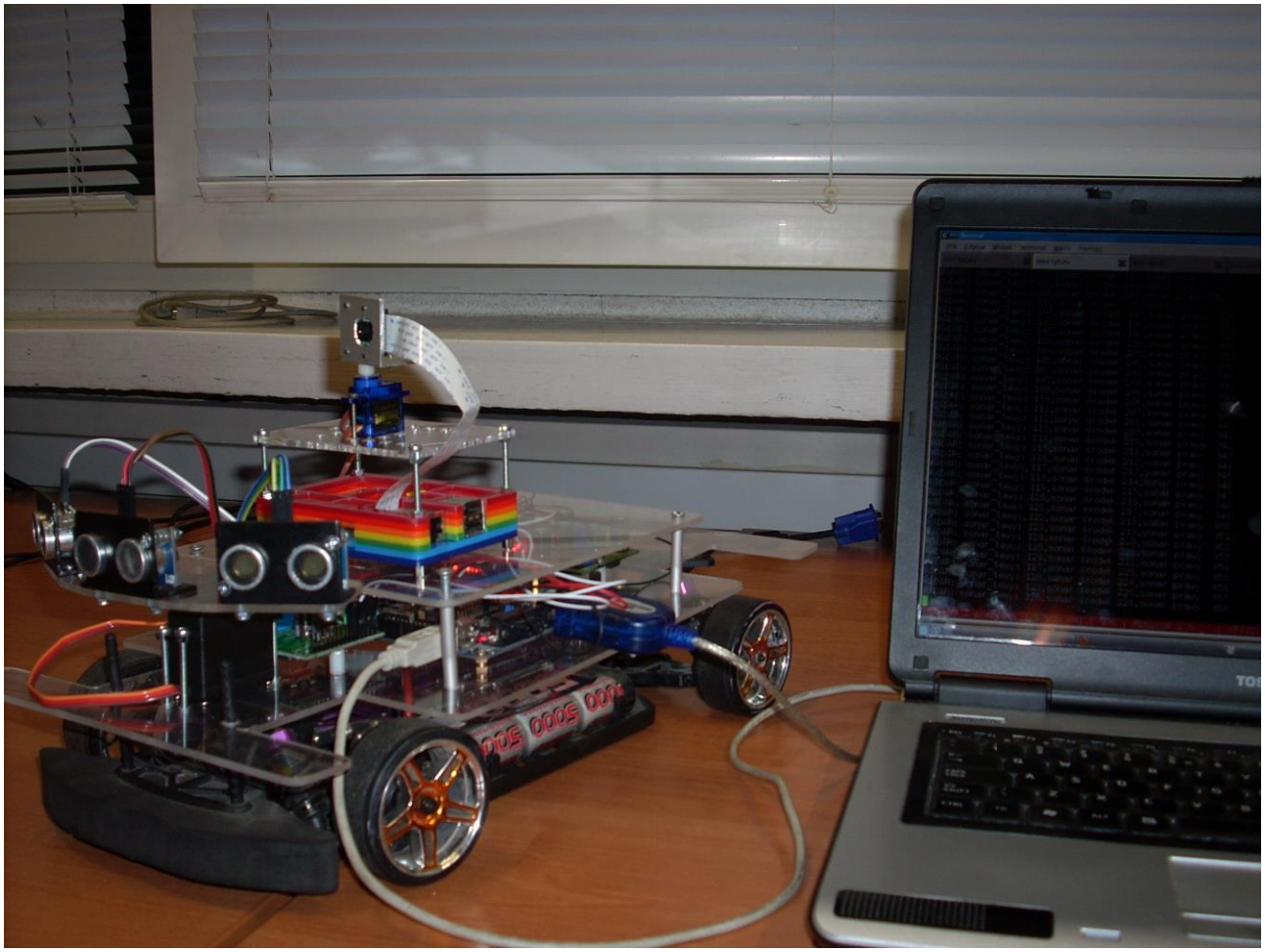


Schemat blokowy sterowania radarem



Testowanie radaru

Testy wykonano z wykorzystaniem przejściówki RS-232 –USB. Podłączono się do LPC w miejscu w którym występuje połączenie z Raspberry PI i wykonano odczyt danych odbieranych z radaru. Cała operacja przebiegała poprawnie.



7. Zakupy zrealizowany w ramach projektu.

7.1. Urządzenie do sterowania kamerą.

W czasie trwania projektu założono budowę ramienia robota w celu sterowania kamerą. Umożliwiło by to łatwy dostęp kamery w miejsca gdzie nie mógłby dojechać pojazd mobilny. Całość miała być sterowana za pomocą mikrokontrolera K20.

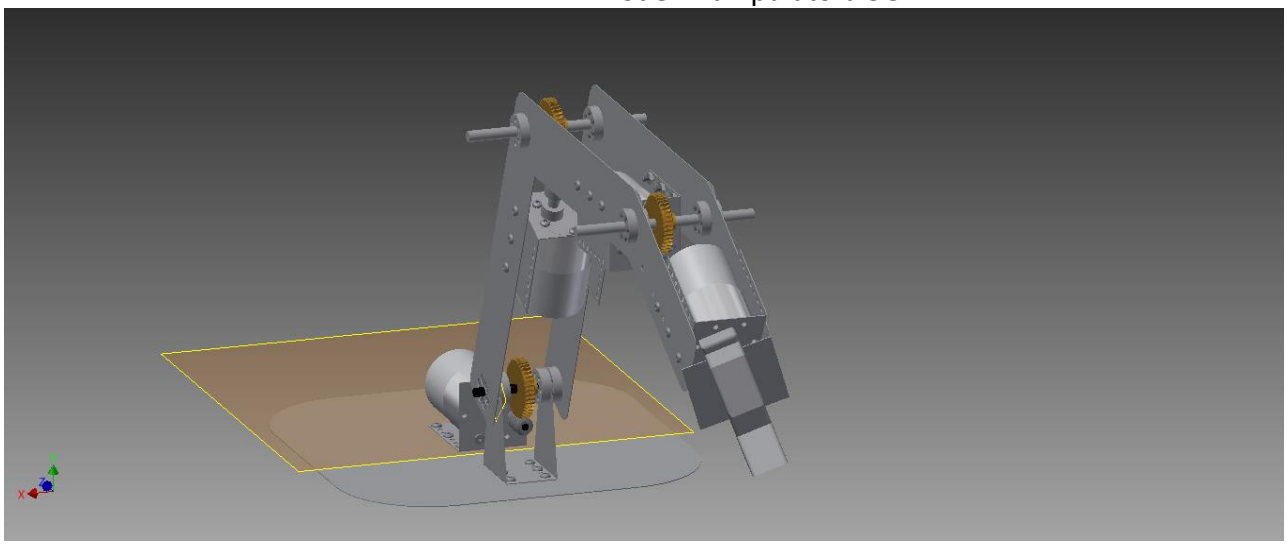
Manipulator ze względu na założenia składa się z 5 łączeń obrotowych. Umożliwia to pełną gammę ruchów. Ze względów na czas realizacja projektu jest na etapie wykonywania elementów ramienia i nie została dokończona, ale będzie kontynuowana.

Na manipulator składają się takie elementy:

- Silniki szczotkowe z przekładniami.
- Silniki bezszczotkowe
- Mostki H oraz sterowniki
- Sprzęgła
- Elementy mocujące
- Pręty fi 4mm oraz 5mm w celu montażu ślimacznicy oraz ślimaka
- Zespół zębatek przekładni ślimakowej
- Chwytak ramienia robota wraz z serwomechanizmem

Zdjęcie całego kompletu

Model Manipulatora 5O



7.2. Platforma Mobilna

Aktualna platforma mobilna składa się z zakupionych w ramach realizacji Projektu elementów. W skład tych elementów wchodzą:

- Raspberry Pi
- Czujnik przepływu prądu: ACS756SCA-100B
- Wifi: PmodWifi
- Sonar: Pololu 1605
- Przewody połączeniowe
- Kamera cyfrowa
- Mikrofony smd: ADMP441ACEZ

8. Literatura:

- „Systemy wizyjne robotów przemysłowych' Ryszard Tadeusiewicz WNT Warszawa 1992, Wydanie I ISBN 83-204-1531-4
- Praca magisterska pt. 'Wstep do komputerowej analizy obrazów' Andrzej Materka, Paweł Strumiłło, Politechnika Łódzka, Łódz 2009
- “Digital Image Processing: PIKS Inside, Third Edition' Wilian K. Pratt, Wydawnictwo: John Wiley & Sons, INC. ISBN 0-471-37407-5
- „Cyfrowe przetwarzanie sygnałów” Tomasz P. Zieliński, WNT Warszawa 2005, Wydanie 1 ISBN 83-206-1596-8
- „Modelowanie i sterowanie mobilnych robotów kołowych” Mariusz J. Giergiel, Zenon Hendzel, Wiesław Żylski, WNT Naukowe PWN Warszawa 2013, Wydanie 1 ISBN 978-83-01-13789-2
- „Podstawy Robotyki” Tadeusz Szkodny, WNT Politechniki Śląskiej Gliwice 2011, ISBN 978-83-7335-769-3

9. Podsumowanie projektu.

- Zrealizowano cel projektu w 60%. Udało się zrealizować budowę sonaru wraz z oprogramowaniem, obsługę Raspberry PI wraz z transportem obrazu na jednostkę docelową. Nie udało się obsłużyć mikrofonów. Wszelkie braki zostaną zrealizowane w przyszłym semestrze.
- Dzięki badaniom czujników w zeszłym semestrze udało się zakupić czujniki oraz stworzyć bardzo dobrą koncepcję radaru, która pozwoli realizować założenie badania otoczenia przed platformą z dużą częstotliwością.
- Realizacja poszczególnych etapów przebiegała bez opóźnień poza budową radaru oraz budową systemu elektronicznego. Ze względu na brak sprzętu etap ten został przesunięty.
- Wyniki uzyskane w projekcie pozwoliły udoskonalić skuteczność prac w kolejnych projektach Koła Naukowego.
- Praca w projekcie nauczyła jego uczestników umiejętności pracy zespołowej oraz udoskonaliła wiedzę z zakresu budowy pojazdów mobilnych, sposobu sterowania, tworzenia algorytmów do autonomicznej jazdy platformy, działania czujników oraz innych urządzeń pomiarowych.
- Wszyscy członkowie projektu sumiennie i rzetelnie wykonali swoje zadania, dzięki temu wyniki projektu są wiarygodne i można się na nich opierać w dalszych projektach.