

AGH

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
Wydział Fizyki i Informatyki Stosowanej

Praca magisterska

Łukasz Hanusiak, Mariusz Nowacki

kierunek studiów: **informatyka stosowana**

specjalność: **informatyka w nauce i technice**

Rozwój platformy robota mobilnego

Opiekun: **dr hab. inż. Marek Idzik**

Kraków, kwiecień 2011

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczanie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

.....
(czytelny podpis)

Kraków, ?? kwiecień 2011

**Tematyka pracy magisterskiej i praktyki dyplomowej Łukasza Hanusiaka
oraz Mariusza Nowackiego studentów V roku studiów kierunku informatyka
stosowana, informatyka w nauce i technice**

Temat pracy magisterskiej: **Rozwój platformy robota mobilnego**

Opiekun pracy: dr hab. inż. Marek Idzik

Recenzenci pracy: ??? ???

Miejsce praktyki dyplomowej: WFiIS AGH, Kraków

Program pracy magisterskiej i praktyki dyplomowej

1. Omówienie realizacji pracy magisterskiej z opiekunem.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Praktyka dyplomowa:
 - zapoznanie się z ideą...,
 - uczestnictwo w eksperymentach/przygotowanie oprogramowania...,
 - dyskusja i analiza wyników...
 - sporządzenie sprawozdania z praktyki.
4. Kontynuacja obliczeń związanych z tematem pracy magisterskiej.
5. Zebranie i opracowanie wyników obliczeń.
6. Analiza wyników obliczeń numerycznych, ich omówienie i zatwierdzenie przez opiekuna.
7. Opracowanie redakcyjne pracy.

Termin oddania w dziekanacie: ?? kwiecień 2011

.....
(podpis kierownika katedry)

.....
(podpis opiekuna)

Na kolejnych dwóch stronach proszę dołączyć kolejno recenzje pracy popełnione przez Opiekuna oraz Recenzenta (wydrukowane z systemu MISIO i podpisane przez odpowiednio Opiekuna i Recenzenta pracy). Papierową wersję pracy (zawierającą podpisane recenzje) proszę złożyć w dziekanacie celem rejestracji co najmniej na tydzień przed planowaną obroną.

Spis treści

Wprowadzenie	8
Rozdział 1. Historia i rozwój robotyki	9
1.1. Obecny rozwój i zakres zastosowań robotyki	15
1.2. Klasyfikacja robotów	16
Rozdział 2. Analiza bazowej konfiguracji robota	18
2.1. Analiza sprzętu	19
2.1.1. Elementy elektroniczne	19
2.1.2. Elementy mechaniczne	21
2.2. Analiza oprogramowania	23
2.2.1. Firmware	23
2.2.2. Aplikacja zarządzająca	24
Rozdział 3. Rozwój robota – środowisko rozwojowe systemu wbudowanego	26
3.1. Narzędzia dla systemu Windows	27
3.1.1. Instalacja WinARM	27
3.1.2. Instalacja sterowników programatora	28
3.1.3. Instalacja Open On-Chip Debugger	29
3.1.4. Konfiguracja zintegrowanego środowiska programistycznego	29
3.2. Narzędzia dla systemu Linux	30
3.2.1. Wybór zintegrowanego środowiska programistycznego	30
3.2.2. Instalacja i konfiguracja Eclipse'a	30
3.2.3. Instalacja i konfiguracja toolchain'a	33
3.2.4. Open On-Chip Debugger – instalacja i konfiguracja	33
Rozdział 4. Rozwój robota – środowiska dla PC i urządzeń mobilnych	37
4.1. Biblioteka zarządzająca dla języka Java	38
4.1.1. Bluetooth a Java SE	38
4.1.2. Opis możliwości biblioteki	38
4.2. Aplikacja zarządzająca na komputery stacjonarne (Java)	39
4.3. Biblioteka programistyczna dla platformy .NET	40

<i>Spis treści</i>	6
4.4. Platforma mobilna (Windows Mobile 6.1)	42
4.4.1. Środowisko rozwojowe	43
4.4.2. Środowisko uruchomieniowe	49
4.5. Platforma mobilna (Java ME)	50
4.5.1. Narzędzia programistyczne	50
4.5.2. Aplikacja mobilna	50
Rozdział 5. Rozwój robota – warstwa sprzętowa	51
5.1. Inercjalny system nawigacyjny	52
5.1.1. Wprowadzenie do INS	52
5.1.2. Elementy IMU robota	53
5.2. Akcelerometr trzyosiowy	55
5.2.1. Rodzaje akcelerometrów	57
5.2.2. Algorytm rozpoznawania kroków	58
5.2.3. Implementacja algorytmu	59
5.3. Żyroskop	62
5.3.1. Zasada działania	62
5.3.2. Kalibracja i odczytywanie wyników	64
5.3.3. Opis zastosowanego elementu i budowy modułu	66
5.4. Magnetometr	69
5.4.1. Zasada działania	70
5.4.2. Opis elementu	70
5.4.3. Budowa modułu	71
5.4.4. Niedana próba zastosowania	72
5.5. Czujnik odległości	74
5.5.1. Algorytm omijania przeszkód	75
5.5.2. Implementacja	78
5.6. Wyświetlacz LCD	81
5.7. Płyta rozszerzeń	83
5.8. Rozbudowa obudowy robota	85
Rozdział 6. Rozwój robota – firmware	87
6.1. Protokół komunikacji bluetooth	88
6.2. Algorytm rekonstrukcji ścieżki powrotnej	92
6.3. Modernizacja sposobu pobierania obrazu z kamery	93
6.4. Lokalizacja twarzy na obrazie	94
6.4.1. Algorytmy	94
6.4.2. Implementacja	96
Podsumowanie	99
Dodatek A. Specyfikacja poleceń protokołu komunikacji	100

A.1. Komunikaty sterujące	101
A.1.1. Sterowanie silnikami	101
A.1.2. Sterowanie serwomechanizmem	101
A.1.3. Akwizycja obrazu z kamery	101
A.1.4. Sterowanie czujnikami	102
A.2. Komunikaty specjalne	103
Dodatek B. Wytrawianie płytEK układów elektronicznych	104
Dodatek C. Plik konfiguracyjny programatora: triton.cfg	107
Kod źródłowy	108
Spis rysunków	109
Spis tabel	112
Bibliografia	113

Wprowadzenie

Rozdział 1

Historia i rozwój robotyki

Robotyka jest stosunkowo młodą dziedziną łączącą różne gałęzie nauk technicznych i nie tylko. Do pełnego zrozumienia zagadnień współczesnej robotyki oraz budowy i zastosowań robotów konieczna jest niejednokrotnie rozległa wiedza na temat elektroniki, mechaniki, inżynierii przemysłowej, matematyki oraz szeroko pojętej informatyki. Ponadto wiele nowopowstałych gałęzi wiedzy zajmujących się rozwojem sztucznej inteligencji, modelowaniem sztucznego życia czy rozwojem inżynierii wiedzy coraz częściej staje się nierozerwalnie związane z problemami współczesnej robotyki. Nie należy zapominać również o tym, że rozwój inżynierii wytwarzania oraz automatyki przemysłowej pozwala na nieustanny rozwój robotyki z którą mamy do czynienia w przemyśle w dniu dzisiejszym.

Pojęcie „robot” w literaturze pojawiło się po raz pierwszy w sztuce czeskiego pisarza Karel'a Capka w roku 1920. Termin „robot” oznacza w języku czeskim pracę lub służbę przymusową. Nieco ponad 20 lat później, amerykański uczony i pisarz Isaac Assimov w jednym ze swoich opowiadań po raz pierwszy używa słowa robotyka. W kolejnych latach Assimov w swojej twórczości niejednokrotnie wraca do problemu robotyki skutkiem czego jest wydanie w 1950 roku zbioru opowiadań pod tytułem „Ja, robot”. Jako ciekawostkę można dodać, że w wydanym w 1942 roku opowiadaniu pod tytułem „Zabawa w berka”, Assimov wprowadza trzy prawa robotyki według których, zdaniem autora, powinny być programowane roboty [5]. Zdefiniowane przez Assimov'a prawa robotyki w pełnym brzmieniu widoczne są na diagramie 1.1.

Nieco później, Isaac Assimov, jako uzupełnienie i prawo nadzędne dodaje prawo zerowe o następującym brzmieniu „Robot nie może szkodzić ludzkości, ani przez zaniedbanie narazić ludzkości na szkodę” [7]. Oprócz wspomnianych powyżej podstawowych praw, zdefiniowano również inne prawa wynikające z prowadzonych w tej dziedzinie badań i rozwoju robotyki.



Rysunek 1.1. Prawa robotyki zdefiniowane przez Issaca Assimov'a

Takim sposobem urządzenie mechaniczne wykonujące zadania w sposób automatyczny otrzymało miano „robota”. [3] Operacje wykonywane przez robota mogą być bezpośrednio kontrolowane przez człowieka na podstawie przygotowanego wcześniej programu zawierającego zestaw reguł które umożliwiają robotowi wykonywanie określonych czynności. Możliwość wyręczenia człowieka przez maszynę w wykonywaniu monotonnych, złożonych i powtarzalnych czynności, niejednokrotnie z dużo większą wydajnością i precyzją, była jednym z podstawowych bodźców który sprzyjał rozwojowi robotyki już od samego jej początku. Pojęcie robot, używane jest również w stosunku do autonomicznie działających urządzeń pobierających informację z otoczenia za pomocą różnego rodzaju czujników, nazywanych sensorami, oraz oddziałujących na swoje otoczenie i reagujące na jego zmiany.

Dzięki gwałtownemu rozwojowi nauki i techniki w czasie II wojny światowej w roku 1956 GC. Devol i JF. Engelberger zainspirowani twórczością Assimov'a zaprojektowali i stworzyli pierwszy w dziejach ludzkości działający egzemplarz robota [7]. Engelberger założył firmę pod nazwą „Unimation” zajmującą się automatyzacją, pierwszym robotem stworzonym przez firmę Engelbergera był robot nazwany „Unimate”. Został on zainstalowany w fabryce General Motors w Trenton gdzie został przystosowany do obsługi wysokociśnieniowej maszyny odlewniczej. W kolejnych latach roboty firmy „Unimation” znalazły swoje zastosowanie w innych gałęziach przemysłu, a sam Engelberger otrzymał miano ojca robotyki. [7]

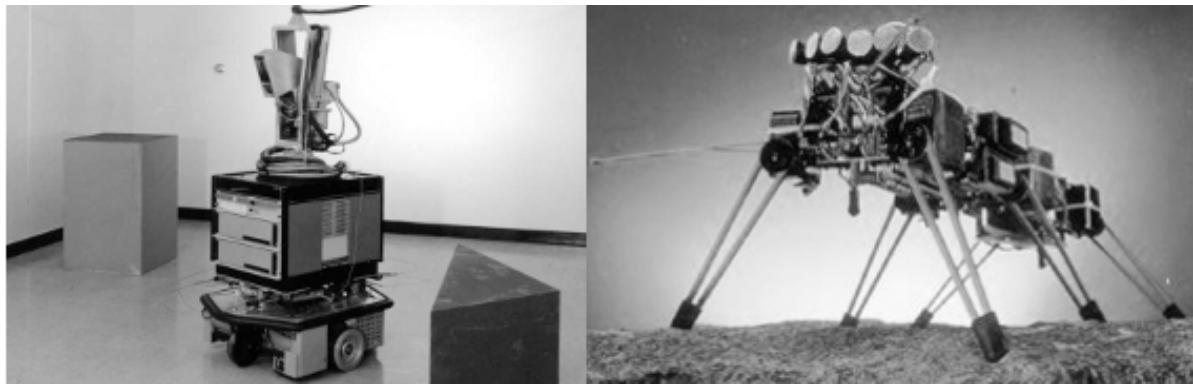
Nieco później, bo w roku 1979 Robotics Industries Association zdefiniowało robota jako wielofunkcyjny, programowalny manipulator zaprojektowany do przenoszenia materiałów, narzędzi, części urządzeń poprzez serię programowalnych ruchów wykonywanych w celu realizacji różnych zadań. W myśl wspomnianej definicji jedną z podstawowych cech

robot jest jego programowalność i możliwość dostosowywania się do zmiennych warunków środowiska pracy.

Pierwsze roboty projektowano z myślą o zastosowaniu ich do realizacji elementarnych zadań związanych z przenoszeniem elementów z jednego punktu do drugiego. Program pracy robota miał więc charakter sekwencji operacji prowadzących do realizacji określonego przez programistę zadania. W miarę rozwoju technicznego, stawiane przed robotami zadania wymusiły użycie przez konstruktorów czujników które pozwalały na zwiększenie poziomu interakcji robotów z otoczeniem, a co za tym idzie umożliwiały realizowanie zadań o wysokim stopniu złożoności. Od tego momentu kierunek rozwoju robotyki obrał sobie za cel stworzenie maszyny na tyle uniwersalnej i niezależnej aby mogła nosić miano androida. Jednak do realizacji tego zadania konieczne jest opracowanie sztucznej inteligencji na którą z pewnością przyjdzie nam jeszcze trochę poczekać.

Pierwszym krokiem na drodze do stworzenia androida było oderwanie robota od stałego miejsca instalacji i pozwolenie mu w miarę możliwości swobodne poruszanie się w dość dużej mu przestrzeni roboczej. Tak powstała klasa robotów które mogą przemieszczać się za pomocą kół, gąsienic czy nawet kończyn lub odnóży. Roboty tego typu potrafią pływać, latać i sprawnie poruszać się po lądzie dodatkowym ich atutem jest fakt iż większość z nich posiada niemal całkowitą autonomię i ograniczona jest jedynie poprzez wielkość otoczenia w jakim zostały umieszczone. Takim sposobem powstała grupa robotów nazywanych robotami mobilnymi. Cechą wspólną wszystkich urządzeń z tej rodziny była umiejętność swobodnego przemieszczania się oraz analiza najbliższego otoczenia. Na tej podstawie maszyna mogła przeprowadzić wnioskowanie pozwalające na podejmowanie dalszych akcji czy też przesłanie użytkownikowi odczytanych parametrów środowiska.

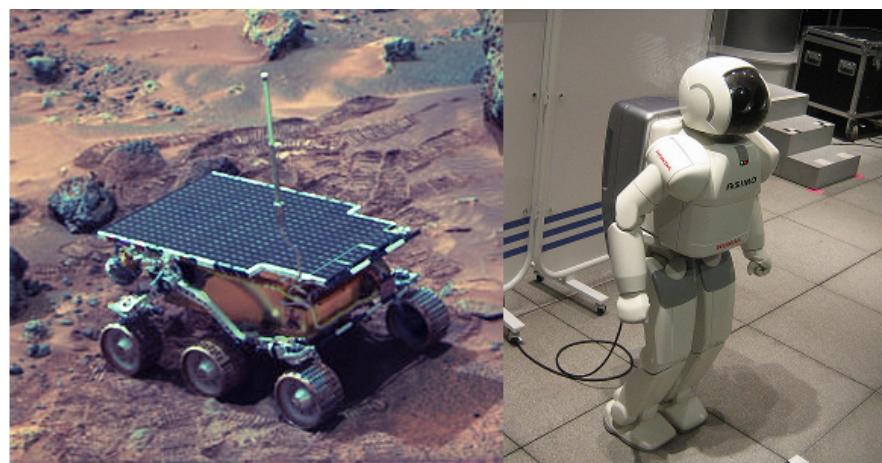
Historię robotów mobilnych zapoczątkował amerykański Uniwersytet w Stanford gdyż w roku 1968 jako pierwszy stworzył w pełni działający model robota mobilnego pod nazwą Shakey. Nazwa została zainspirowana szarpanymi ruchami z jakimi robot się poruszał. Głównym zadaniem robota było modelowanie otoczenia w którym się znajdował. W ślad za Uniwersytetem w Stanford ruszyło MIT. W roku 1983 posiadali już pierwszy działający model robota swobodnie skaczącego. Niecałe 6 lat później MIT stworzyło pierwszego robota kroczącego wzorowanego na owadach. Robot ten sterowany był za pomocą wielowarstwowych automatów o stanach skończonych, a ze światem zewnętrznym komunikował się za pomocą czułek, inklinometrów, czujników zbliżeniowych na podczerwień. Genghis, bo tak nazwany został robot, posiadał na pokładzie 4 ośmioribowe jednostki obliczeniowe, ważył niecały kilogram i miał 35 cm długości.



Rysunek 1.2. Od lewej: Shakey (Stanford), Genghis (MIT)

Po sukcesie wspomnianych projektów, rozwój robotów mobilnych następował już bardzo dynamiczne. W latach 90 powstawało wiele różnych modeli robotów o bardzo różnorodnych rodzajach napędów oraz zestawach czujników umożliwiających interakcje ze światem zewnętrznym.

Swoistym ukoronowaniem prac było w 1997 roku stworzenie przez NASA robota o nazwie Pathfinder. Robot wyposażony był w czujniki laserowe, stereowizję, żyroskopy i inne rodzaje czujników o charakterze badawczym. Zasilany był on bateriami słonecznymi które pozwoliły mu na 83 dni nieprzerwanej pracy podczas której robot przebyło około 100 metrów i wykonał 230 manewrów. W ostatnich latach do największych osiągnięć robotyki mobilnej można z pewnością zaliczyć powstanie robotów humanoidalnych takich jak japoński ASIMO.



Rysunek 1.3. Kolejno: Pathfinder (NASA), Asimo (Honda)

Robot ten ważył 54 kg i posiadał 130 cm wysokości. Wersja z roku 2005 potrafiła biegając osiągnąć prędkość dochodzącą nawet do 6 km/h. Ponad to robot potrafił wchodzić w interakcję z otaczającymi go ludźmi i przedmiotami. Urządzenie stworzone przez inżynierów z firmy Honda potrafiło rozpoznawać gesty takie jak podanie ręki, wskazanie

kierunku czy machanie ręką na pożegnanie. Robot równie dobrze radził sobie z rozpoznaniem twarzy, dźwięków i analizą otaczającego go środowiska. Potrafił on rozpoznać i omijać niebezpieczeństwwa postawione na jego drodze jak na przykład schody czy osoby poruszające się w jego kierunku.

W roku 2006 podczas Robot World w Seulu grupa południowokoreańskich inżynierów prezentuje swojego robota EveR-2. EveR są robotami posiadającymi wygląd typowej dwudziestoletniej koreanki. Urządzenie potrafi rozmawiać i śpiewać dzięki wbudowanemu „silnikowi dialogu” (ang. embedded dialogue engine). Android EveR-2 w odróżnieniu od swojej poprzedniczki ma udoskonalony system wizyjny oraz możliwość wyrażania emocji takich jak znudzenie, zadowolenie, żal czy radość. Robot ma 170cm wzrostu i waży około 60kg. Twarz androidy posiada wysoką elastyczność i poruszana jest przy pomocy 29 silniczków i licznych stawów które pozwalają na pełną swobodę w wyrażaniu emocji. Wbudowany system rozpoznawania i syntezy mowy w połączeniu z możliwością wyrażania się za pomocą gestów pozwala na niemal całkowitą swobodę podczas rozmowy z robotem. Możliwości komunikacyjne robota są tak znakomite, że EveR-2 została pierwszą piosenkarką androidem. Jej pierwszy występ odbył się podczas jej prezentacji w Seulu gdzie na oczach publiczności odśpiewała koreańską balladę pt. „I will close my eyes for you”.

Lata 2008 i 2009 zaowocowały powstaniem wielu robotów naśladowujących w swoim zachowaniu niektóre zwierzęta. Bardzo imponującym przykładem takiego robota jest wyprodukowany przez firmę AeroVironment latający robot o nazwie Mercury. Urządzenie to potrafi „zawisnąć” w powietrzu poruszając jedynie skrzydłami dokładnie w taki sam sposób jak robią to prawdziwe kolibry. Lata te były również obfite w sukcesy w dziedzinie rozwoju sztucznego mózgu i sztucznej inteligencji. W roku 2008 grupa naukowców z University of Reading zbudowała robota sterowanego w całości za pomocą biologicznego mózgu z wyhodowanych neuronów. Biologiczny mózg w który wyposażony został zainstalowany w macierzy wieloelektrodowej (MEA). MEA jest to pewnego rodzaju naczynie które za pomocą około 60 elektrod przechwytuje sygnały elektryczne generowane przez komórki nerwowe i przekłada je na ruchy robota. W przypadku gdy robot napotka na swojej drodze przeszkody wspomniane wcześniej elektrody stymulują komórki nerwowe które tą samą drogą udzielają instrukcji na zachowania kół które pozwoliłyby na ominięcie wykrytej przeszkody. Robot jest całkowicie samodzielny i w całości sterowany przez własny mózg.

Obserwując postęp w dziedzinie robotyki można odnieść wrażenie iż w dzisiejszych czasach roboty znalazły dla siebie zastosowanie niemal w każdej dziedzinie życia. Od wielu lat sprawdzają się już w przemyśle, transporcie, budownictwie oraz są

niezastąpione w środowiskach nieprzyjaznych człowiekowi, takich jak podmorskie głębiny czy otchłań kosmosu. Obszar zastosowań robotów jest tak szeroki iż wydawać się może, że jedynym czynnikiem ograniczającym rozwój współczesnej robotyki są względy czysto technologiczne. Nie staje to jednak na przeszkodzie do projektowania i tworzenia przez konstruktorów z całego świata rozwiązań coraz bardziej przybliżających ludzkość do stworzenia w pełni samodzielnego oraz inteligentnego androida.

1.1. Obecny rozwój i zakres zastosowań robotyki

Robotyka zawładnęła wieloma dziedzinami życia człowieka od przemysłu poprzez zastosowania medyczne, wojskowe aż po urządzenie stosowane w gospodarstwach domowych. Roboty znalazły dla siebie zastosowanie w wykonywaniu zadań wymagających dużej szybkości, dokładności i wytrzymałości której nie może zapewnić praca wykonywana przez ludzi. W rezultacie wiele zadań wykonywanych w produkcyjnych zakładach pracy, dawniej wykonywane przez ludzi, zostało zastąpione przez roboty. Efektem tego jest zmniejszenie kosztów produkcji towarów masowych, szczególnie widocznych w przypadku części samochodowych i elektroniki. Zastosowanie robotów powoduje więc wzrost efektywności ekonomicznej i skraca czas uruchomienia produkcji, a co za tym idzie jest głównym czynnikiem ekonomicznym wspomagającym rozwój robotyki.

Istnieje szereg zadań które człowiek wykonuje lepiej niż maszyna, ale ze względu na męczący charakter pracy lub niebezpieczne środowisko jej wykonywania praca ludzka jest zastępowana przez maszyny. Stały rozwój technologii stosowanych w robotyce skutkuje w powstawaniu coraz bardziej zaawansowanych systemów co sprzyja powszechniejszemu ich stosowaniu. Zastosowanie robotów w życiu codziennym prowadzi do zwiększenia bezpieczeństwa pracy szczególnie na stanowiskach pracy zagrażających zdrowiu i życiu człowieka. Co więcej stały rozwój robotyki pozwala na prowadzenie badań w lokalizacjach fizycznie niedostępnych dla człowieka. Bardzo dobrym tego przykładem jest eksploracja odległych planet czy też wnętrz wulkanów. Jak można więc zauważyć rozwój robotyki stał się w chwili obecnej samonapędzającą się maszyną, w której powstawanie coraz doskonalszych robotów zwiększa na nie zapotrzebowanie, a to z kolei wymusza dalszy ich rozwój.

1.2. Klasyfikacja robotów

Współczesna różnorodność robotów doprowadziła do powstania wielu podziałów robotów ze względu na szereg różnych parametrów. Jednym z bardzo często spotykanych podziałów jest klasyfikacja ze względu na sposób programowania i możliwości komunikacyjne. Każda z grup tworzy tzw. generacje robotów. Można wyróżnić trzy generacje robotów uwzględniając różnice w ich układzie sterowania oraz dostępne sensory [7].

Robot I generacji

- Urządzenia zaprogramowane do wykonywania określonej sekwencji czynności z możliwością ich przeprogramowania. W robotach tej generacji stosuje się otwarty układ sterowania który charakteryzuje się całkowitym brakiem możliwości pobierania informacji ze świata zewnętrznego. Do robotów pierwszej generacji zaliczyć można roboty przemysłowe przeznaczone do podawania i odbierania obiektów z linii produkcyjnej.

Robot II generacji

- Maszyny wyposażone w zamknięty układ sterowania. Oznacza to, że posiadają one zestaw czujników umożliwiających dokonywanie pomiarów stanu robota oraz jego otoczenia. Pozwala to robotowi na rozpoznanie żądanego obiektu nawet wówczas, gdy przemieszcza się wśród innych obiektów. Robot tej generacji powinien być w stanie podjąć decyzję na temat sposobu realizacji zadania na podstawie aktualnego stanu otoczenia i obiektu manipulacji.

Robot III generacji

- Urządzenia również wyposażone w zamknięty układ sterowania oraz zestaw czujników umożliwiających dokonywanie skomplikowanych pomiarów i klasyfikację obiektów o wysokim stopniu złożoności. System sterowania robota tej generacji powinna pozwalać na jego adaptację w nieznanym otoczeniu.

Rysunek 1.4. Klasyfikacja robotów ze względu na ich generację

Istnieje szereg różnych klasyfikacji robotów bazujących na podziale ze względu na ich parametry techniczne. Jednym z bardziej znanych podziałów jest przedstawiona poniżej klasyfikacja ze względu na rodzaj środowiska w jakim się poruszają i sposób ich poruszania się.

- roboty podwodne i poruszające się na wodzie,
- roboty lądowe oraz wodno-lądowe
- roboty powietrzne,

Wśród robotów lądowych bardzo popularną grupą są roboty mobilne, podział których można przeprowadzić na podstawie sposobie poruszania się. Za pomocą takiego kryterium możemy wyróżnić roboty kołowe, kroczące, skaczące oraz pełzające. Innym popularnym sposobem klasyfikacji robotów jest podział ze względu na obszar ich zastosowania. Stosując kryterium takiego rodzaju istnieje możliwość wyróżnienia następujących grup:



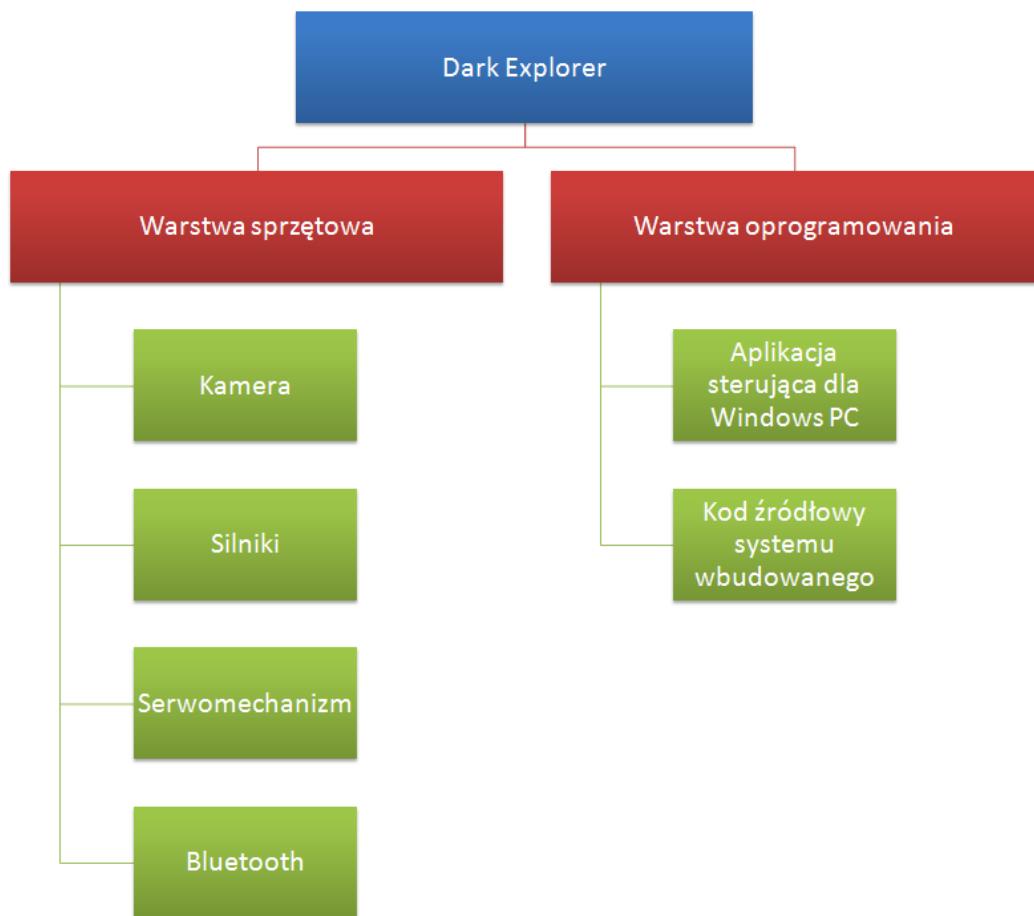
Rysunek 1.5. Podział robotów ze względu na obszar ich zastosowania

Omówione sposoby klasyfikacji nie wyczerpują w pełni problemu podziału robotów i ich zastosowań. Istnieje bowiem wiele charakterystyk opartych o takie parametry jak na przykład kształt czy rodzaj układu napędowego o których w tym rozdziale nie wspomniano. Celem przedstawionych informacji było zakreślenie obszaru zastosowań robotów oraz bogactwa ich różnorodności, a co za tym idzie uświadomienie czytelnikowi obszarów zastosowań robotyki we współczesnym świecie jak również potencjalnych problemów z jakim na codzień zmagają się ludzie projektujący i budujący roboty.

Rozdział 2

Analiza bazowej konfiguracji robota

W tym rozdziale zostanie opisana konfiguracja pierwotna robota z wyszczególnieniem elementów, które mogą być problemem przy dalszym rozwoju możliwości tego urządzenia.



Rysunek 2.1. Struktura platformy robota mobilnego zrealizowanej w ramach poprzedniej pracy magisterskiej [12]

2.1. Analiza sprzętu

Dark Explorer jest autonomicznym robotem mobilnym z wbudowaną kolorową kamerą cyfrową VGA. Jego podstawowe możliwości to:

- jazda ze zmieniąną prędkością w przód, tył, lewo oraz prawo
- komunikacja z urządzeniami zewnętrznymi przy pomocy technologii bluetooth
- wykonywanie zdjęć z maksymalną rozdzielcością 160x100 pikseli w kolorze oraz 320x200 pikseli w odcieniach szarości
- poruszanie wieżyczką na której zainstalowana jest kamera
- wykrywanie prostych wzorców przy pomocy analizy obrazu oraz podążanie za nimi

2.1.1. Elementy elektroniczne

Dark Explorer został wyposażony w mikrokontroler zarządzający AT91Sam7s256 o częstotliwości pracy zegara maksymalnie do 50 MHz oraz wbudowanej szybkiej pamięci SRAM 64 kB. Mikrokontroler ten jest bogaty w różnego rodzaju urządzenia peryferyjne takie jak na przykład: TWI¹, RTT², PDC³, AIC⁴, PWM⁵, ADC⁶. To tylko część z nich. Dzięki tak wielkiemu wyborowi urządzeń peryferyjnych mikrokontroler ten daje nam duże możliwości rozwoju konfiguracji. Częstotliwość pracy mikrokontrolera ARM7 jest wystarczająca do zadań przez niego wykonywanych.

Większość elementów elektronicznych robota jest umieszczonych na płycie głównej zaprojektowanej przez autora projektu robota. Jest ona bardzo dobrze przemyślana ponieważ pozwala na wykorzystywanie urządzeń peryferyjnych mikrokontrolera w praktycznie dowolny sposób. Większość wyjść oraz wejść mikrokontrolera nie jest połączona na stałe z konkretnymi podzespołami Dark Explorera lecz poprzez zworki, które pozwalają podłączanie i odłączanie poszczególnych urządzeń.

Twórca Dark Explorera wyposażył go kamerą cyfrową o maksymalnej rozdzielczości 640x480 pikseli. Jest to urządzenie PO6040 firmy Pixelplus. Można go konfigurować przy pomocy interfejsu I^2C . W konfiguracji pierwotnej robot potrafi odbierać zdjęcia o

¹ TWI – Two Wire Interface znany również jako I^2C , interfejs szeregowej komunikacji danych

² RTT – Real Time Timer, służy do odmierzania dłuższych odcinków czasu

³ PDC – Peripheral DMA Controller, kontroler DMA

⁴ AIC – Advanced Interrupt Controller, kontroler przerywań

⁵ PWM – Pulse Width Modulation Controller

⁶ ADC – Analog to Digital Converter, konwerter analogowo cyfrowy



Rysunek 2.2. Obraz wykonany przy pomocy kamery zamontowanej w robocie. 160 x 100 pikseli w kolorze. [12]

maksymalnej rozdzielczości 320×200 pikseli w odcieniach szarości(rys. 2.3) oraz 160×100 pikseli w kolorze (rys. 2.2). Tak niskie rozdzielczości nie są wystarczające do wykrywania i tym bardziej rozpoznawania twarzy na obrazie dlatego też konieczna jest poprawa tych parametrów.

W skład robota wchodzą także silniki napędowe, serwomechanizm, dioda mocy oraz moduł bluetooth. Wszystkie te elementy muszą być podłączone do mikrokontrolera przez konkretne wejścia/wyjścia. Z tego powodu do dyspozycji osób przeprowadzających dalszy rozwój robota pozostało jedynie pięć wejść/wyjść cyfrowych oraz trzy wejścia analogowe. Jest to kolejna przeszkoda którą trzeba będzie pokonać.



Rysunek 2.3. Obraz wykonany przy pomocy kamery zamontowanej w robocie. 320×200 pikseli w odcieniach szarości [12]

Robot mobilny komunikuje się z urządzeniami zewnętrznymi przy pomocy modułu bluetooth BTM-222 firmy Rayson. Maksymalna przepustowość danych z jaką potrafi działać wynosi 3Mb/s . Obsługiwane przez niego interfejsy to: USB, UART oraz PCM. Mikrokontroler komunikuje się z modułem bluetooth przy pomocy interfejsu UART o maksymalnej przepustowości $460,8\text{ kb/s}$. Jak widać takie rozwiązanie nie wykorzystuje

pełnych możliwości modułu BTM-222. Bardziej efektywne byłoby skorzystanie z interfejsu USB który potrafi przesyłać dane z prędkością maksymalną od 1,5Mb/s w standardzie USB 1.0 do 5Gb/s w standardzie USB 3.0. Aby dokonać zmiany interfejsu komunikacyjnego pomiędzy mikrokontrolerem a modułem bluetooth konieczna jest ingerencja w konstrukcję płyty głównej robota, gdyż BTM-222 jest przylutowany bezpośrednio do niej (rys. 2.4).



Rysunek 2.4. Zdjęcie modułu bluetooth zamontowanego na płytce głównej robota.

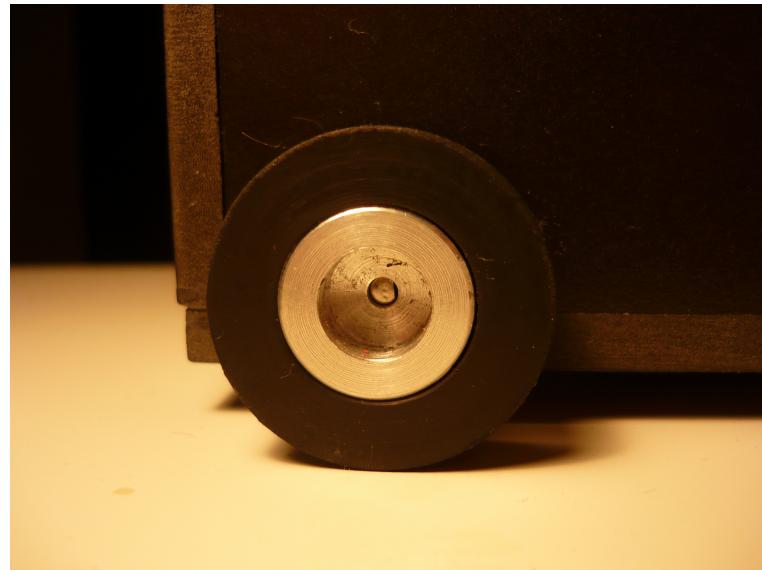
2.1.2. Elementy mechaniczne

Konstrukcja obudowy Dark Explorera została wykonana z tworzywa sztucznego i jest dopasowana do elementów które zostały zaprojektowane przez autora. Wewnątrz obudowy nie ma miejsca na jakiekolwiek nowe podzespoły, dlatego też konieczna będzie jej modyfikacja (rys. 2.5).



Rysunek 2.5. Widok na płytę główną Dark Explorer'a

Podczas testów konfiguracji pierwotnej zauważono lekkie kłopoty robota z poruszaniem się po gładkich powierzchniach. Najprawdopodobniej jest to spowodowane kółkami (rys. 2.6) zamontowanymi przy robocie, które tracą przyczepność na nieco bardziej śliskim podłożu. Możliwe, że podczas rozwoju robota, konieczna będzie ich wymiana w celu zapewnienia dobrej przyczepności i poprawnego toru jazdy urządzenia.



Rysunek 2.6. Koło napędowe Dark Explorer'a

2.2. Analiza oprogramowania

2.2.1. Firmware

Jako firmware określane jest oprogramowanie którego komendy są wykonywane przez mikrokontroler. W tym podrozdziale zajmiemy się kodem źródłowym opisującym sposób działania robota.

Pliki składające się na kod źródłowy robota możemy podzielić na dwa rodzaje:

- napisane przez autora
- dostarczone przez producenta mikrokontrolera

Pliki dostarczone przez producenta są po prostu biblioteką, dzięki której programista może kontrolować działanie różnych urządzeń peryferyjnych mikrokontrolera. Biblioteka wykorzystana przez twórcę Dark Explorera bazuje na prostych funkcjach wpisujących podawane wartości do odpowiednich rejestrów mikrokontrolera. Można powiedzieć, że jest to biblioteka nisko poziomowa. Wymaga ona dużej wiedzy na temat urządzeń peryferyjnych z których mamy zamiar korzystać oraz rejestrów które nimi kontrolują. Praktycznie nie jest możliwe oprogramowanie robota przy pomocy tej biblioteki bez wcześniejszego dokładnego zaznajomienia się z notą katalogową mikrokontrolera AT91Sam7S256. Podczas rozwoju robota przydatnym byłoby wykorzystanie innej biblioteki, która jest bardziej przyjazna dla programisty.

Bazowa wersja firmware'u napisanego przez autora Dark Explorera została podzielona na sześć plików:

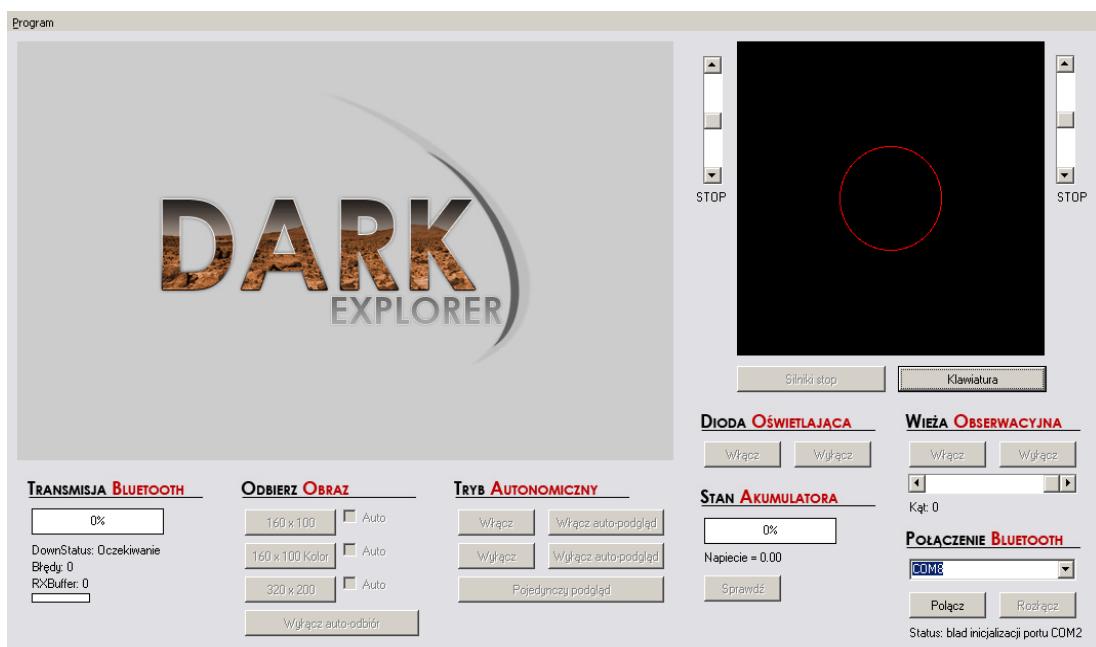
- board.h – plik nagłówkowy z informacjami dotyczącymi mikromodułu mikrokontrolera
- pio.h – definicje określające wejścia i wyjścia ogólnego przeznaczenia zamontowane na płycie głównej robota
- main.c – inicjalizacje początkowe, obsługa przerwań systemowych, interpretacja komend bluetooth, pętla główna programu
- peripherals.c – funkcje do obsługi urządzeń peryferyjnych
- rozpoznawanie.c – analiza i rozpoznawanie obrazu
- utils.c – procedury sterujące i obliczeniowe wyższego poziomu

Biorąc pod uwagę ilość kodu znajdującej się w plikach, taki podział wydaje się być wystarczający. W przyszłości trzeba zadbać o gęstsze partycjonowanie kodu źródłowego na logiczne bloki w celu zapewnienia przejrzystości kodu źródłowego.

2.2.2. Aplikacja zarządzająca

Do kontrolowania robota została stworzona aplikacja graficzna działająca pod systemem Windows. Interfejs graficzny aplikacji jest atrakcyjny i przejrzysty. Pozwala ona na kontrole następujących funkcji robota:

- kontrola kierunku i szybkości jazdy robota za pomocą wektora wodzącego oraz klawiatury
- kontrola wierzy obserwacyjnej
- włączanie oraz wyłączanie diody oświetlającej
- informacja o stanie akumulatorów
- zarządzanie trybem autonomicznym robota
- konfiguracja oraz status połączenia bluetooth



Rysunek 2.7. Okno aplikacji zarządzającej Dark Explorera

Aplikacja zarządzająca została napisana tylko na jeden rodzaj systemu operacyjnego. Nie ma możliwości kontrolowania robota przy pomocy komputera z zainstalowanym systemem operacyjnym innym niż Microsoft Windows. Dlatego też, konieczne będzie stworzenie aplikacji zarządzającej, którą będzie można modyfikować oraz uruchomić na dowolnym systemie.

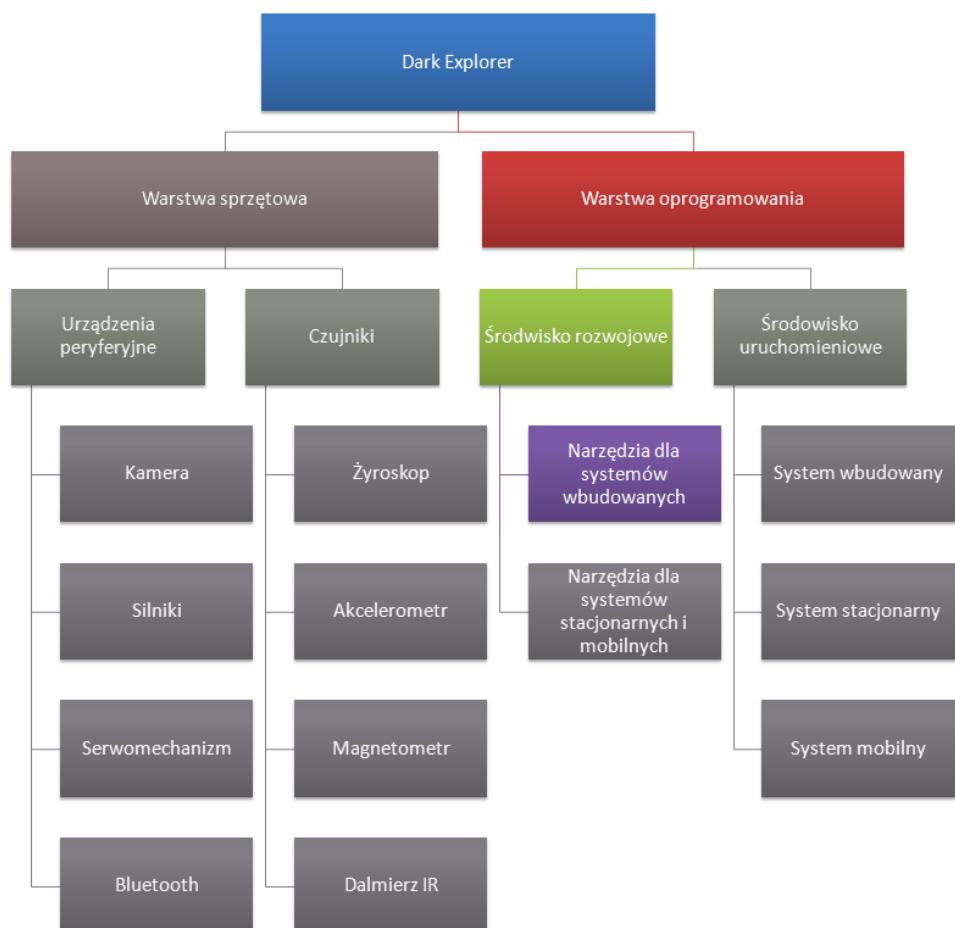
W celu komunikacji z robotem, aplikacja zarządzająca wysyła do robota odpowiednie komunikaty. Komunikaty te składają się z pojedynczego znaku i liczby. Możliwe, że w przypadku obsługi większej ilości urządzeń na Dark Explorerze, będzie konieczne stworzenie bardziej zaawansowanego protokołu komunikacyjnego. Zapewni to możliwość sprawdzenia czy dana komenda jest poprawnie skonstruowana, czy może zawiera jakieś błędy które pojawiły się podczas transmisji.

Zarówno w aplikacji zarządzającej jak i na robocie został zastosowany mechanizm retransmisji pakietów z obrazem w przypadku błędów w trakcie połączenia. Jest to bardzo dobry pomysł w szczególności w przypadku przesyłania takiej ilości danych jaką generuje kamera.

Rozdział 3

Rozwój robota – środowisko rozwojowe systemu wbudowanego

Podczas przygotowywania się do rozpoczęcia pracy z nową technologią, każdy programista powinien być świadomy tego jakie są istniejące narzędzia, które mogą mu pomóc w pracy. W tym rozdziale opisany jest sposób instalacji oraz używania narzędzi wykorzystywanych przez autorów tej pracy.



Rysunek 3.1. Struktura platformy robota mobilnego po zakończeniu prac. Kolorem oznaczono zakres prac opisanych w bierzącym rozdziale.

3.1. Narzędzia dla systemu Windows

Pierwszym krokiem do przygotowania środowiska rozwojowego umożliwiającego rozwijanie oprogramowania sterującego robotem jest instalacja sterowników wymaganych przez system Windows do obsługi interfejsu JTAG za pomocą którego odbywa się proces wgrywania przygotowanego oprogramowania do pamięci robota. Kolejnym wymaganym krokiem jest instalacja i konfiguracja narzędzi umożliwiających stworzenie pliku binarnego umożliwiającego uruchomienie przygotowanej na platformie sprzętowej robota. Ostatnim etapem przygotowań jest instalacja oprogramowania umożliwiającego programowanie układu za pomocą wspomnianego interfejsu JTAG oraz debugowanie aplikacji w trakcie jej działania na robocie.

3.1.1. Instalacja WinARM

Do komplikacji kodu źródłowego oprogramowania zajmującego się sterowaniem podzespołami robota wykorzystany został zestaw narzędzi znany pod nazwą WinARM. WinARM jest zestawem narzędzi umożliwiających tworzenie oprogramowania dla kontrolerów opartych na platformie ARM. W odróżnieniu od innych dostępnych obecnie rozwiązań, środowisko to, nie wymaga dodatkowej instalacji narzędzi udostępnianych w ramach MinGW czy też Cygwina. Wszystkie potrzebne narzędzia dostarczane są w ramach SDK¹. Narzędzia WinARM pomyślnie przeszły testy z kontrolerami Atmel AT91SAM7S64, AT91SAM7S256, AT91RM9200 ARM7TDMI oraz Philips LPC2106, Philips LPC2129, Philips LPC2138, Philips LPC2148. Dodatkowo dostarczane w ramach środowiska kompilatory i narzędzia powinny prawidłowo współpracować ze wszystkimi mikrokontrolerami opartymi o architekturę ARM(-TDMI/Thumb itp.).

Instalację środowiska WinARM należy rozpocząć od pobrania archiwum z najnowszą wersją narzędzi ze strony http://gandalf.arubi.uni-kl.de/avr_projects/arm_projects/. W chwili pisania pracy dostępna była wersja środowiska WinARM w wersji 20060606. Po zakończeniu procesu pobierania, archiwum należy rozpakować w taki sposób aby wszystkie podstawowe narzędzia dostępne były w katalogu C:\WinARM\bin. Umieszczenie katalogu z narzędziami WinARM w innej lokalizacji jest również możliwe, ale może wymagać wykonania dodatkowych operacji konfiguracyjnych w celu zapewnienia poprawności działania wszystkich narzędzi. Aby udostępnić narzędzia WinARM z linii poleceń

¹ SDK (z ang. Software Development Kit) - Zestaw narzędzi do rozwoju oprogramowania

systemu Windows konieczne jest dodanie do zmiennej systemowej PATH ścieżki do katalogów z plikami wykonywalnymi biblioteki. W przypadku instalacji w podanym powyżej katalogu wartości powinny być następujące C:\WinARM\bin;C:\WinARM\utils\bin;.

3.1.2. Instalacja sterowników programatora

Programowanie oraz debugowanie aplikacji robota może zostać zrealizowane za pomocą dowolnego programatora kompatybilnego z interfejsem JTAG. Programatory oparte o interfejs LPT nie wymagają od użytkownika żadnej dodatkowej konfiguracji. Nieco inaczej wygląda sytuacja z programatorami opartymi o interfejs USB, które to wymagają przed pierwszym użyciem zainstalowania sterowników umożliwiających prawidłowe rozpoznanie programatora przez system Windows. Jednym z bardziej popularnych programatorów USB jest TriTon JTAG. TriTon JTAG to programator przeznaczony dla procesorów zbudowanych w oparciu o rdzeń ARM podłączany do komputera za pomocą portu USB. TriTon JTAG posiada standardowe 20 pinowe złącze JTAG wraz z wyprowadzeniami sygnałów RxD i TxD interfejsu UART. TriTon JTAG współpracuje z OpenOCD, pozwalaając na programowanie oraz debugowanie działającej na urządzeniu aplikacji. Urządzenie oparte jest o układ FT2232 który umożliwia jego współpracę także z innymi środowiskami rozwoju oprogramowania dla platformy ARM kompatybilnymi z FT2232.

Instalację programatora należy rozpocząć od pobrania sterowników do układu FT2232 ze strony <http://www.ethernut.de/en/download/>. Na stronie dostępne są sterowniki przeznaczone dla systemów Windows 2000, XP, Server 2003, Vista oraz Server 2008. Po rozpakowaniu archiwum ze sterownikami należy za pomocą interfejsu USB podłączyć programator do komputera. Po wykryciu system Windows trzykrotnie poprosi o podanie ścieżki do sterowników do urządzeń Triton JTAG, Triton USB RS232 Adapter oraz USB Serial Port. Należy wtedy skazać ścieżkę do katalogu w którym rozpakowane zostały sterowniki pobrane ze strony wspomnianej wcześniej. Po poprawnym zakończeniu instalacji w Menadżerze urządzeń systemu Windows widoczne będą następujące elementy

- Triton USB JTAG Adapter,
- Triton USB RS232 Adapter,
- Triton JTAG

3.1.3. Instalacja Open On-Chip Debugger

OpenOCD zostało zapoczątkowane przez Dominika Rath w ramach pracy dyplomowej realizowanej na uniwersytecie w Augsburg. Od tamtego czasu OpenOCD bardzo się rozwinęło i urosło do rozmiarów aktywnego projektu open-sourcowego wspieranego przez programistów z całego świata. Celem OpenOCD jest dostarczenie uniwersalnego narzędzia umożliwiającego debugowanie i programowanie systemów wbudowanych.

Strona projektu OpenOCD dostępna jest pod adresem <http://openocd.berlios.de/web/>. Dostępne są tam zarówno źródła jak i dokumentacja do projektu. Niestety w chwili pisania pracy magisterskiej autorzy nie udostępniali wersji skompilowanej dla systemu Windows. Dlatego też użyta została niezależna wersja OpenOCD z przygotowanym instalatorem dla systemu Windows. Instalator OpenOCD dla Windows jest do pobrania ze strony <http://www.ethernut.de/en/download/>. Po uruchomieniu instalatora wybieramy lokalizację docelową w której OpenOCD ma zostać zainstalowane. Po zakończeniu instalacji konieczne jest uzupełnienie wartości zmiennej systemowej PATH ścieżką do miejsca instalacji OpenOCD, domyślnie C:\ethernut\nut\tools\win32.

3.1.4. Konfiguracja zintegrowanego środowiska programistycznego

Poprawne zainstalowanie pakietów WinARM oraz OpenOCD dostarcza wszystkich niezbędnych narzędzi potrzebnych do rozwijania aplikacji dla platform wbudowanych. Niemniej jednak korzystanie z nich wymaga bezpośredniej interakcji z linią poleceń systemu Windows, co dla niektórych programistów może być uciążliwe. Możliwe jest napisanie skryptów systemowych pozwalających na uruchamianie sekwencji procedur wymaganych np. do zaprogramowania robota, ale jest to rozwiązanie nieprzenośne i słabo konfigurowalne. Dlatego też zaleca się instalację zintegrowanego środowiska programistycznego które oprócz edytora kodu umożliwia automatyzację najczęściej wykonywanych zadań. Wybór rodzaju środowiska programistycznego zależy niemal w całości od preferencji programisty gdyż większość potrzebnych narzędzi można bez większych trudności zintegrować z ulubionym edytorem. Na potrzeby tej pracy omówiona zostanie konfiguracja dla środowiska Eclipse. Wybór podyktowany został faktem iż jest to obecnie jedna z najpopularniejszych platform do rozwoju oprogramowania, a co więcej jej konfiguracja przebiega identycznie dla systemu Windows jak i Linux. Z tego względu szczegółowy opis procedury konfiguracji zamieszczony został w rozdziale poświęconym narzędziom dla systemu Linux.

3.2. Narzędzia dla systemu Linux

Jednym z wymagań pracy dyplomowej było przygotowanie zestawu narzędzi dla systemu Linux, dzięki którym będzie możliwy dalszy rozwój robota. Zbiór programów potrzebnych do rozwoju projektu to: zintegrowane środowisko programistyczne (IDE), kompilator oraz oprogramowanie pozwalające zaprogramować mikrokontroler. Bieżący rozdział opisuje sposób instalacji i wykorzystania poszczególnych narzędzi, a także daje światło na inne tego typu oprogramowanie, które było brane pod uwagę podczas pracy nad projektem, jednak nie zostało użyte.

3.2.1. Wybór zintegrowanego środowiska programistycznego

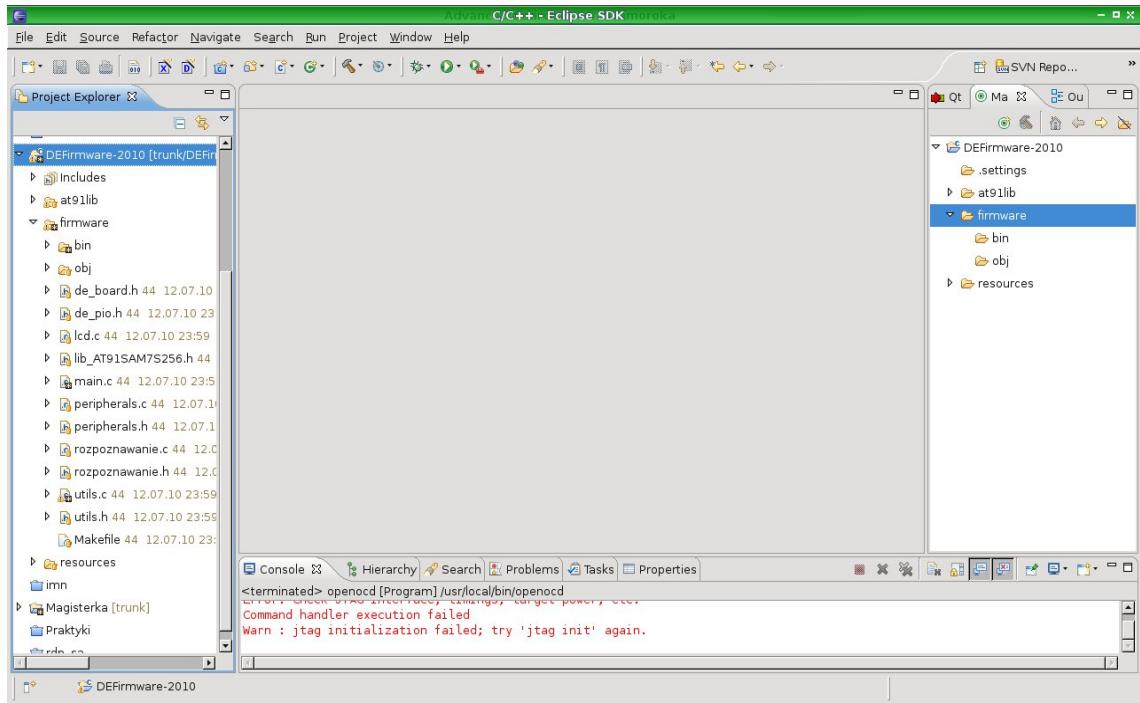
We wcześniejszej wersji oprogramowanie robota było rozwijane na systemie Microsoft Windows. Niestety zintegrowane środowisko programistyczne używane do tej pory nie jest multiplatformowe. Konieczne było zatem dobranie nowego IDE, które umożliwiałoby będzie rozwijanie stworzonego wcześniej kodu pod systemem Linux. Oczywiście biorąc pod uwagę niski budżet projektu, wszelkie płatne rozwiązania zostały prawie od razu odrzucone. Rozważaniom zostały poddane następujące środowiska: Eclipse, Netbeans, CodeWarrior, Kile, ARM Workbench IDE.

Ostatecznie zostało wybrane środowisko Eclipse, które jest dostępne na zasadach licencji: Eclipse Public License, odpowiadającej wymaganiom projektu. Największymi zaletami tego rozwiązania jest łatwość instalacji, konfiguracji oraz użytkowania. W podjęciu ostatecznej decyzji równie istotne było to, iż rozwiązania płatne takie jak np. CodeWarrior są bazowane na Eclips'ie. Plusem był także fakt iż dostępne są różnego rodzaju dodatki do Eclipse'a dedykowane do rozwoju oprogramowania na mikrokontrolery ARM.

3.2.2. Instalacja i konfiguracja Eclipse'a

Wybrane środowisko programistyczne (Eclipse) jest udostępniane pod adresem: <http://www.eclipse.org/downloads/>. Ściągnięte archiwum rozpakowujemy w wybranym przez nas miejscu. Przechodzimy następnie do katalogu który został wydobyty z archiwum i uruchamiamy Eclipse'a.

Oprogramowanie zaraz po pierwszym uruchomieniu zapyta nas o miejsce w którym będą przechowywane źródła naszego projektu tzw. Workspace. Dobrym pomysłem jest potwierdzenie ustawień domyślnych i zapamiętanie tej ścieżki.



Rysunek 3.2. Okno główne IDE Eclipse z dodanym projektem

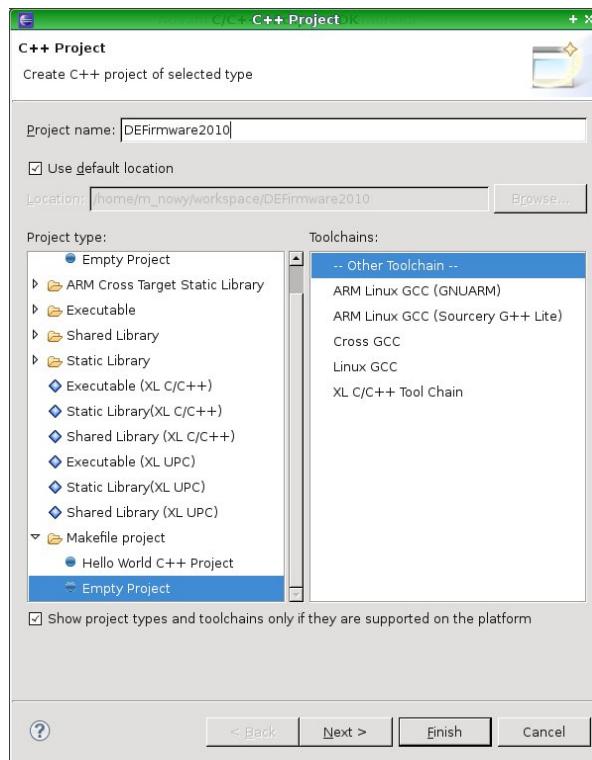
Dodawanie projektu z firmware'm Dark Explorer'a

Do workspace'u Eclipse'a należy skopiować katalog z kodem sterującym robota. Następnie dodajemy nowy projekt w IDE wybierając kolejno *File -> New -> C++ Project*. W oknie dialogowym *C++ Project* (rysunek 3.3), należy podać nazwę projektu która powinna być zgodna z nazwą katalogu zawierającego kod robota. Następnie z listy Project type, wybieramy *Makefile project -> Empty Project*, a jako Toolchain wybieramy *Other toolchain*. Całą operację zatwierdzamy przyciskiem *Finish*.

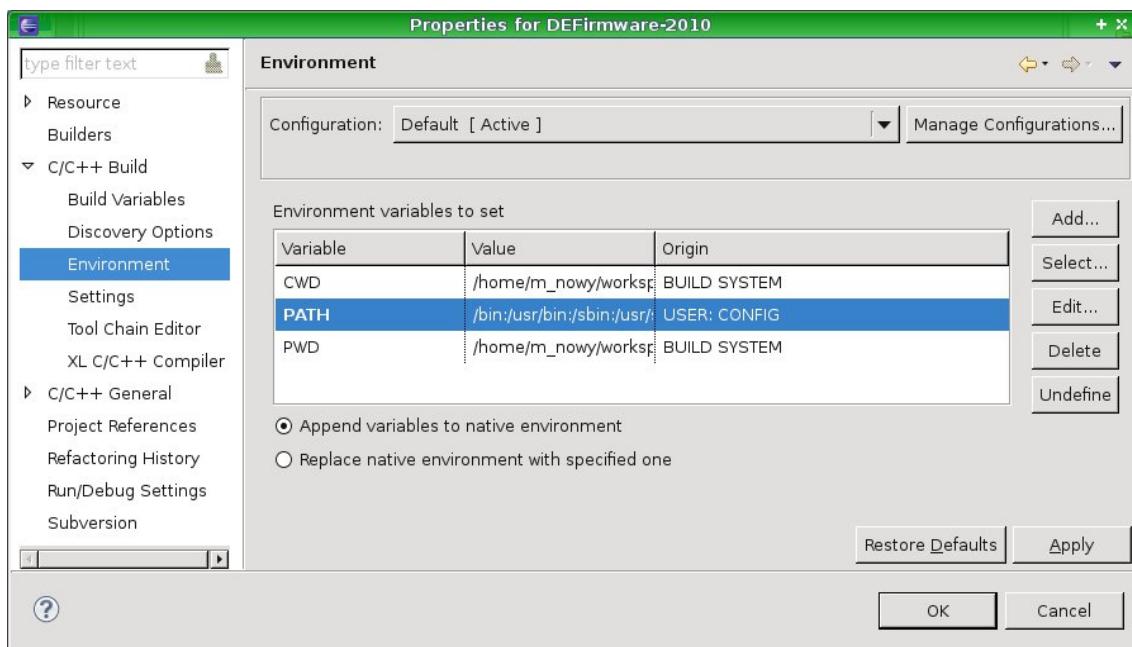
Po poprawnym dodaniu projektu naszym oczom powinno się ukazać okno podobne do tego na rysunku 3.2

W przypadku gdy przed uruchomieniem Eclipse'a nie ustawiliśmy ścieżki do odpowiedniego toolchain'a w zmiennych środowiskowych systemu. podjąć dodatkowe kroki. Eclipse umożliwia konfigurowanie tych zmiennych dla pojedynczych projektów. W celu ustawienia wymaganej ścieżki klikamy prawym klawiszem myszy na nazwie naszego projektu w *Project Explorer'ze* i wybieramy *Properties*.

Po ukazaniu się okna dialogowego *Properties* (rysunek 3.4) wybieramy *C/C++ Build -> Environment*. Następnie modyfikujemy zmienną *PATH*, dodając na końcu ścieżkę do naszego toolchain'a.



Rysunek 3.3. Okno dialogowe C++ Project



Rysunek 3.4. Okno dialogowe Properties

W celu przetestowania poprawności naszej konfiguracji możemy spróbować skompilować kod, klikając prawym klawiszem myszy na nazwę projektu, a następnie wybierając z menu kontekstowego opcję *Build Project*. W wyniku powinniśmy otrzymać dwa pliki

binarne w katalogu *firmware/bin*. Są to pliki gotowe do umieszczenia w pamięci flash robota.

3.2.3. Instalacja i konfiguracja toolchain'a

W celu przetworzenia kodu do formy współpracującej z mikrokontrolerem ARM niezbędny nam jest odpowiedni toolchain, czyli zestaw narzędzi generujących pliki wykonywalne oraz pomagających w debugowaniu utworzonego oprogramowania. Tworzony kod był komplikowany przy pomocy GNU ARM toolchain w wersji 3.4.3 dostępnej na stronie http://www.gnuarm.com/bu-2.15_gcc-3.4.3-c-c--java_nl-1.12.0_gi-6.1.tar.bz2

Pościągnięciu archiwum ze strony producenta należy rozpakować je do dowolnego katalogu, na potrzeby tej pracy założymy że będzie to katalog */usr/local/* W celu zapewnienia dostępu do toolchain'a wszystkim programom wskazane jest dodanie ścieżki */usr/local/bin* do zmiennej środowiskowej PATH (komenda `export PATH=$PATH:/usr/local/bin`).

W celu sprawdzenia poprawności instalacji należy wykonać komendę:

```
arm-elf-gcc --version
```

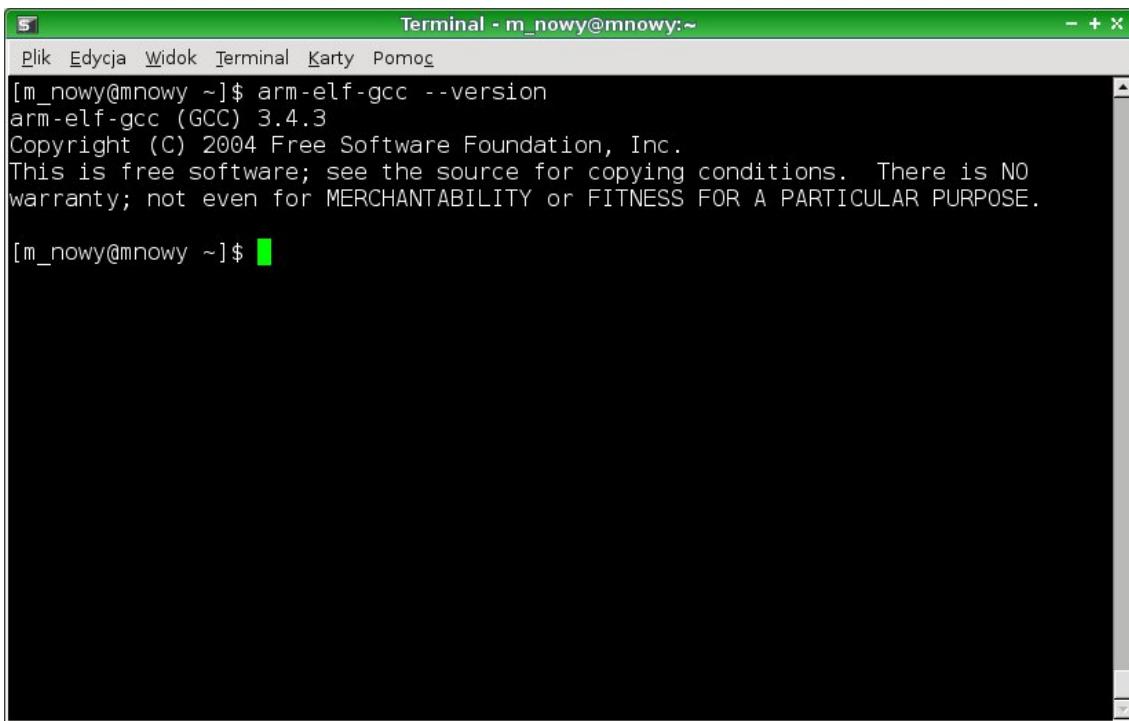
której wynikiem powinien być komunikat podobny do tego na rysunku 3.5.

Trzeba wziąć pod uwagę to, iż toolchain o którym mowa był przygotowany pod system 32-bitowy. W przypadku konfiguracji na systemie 64-bitowym konieczne jest zaopatrzenie się w 64-bitową wersję binarną toolchain'a lub skompilowanie go samodzielnie. Ewentualne dodatkowe informacje można znaleźć pod adresem <http://www.gnuarm.com>.

3.2.4. Open On-Chip Debugger – instalacja i konfiguracja

Pamięć robota była programowana przy pomocy tego samego narzędzia które służy do sprawdzania poprawności działania napisanego kodu. Mowa tu o oprogramowaniu Open On-Chip Debugger.

Źródła programu należy pobrać ze strony: <http://sourceforge.net/projects/openocd/>. Autorzy programu nie zamieścili wersji binarnych, wiec komplikacje będziemy musieli przeprowadzić sami.



The screenshot shows a terminal window titled "Terminal - m_nowy@mnowy:~". The menu bar includes "Plik", "Edycja", "Widok", "Terminal", "Karty", and "Pomoc". The terminal content displays the output of the command "arm-elf-gcc --version":

```
[m_nowy@mnowy ~]$ arm-elf-gcc --version
arm-elf-gcc (GCC) 3.4.3
Copyright (C) 2004 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[m_nowy@mnowy ~]$
```

Rysunek 3.5. Okno pokazujące odpowiedź prawidłowo zainstalowanego kompilatora

Pościągnięciu i rozpakowaniu źródeł uruchamiamy skrypt configure z odpowiednimi argumentami przy pomocy komendy:

```
./configure --prefix=/usr/local --enable-ft2232_libftdi
```

Dodatkowy argument powoduje włączenie obsługi urządzeń bazujących na układzie FT2232 używając biblioteki libftdi. Jest to niezbędne przy korzystaniu z programatora Triton JTAG A. W przypadku wykorzystywania programatora innej firmy możliwa będzie konieczność wprowadzenia innego argumentu do skryptu konfiguracyjnego. Argument prefix określa ścieżkę docelową instalacji oprogramowania.

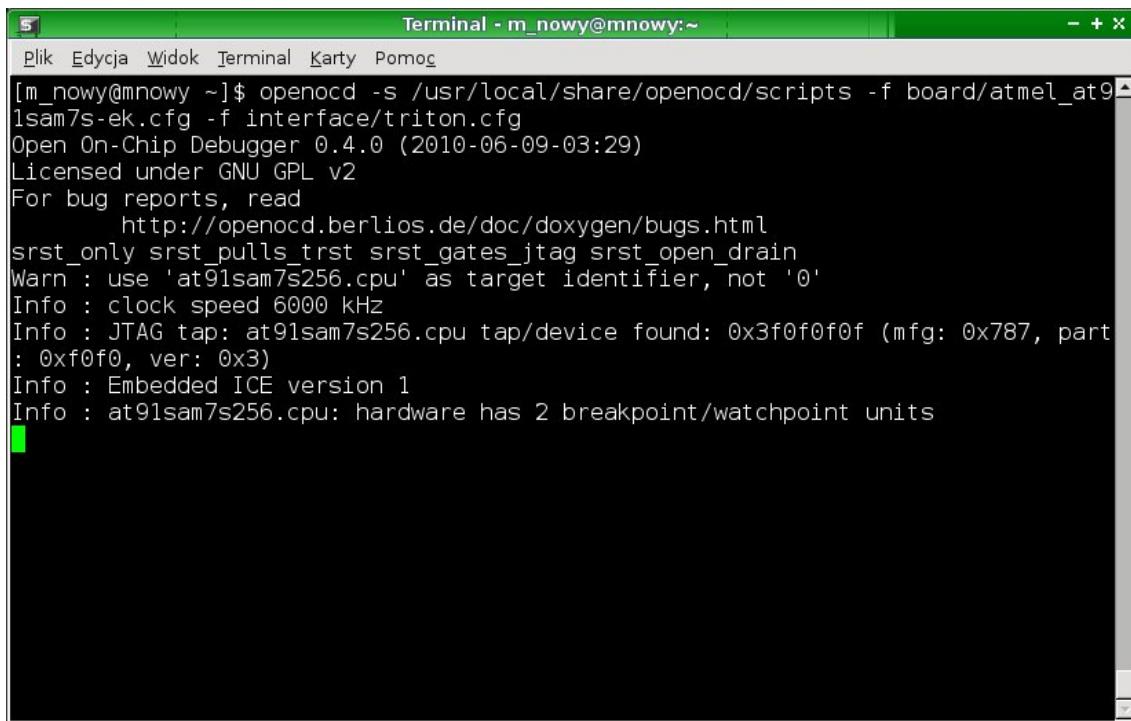
Gdy skrypt konfiguracyjny zakończy działanie z powodzeniem, możemy wywołać komendę `make && make install` w celu komplikacji i instalacji oprogramowania.

Po zakończeniu powyższych czynności należy skopiować plik triton.cfg (dodatek C) konfigurujący połączenie przy pomocy programatora Triton JTAG A do katalogu `/usr/local/openocd/interface`.

W celu przetestowania działania Open On-Chip Debugger'a należy podłączyć programator Triton JTAG A do portu usb komputera oraz portu JTAG robota. Po upewnieniu się że robot jest włączony wykonujemy komendę:

```
openocd -s /usr/local/share/openocd/scripts -f board/atmel_at91sam7s-ek.cfg
```

```
-f interface/triton.cfg
```

A screenshot of a terminal window titled "Terminal - m_nowy@mnowy:~". The window has a green header bar with menu options: Plik, Edycja, Widok, Terminal, Karty, Pomoc. The main area displays the output of the "openocd" command. The output shows the version of the Open On-Chip Debugger (0.4.0), the date it was compiled (2010-06-09-03:29), and the fact that it is licensed under GNU GPL v2. It also provides information about the target identifier (at91sam7s256.cpu) and its clock speed (6000 kHz). The JTAG tap (at91sam7s256.cpu) is found at address 0x3f0f0f0f (mfg: 0x787, part: 0xf0f0, ver: 0x3). The Embedded ICE version is 1, and the CPU has 2 breakpoint/watchpoint units.

Rysunek 3.6. Poprawnie uruchomiony openocd

Programowanie pamięci Dark Explorer'a

W celu wgrania programu do pamięci robota musimy uruchomić dwa narzędzia: telnet oraz openocd. Procedura instalacji i uruchamiania openocd została opisana wcześniej w rozdziale 3.2.4. W pierwszym kroku uruchamiamy openocd w celu podłączenia się do robota. Następnie startujemy telnet za pomocą polecenia:

```
telnet localhost 4444
```

Zapewnia nam to możliwość wysyłania komend sterujących do openocd. W celu zaprogramowania pamięci flash Dark Explorer'a i uruchomienia nowej wersji firmware'u należy wykonać zestaw komend:

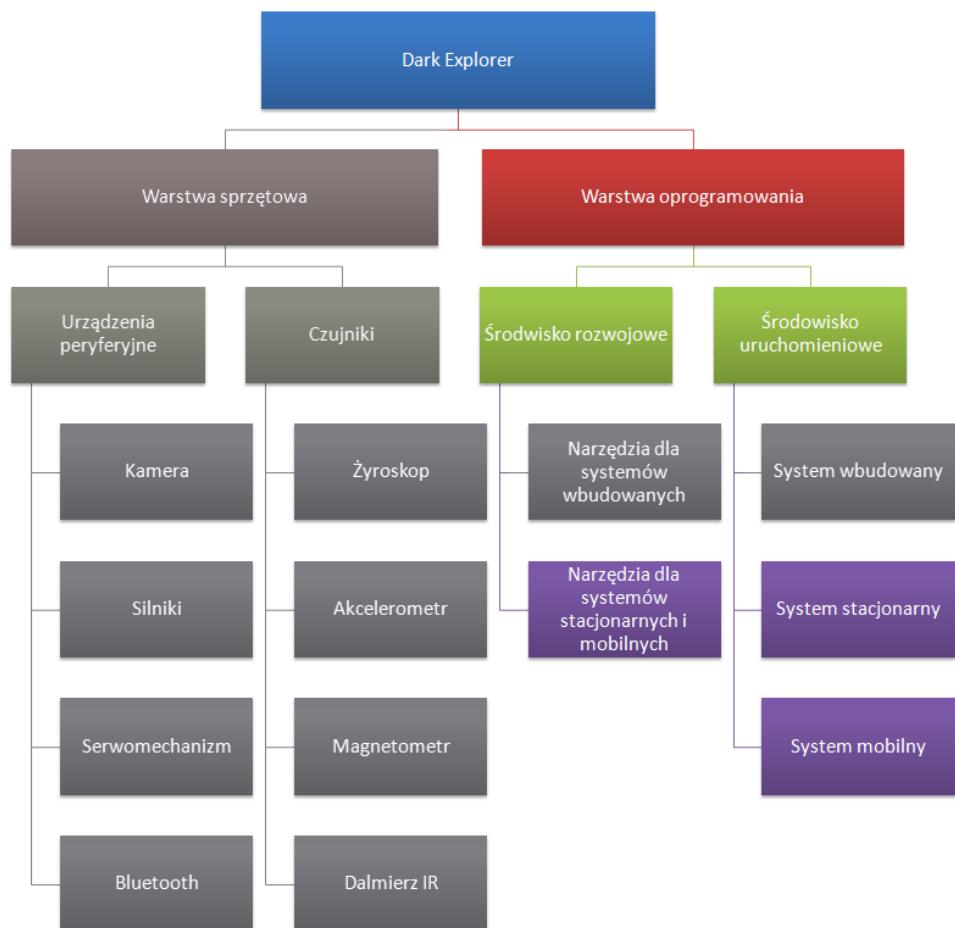
```
halt  
flash write_image {ściezka_do_pliku_elf}  
reset init  
resume
```

Natomiast w przypadku programowania pamięci RAM robota wykonujemy następujący zestaw poleceń:

```
halt  
load_image {ściezka_do_pliku_bin} {początkowy_adres_pamięci}  
reset init  
resume
```

Rozdział 4

Rozwój robota – środowiska dla PC i urządzeń mobilnych



Rysunek 4.1. Struktura platformy robota mobilnego po zakończeniu prac. Kolorem oznaczono zakres prac opisanych w bierzącym rozdziale.

4.1. Biblioteka zarządzająca dla języka Java

4.1.1. Bluetooth a Java SE

4.1.2. Opis możliwości biblioteki

4.2. Aplikacja zarządzająca na komputery stacjonarne (Java)

4.3. Biblioteka programistyczna dla platformy .NET

Jednym z poważniejszych problemów zauważonych podczas analizy pierwotnej konfiguracji robota był brak bibliotek umożliwiających tworzenie oprogramowania które pozwalałoby na dowolne wykorzystanie możliwości oferowanych przez konfigurację sprzętową robota. Brak tego rodzaju narzędzi znaczco ogranicza możliwości rozwoju robota gdyż każda próba tworzenia nowego oprogramowania wymaga od programisty zagłębiania się w szczegóły implementacji systemu wbudowanego który kontroluje działanie robota. Aby usunąć tak poważne ograniczenie zaprojektowana została biblioteka mająca na celu udostępnienie narzędzi pozwalających programiście na skupienie się jedynie wysokopoziomowej funkcjonalności bez konieczności szczegółowej analizy protokołu komunikacji i zasad działania poszczególnych funkcji robota.

Przed przystąpieniem do implementacji konieczne jest podjęcie rozważnej decyzji z wyborem środowiska i języka programowania w jakim biblioteka zostanie napisana. Biorąc pod uwagę ciągle rosnącą popularność obiektowych języków programowania oczywistym wydaje się być wybór języka właśnie z tej rodziny. Decydującym aspektem, wpływającym na ostateczny wybór docelowej platformy rozwojowej, jest więc ilość dostępnych bibliotek oraz przenośność kodu pomiędzy dostępnymi na rynku platformami sprzętowymi. Po przeprowadzeniu wnikliwej analizy ostateczny wybór padł na język C# oraz platformę .NET. Wybór motywowany jest faktem iż platforma .NET jest jednym z najdynamiczniej rozwijających środowisk programistycznych ostatnich lat. Firma Microsoft dostarcza szereg bibliotek dodatkowych oraz narzędzi pozwalających na szybkie tworzenie oprogramowania działającego zarówno na urządzeniach mobilnych jak i stacjonarnych. Dzięki pracy programistów w ramach projektu Mono¹ powstała platforma umożliwiająca uruchamianie aplikacji napisanych w języku C# nie tylko pod kontrolą systemu Windows, ale również pod systemami z rodziny Linux i Mac. Dodatkowym atutem platformy .NET jest bardzo duża liczba bibliotek dodatkowych dostarczanych przez środowiska programistów opensource.

W ramach pracy magisterskiej zaprojektowana została biblioteka programistyczna w języku C#. Docelową platformą uruchomieniową dla przygotowanej biblioteki są systemy z rodziny Windows, Windows Mobile oraz Linux. Przy tworzeniu biblioteki brane pod uwagę były najnowsze trendy w dziedzinie programistycznych wzorców projektowych przy jednoczesnym zachowaniu spójności i uniwersalności kodu dla poszczególnych środowisk uruchomieniowych. W wyniku implementacji powstała wielowątkowa biblioteka oparta

¹ Więcej informacji na temat projektu Mono można znaleźć pod adresem strony internetowej <http://www.mono-project.com/>

na zdarzeniach pozwalająca na dostęp do wszystkich funkcji robota za pomocą intuicyjnego interfejsu programistycznego. Biblioteka udostępnia szerokie spektrum metod pozwalających na swobodne sterowanie i zarządzanie dostępnymi funkcjami robota. Wśród metod bibliotecznych znaleźć można funkcje pozwalające na bezpośrednią interakcję z poszczególnymi podzespołami bazowymi, jak również takie które umożliwiają wykonywanie predefiniowanych sekwencji zadań przewidzianych przez autorów projektu. Biblioteka pokrywa swoją funkcjonalnością nie tylko wsparcie dla wszystkich dostępnych rozszerzeń sprzętowych robota, ale również dostarcza interfejs pozwalający na automatyczną obsługę połączenia z robotem z wykorzystaniem technologii bluetooth. Fakt ten jest o tyle istotny, że proces komunikacji okazał się być najbardziej wrażliwym elementem podczas migracji biblioteki pomiędzy poszczególnymi platformami softwareowymi. Szczegółowa lista wszystkich dostępnych funkcji wraz z niezbędnym komentarzem zamieszczona została w dodatku poświęconym kodu źródłowemu stworzonego w ramach pracy magisterskiej. Taki sposób implementacji pozwala na tworzenie oprogramowania współpracującego z nową wersją robota nawet przez osoby nie posiadające dostatecznej wiedzy i umiejętności tworzenia oprogramowania do obsługi systemów wbudowanych.

W ramach pracy magisterskiej stworzona została również przykładowa aplikacja sterująca prezentująca wszystkie możliwości oferowane zarówno przez warstwę aplikacyjną jak i sprzętową robota Dark Explorer.

TODO: zrzut ekranu aplikacji sterującej dla .net'a

4.4. Platforma mobilna (Windows Mobile 6.1)

Mobilne urządzenia przenośne z dnia na dzień zyskują na popularności. Każdego dnia spotykamy się z nimi w domu, w pracy czy spacerując po parku. Z całą pewnością można stwierdzić, iż większa część społeczeństwa obecnie jest w posiadaniu telefonu komórkowego, komputera przenośnego czy też jakiegoś innego urządzenia mobilnego. Wszystkie z wspomnianych urządzeń posiadają charakterystyczną dla siebie platformę software'ową. Do najlepiej znanych we współczesnym świecie zaliczyć można między innymi: Windows Mobile, iPhone, BlackBerry, Symbian OS, Android, Maemo, OpenMoko itp. Każda z wymienionych platform posiada inną genezę jak również ma swoje mocne i słabe strony.

Platformy takie jak Windows Mobile, BlackBerry czy iPhone ograniczone są do urządzeń dedykowanych docelowo do współpracy z wspomnianymi środowiskami. Obok różnorakich problemów z jakimi zmagają się wspomniane wcześniej platformy do jednego z najpoważniejszych zaliczyć można bardzo ograniczone w niektórych aspektach API². Nawet tak przenośna platforma jak Java na urządzeniach przenośnych nie zawsze się sprawdza ze względu na liczne braki oraz różnice w API zmuszające programistów do tworzenia kodu dedykowanego dla konkretnego urządzenia. Symbian oraz Windows Mobile wypadają na tym tle nieco lepiej ponieważ wspierają szerszą gamę urządzeń jak również ich API daje więcej możliwości niż ma to miejsce na przykład w przypadku Javy. Głównym powodem takiego stanu rzeczy jest bardzo szeroki i różnorodny asortyment platform sprzętowych utrudniający stworzenie jednolitej i w pełni wykorzystującej wszystkie możliwości urządzenia platformy programistycznej. Dostępne w chwili obecnej OpenSource'owe i wieloplatformowe rozwiązania znajdują się ciągle we wczesnej fazie rozwoju i nie są jeszcze powszechnie znane przez środowiska twórców oprogramowania.

Firma Microsoft wypuściła po raz pierwszy na światło dzienne swoją platformę mobilną w latach 90-tych [13]. Natomiast w roku 2002 pojawiła się pierwsza platforma Windows CE.NET. Zapoczątkowało to popularyzację urządzeń Pocket PC opartych o system Windows CE 3.0 oraz późniejsze wersje. Dalszy rozwój bezprzewodowych technologii telekomunikacyjnych pozwolił na integracje telefonu z komputerem osobistym. Wspomniane urządzenia Pocket PC z 2002 roku wspierały między innymi standard GSM³, GPRS⁴, bluetooth oraz umożliwiały użytkownikom dostęp do sieci bezprzewodowych. W między

² Application Programming Interface - interfejs programowania aplikacji, specyfikacja instrukcji pozwalających na dostęp do zasobów dostarczanych przez zewnętrzny program

³ Global System for Mobile Communication, pierwotnie Groupe Special Mobile - najpopularniejszy obecnie standard telefonii komórkowej

⁴ General Packet Radio Service - technologia pakietowego przesyłania danych popularnie stosowana w sieciach GSM

czasie rozwojowi ulegały urządzenia typu SmartPhone które koncepcyjnie były bardzo zbliżone do Pocket PC jednakże były one bardziej zbliżone do telefonu niż komputera osobistego. Podstawową różnicą pomiędzy Smartphone i Pocket PC jest fakt iż urządzenia Pocket PC posiadają ekran dotykowy, a Smartphone wyposażone są jedynie w przyciski umożliwiające sterowanie urządzeniem. Każde z tych urządzeń posiadało inny zestaw aplikacji pomocniczych oraz wspierało inne standardy i technologie.

W chwili obecnej większość urządzeń Pocket PC oraz Smartphone działają w oparciu o system Windows Mobile 5 oraz Windows Mobile 6. Nowoczesne urządzenia Pocket PC wyposażone są w procesor o taktowaniu 500-600 MHz oraz od 64-128 MB pamięci RAM. Najnowsze urządzenia z tej grupy wyposażane są w 1 GHz procesor oraz 512 MB pamięci.

4.4.1. Środowisko rozwojowe

Tworzenie aplikacji działających na urządzeniach pod kontrolą systemu Windows Mobile jest niemal tak samo proste jak tworzenie zwykłych aplikacji na komputery stacjonarne. Niemniej jednak do stworzenia w pełni funkcjonalnego środowiska rozwojowego konieczne jest przejście przez kilka kroków przygotowawczych związanych z instalacją potrzebnych aplikacji narzędziowych.



Rysunek 4.2. Elementy składowe środowiska rozwojowego Windows Mobile

Przed rozpoczęciem przygody z tworzeniem aplikacji dla systemu Windows Mobile konieczne jest zainstalowanie Microsoft Visual Studio. Zaleca się aby Visual Studio było w wersji 2005 lub 2008. Niestety narzędzia umożliwiające rozwijanie aplikacji mobilnych nie są poprawnie wykrywane przez Visual Studio 2010 oraz poprzednie wydania w wersji Express. Dlatego też koniecznością jest instalacja środowiska w wersji Standard lub Professional. Każda z tych wersji może zostać pobrana w wersji czasowej ze stron firmy Microsoft lub w wersji pełnej z MSDNAA⁵. Visual Studio posłuży nam nie tylko do edycji kodu aplikacji ale pozwoli również w prosty sposób budować, debugować oraz przygotować instalator finalnej wersji aplikacji. Po poprawnym zainstalowaniu środowiska rozwojowego konieczne jest pobranie i zainstalowanie dostępnych paczek serwisowych dostępnych dla wybranej wersji Visual Studio. Pozwoli to uniknąć nieprzyjemnych niespodzianek podczas instalacji bibliotek narzędziowych i późniejszej pracy.

Jeżeli posiadamy już zainstalowaną kopię Visual Studio możemy przystąpić do instalacji narzędzi pomocniczych które pomogą nam w tworzeniu aplikacji. Pierwszą niezbędną biblioteką jest .NET Compact Framework 2.0 SP1. Jest to zestaw narzędzi wykorzystywanych do uruchamiania aplikacji na platformach opartych o Windows Mobile. Aby ułatwić sobie proces budowania, debugowania i uruchamiania aplikacji na urządzeniu konieczne jest zainstalowanie w systemie Windows ActiveSync. Dzięki ActiveSync możliwe stanie się uruchamianie projektowanej aplikacji, bezpośrednio z IDE, nie tylko na prawdziwym urządzeniu ale również emulatorze.

Ostatnim, ale i zarazem najważniejszym krokiem jest instalacja Windows Mobile SDK⁶. Na stronach firmy Microsoft dostępne są dwie wersje SDK, Standard oraz Professional. Wersja Standard zawiera w sobie tylko wsparcie dla urządzeń z Windows Mobile Classic lub Standard natomiast wersja Professional obejmuje wszystkie dostępne środowiska. Wybór SDK można sprowadzić do następującej zasady. Jeżeli zamierzamy tworzyć oprogramowanie dla urządzeń Smartphone bez ekranu dotykowego w zupełności wystarczy nam wersja standardowa. Jeżeli natomiast planujemy napisane aplikacje uruchamiać na PocketPC lub dotykowych SmartPhon'ach będziemy potrzebować bibliotek systemu Windows Mobile Classic lub Professional, tak więc konieczne jest użycie SDK w wersji Professional. Tak jak w przypadku Visual Studio, również tutaj zaleca się instalację wszystkich

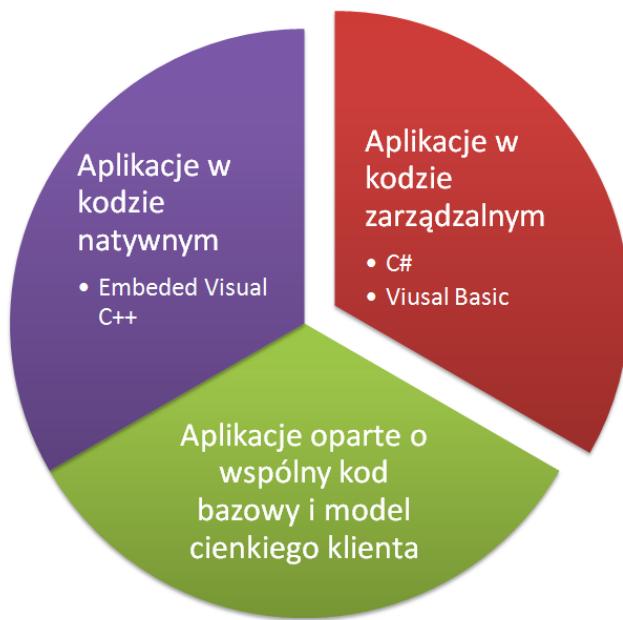
⁵ Microsoft Developer Network Academic Alliance - program firmy Microsoft skierowany do studentów i pracowników naukowych w ramach którego uczestnicy mogą pozyskać darmowe kopie oprogramowania firmy Microsoft

⁶ Software Development Kit - zestaw narzędzi programistycznych niezbędnych do tworzenia aplikacji korzystających z funkcjonalności dostarczonej przez daną bibliotekę

dostępnych na stronie producenta aktualizacji i poprawek. Jest to szczególnie istotne podczas pracy z emulatorami urządzeń. Po zrealizowaniu tych kroków otrzymujemy w pełni funkcjonalne środowisko do rozwoju aplikacji mobilnych dla urządzeń smartphone.

Modele aplikacji

Istnieje kilka modeli rozwoju aplikacji dla Windows Mobile, a wybór docelowego modelu został pozostawiony programiście. Pierwszy z nich służy do tworzenia aplikacji w kodzie natywnym. Aplikacje pisane zgodnie z tym modelem cechują się wysoką wydajnością, bezpośrednim dostępem do sprzętu oraz małym zużyciem zasobów. Do rozwoju tego rodzaju aplikacji korzysta się z reguły ze środowiska do rozwijania aplikacji z użyciem Embedded Visual C++. Główną wadą tego modelu jest niska przenośność pomiędzy różnymi platformami zwłaszcza jeżeli aplikacja korzysta z urządzeń specyficznych dla danego modelu urządzenia. Docelowo więc za pomocą tego modelu tworzy się biblioteki i narzędzia ułatwiających tworzenie bardziej skomplikowanych aplikacji. Jeżeli więc interesuje nas tworzenie wysokopoziomowych aplikacji z GUI, skierowanych bezpośrednio do użytkowników, zaleca się tworzenie tego typu aplikacji za pomocą kodu zarządzalnego z użyciem takich języków jak C# czy Visual Basic.



Rysunek 4.3. Modele tworzenia aplikacji dla Windows Mobile.

Rozwijanie aplikacji w oparciu o kod zarządzalny pozwala na stworzenie programu który będzie mógł w pełni wykorzystywać możliwości oferowane przez Microsoft .NET Compact Framework. Umożliwia to programiście tworzenie rozproszonych systemów mobilnych pracujących zarówno w modelu ze stałym połączeniem jak i bez. Spora część

narzędzi dostępnych w ramach .NET Compact Framework jest również wykorzystywana do rozwoju aplikacji na komputery stacjonarne. Biblioteka została zaprojektowana do celowo na urządzenia o ograniczonych zasobach co w połączeniu z możliwościami języków z rodziny .NET oraz integracją z Visual Studio daje nam profesjonalny zestaw narzędzi do tworzenia aplikacji mobilnych.

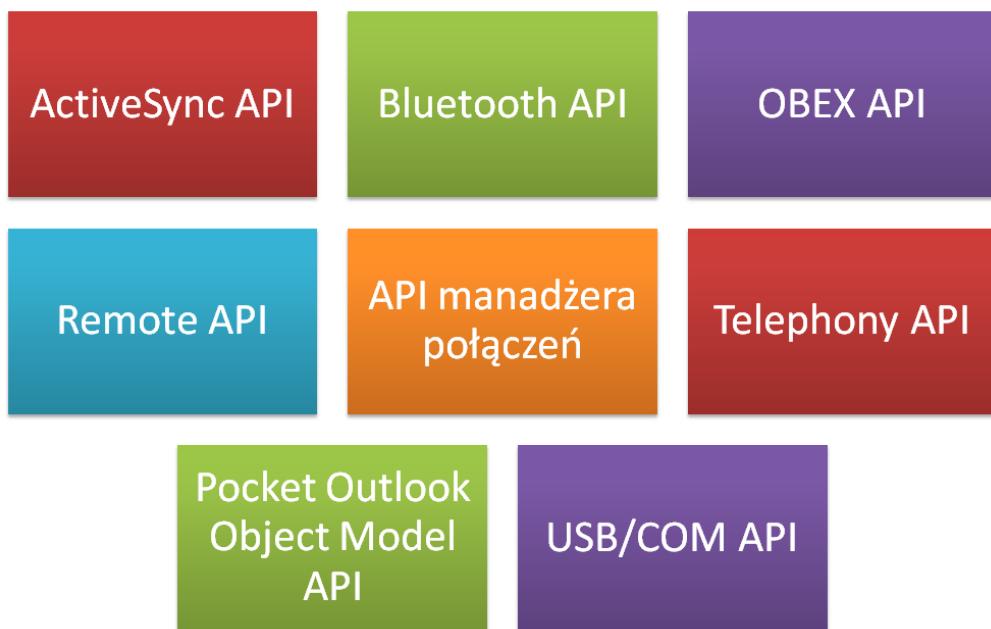
Trzecim modelem tworzenia programów pod Windows Mobile jest wykorzystywanie kodu serwera do pracy z wieloma różnymi typami urządzeń poprzez jeden wspólny kod bazowy i model cienkiego klienta. Oczywiście tego typu podejście ma sens jedynie gdy możemy zagwarantować stabilny kanał komunikacyjny pomiędzy urządzeniem klienta, a serwerem. Każdy z przedstawionych modeli idealnie sprawdza się jeżeli tylko w prawidłowy sposób wybierzemy model najbardziej odpowiadający potrzebom naszej aplikacji.

Graficzny interfejs użytkownika

Dzięki wygodnemu systemowi projektowania GUI dostępnego w Visual Studio tworzenie interfejsu użytkownika dla platformy mobilnej jest niemal tak proste jak w przypadku tradycyjnych aplikacji, a jedyną różnicą jest ilość i rodzaj dostępnych kontrolek. Różnice te wynikają z faktu iż niektóre urządzenia mobilne posiadają ekran dotykowe, a inne nie. Co za tym idzie, rozwój interfejsu użytkownika staje się bardziej skomplikowany zwłaszcza jeżeli interesuje nas rozwój aplikacji wspólnej dla obydwóch platform sprzętowych. W tym miejscu nie może zabraknąć informacji, że oprogramowanie zbudowane dla PocketPC nie uruchomi się na urządzeniach SmartPhone natomiast sytuacja odwrotna jest możliwa do momentu w którym aplikacja nie zacznie korzystać ze specyficznych funkcji SmartPhone. Naturalnym stanem rzeczy wydaje się fakt, iż wiele funkcji i komponentów graficznych znanych z aplikacji desktopowych zostało usunięte z bibliotek Windows Mobile, aby zapewnić jej wydajność i niewielki rozmiar. Dlatego też pozostawiono tylko niezbędne, najprostsze komponenty. Ponieważ wydajność i zasoby pamięciowe urządzeń stale rosną, również ilość narzędzi dostępnych w SDK jest systematycznie zwiększana, a co za tym idzie różnice pomiędzy kolejnymi wersjami .NET Compact Framework są bardzo duże. Dlatego też posiadanie jak najbardziej aktualnej wersji SDK ma niebagatelne znaczenie dla wygody tworzenia aplikacji. Podsumowując rozwój GUI dla platformy mobilnej nie różni się bardzo od tworzenia interfejsu użytkownika dla aplikacji dekstopowej. Istnieje również możliwość rozwijania GUI w oparciu o silniki 3D. W chwili obecnych dostępne są takie rozwiązania jak GAPI (Game API), OpenGL ES (Embedded Systems), Open VG (Vector Graphics). Jednakże rozwój takich aplikacji jest niezwykle trudny gdyż wymaga od programisty tworzenie maksymalnie optymalnego kodu, ze względu na ograniczone możliwości niektórych urządzeń.

Komunikacja

Nowoczesne urządzenia mobilne posiadają szeroką gamę możliwości komunikacyjnych. Posiadają one dostęp do szybkich sieci bezprzewodowych w standardzie 802.11 WiFi. Umożliwiają one również komunikację za pomocą portu podczerwieni, bluetooth czy USB. Podejmując decyzję na temat wyboru kanału komunikacji należy brać pod uwagę nie tylko parametry techniczne, ale również liczbę standardów i protokołów dostępnych dla danego kanału komunikacji. Firma Microsoft dostarcza szereg interfejsów programistycznych (API) umożliwiających niemal błyskawiczny dostęp do funkcji komunikacyjnych urządzenia. ActiveSync API dostarcza funkcjonalność umożliwiającą komunikację za pomocą protokołu synchronizacji. Natomiast Bluetooth API dostarcza zestaw narzędzi umożliwiających nawiązywanie komunikacji bezprzewodowej pomiędzy telefonami jak i urządzeniami periferyjnymi.



Rysunek 4.4. Lista dostępnych na platformie Windows Mobile API komunikacyjnych

API Managera połączeń dostarcza zestaw usług umożliwiających automatyzację procesu nawiązywania połączenia oraz zarządzanie ich aktywnością. API wymiany obiektów (OBEX API) dostarcza funkcjonalność umożliwiającą wymianę danych pomiędzy urządzeniami za pomocą efektywnego, kompaktowego protokołu binarnego dedykowanego dla urządzeń z ograniczonymi zasobami. Remote API (RAPI) dostarcza funkcję do zarządzania oraz zdalnego wywoływanie metod po stronie urządzenia klienta. Dostępne są m.in. funkcje dostępu do rejestru, plików, bazy danych czy też konfiguracji urządzenia. Najważniejszą funkcjonalnością jest jednak możliwość zdalnego wywoływania procedur.

Z pomocą funkcji CeRapiInvoke() przesyłamy do urządzenia nazwę biblioteki dynamicznej wraz z nazwą metody która ma zostać wywołana na urządzeniu mobilnym. Kolejnym zestawem narzędzi jest Pocket Outlook Object Model API dostarcza funkcje do zarządzania obiektami Pocket Outlook co umożliwia synchronizację zadań, kalendarza czy kontaktów za pomocą prostego i intuicyjnego interfejsu. Dostępne jest również Telephony API (TAPI) które zawiera w sobie biblioteki umożliwiające zarządzanie kartą SIM oraz wiadomościami SMS. TAPI udostępnia również zestaw funkcji umożliwiających dostęp do funkcji telefonowania oraz protokołu WAP. Nie zabrakło również narzędzi do pracy z portami USB oraz COM. Część z dostępnych portów COM jest zarezerwowana dla urządzeń wewnętrznych, ale pozostałe dostępne są do pełnej dyspozycji użytkownika.

Debugowanie

Microsoft Visual Studio umożliwia debugowanie aplikacji działających pod kontrolą Windows Mobile niemal w taki sam sposób jak ma to miejsce w przypadku tradycyjnych aplikacji desktopowych. Ponadto programista ma do swojej dyspozycji następujące narzędzia: emulator, panel zarządzania emulowanymi urządzeniami, panel punktów przerwań i wątków. Niestety w Visual Studio nie uda się nam jednocześnie debugować kodu natywnego i zarządzalnego. Możliwe jest natomiast uruchomienie zarówno projektu napisanego w Visual C++ jak i projektu opartego o kod zarządzalny, a dzięki funkcjonalności „Dołącz do procesu” możliwe jest zdalne dołączenie się i monitorowanie procesu działającego na urządzeniu lub emulatorze urządzenia. Narzędziem umożliwiającym komunikację pomiędzy urządzeniem a systemem jest ActiveSync instalowany wraz ze środowiskiem rozwojowym. Za pomocą narzędzia ActiveSync możemy łączyć się nie tylko z rzeczywistymi urządzeniami ale również z emulatorami. Umożliwia to pełną wirtualizację urządzeń mobilnych i znacznie ułatwia testowanie funkcjonalności zwłaszcza pomiędzy różnymi platformami urządzeń (SmartPhone, PocketPC). Jedynym ograniczeniem tego procesu jest możliwość utrzymania tylko jednego aktywnego połączenia co uniemożliwia debugowanie na wielu urządzeniach jednocześnie. Co więcej Visual Studio umożliwia debugowanie jedynie aplikacji stworzonej przez programistę, nie możliwa jest z poziomu IDE debugowanie aplikacji i usług systemowych działających na urządzeniu. Do tego typu debugowania konieczne byłoby zbudowanie własnej wersji systemu Windows Mobile przy użyciu Platform Buildera. Narzędzie to umożliwia również tworzenie własnego SDK dla Visual Studio i platformy Windows CE. Dodatkową możliwością dostępną z poziomu emulatora jest emulowanie połączenia z siecią GSM oraz wsparcie dla GPS. Umożliwia to testowanie, debugowanie i rozwijanie szerokiego spektrum aplikacji bez konieczności posiadania urządzenia fizycznie.

4.4.2. Środowisko uruchomieniowe

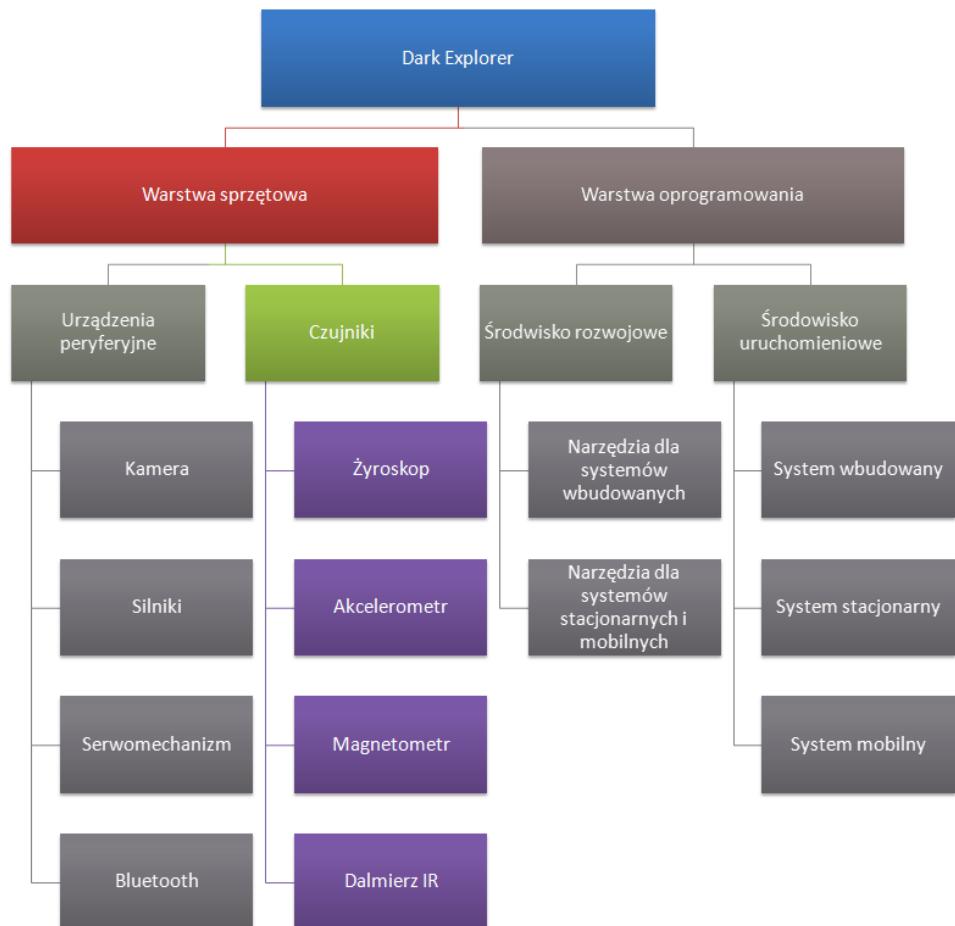
4.5. Platforma mobilna (Java ME)

4.5.1. Narzędzia programistyczne

4.5.2. Aplikacja mobilna

Rozdział 5

Rozwój robota – warstwa sprzętowa



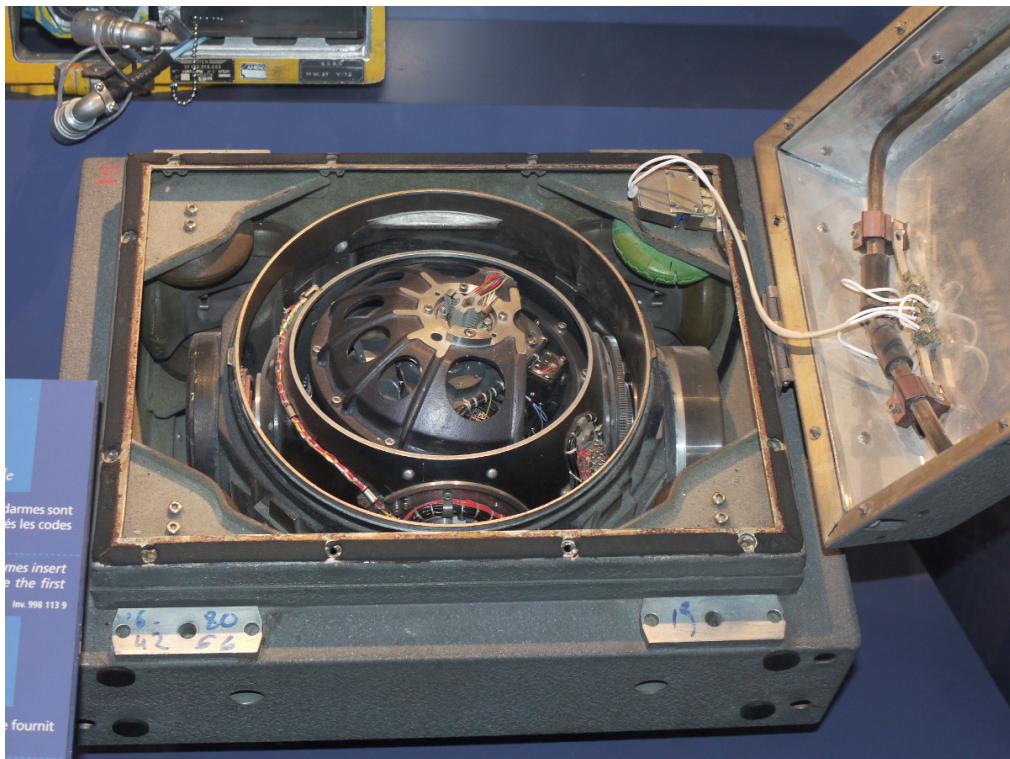
Rysunek 5.1. Struktura platformy robota mobilnego po zakończeniu prac. Kolorem oznaczono zakres prac opisanych w bierzącym rozdziale.

5.1. Inercjalny system nawigacyjny

W celu rozwinięcia możliwości robota, zamontowano w nim elementy przy pomocy których, została podjęta próba stworzenia inercjalnego systemu nawigacyjnego (INS¹). Założeniem było, aby robot mobilny zapamiętał tor ruchu po jakim się porusza, gdy jest niesiony na ręce osoby operującej nim. Następnie na podstawie wykonanych pomiarów robot miał powrócić po zapamiętanym torze w miejsce początkowe. Poniższy rozdział omawia pokrótce czym jest INS, opisuje zasadę działania jego elementów oraz sposób wykorzystania tych podzespołów do osiągnięcia wystarczająco dobrych efektów.

5.1.1. Wprowadzenie do INS

Inercjalny system nawigacyjny jest to narzędzie służące do określenia położenia, prędkości oraz orientacji obiektu w przestrzeni, bez korzystania z żadnych zewnętrznych elementów naprowadzających, które byłyby dla niego punktem odniesienia. Wykorzystuje on jedynie elementy wbudowane, składające się na inercjalną jednostkę pomiarową (IMU²).



Rysunek 5.2. INS francuskiego IRBM S3³

¹ Inertial Navigation System

² Inertial Measurement Unit

³ źródło: http://en.wikipedia.org/wiki/Inertial_navigation_system

Inercjalne systemy nawigacyjne mają zastosowanie tam, gdzie jest potrzebna informacja o aktualnym położeniu obiektów, natomiast nie ma możliwości odbioru sygnału zewnętrznego, wymaganego do działania takich urządzeń jak na przykład urządzeń GPS. Systemy tego typu są stosowane w: samolotach, statkach, łodziach podwodnych, pojazdach bezzałogowych czy statkach kosmicznych. Pierwsze rozwiązania tego typu były drogie i bazowały na bardzo dużych gabarytowo elementach mechanicznych.



Rysunek 5.3. Żyrokompas samolotowy⁴

Rozwój miniaturowych układów elektromechanicznych MEMS⁵ otworzył przed nami cały wachlarz potencjalnych nowych zastosowań INS, np. do śledzenia ruchów ludzi bądź zwierząt.



Rysunek 5.4. IMU stworzone przy pomocy układów MEMS, cena ok.: 80\$⁶

5.1.2. Elementy IMU robota

System nawigacyjny o którym mowa w tym rozdziale oblicza swoje położenie na podstawie ciągłego badania przyspieszenia liniowego oraz prędkości kątowej. INS musi

⁴ źródło: <http://www.gyroscopes.org/uses.asp>

⁵ Microelectromechanical Systems

⁶ źródło: <http://www.flytron.com>

otrzymać na starcie wartości początkowe położenia oraz prędkości z jaką się porusza, aby móc zacząć wyznaczać dalsze przemieszczenie i zmiany w orientacji.

Robot mobilny został wyposażony w IMU składające się z następujących elementów MEMS:

- cyfrowy żyroskop trójosiowy L3G4200D firmy STMicroelectronics
- analogowy akcelerometr trójosiowy MMA7260 firmy Freescale
- cyfrowy magnetometr dwuosiowy firmy MMC2120MG firmy Memsic

Elementy użyte do wykonania IMU zostały wybrane pod kątem walorów ekonomicznych. Nie były przeprowadzane testy porównawcze pomiędzy podzespołami danego typu. Zasada działania poszczególnych elementów oraz opis ich wykorzystania można znaleźć w kolejnych podrozdziałach.

5.2. Akcelerometr trzyosiowy

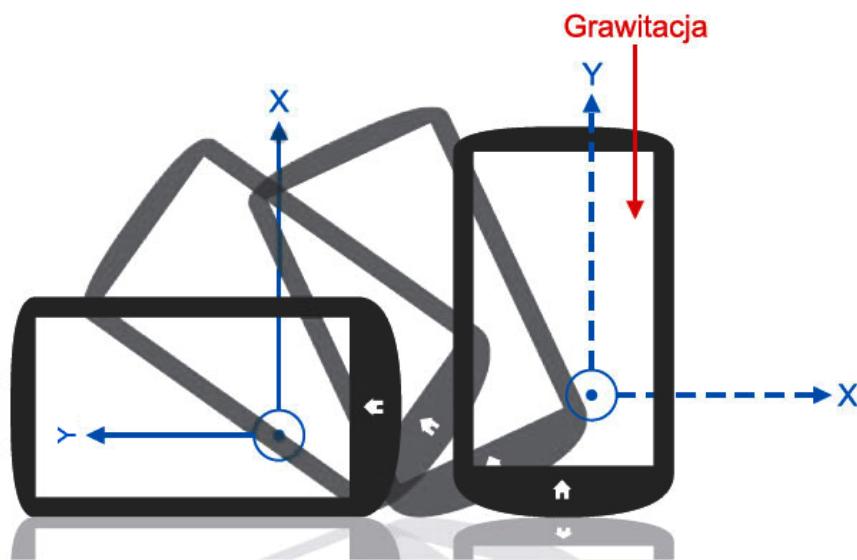
W latach 90 akcelerometry instalowano jedynie w roli mechanizmów służących do uruchamiania poduszek powietrznych w samochodzie w przypadku wystąpienia zderzenia. W chwili obecnej akcelerometry wykonane w technologii MEMS można znaleźć w większości urządzeń codziennego użytku tj. telefony komórkowe, konsole do gier, komputery przenośne czy cyfrowe aparaty fotograficzne. Powodem tak spektakularnego wzrostu popularności było pełne rozwinięcie się technologii MEMS. W dobie miniaturyzacji fakt, iż w miejsce do niedawna używanych urządzeń można wstawić pojedynczy układ scalony jest jednym z podstawowych powodów dla których, między innymi, akcelerometry cieszą się tak szerokim spektrum zastosowań. Jednakże, ceną tak daleko idącej miniaturyzacji jest spadek duży dokładności pomiarowej w porównaniu do akcelerometrów opartych o tensometry. Jak się jednak okazuje, do codziennego użytku precyzaja oferowana przez tego typu czujniki jest w zupełności wystarczająca.

Bardzo dobrym przykładem zastosowania akcelerometru jest dostępna w większości aparatów cyfrowych funkcja optycznej stabilizacji obrazu (Optical Image Stabilization). Funkcja ta polega na redukcji zniekształceń spowodowanych drżeniem rąk fotografa w trakcie akwizycji obrazu. Problem jest tym poważniejszy iż nieustanna miniaturyzacja urządzeń dodatkowo wzmacnia mimowolne⁷ drżenie rąk użytkownika. Dlatego też większość aparatów wyposażonych jest czujniki które mierzą wspomniane drżania, a następnie uzyskane informacje przekazują do modułu sterującego położeniem soczewek i przetwornika obrazu. Następnie moduł ten dostosowuje parametry pracy aparatu tak aby zminimalizować wpływ nieporządków przeunięć na efekt końcowy pracy jakim jest zdjęcie.

Innym popularnym przykładem zastosowania akcelerometrów są pedometry, potocznie nazywane krokomierzami. Dzięki zastosowanej technologii czujniki w krokomierzach mogą dokonywać równocześnie pomiaru względem trzech osi. Co więcej tego typu urządzenia umożliwiają dokonywanie pomiarów niemal całkowicie niezależne od swojej orientacji, a ich niewielkie rozmiary pozwalają na ich integrację z dowolnymi urządzeniami mobilnymi. Akcelerometry wykonane w technologii MEMS najlepiej sprawdzają się podczas pomiaru przyspieszenia statycznego pozwalającego jednoznacznie wyznaczyć kąt odchylenia urządzenia względem pionu. Można nimi dokonywać również pomiaru przyspieszenia dynamicznego pojawiającego się na skutek wibracji, uderzenia czy innego rodzaju ruchu.

⁷ Ręka przeciętnego człowieka drży mimowolnie z częstotliwością od 10 do 20 Hz

Niestety, pomimo swoich licznych zalet akcelerometry MEMS posiadają pewne ograniczenia które mogą być bardzo istotne podczas projektowania bardziej skomplikowanych systemów. Jednym z podstawowych problemów jest brak możliwości uzyskania jednoznacznej informacji na temat orientacji urządzenia zarówno w pionie jak i poziomie. W ramach przykładu spróbujmy przeanalizować wykorzystanie akcelerometru do przelączania widoku na ekranie w zależności od jego orientacji. W przypadku gdy osi X lub Y są równoległe z wektorem grawitacji na podstawie informacji z akcelerometru można w bardzo prosty sposób ustalić w jakim położeniu znajduje się aktualnie urządzenie.



Rysunek 5.5. Pokrywanie się osi ekranu z kierunkiem wektora grawitacji pozwala na precyzyjne wyznaczenia orientacji urządzenia [10]

Natomiast w przypadku gdy osie ekranu ustawią się prostopadle do wektora grawitacji dane pozyskane z akcelerometru nie pozwalają na wyznaczenie jednoznacznej pozycji w jakiej znalazło się urządzenie [10]. Co więcej w przypadku aplikacji w których akcelerometr jest źródłem informacji o odchyleniu od pionu konieczne jest rozdzielenie zmian związanych wpływem grawitacji od szumów związanych z przyspieszeniem dynamicznym. Tak pozyskane informacje będą służyć jako punkt odniesienia pozwalającego na wyznaczenie kąta wychylenia. Głównym problemem jest fakt, że sygnał taki można wyróżnić jedynie w chwili gdy akcelerometr jest w stanie spoczynku, co w przypadku niektórych rodzajów zastosowań może dawać nieoczekiwane rezultaty i zachwiać stabilność pracy całego systemu.

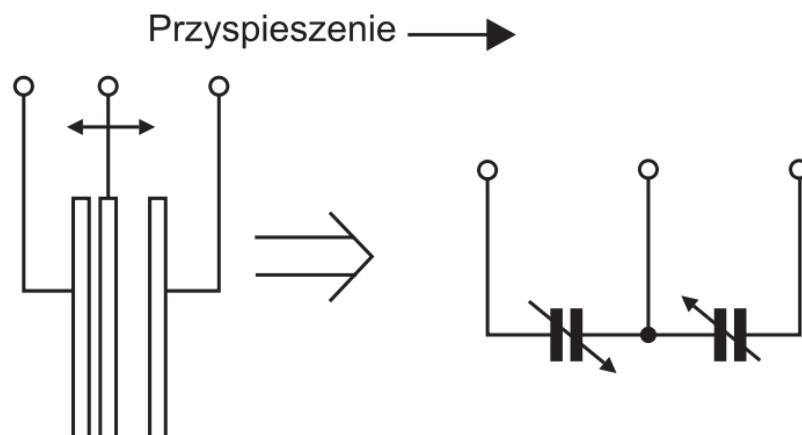
Jeżeli założymy, że zmiany odpowiadające przyspieszeniu dynamicznemu mają dużą częstotliwość to możemy przy użyciu filtra dolnoprzepustowego usunąć zakłócenia związane z drganiami wywołanymi przez użytkownika. Oczywistą implikacją powyższego założenia jest założenie, że zmiana kąta nachylenia względem pionu są wolne. W przeciwnym

wypadku informacja o zmianie odchylenia zostaną usunięte przez wspomniany filtr. Aby pominięcie wspomnianych ograniczeń stało się możliwe konieczne jest zastosowanie dodatkowych czujników takich jak np. żyroskop które dostarczą uzupełniających informacji pozwalających na stabilną implementację funkcjonalności.

5.2.1. Rodzaje akcelerometrów

Istnieje wiele różnych rodzajów akcelerometrów. Akcelerometry mechaniczne przypominają po części zachowanie pasażera w samochodzie który gwałtownie przyspiesza i zwalnia. Posiadają one element masy przyczepiony do elementu sprężynowego umieszczonego w całości w zewnętrznej obudowie. Kiedy taki akcelerometr zacznie przyspieszać obudowa zewnętrzna przemieści się podczas gdy punkt masy wychyli się w kierunku przeciwnym do przyspieszenia co spowoduje rozciągnięcie się sprężyny wprost proporcjonalne do siły która spowodowała przyspieszenie. Mierząc więc odległość na jaką nastąpiło wychylenie jesteśmy w stanie wyznaczyć wartość działającej siły, a co za tym idzie również wartość przyspieszenia. Opisana powyżej zasada pomiaru leży u podstaw działania sejsmografów które za pomocą masy dołączonej do elementu piśmiennego rejestrują siły występujące w chwili trzęsienia ziemi.

Alternatywnym podejściem jest wykorzystanie sygnałów elektrycznych i magnetycznych do pomiaru przyspieszenia. Wśród tego rodzaju akcelerometrów możemy wyróżnić następujące trzy typy: akcelerometr piezorezystywny, akcelerometr pojemościowy oraz akcelerometr piezoelektryczny. Istnieją również akcelerometry które dokonują pomiaru przyspieszenia w oparciu o efekt Halla.



Rysunek 5.6. Schemat ideowy akcelerometru pojemościowego dokonującego pomiaru wzduż jednej osi [15]

Przyspieszomierze piezorezystywne mają punkt masy dołączony do potencjometru który zwiększa i zmniejsza przepływ prądu w zależności od siły jaka oddziałuje na czujnik. Bardzo podobną zasadę działania wykazują akcelerometry pojemnościowe te jednak wykorzystują efekt zmiany pojemności kondensatorów zastosowanych w miejscu w którym poprzednio użyty został potencjometr. Ostatnim rodzajem akcelerometrów są urządzeniami bazującymi na piezoelektrycznych kryształach takich jak na przykład kwarc. W urządzeniach tych punkt masy, pod wpływem przyspieszenia, naciska na kryształ co powoduje, powstawanie napięcia które jest wykorzystywane do wykonywania pomiaru. Cechą charakterystyczną tego rodzaju czujników jest brak wskazań wartości przyspieszenia statycznego. [14]

5.2.2. Algorytm rozpoznawania kroków

Do poprawnego działania algorytmu zaprezentowanego w ramach noty aplikacyjnej [9] wymagane jest wykorzystanie akcelerometru trójosiowego. Jak już wspomniano wcześniej, dzięki zastosowaniu urządzenia tego typu jesteśmy w stanie wyeliminować problemy wynikające z konieczności uwzględnienia aktualnego wychylenia akcelerometru względem wektora grawitacji. W przypadku gdyby możliwe byłoby korzystanie tylko z pojedynczych osi poprawne działanie algorytmu byłoby zapewnione jedynie w ścisłe określonej pozycji co jest znaczącym utrudnieniem biorąc pod uwagę niehomogeniczny charakter ruchów wykonywanych podczas chodzenia. Aby wyeliminować wspomniany problem do ostatecznego rozwiązania brana jest jedynie wartość stanowiąca złożenie wartości przyspieszenia dla poszczególnych osi. Do wyznaczenia bezwzględnego przyspieszenia na podstawie danych z osi X, Y, Z użyte zostało równanie 5.1.

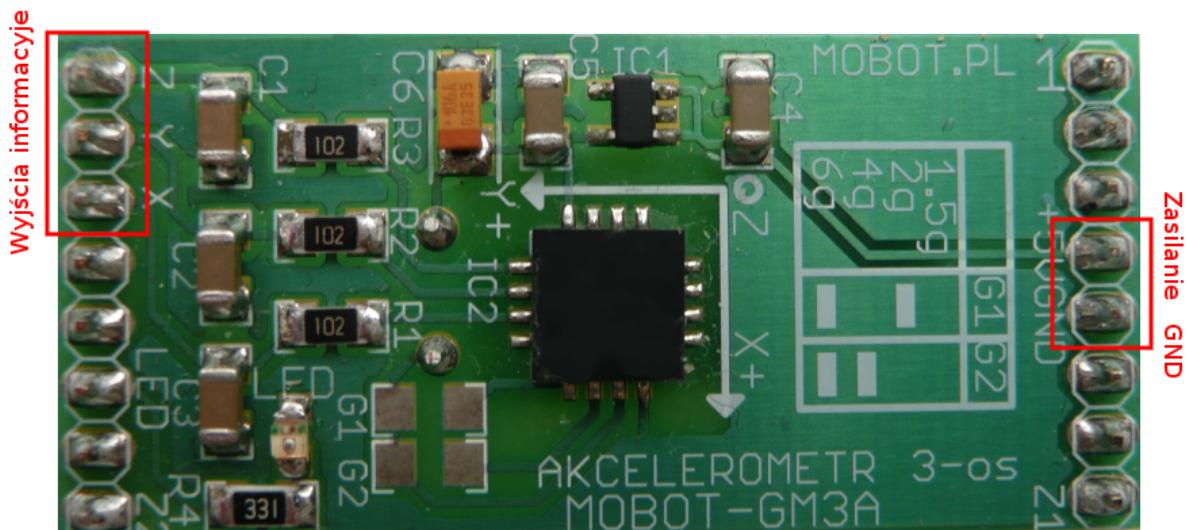
$$a_{xyz} = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (5.1)$$

Zaimplementowany w ramach pracy magisterskiej algorytm wykrywania i zliczania kroków bazuje na założeniu iż sumaryczna wartość przyspieszenia wykrywanego przez akcelerometr w trakcie chodzenia oscyluje wokół wartości 1G⁸. Podstawową zasadą działania algorytmu jest więc wykrywanie cykli w jakich następuje kolejno spadek wartości odczytywanego przyspieszenia, a następnie jego wzrost ponad wartość spoczynkową. Do poprawnego działania algorytmu konieczne jest dobranie nie tylko odpowiedniej stałej czasowej w której wykryty cykl będzie zliczany jako krok, ale również ustalenie progów przyspieszenia poniżej i powyżej wartości spoczynkowej które będą traktowane jako odpowiednio początek i koniec cyklu.

⁸ 1G - Jednokrotność przyspieszenia ziemskiego wynoszącego w przybliżeniu 9.81 $\frac{m}{s^2}$

5.2.3. Implementacja algorytmu

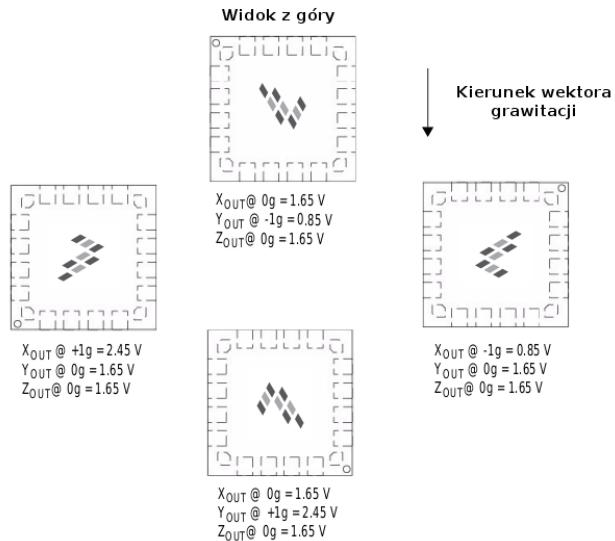
Do praktycznej implementacji algorytmu wykrywania kroków wykorzystany został moduł akcelerometru trójosiowego firmy Freescale Semiconductor. Czujnik MMA7260 [16] został zainstalowany w robocie jako element modułu MOBOT-GM3A. Moduł akcelerometru jest to jedyny moduł w całości dostarczony przez zewnętrznego dostawcę, a wybór takiej strategii podyktowany był jedynie względami ekonomicznymi i dostępnością tego typu urządzeń na polskim rynku elektronicznym. Na ilustracji zamieszczone zostało zdjęcie gotowego układu wraz z zaznaczonymi wyprowadzeniami.



Rysunek 5.7. Moduł akcelerometru trójosiowego z czujnikiem przyspieszenia MMA7260

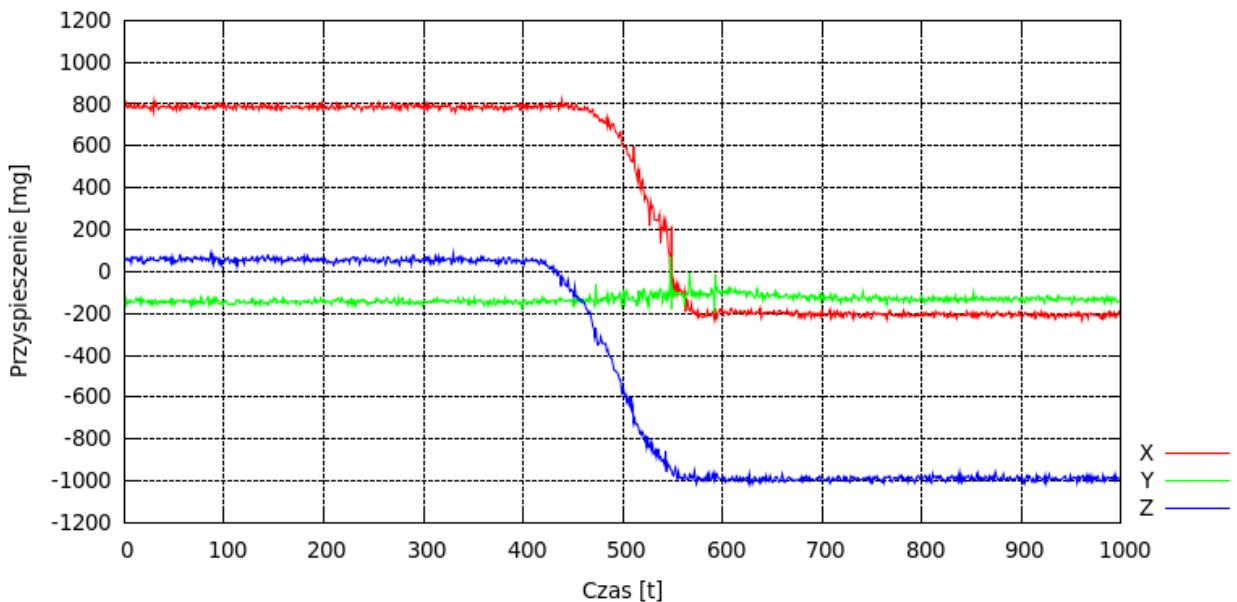
Wykorzystany czujnik przyspieszenia oferuje cztery zakresy czułości których zmiana odbywa się poprzez konfigurację odpowiednich zworek w module MOBOT - GM3A. W ramach implementacji wybrany został zakres czułości od $-1.5g$ do $1.5g$ gdyż udziela on najdokładniejszej informacji na temat przyspieszenia w przedziale w jakim mieści się przyspieszenia związane z chodzeniem. W wybranym trybie czułości akcelerometr wykazuje czułość rzędu 800mV/g , co przy wartości przyspieszenia równej zero daje środek przedziału czułości na poziomie 1.65V . Ze względu na analogową charakterystykę sygnału wyjściowego z modułu akcelerometru konieczne jest wykorzystanie konwertera analogowo-cyfrowego w celu odczytania danych pomiarowych otrzymywanych na wyjściach urządzenia. Szczegółowy opis działania wbudowanego w robota ADC można znaleźć w rozdziale poświęconym dalmierzom IR.

Przed rozpoczęciem praktycznej implementacji algorytmu wykrywania kroków konieczne okazało się przeprowadzenie kalibracji modułu, gdyż już wstępne testy czujnika zwróciły spore różnice pomiędzy wartościami podanymi w napisie katalogowej, a stanem



Rysunek 5.8. Wartości przyspieszenia statycznego dla układu MMA7260 dostarczone w ramach specyfikacji technicznej urządzenia [16]

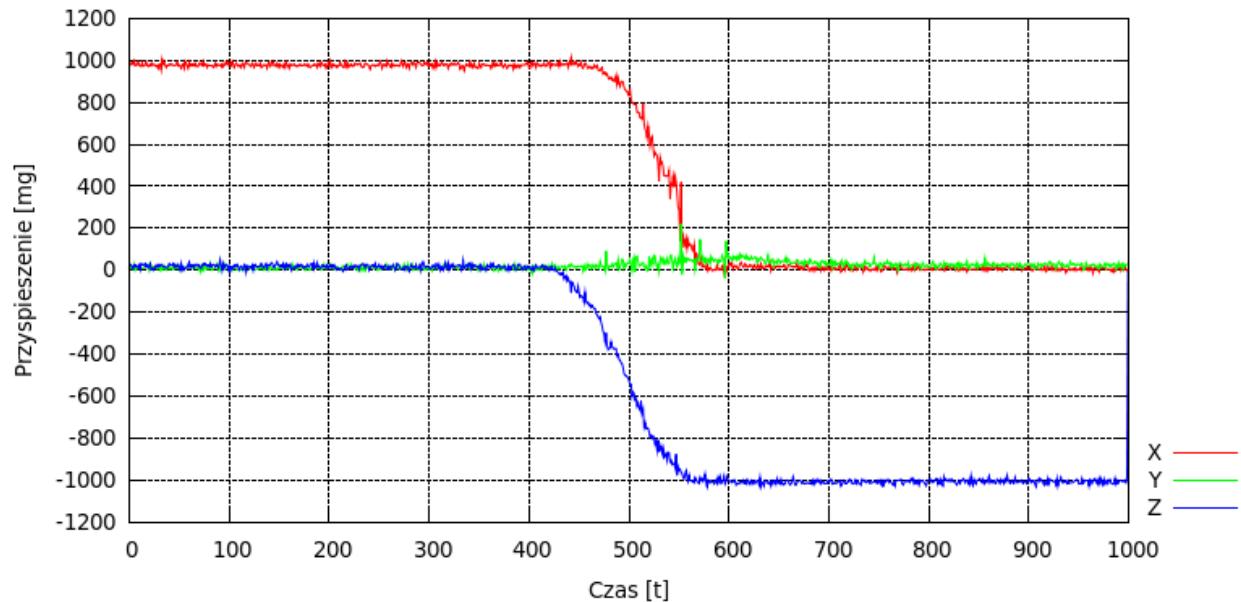
zmierzonym na wyjściach akcelerometru. Zgodnie z rysunkiem znalezionym w dokumentacji udostępnionej przez producenta czułość oraz zakres napięć na poszczególnych osiach powinien być w stałych, jednakowych przedziałach. Niestety jak się okazało poszczególne osie miały bardzo znaczące różnice w zakresie swojej czułości co mogłoby bardzo niekorzystnie wpływać na stabilność działania zaproponowanego algorytmu zliczania kroków.



Rysunek 5.9. Wartości przyspieszenia odczytywane na wyjściach akcelerometru przed przeprowadzeniem kalibracji

Wspomniany proces kalibracji polegał na ustawieniu czujnika na płaskiej, poziomej powierzchni oraz odczytaniu wartości przyspieszenia statycznego dla każdej z osi z osobna.

Procedurę taką powtarzano wielokrotnie zmieniając orientację modułu względem wektora grawitacji. Po zebraniu wszystkich danych pomiarowych dla każdej osi z osobna została wyznaczona rzeczywista czułość, a następnie w module odpowiedzialnym za akwycję danych o przyspieszeniu zostały zaprogramowane funkcje korygujące otrzymane dane wejściowe o wartości wyliczone podczas kalibracji urządzenia. Na poniższych wykresach znajdują się informacje o stanie wyjść przed i po przeprowadzeniu kalibracji.



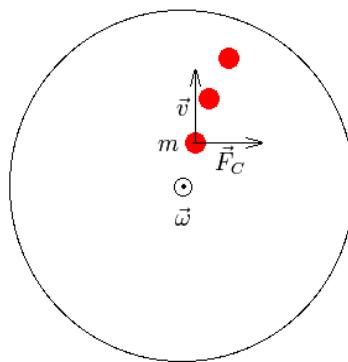
Rysunek 5.10. Wartości odczytywane na wyjściach akcelerometru po przeprowadzeniu kalibracji

5.3. Żyroskop

Do budowy inercjalnej jednostki pomiarowej został użyty żyroskop trójosiowy wykonany w technologii MEMS. Żyroskopy wyprodukowane w tej technologii są wykorzystywane np.: w stabilizatorach obrazu do kamer lub kontrolerach do gier wideo. Ich główną zaletą jest mały rozmiar. W poniższym podrozdziale zostanie przedstawiona zasada działania takiego żyroskopu oraz sposób wykorzystania go w robocie mobilnym.

5.3.1. Zasada działania

Żyroskopy MEMS korzystają z pozornej siły Coriolis'a do pomiaru prędkości kątowej z jaką obraca się ciało. Założmy, że ciało o masie m porusza się z prędkością \vec{v} oraz układ

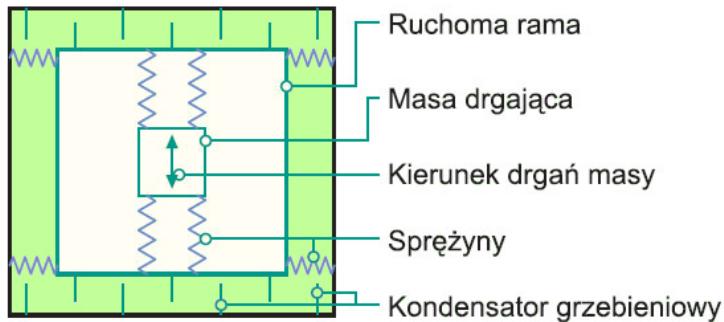


Rysunek 5.11. Siła Coriolis'a

w którym przemieszcza się to ciało obraca się z prędkością kątową $\vec{\omega}$. W takich warunkach ciało o którym mowa zostanie odchylone od kierunku przemieszczania się wyznaczonego przez wektor prędkości \vec{v} . Odchylenie to będzie spowodowane siłą Coriolis'a \vec{F}_c którą można opisać przy pomocy wzoru 5.2. Żyroskopy MEMS wykorzystują to zjawisko określając przemieszczenie drgającego ciała, które jest tak naprawdę jedną z okładek kondensatora, jako zmianę pojemności.

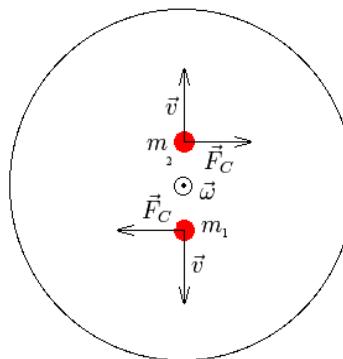
$$\vec{F}_c = -2m (\vec{\omega} \times \vec{v}) \quad (5.2)$$

Żyroskopy tego typu posiadają dwa oscylujące miniaturowe elementy które poruszają się w przeciwnych względem siebie kierunkach (rys. 5.12). Jeżeli urządzenie do którego przy mocowany jest żyroskop zacznie się obracać, spowoduje to wychylenie się oscylujących elementów w przeciwnych kierunkach(rys. 5.13). Wychylenie to z kolei powoduje zmianę pojemności, a różnica pomiędzy pojemnościami zmierzonymi przy pomocy obydwu ruchomych elementów jest proporcjonalna do prędkości kątowej. W ten sposób zmierzona



Rysunek 5.12. Budowa elementu pomiarowego żyroskopu [11]

szybkość obrotu jest następnie reprezentowana jako wynik analogowy (napięcie) lub cyfrowy (wartość liczbową).



Rysunek 5.13. Dwie oscylujące masy odchylane przez siłę Coriolis'a

Wynik działania żyroskopu MEMS jest obojętny na zmianę przyspieszenia liniowego, w szczególności na przyspieszenie ziemskie. Przyspieszenie liniowe wywoła wychylenie się jednocześnie obydwu elementów oscylujących w tym samym kierunku. W ten sposób nie będzie wykryta żadna różnica pojemności. Prędkość kątowa wskazywana przez żyroskop nie ulegnie zmianie. Dzięki temu żyroskopy MEMS są odporne na wstrząsy, uderzenia oraz wibracje.

Podstawowe wielkości charakteryzujące żyroskop:

- zakres [dps^9] – definiuje wartość maksymalną i minimalną jaką żyroskop jest w stanie zmierzyć. Często żyroskop ma kilka zakresów z których możemy wybrać jeden odpowiadający naszym potrzebom.
- czułość [$\frac{mdps}{digit}$] – wielkość określająca wartość minimalną jaką żyroskop może zmierzyć

⁹ dps - stopień na sekundę

- zmiana czułości względem temperatury [%] – określa jak bardzo pomiary urządzenia są podatne na zmianę temperatury
- zakres poziomu zero [dps] – określa zakres w jakim pomiary mogą się wachać w chwili gdy żyroskop pozostaje w spoczynku
- zmiana zakresu poziomu zero względem temperatury [$\frac{dps}{^{\circ}C}$] – definiuje zmiany pomiarów żyroskopu pozostającego w spoczynku względem temperatury
- nieliniowość [%FS] – parametr określa maksymalne procentowe odchylenie wartości na wyjściu żyroskopu od dopasowanej do nich linii prostej
- przepustowość [Hz] – definiuje częstotliwość z jaką są wykonywane pomiary

5.3.2. Kalibracja i odczytywanie wyników

Żyroskopy są zazwyczaj fabrycznie testowane i kalibrowane pod kątem zakresów pomiarów poziomu zerowego oraz czułości. Jednak po umieszczeniu elementu ma płytce PCB zostaje on poddany naprężeniom przez co może być potrzebna dodatkowa kalibracja układu.

Wartości wyjściowe żyroskopu można przedstawić za pomocą równania:

$$\omega_t = \omega_m - \omega_0 \quad (5.3)$$

gdzie:

- ω_t – rzeczywista wartość prędkości kątowej,
- ω_m – pomiar z żyroskopu,
- ω_0 – wartość zerowa reprezentująca brak ruchu,

W celu kompensacji niestabilności żyroskopu należy pozostawić urządzenie nieruchome i wykonać ok 1000 pomiarów. Następnie konieczne jest obliczenie wartości średniej z wykonanych pomiarów. W ten sposób określmy średnie wychylenie od wartości zerowej żyroskopu, czyli nasze ω_0 .

Podczas używania żyroskopu do pomiaru bardzo małych kątów należy jeszcze wziąć pod uwagę minimalne odchylenia wartości w zależności od zmiany temperatury.

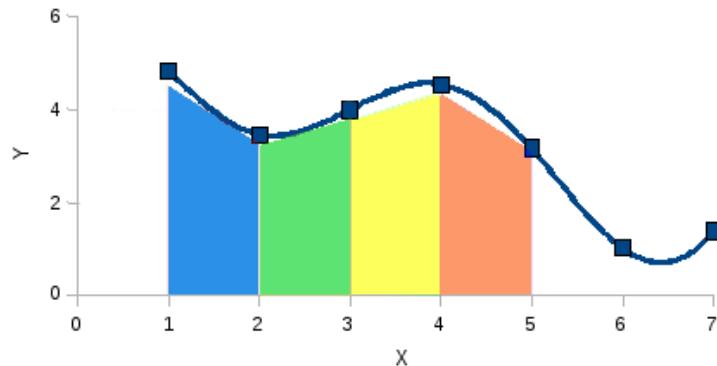
Prędkość kątową opisujemy wzorem:

$$\omega = \frac{d\varphi}{dt} \quad (5.4)$$

Wzór na kąt o jaki zostało obrócone ciało poruszające się z prędkością kątową ω wyprowadzamy całkując obustronnie równanie 5.4.

$$\varphi = \int \omega dt \quad (5.5)$$

Do obliczenia kąta można wykorzystać jedną z numerycznych metod całkowania, np. metodę trapezów. Metoda ta polega na podzieleniu pola pod wykresem całkowanej funkcji na wąskie trapezy, a następnie zsumowanie pól tych trapezów.



Rysunek 5.14. Całkowanie przy pomocy metody trapezów¹⁰

Po wykorzystaniu metody trapezów, wzór na kąt o jaki żyroskop został obrócony w czasie pomiędzy dwoma pomiarami, można zapisać następująco:

$$\varphi = C_f \cdot \frac{\omega_{i-1} + \omega_i}{2} \cdot \Delta t \quad (5.6)$$

gdzie:

C_f – współczynnik korygujący,

Δt – czas jaki upłynął pomiędzy dwoma pomiarami,

ω_{i-1} – szybkość kątowa odczytana z poprzedniego pomiaru,

ω_i – szybkość kątowa odczytana z obecnego pomiaru,

W taki sposób otrzymujemy wzór na kąt o jaki ciało zostało obrócone w czasie Δt . Kąt całkowity pomiędzy orientacją początkową a orientacją końcową ciała jest sumą

¹⁰ źródło: <http://mateusz-lach.blogspot.com/2010/09/cakowanie-numeryczne.html>

wszystkich wykonanych pomiarów kątów.

$$\varphi_c = \varphi_0 + \sum_{i=1}^n \varphi_i \quad (5.7)$$

gdzie:

φ_0 – kąt początkowy,

φ_i – wartość kąta z pojedynczego pomiaru

Do otrzymania jak najlepszych wyników konieczne jest określenie współczynnika korygującego pomiar żyroskopu. W tym celu porównujemy wyniki pomiarów żyroskopu z pomiarami innych przyrządów np. magnetometru wykorzystując wzór:

$$C_f = \frac{\varphi_o}{\varphi_g} \quad (5.8)$$

gdzie:

C_f – współczynnik korygujący,

φ_o – kąt obliczony przy pomocy urządzenia zewnętrznego,

φ_g – kąt wyznaczony przez żyroskop,

Niestety z powodu ciągle nakładających się błędów całkowania oraz niedoskonałości samego żyroskopu, wskazywany kąt odchylenia od ułożenia początkowego będzie się od-dalał od wartości rzeczywistej wraz z upływem czasu. W przypadku tej pracy dyplomowej nie jest jednak potrzebna bardzo wysoka stabilność i precyzaja wartości mierzonych przez żyroskop.

5.3.3. Opis zastosowanego elementu i budowy modułu

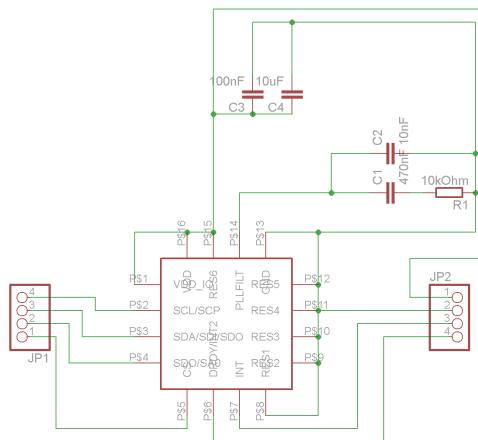
W robocie został użyty ultra-stabilny trójosiowy żyroskop cyfrowy L3G4200D firmy STMicroelectronics. Jego główną zaletą jest bardzo dobra jakość pomiarów osiągnięta dzięki zastosowaniu innowacyjnej struktury mechanicznej. Zazwyczaj urządzenia tego typu stosują dwie lub trzy struktury odpowiedzialne za wykonywanie pomiarów. Żyroskop firmy STMicroelectronics natomiast posiada jeden taki element odpowiedzialny za pomiar na wszystkich osiach. Takie rozwiązanie eliminuje zakłócenia pomiędzy osiami, powodowane przez same elementy pomiarowe.

Zgodnie z notą katalogową [17] żyroskop zamontowany w robocie posiada charakterystykę mechaniczną taką jak w tabeli 5.1:

Tabela 5.1. Charakterystyka mechaniczna żyroskopu L3G4200D dla napięcia zasilania 3.0V i temperatury pracy 25°C

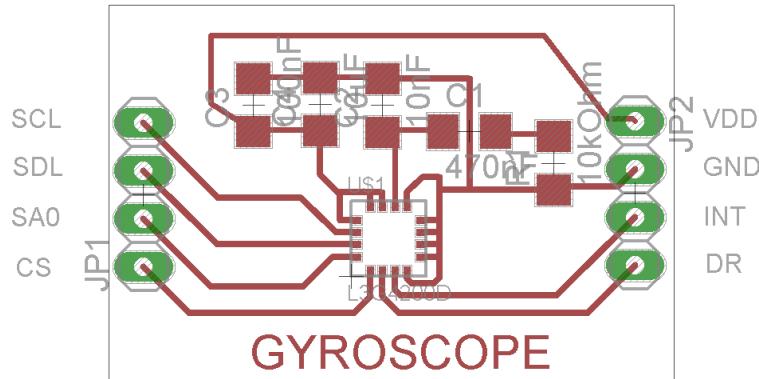
Symbol	Parametr	Warunki testowe	Wartość	Jednostka
FS	Skala		±250	
			±500	dps
			±2000	
So	Czułość	$FS = 250\text{dps}$	8.75	
		$FS = 500\text{dps}$	17.50	$\frac{\text{mdps}}{\text{digit}}$
		$FS = 2000\text{dps}$	70	
SoDr	Zmiana czułości względem temperatury	-40°C do $+85^{\circ}\text{C}$	±2	%
DVoff	Zakres poziomu zero	$FS = 250\text{dps}$	±10	
		$FS = 500\text{dps}$	±15	dps
		$FS = 2000\text{dps}$	±75	
OffDr	Zmiana zakresu poziomu zero względem temperatury	$FS = 250\text{dps}$	±0.03	dps
		$FS = 2000\text{dps}$	±0.04	
NL	Nieliniowość		±0.2	% FS
Rn	Współczynnik szumów	$BW = 50\text{Hz}$	±0.03	$\frac{\text{dps}}{\sqrt{\text{Hz}}}$
ODR	Częstotliwość odświeżania danych na wyjściu		100/200	Hz
			400/800	
Top	Zakres temperatur operacyjnych		-40°C do $+85^{\circ}\text{C}$	°C

Do żyroskopu została zaprojektowana płytka podłączeniowa PCB w programie Eagle.



Rysunek 5.15. Schemat płytki PCB modułu żyroskopu

Schemat przedstawiony na rysunku 5.15 został stworzony na podstawie noty katalogowej.

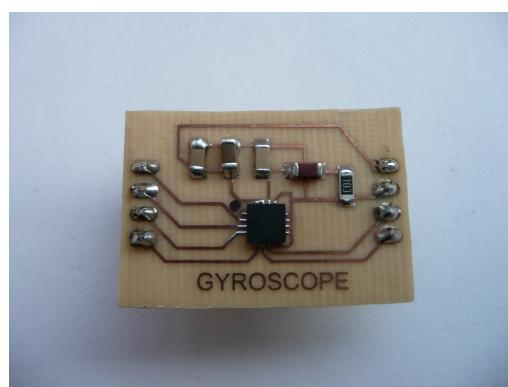


Rysunek 5.16. Płytki PCB modułu żyroskopu trójosiowego

Płytki modułu zostały zaprojektowane z wykorzystaniem elementów SMD w obudowie 1206. Rozmiar obudowy tych elementów został wymuszony przez brak kondensatorów ceramicznych o pojemności $10\mu F$ w obudowie mniejszej. Cały proces tworzenia płytka modułu żyroskopu był przeprowadzony metodami domowymi ze względu na niski koszt oraz brak konieczności czekania kilku tygodni na wykonanie takiej płytki przez specjalistyczną firmę. Opis tworzenia płytka można znaleźć w dodatku B.

Tabela 5.2. Opis wyprowadzeń płytka modułowej żyroskopu

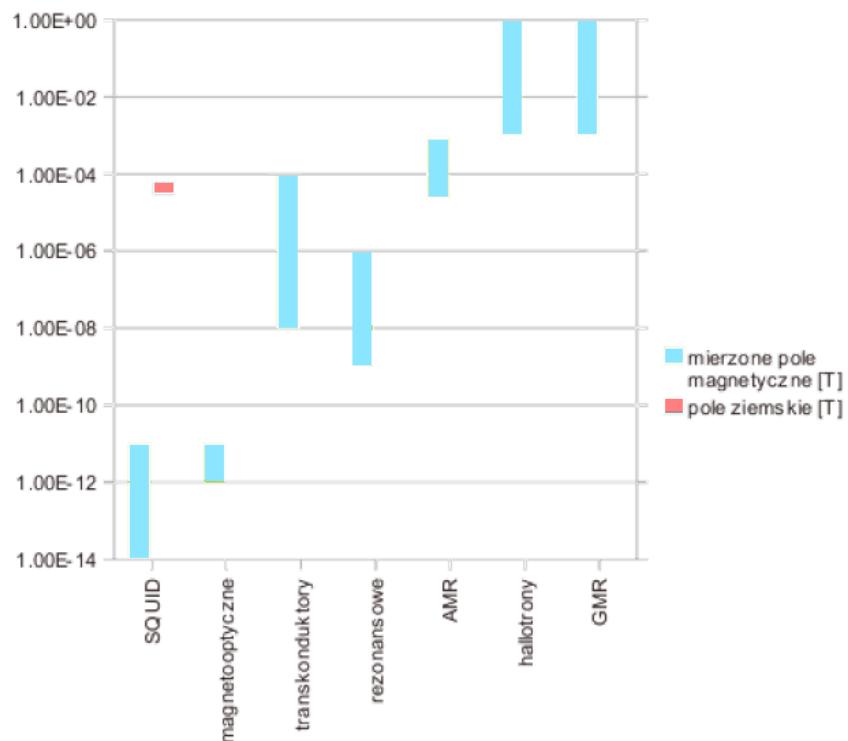
Nazwa	Typ	Opis
VDD	Zasilanie	Napięcie zasilania 3.3V
GND	Zasilanie	Masa
INT	Wyjście	Konfigurowalny pin przerywania
DR	Wyjście	Pin informujący o nowych danych do pobrania
SCL	Wejście	Zegar sterujący I^2C
SDL	We / Wy	Szyna do przesyłania danych I^2C
SA0	Wejście	Najmniej znaczący bit adresu urządzenia
CS	Wejście	Przełącznik trybu działania I^2C lub SPI



Rysunek 5.17. Gotowy moduł żyroskopu

5.4. Magnetometr

Jednym z elementów większości Inercjalnych Systemów Nawigacyjnych jest magnetometr. Wykorzystuje się go do określenia kierunku w którym porusza się dane ciało. Typów czujników mierzących pole magnetyczne jest wiele. Dla INS istotny jest jedynie czujnik, będący w stanie wykonać pomiary w zakresie indukcji pola magnetycznego ziemi, która wynosi od $30\mu T$ do $60\mu T$.



Rysunek 5.18. Zakresy czujników pola magnetycznego różnego typu [4]

Na rysunku 5.18 przedstawione są zakresy pomiarowe czujników pola magnetycznego różnego typu. Jak widać tylko dwa rodzaje magnetometrów z wyżej wymienionych działają w przedziale obejmującym zakres indukcji pola magnetycznego ziemi. Są to magnetometry transdukторowe oraz AMR¹¹. Z powodu małych rozmiarów, przystępnej ceny i dostępności, w tej pracy wykorzystany został magnetometr AMR.

¹¹ AMR - Anisotropic Magneto Resistance, anizotropowy magnetoopór

5.4.1. Zasada działania

Czujnik AMR wykorzystuje zjawisko anizotropowego magnetooporu w celu pomiaru indukcji pola magnetycznego. Zjawisko to objawia się zmianą rezystancji materiału pod wpływem zmiany orientacji działającego pola magnetycznego względem kierunku prądu płynącego przez ten materiał.

Czujniki tego typu charakteryzują się wysoką dokładnością i szybkością działania jednocześnie pobierając mniejszy prąd w stosunku do alternatywnych technologii.

5.4.2. Opis elementu

W celu stworzenia kompasu został wykorzystany magnetometr dwuosiowy MMC2120 firmy Memsic. Dane z magnetometru są przesyłane przy pomocy interfejsu I^2C w postaci dwóch 12-bitowych liczb. Jedna z nich określa wartość indukcji pola magnetycznego zmierzonego na osi X druga zaś na osi Y .

Tabela 5.3. Parametry magnetometru MMC2120

Parametr	Warunki testowe	Wartość	Jednostka
Zakres pomiarów	Wypadkowe pole magnetyczne	od -2 do 2	gauss
Nieliniowość	± 1 gauss	0.1	%FS
	± 2 gauss	0.5	%FS
Dokładność		od ± 2 do ± 5	deg
Czułość		od $\frac{1}{461}$ do $\frac{1}{563}$	gauss
Wartość na wyjściu przy braku pola magnetycznego		od -0.2 do 0.2 2048	gauss wartość zerowa

Dla poprawnego działania kompasu niezbędna jest kalibracja magnetometru. Przeprowadzamy ją poprzez obracanie magnetometru we wszystkich możliwych stopniach swobody jednocześnie zapisując wartość maksymalną i minimalną wskazywaną zarówno na osi X jak i Y . Dzięki tym wartośćmi będziemy w stanie wyznaczyć, niezależnie dla obydwu osi, przesunięcie wartości zerowej oraz ich czułość. **TODO: WYKRES POMIARU Z NIESKALIBROWANEGO MAGNETOMETRU!** Kalibrację należy wykonać po umieszczeniu modułu w robocie w celu wyeliminowania zakłóceń spowodowanych metalowymi elementami robota. MMC2120 jako wartość zerową przyjmuje liczbę 2048. Jest to środek przedziału liczb składającego się z 12 bitów. Wartości poniżej tej liczby są uznawane za ujemne, a wartości powyżej za dodatnie.

TODO: WYKRES POMIARU ZE SKALIBROWANEGO MAGNETOMETRU!

Niestety z powodu niedokładności samego układu, a także negatywnego wpływu metalowych elementów konstrukcji robota i modułu, wartość zerowa magnetometru może ulec przemieszczeniu. Dlatego też musimy obliczyć odchylenie o jaki należy przesunąć każdy pomiar w celu otrzymania prawidłowego wyniku. Wspomniane odchylenie ten wyznaczamy dla każdej osi jako wartość środkową pomiędzy maksymalnym a minimalnym odczytem podczas kalibracji.

TODO: WYKRES POMIARU ZE SKALIBROWANEGO I NIESKALIBROWANEGO MAGNETOMETRU Z ZAZNACZONYM OFFSETEM!

Dopiero po właściwym skalibrowaniu magnetometru jesteśmy w stanie wykorzystać go jako kompas. W celu wyznaczenia azymutu, czyli kąta pomiędzy kierunkiem wskazywanym przez oś X , **TODO: SPRAWDZIĆ KTÓRĄ OŚ FAKTYCZNIE!** a kierunkiem północnym, należy skorzystać z prostego wyrażenia trygonometrycznego przedstawionego na wzorze 5.9.

$$\alpha = \arctg \left(\frac{y}{x} \right) \quad (5.9)$$

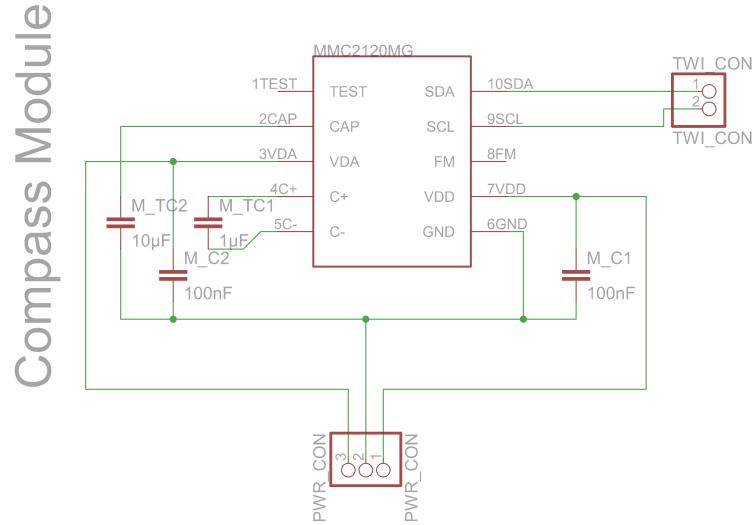
Podczas odczytu wskazań kompasu istotne jest także wzięcie pod uwagę takich czynników jak odchylenie osi X i Y czujnika od kierunku równoległego do kierunku pola magnetycznego ziemi. **TODO: WYKRES POMIARU ZE SKALIBROWANEGO MAGNETOMETRU ALE NIERÓWNOLEGLE DO POLA!**

5.4.3. Budowa modułu

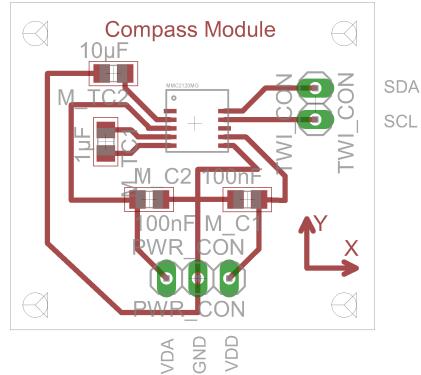
Płytką modułu kompasu została zaprojektowana przy pomocy darmowej wersji programu Eagle na podstawie danych zawartych w nacie katalogowej czujnika MMC2120. Na rysunkach 5.20 oraz 5.19 przedstawiono schemat oraz layout płytka modułu kompasu.

Tabela 5.4. Opis wyprowadzeń modułu kompasu

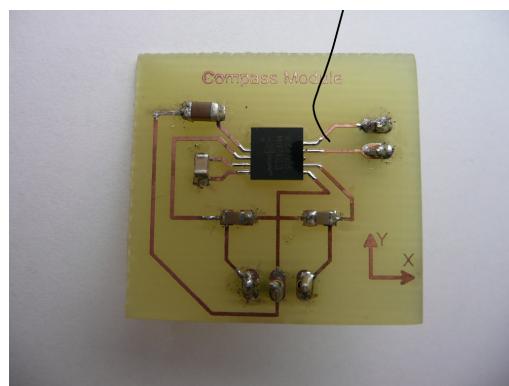
Nazwa	Typ	Opis
VDA	Zasilanie	Napięcie zasilania 3.3V
GND	Zasilanie	Masa
SCL	Wejście	Zegar sterujący I^2C
SDA	We / Wy	Szyna do przesyłania danych I^2C
VDD	Zasilanie	Napięcie zasilania szyny danych 3.3V



Rysunek 5.19. Schemat modułu kompasu



Rysunek 5.20. Projekt płytka kompasu elektronicznego



Rysunek 5.21. Gotowy moduł kompasu

5.4.4. Nieudana próba zastosowania

Docelowo moduł kompasu miał być wykorzystywany jako element Inercjalnego Systemu Nawigacji w celu określania kierunku w którym robot się przemieszcza. Zasada

działania miała być prosta. Robot miał zapamiętywać aktualny azymut z jakim się poruszał będąc na rękach operatora. Następnie po postawieniu na podłożu miał odtwarzać tor ruchu obracając wszystkie zapamiętane azymuty o 180° .



Rysunek 5.22. Busola w położeniu początkowym



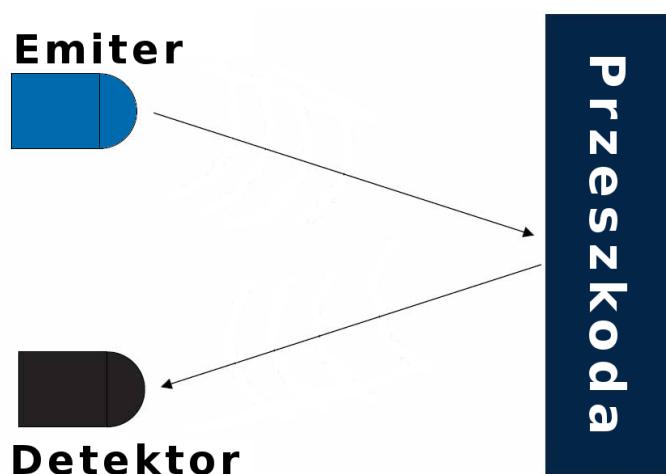
Rysunek 5.23. Busola przesunięta równolegle o 25 cm,

Niestety wykonane testy pokazały, iż nie jest możliwe wykorzystanie kompasu w celu określenia azymutu robota poruszającego się po podłodze budynku. Jest to spowodowane stosowaniem metalowych prętów zbrojeniowych w stropach budynków, które zakłócają pracę kompasu. Testy zostały wykonane zarówno przy pomocy modułu elektronicznego kompasu jak i tradycyjnej wojskowej busoli. Rysunki 5.22 oraz 5.23 przedstawiają odchylenie azymutów wyznaczonych na podłodze budynku. Busola przedstawiona na zdjęciach została przesunięta równolegle o 25 cm w prawo. Wskazywany kierunek północy zmienił się o ponad 90° . Jak widać przy tak dużych wahaniach nie jest możliwe wykorzystanie magnetometru jako elementu jednoznacznie określającego kierunek w którym robot ma się poruszać. Po wielu nieudanych próbach wyeliminowania zakłóceń moduł kompasu został zastąpiony modułem żyroskopu.

5.5. Czujnik odległości

Bardzo popularnym rozwiążaniem problemu omijania przeszkód jest wykorzystanie czujników ultradźwiękowych. Dane pomiarowe pobrane w pierwszym kroku z sonaru wykorzystywane są do stworzenia lokalnej reprezentacji otoczenia pozwalającej na późniejsze sterowanie robotem [8]. Dla tak zdefiniowanego problemu możemy wyróżnić dwa rodzaje reprezentacji środowiska. Pierwszy rodzaj reprezentacji opiera się o siatkę dzięki której środowisko podzielone jest na skończoną liczbę komórek które mogą posiadać stan świadczący o pustej przestrzeni lub obecności przeszkody w danej komórce. Drugim sposobem reprezentacji jest przedstawienie otoczenia za pomocą zbioru właściwości takich jak punkty, linie oraz płaszczyzny. Wybór sposobu reprezentacji jest z reguły podyktowany rodzajem problemu jaki stawiany jest przed robotem mobilnym.

Innym rodzajem czujników bardzo dobrze sprawdzających się w rozwiązywaniu problemu omijania przeszkód są dalmierze oparte o czujnik podczerwieni. Często zdarza się, iż dalmierze IR¹² są wybierane ze względu na ich bardzo krótki czas odpowiedzi w porównaniu do czujników ultradźwiękowych. Niestety jakość pomiaru w przypadku czujników podczerwieni jest w dużym stopniu zależna od współczynnika odbicia powierzchni przeszkody, jak również odległości od przeszkody oraz położenia nadajnika i odbiornika w stosunku do powierzchni przeszkody. Zdarza się więc, że brak precyzyjnych danych o lokalizacji przeszkody i jej właściwościach wyklucza dalmierze IR z niektórych zastosowań. Jednakże szeroka dostępność tego typu urządzeń oraz łatwość ich wykorzystania spowodowała, że są one najczęściej stosowane w robotach zajmujących się podążaniem za ścianą czy omijaniem i liczeniem przeszkód.



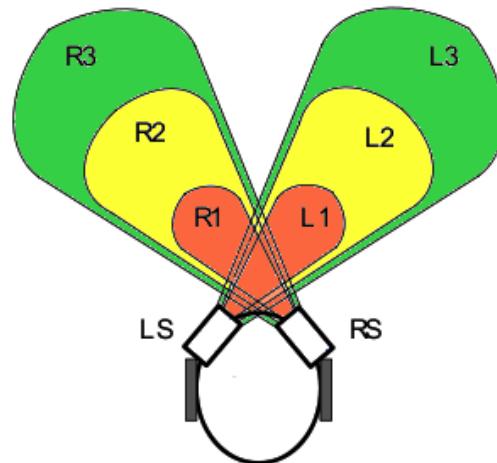
Rysunek 5.24. Zasada działania dalmierza wykorzystującego podczerwień

¹² IR (ang. infrared) - skrótowe oznaczenie urządzeń wykorzystujących podczerwień do swojego działania

Typowy dalmierz IR składa się z dwóch elementów. Pierwszym z nich jest dioda emitująca promieniowanie podczerwone, natomiast drugim elementem jest detektor najczęściej fotodioda lub fototranzystor. Zasada działania tak zbudowanego czujnika odległości opiera się o zjawisko odbicia. Światło wyemitowane przez emiter odbija się od przeszkody i trafia do detektora. Niestety pomiar tego rodzaju jest niezwykle wrażliwy na zmiany w oświetleniu oraz rodzaj i kształt napotkanej przeszkody.

5.5.1. Algorytm omijania przeszkód

Algorytm sterujący ruchami robota bazuje w całości na danych pomiarowych otrzymanych z czujników robota. Opisane podejście jest adaptacją metodologii opisanej w ramach pracy „Obstacles Avoidance Method for an Autonomous Mobile Robot” [8]. Zaproponowane rozwiązanie bazuje na prostej maszynie stanów w której przejścia pomiędzy kolejnymi stanami odbywają się poprzez zmianę poziomów na czujnikach odległości. Autorzy proponują podłączenie do robota dwóch dalmierzy jednego po lewej (LS), a drugiego po prawej stronie (RS) robota w taki sposób aby przed robotem nie było martwego punktu. Sposób montażu czujników został przedstawiony na rysunku poniżej.



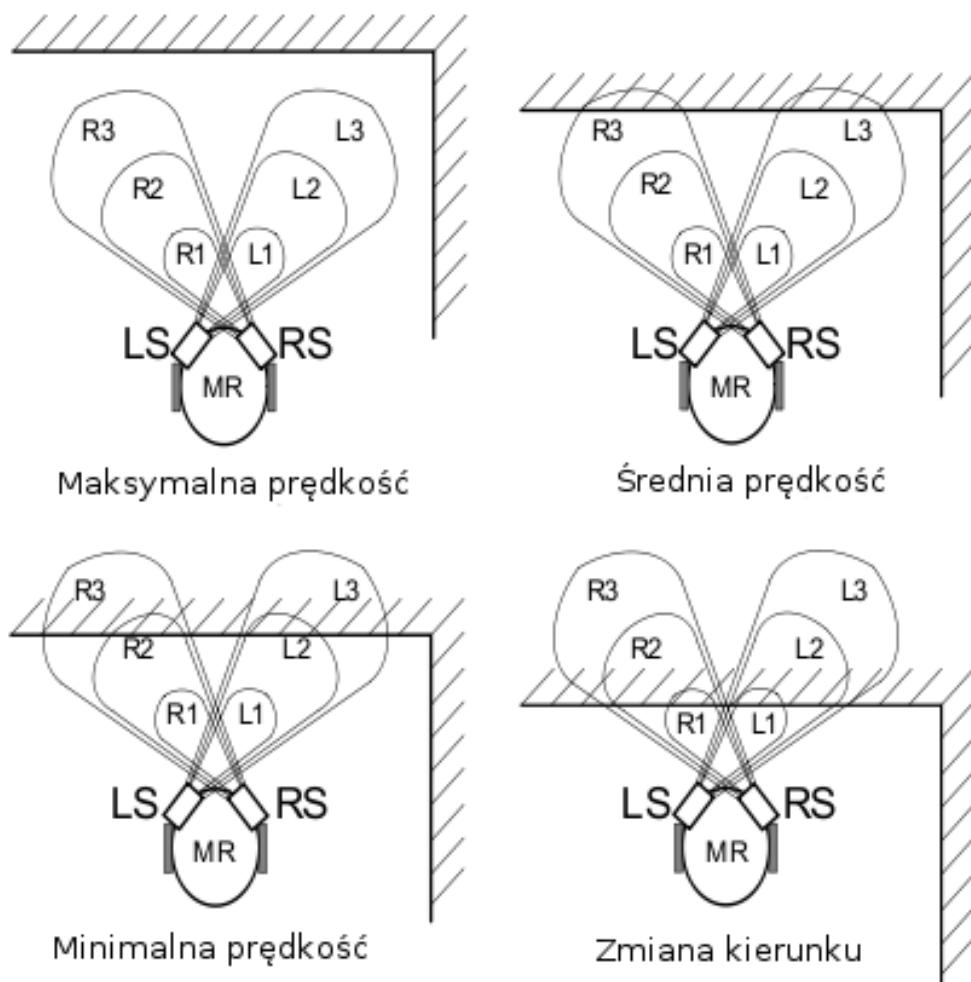
Rysunek 5.25. Sposób montażu czujników podczerwieni na robocie [8]

Każdy z czujników składa się z nadajnika i odbiornika podczerwieni, a ich przedział czułości, zgodnie z zaleceniami autorów, podzielony został na trzy poziomy oznaczone symbolami L1, L2, L3 dla czujnika lewego i odpowiednio R1, R2, R3 dla czujnika prawego. Istotne jest aby kolejne poziomy ułożone były w porządku rosnącym, a przerwa pomiędzy kolejnymi była na tyle szeroka, aby zminimalizować wpływ niedokładności pomiaru odległości. W chwili gdy robot znajduje się w ruchu na każdym z czujników jesteśmy w stanie odczytać wartość odległości od przeszkody. Dzięki takiej informacji możemy jednoznacznie

kontrolować zachowanie robota w zależności od poziomu jaki w danej chwili ustalił się na poszczególnych dalmierzach.

Opierając się na takich założeniach, jeżeli oba czujniki nie wykrywają na swojej drodze przeszkody to robot porusza się z maksymalną możliwą prędkością. Gdy jeden z czujników wykryje obecność przeszkody na poziomie L3 lub/i R3 prędkość poruszania się robota zostanie zmniejszona o połowę. Gdy przeszkoda znajdzie się w odległości poziomu L2 lub/i R2 prędkość z jaką porusza się robot zostanie ustalana na 1/4 prędkości maksymalnej. Jeżeli natomiast przeszkoda zostanie wykryta na poziomie L1 lub/i R1 robot skruci w prawo lub lewo w zależności od położenia przeszkody i wyznaczonego celu.

Na poniższym rysunku zaprezentowane zostały cztery najbardziej typowe ze wszystkich z możliwych sytuacji oraz definicja manewru jaki robot podejmie, aby skutecznie ominąć napotkaną przeszkodę.



Rysunek 5.26. Przykładowe sytuacje i definicja reakcji robota

Uogólniając zasadę działania algorytmu, robot dostosowuje swoją prędkość na podstawie aktualnej odległości od przeszkody odczytanej z czujników odległości. W chwili gdy robot znajdzie się bezpośrednio przed przeszkodą skręci on w prawo lub lewo aby ją ominąć. Kąt o jaki wykonany zostanie zakręt zależy od tego na którym czujniku przeszkoda została wykryta oraz aktualnego celu do którego robot zmierza. Szczegółowy opis poszczególnych możliwych sytuacji oraz rodzaj akcji podejmowanej przez robota jest przedstawiony w tabeli na następnej stronie.

Podane w tabeli wartości mają charakter orientacyjny i w gdy robot ma wyznaczony cel do którego zmierza kierunki skrętu mogą w niektórych przypadkach ulec zmianie. Mogą również pojawić się dodatkowe zmiany kierunku jazdy umożliwiające powrócenie na ścieżkę do wyznaczonego punktu docelowego.

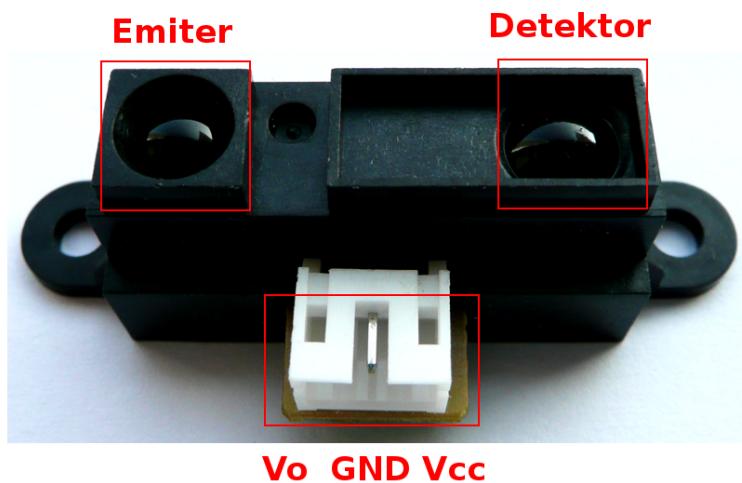
W pierwszych sześciu kolumnach tabeli prezentowany jest stan ustalony na poszczególnych poziomach odległości z podziałem na czujnik prawy i lewy. Jedynka w odpowiedniej kolumnie oznacza obecność przeszkody w danym przedziale odległości, natomiast zerem oznaczany jest brak przeszkody. W ostatniej kolumnie prezentowana jest akcja jaką podejmie robot po wykryciu danej konfiguracji stanów.

Tabela 5.5. Zachowanie robota w zależności od stanu wykrywanego na czujnikach odległości

R3	L3	R2	L2	R1	L1	Akcja robota
0	0	0	0	0	0	Maksymalna prędkość
0	1	0	0	0	0	Średnia prędkość
0	1	0	1	0	0	Minimalna prędkość
0	1	0	1	0	1	Skrót w lewo o 45°
1	0	0	0	0	0	Średnia prędkość
1	0	1	0	0	0	Minimalna prędkość
1	0	1	0	1	0	Skrót w prawo o 45°
1	1	0	0	0	0	Średnia prędkość
1	1	1	0	0	0	Minimalna prędkość
1	1	1	0	1	0	Skrót w prawo o 45°
1	1	0	1	0	0	Minimalna prędkość
1	1	0	1	0	1	Skrót w lewo o 45°
1	1	1	1	0	0	Minimalna prędkość
1	1	1	1	1	0	Skrót w prawo o 45°
1	1	1	1	1	1	Skrót w lewo o 90°

5.5.2. Implementacja

W zrealizowanej w ramach pracy magisterskiej implementacji robot, Dark Explorer, wyposażony został w dwa dalmierze Sharp GP2D12 o efektywnym zasięgu od 10 do 80 cm. Każdy z dalmierzy wyposażony jest w nadajnik i odbiornik IR. Czujnik GP2D12 samodzielnie dokonuje pomiaru odległości i zwraca go w postaci sygnału analogowego. Dlatego też, robot wykorzystuje konwerter analogowo-cyfrowy do przetwarzania wyniku pomiaru otrzymanego z czujnika. Wartość napięcia otrzymana na wyjściu z konwertera jest za pomocą równania, otrzymanego na podstawie charakterystyki wyjściowej dalmierza, zamieniana na odległość czujnika od przeszkody. Na zdjęciu poniżej zaznaczone zostały poszczególne elementy składowe czujnika oraz kolejne wyprowadzenia.

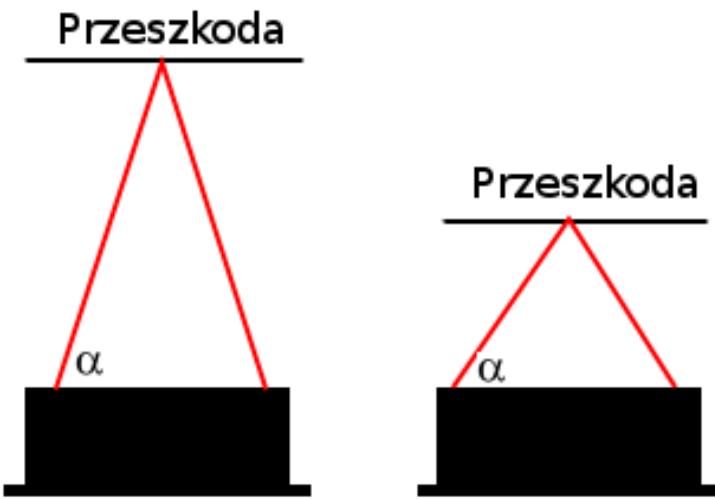


Rysunek 5.27. Budowa czujnika GP2D12 wraz z wyprowadzeniami

Zastosowany czujnik odległości został wyposażony w unowocześniony sposób dokonywania pomiarów odległości. Użyty dalmierz wykorzystuje triangulację oraz małą liniową matrycę CCD aby obliczyć odległość od przeszkody w polu widzenia sensora [2]. Podstawową zasadą działania nowego systemu pomiaru odległości jest wysłanie impulsu światła podczerwonego który po odbiciu od przeszkody wraca do detektora i tworzy trójkąt pomiędzy emiterem, detektorem a przeszkodą. Kąty w tym trójkącie zmieniają się razem z odlegością od przeszkody, co zostało zaprezentowane na rysunku 5.28.

Kluczem do poprawnego działania takiego rozwiązania jest soczewka która dba o to, aby odbite światło trafiało na odpowiednie pole matrycy CCD w zależności od wartości kątów w opisany powyżej trójkącie. Dzięki takiej budowie soczewki, czujnik na podstawie danych z matrycy CCD może określić kąt pod jakim światło odbite wróciło, a dzięki temu jest w stanie obliczyć aktualną odległość od przeszkody. Zastosowana w dalmierzu metoda pomiaru odległości jest niemal całkowicie odporna na zakłócenia

spowodowane światłem zewnętrznym. Dodatkowo eliminuje błędy pomiarowe związane z kolorem powierzchni przeszkody. Dzięki zastosowaniu takiego podejścia możliwe jest wykrycie całkowicie czarnej ściany w całości oświetłonej przez intensywne światło słoneczne, co w przypadku starszych dalmierzy praktycznie nie jest możliwe.



Rysunek 5.28. Zasada przeprowadzania pomiaru przez czujnik GP2D12

Zgodnie z założeniami algorytmu obszar zasięgu czujników został podzielony na trzy poziomy. Poziom trzeci został wyznaczony w odległości 50 cm, drugi poziom wyznaczony został w odległości 25 cm, a poziom pierwszy wyznaczony został w odległości 12 cm od przeszkody. Program zarządzający działaniem robota odczytuje kolejno dane o odległości otrzymane z czujnika prawego i lewego i kontroluje zachowanie robota zgodnie z zasadami przedstawionymi w tabeli 5.5. Z praktycznego punktu widzenia implementacja tego rodzaju algorytmu sprowadza się do stworzenia tablicy w której przechowywane są prędkości poruszania się każdego z czterech kół z uwzględnieniem lokalizacji czujnika oraz progu odległości w jakim znajduje się przeszkoda.

Aby móc praktycznie wykorzystać dane analogowe uzyskane z dalmierza wykorzystany został wbudowany w robota przetwornik analogowo-cyfrowy (ADC). Sposób obsługi przetwornika analogowo-cyfrowego, na potrzeby omijania przeszkód, został zrealizowany w oparciu o dokumentację i przykłady zawarte w książce J. Augustyna pt. „Projektowanie systemów wbudowanych na przykładzie rodziny SAM7S z rdzeniem ARM7TDMI”. Zgodnie ze specyfikacją, przetwornik, wbudowany w procesor SAM7S pozwala na przetwarzanie napięcia w granicach od 0V do napięcia referencyjnego które w przypadku robota Dark Explorer jest równe napięciu zasilania procesora tj. 3.3V. Aby uzyskać jak najdokładniejszy wynik pomiaru przetwornik został skonfigurowany do pracy z rozdzielczością 10-bitową. Wydłuża to czas potrzebny na konwersję danych jednakże przy prędkościach z

jakimi robot się porusza nie ma to większego wpływu na jakość działania aplikacji gdyż wykorzystany przetwornik ADC mierzy i zwraca wartość chwilową napięcia [6].

W trakcie działania aplikacji uruchamiany jest proces konwersji na kanałach do których podłączone są czujniki odległości, a następnie aplikacja oczekuje na wywołanie przerwania związanego z wpisaniem wartości na temat napięcia do odpowiednich rejestrów. Po otrzymaniu wartości napięcia widocznego na wyjściu z dalmierzy, obliczana jest odległość w jakiej znajduje się przeszkoda. Ze względu na zastosowaną metodę pomiarową napięcie wyjściowe z czujników odległości nie ma charakterystyki liniowej. Jest to związane obliczeniami trygonometrycznymi wykonywanymi w celu wyznaczenia odległości od przeszkody. W tym celu, udostępniona w ramach dokumentacji czujnika [1] charakterystyka wyjściowa, posłużyła do wyznaczenia progów napięcia dla zdefiniowanych przez algorytm progów odległości.

W ramach testów przeprowadzono kilka symulacji działania implementacji dla różnych pozycji startowych przy zbliżaniu się robota do brzegu przeszkody. Już pierwsze testy wykazały, że gdy robot porusza się prostopadle do płaszczyzny ściany algorytm działa bardzo dobrze, jednakże w przypadku gdy ruch ten nie jest prostopadły w niektórych przypadkach pojawiają się problemy z działaniem algorytmu. Wspomniane problemy związane są z w niektórych przypadkach z odbieraniem przez dalmierz dodatkowych sygnałów wysłanych z sąsiedniego czujnika. Rozwiążaniem takiego problemu mogłoby być zastosowanie czujników z kodowaniem sygnału. Co więcej zasada działania dalmierzy IR nie pozwala na wykrycie małych i wąskich obiektów stawianych na drodze robota. Dodatkowym problemem jest nieliniowość charakterystyki czujnika która przy odległości mniejszej od 10cm wskazuje napięcia które mogą być mylnie zinterpretowane.

5.6. Wyświetlacz LCD

Do sygnalizacji wykonywanych czynności oraz polepszenia komunikacji pomiędzy człowiekiem a robotem został wykorzystany wyświetlacz LCD. Użyty model to wyświetlacz LCD 2x16, ze złączem pasującym do płyt systemu EVBXXX, z podświetleniem biało-niebieskim. Element ten został wybrany ze względu na stosunkowo niską cenę oraz możliwość podłączenia do płyty ewaluacyjnej EVBsam7s wykorzystywanej podczas rozwoju robota.

Na wyświetlaczu pokazywane są czynności aktualnie wykonywane przez robota oraz informacje pobierane z czujników zamontowanych na nim. Przykładowe komunikaty pokazywane przez Dark Explorera mówią o: aktualnej temperaturze, ilości kroków które wykonał użytkownik, kierunku w jakim podąża.



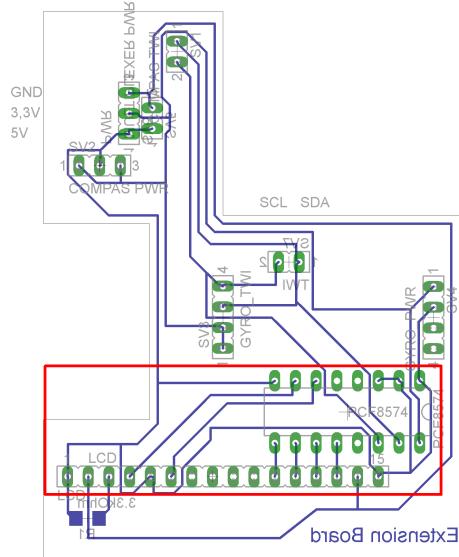
Rysunek 5.29. Wyświetlacz LCD z komunikatem powitalnym

W celu wykorzystania wyświetlacza LCD konieczne było zastosowanie jakiegoś sposobu na redukcję wejść oraz wyjść wykorzystywanych przez to urządzenie. W konfiguracji podstawowej Dark Explorer udostępniał jedynie pięć złącz GPIO¹³ natomiast sam wyświetlacz potrzebuje takich złącz sześć. Z pomocą przyszedł tutaj 8-bitowy ekspander GPIO z interfejsem I^2C ¹⁴. Wykorzystany element to PCF8574 firmy Philips. Układ ten posiada ośmio bitowy rejestr w którym przechowuje słowo odebrane poprzez interfejs I^2C . Każdy bit tego rejestru jest odzwierciedlany jako stan wysoki lub niski na odpowiednich wyjściach GPIO. W taki sposób możliwe jest sterowanie wyświetlaczem LCD. Wykorzystując ten

¹³ GPIO – General Purpose Input/Output, złącza wejścia wyjścia ogólnego przeznaczenia

¹⁴ I^2C – szeregowa dwukierunkowa magistrala służąca do przesyłania danych w urządzeniach elektronicznych

pomysł można znaczco rozszerzyć ilość portów GPIO na robocie i sterować dowolnymi urządzeniami.

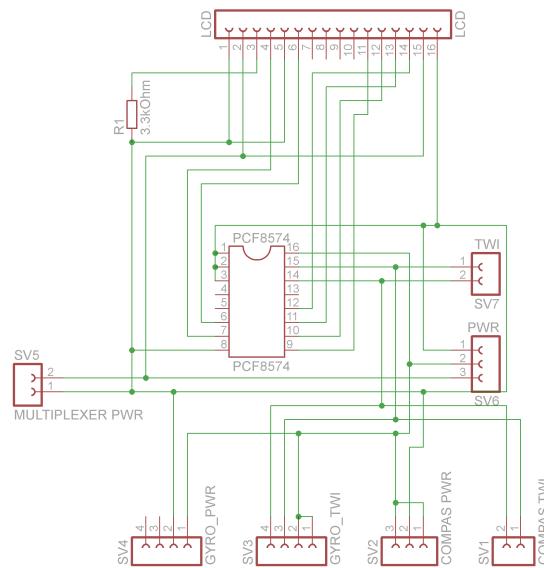


Rysunek 5.30. Płyta rozszerzeń z zaznaczonymi elementami odpowiedzialnymi za obsługę wyświetlacza LCD

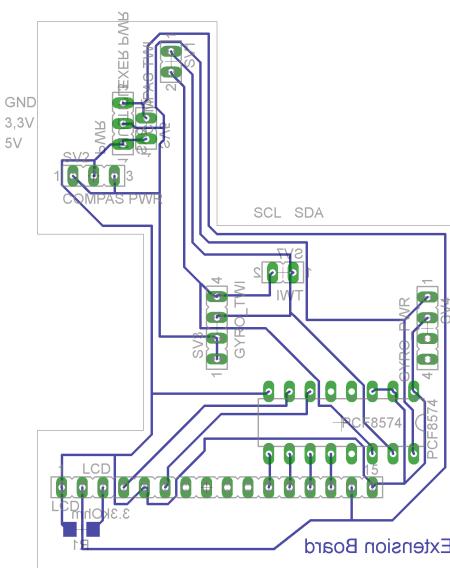
Wyświetlacz został zakupiony jako moduł z wtykiem 16 pinowym. Można go podłączyć do gniazda LCD na płycie rozszerzeń Dark Explorera.

5.7. Płyta rozszerzeń

Wszystkie elementy elektroniczne stworzone na rzecz tej pracy magisterskiej zostały połączone przy pomocy płyty rozszerzeń. Doprowadza ona zasilanie oraz odpowiedni interfejs komunikacyjny do poszczególnych urządzeń. Płyta ta została przygotowana przy pomocy oprogramowania Eagle¹⁵ w wersji edukacyjnej. Niestety z powodu ograniczeń na maksymalny rozmiar płytki stworzonej za pomocą tego oprogramowania z licencją edukacyjną, konieczne było podzielenie płyty rozszerzeń na dwie części.



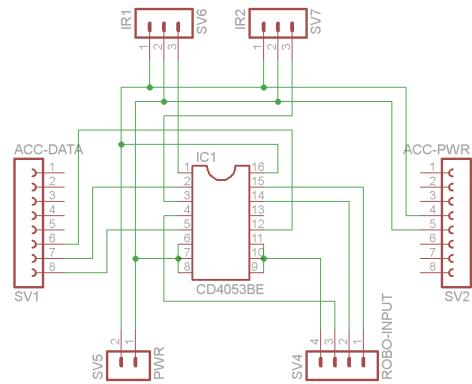
Rysunek 5.31. Schemat części cyfrowej płyty rozszerzeń



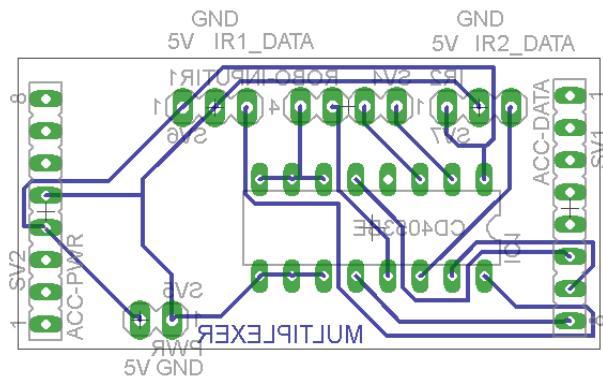
Rysunek 5.32. Layout cyfrowej części płyty rozszerzeń

¹⁵ Eagle – Easily Applicable Graphical Layout Editor

Jedna część płyty rozszerzeń (rys. 5.31) jest odpowiedzialna za wszystkie elementy cyfrowe które komunikują się z robotem przy pomocy interfejsu I^2C . Druga natomiast (rys. 5.33 pozwala na podłączanie elementów analogowych których sygnały są interpretowane przez przetwornik analogowo cyfrowy będący jednym z urządzeń peryferyjnych mikrokontrolera ARM.



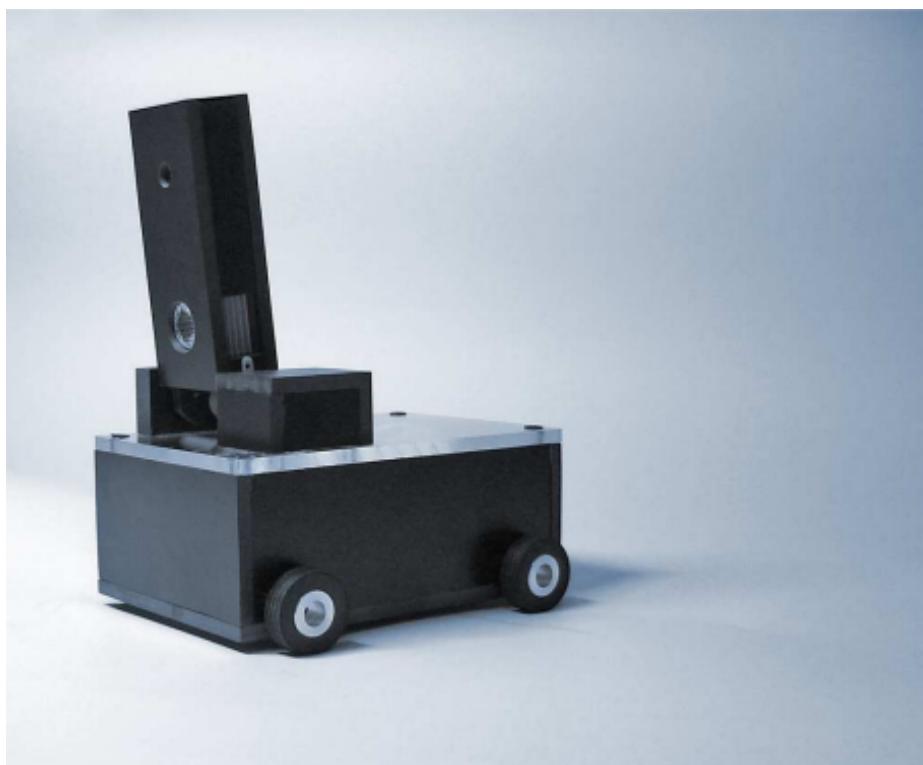
Rysunek 5.33. Schemat części analogowej płyty rozszerzeń



Rysunek 5.34. Layout analogowej części płyty rozszerzeń

5.8. Rozbudowa obudowy robota

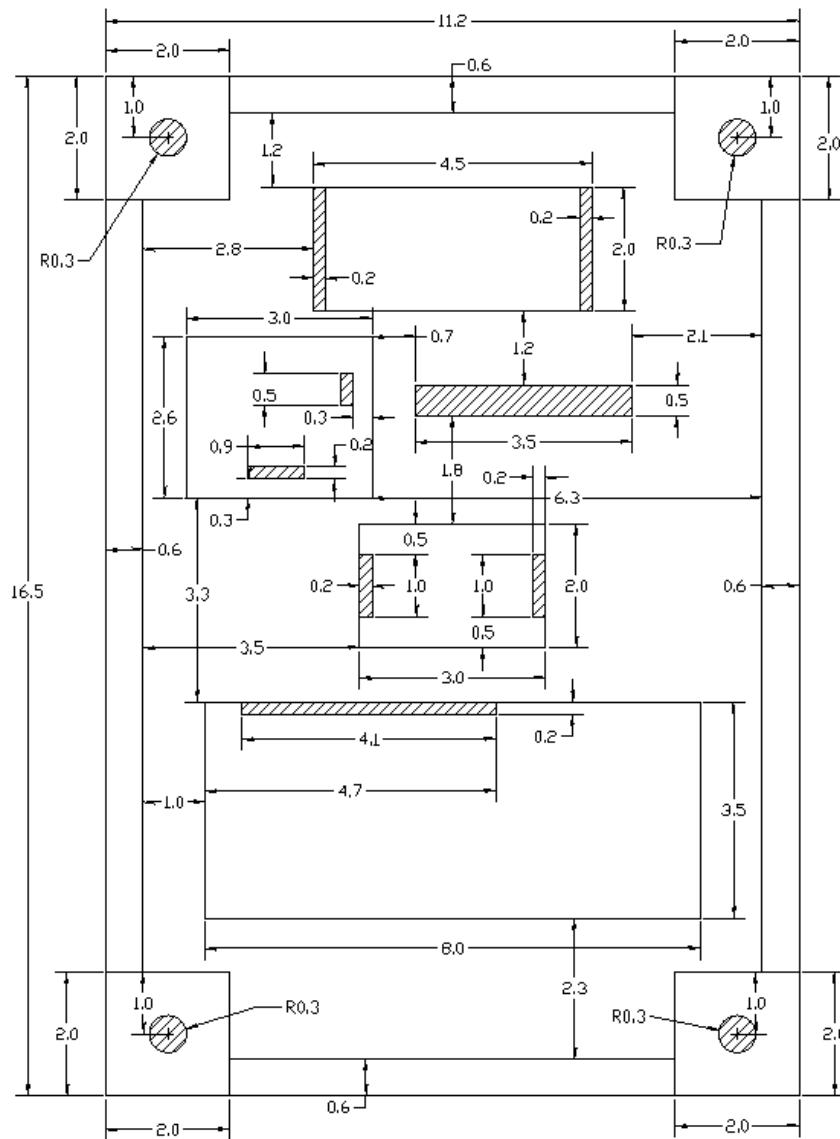
Obudowa robota Dark Explorer stworzona w ramach poprzedniej pracy magisterskiej [12] składała się z dwóch elementów bazowych. Pierwszym elementem jest podwozie wykonane ze spienionego PCW o grubości 6mm. Głównym jego zadaniem jest zapewnienie ochrony wszystkim podzespołom elektronicznym zainstalowanym w robocie, jak również dostarczenie punktów mocowania pozwalających na integrację poszczególnych grup funkcjonalnych. Drugim ważnym elementem jest pokrywa wierzchnia wykonana z przeźeczystej płyty pleksi na której zamontowany został serwomechanizm z wieżą na której zamontowana została kamera. Całość po zmontowaniu prezentuje się w sposób pokazany na rysunku 5.35.



Rysunek 5.35. Wygląd robota po zmontowaniu wszystkich elementów

Ze względu na dużą ilość czujników dodatkowych których obsługa została dodana w obecnej wersji robota, wszelkie próby wykorzystania istniejącej obudowy zakończyły się niepowodzeniem. Dlatego też zaprojektowany został ekspander pozwalający w łatwy sposób zintegrować poprzednią konstrukcję z zestawem nowych czujników i urządzeń peryferyjnych. Wspomniany ekspander ma postać prostopadłościanu wykonanego w całości z płyt bezbarwnej pleksi. Wszystkie elementy elektroniczne zostały rozmieszczone na spodniej płycie ekspandera, a potrzebne połączenia zostały wyprowadzone poprzez otwory

z gniazdami zainstalowanymi w miejscu podłączenia czujnika. Schemat projektowy dolnej płyty ekspandera widoczny jest na rysunku 5.36.

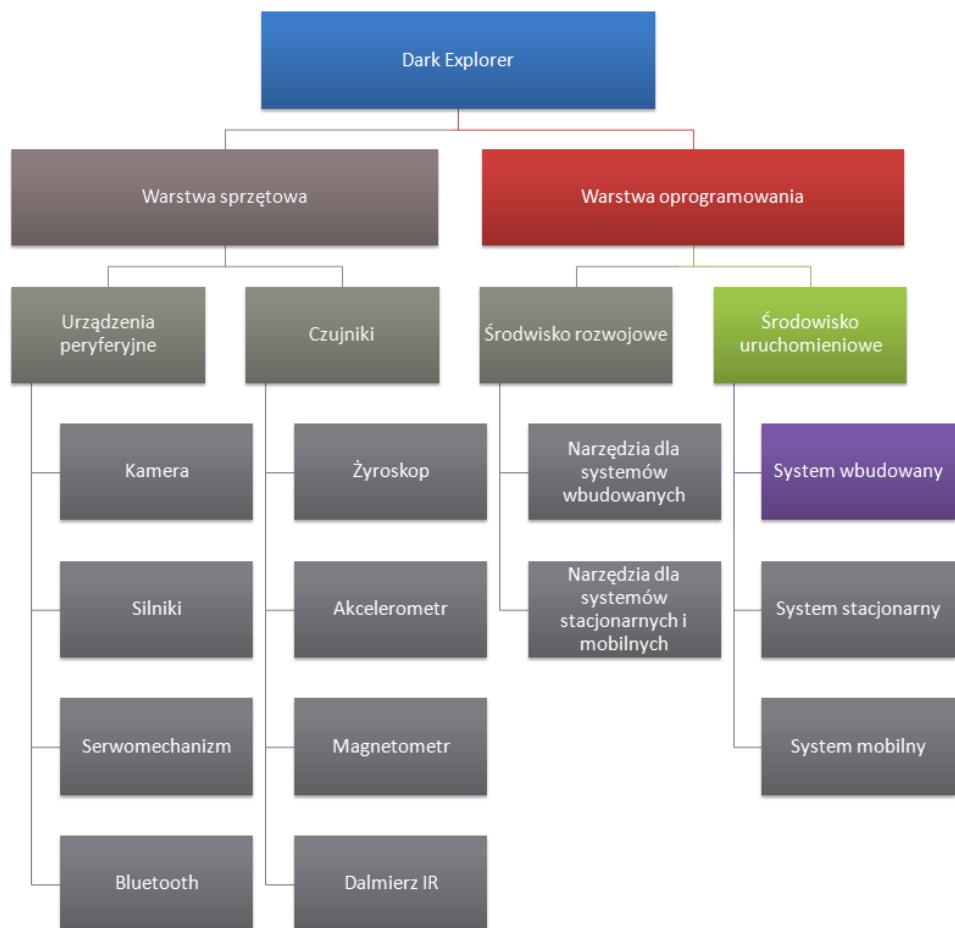


Rysunek 5.36. Wygląd robota po zmontowaniu wszystkich elementów

W centrum ekspandera umiejscowiony został żyroskop, celem takiego zabiegu było wyeliminowanie potencjalnych problemów związanych z pomiarem kąta o jaki robot się obrócił w przypadku gdy czujnik pomiarowy nie znajduje się na osi wzdłuż której obrót jest dokonywany. W południowej części umieszczony został wyświetlacz LCD za pomocą którego użytkownik będzie mógł na bieżąco monitorować aktualne działania robota. W północnej części zamontowany został akcelerometr oraz magnetometr. Taka konfiguracja pozwoliła na ograniczenie ilości i długości przewodów potrzebnych do podłączenia wszystkich elementów do płyty głównej robota.

Rozdział 6

Rozwój robota – firmware



Rysunek 6.1. Struktura platformy robota mobilnego po zakończeniu prac. Kolorem oznaczono zakres prac opisanych w bierzącym rozdziale.

6.1. Protokół komunikacji bluetooth

Robot Dark Explorer komunikuje się z otaczającym go światem przy użyciu technologii Bluetooth. Zaimplementowany w pierwotnej wersji oprogramowania robota protokół komunikacji bazuje na ramce składającej się z 8 bitów nagłówka oraz 8 bitów danych. Za pomocą informacji zawartych w nagłówku robot rozpoznaje rodzaj akcji którą należy podjąć, natomiast po odczytaniu danych z kolejnego przesłanego bajtu urządzenie jest w stanie uruchomić odpowiedni wariant metody zdefiniowanej w nagłówku przesłanej ramki. Jeżeli wysłane polecenie wymusza odesłanie danych zwrotnych są one transmitowane przez robota w postaci czystego strumienia bajtów bez żadnych dodatkowych metadanych.

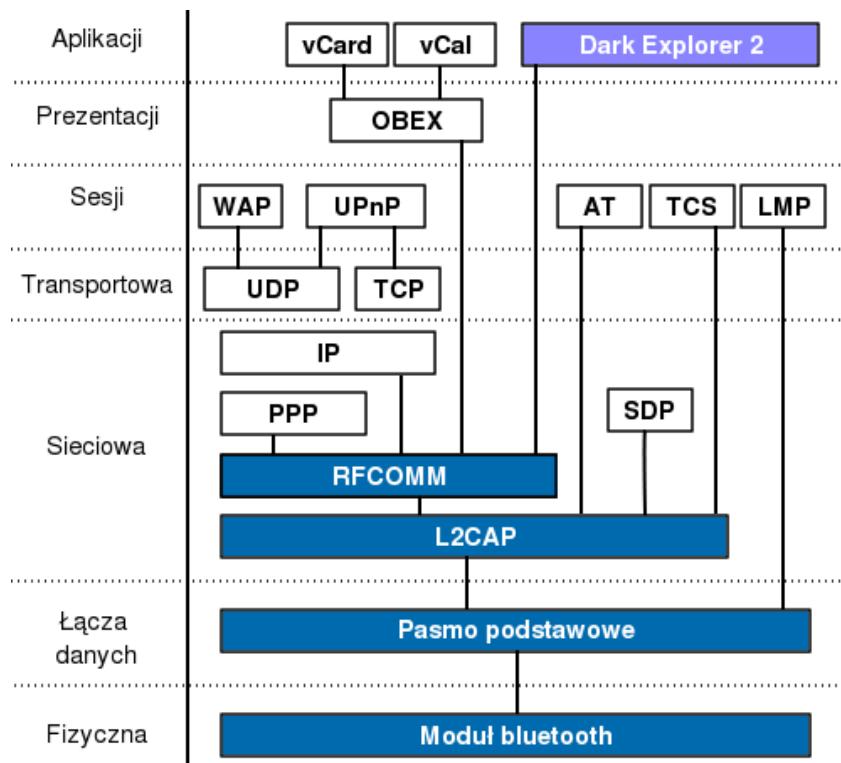


Rysunek 6.2. Schemat ramki komunikacyjnej w pierwotnej wersji robota

Zaprezentowane tutaj podejście jest bardzo wydajne i nie ogranicza efektywnej przepustowości łącza poprzez konieczność przesyłania dodatkowych danych związanych z obsługą protokołu komunikacji. Nie mniej jednak tego rodzaju komunikacja wymaga, od programisty tworzącego aplikacje klienckie, głębszej znajomości sposobu realizacji poszczególnych poleceń, tak aby mógł on synchronizować komunikację zarówno pod względem czasowym jak i logicznym. Ma to szczególne znaczenie w przypadku wykonywania poleceń w sposób sekwencyjny gdzie aplikacja klienta powinna oczekiwana na zakończenie wykonania poprzedniego polecenia, jak ma to miejsce np. podczas pobierania obrazu z kamery. Kolejną znaczącą niedogodnością jest fakt, iż bardzo często nawet najdrobniejsze zmiany w oprogramowaniu robota wymuszają wprowadzanie zmian w aplikacji klienta pomimo tego, że sposób wymiany komunikatów pozostał niezmieniony. Swego rodzaju kłopotliwym problemem było również interpretowanie odpowiedzi przychodzących z robota, gdyż aplikacja klienta musiała przechowywać informację na temat rodzaju i formatu odpowiedzi która została odebrana w wyniku wykonania akcji. Co więcej dotychczasowy sposób wymiany danych nie gwarantował również mechanizmów wykrywania i zapobiegania błędem transmisji na poziomie aplikacji.

Zastosowana do wymiany danych technologia bluetooth posiada szereg standardowych protokołów komunikacyjnych działających w ramach różnych warstw modelu referencyjnego OSI. Niestety protokoły dostępne w warstwie aplikacji ograniczają się jedynie do wymiany danych na temat kontaktów i synchronizacji danych kalendarzowych,

a co za tym idzie nie nadają się do rozwiązania problemów zdalnego sterowania urządzeń. Pełna lista protokołów dostępnych w ramach technologii bluetooth widoczna jest na rysunku 6.3. Niezmiernie ważne jest aby przy projektowaniu schematu komunikacji w warstwie aplikacji, tak dobrąć protokół warstwy niższej aby realizował on jak najwięcej wymagań postawionych przed systemem komunikacji.



Rysunek 6.3. Diagram protokołów bluetooth z podziałem na warstwy

Aby zaadresować wszystkie napotkane w czasie rozwoju robota problemy konieczne okazało się zaprojektowanie i zaimplementowanie nowego protokołu warstwy aplikacji który ułatwiłby tworzenie oprogramowania współpracującego z rozbudowaną wersją robota Dark Explorer. Konieczność taka wynika z faktu iż w warstwie aplikacji nie istnieją protokoły które można byłoby zaadaptować na potrzeby sterowania robotem. Do implementacji protokołu komunikacji wykorzystany został protokół RFCOMM¹ gdyż gwarantuje on nam korekcję błędów na poziomie pakietów. Wszystkie powiązane z nim protokoły niższego poziomu zostały oznaczone kolorem na rysunku 6.3. Projekt protokołu komunikacji bazuje na 16 bitowych ramkach bazowych. Każdy przesyłany komunikat musi posiadać 16 nagłówków w odpowiednim formacie który pozwoli na jego poprawne zinterpretowanie i przetworzenie. Komunikaty w niedozwolonym formacie lub zawierające nieprawidłowe dane będą przez robota ignorowane.

¹ Radio Frequency Communication

W ramach opracowanego standardu wyróżnić można dwa rodzaje ramek. Do pierwszej grupy zaliczyć można ramki żądań wysyłane przez urządzenia zewnętrzne w celu zlecenia robotowi wykonania manewru lub rozpoczęcia kolejnych etapów bardziej złożonej procedury. Diagram przedstawiający strukturę funkcjonalną ramki żądania widoczny jest na rysunku 6.4.



CMD – unikalny identyfikator polecenia [0 – 63]

ACK – 1 jeśli polecenie wymaga potwierdzenia, 0 w przeciwnym wypadku

EXT – 1 jeśli polecenie zawiera dodatkowe dane konfiguracyjne, 0 w przeciwnym wypadku

PARAM – podstawowa konfiguracja parametrów polecenia

EXT PARAM – dodatkowa (opcjonalna) konfiguracja polecenia, uwzględniana jedynie dla EXT = 1

Rysunek 6.4. Schemat funkcjonalny ramki komunikacyjnej z żądaniem

Pierwsze 6 bitów zostało zarezerwowane na unikalny identyfikator polecenia które robot ma wykonać. Pozwala to na zdefiniowanie 64 niezależnych funkcjonalnie komend w ramach których, w wersji podstawowej, możliwe jest uruchomienie 256 niezależnych wariantów polecenia. W przypadku wykorzystania wersji rozszerzonej nagłówka programisty ma do dyspozycji 24 bity które może przeznaczyć na dane polecenia i identyfikator wariantu komendy w proporcjach odpowiadających wymaganiom programisty. Kolejny bit nagłówka przeznaczony został na flagę potwierdzenia. Umieszczenie 1 na wspomnianym bieżie spowoduje iż robot po zakończeniu wykonywania żądania prześle ramkę potwierdzającą ze statusem wykonania akcji. Ułatwia to programistę wykonywanie czynności sekwencyjnych takich jak np. wykonanie zdjęcia za pomocą kamery i przesłanie go do aplikacji klienta. Ósmy bit nagłówka zarezerwowany został na flagę z informacją o użyciu rozszerzonej wersji nagłówka. Wymuszenie 1 na tym bieżie spowoduje, że robot będzie oczekiwany na dodatkowe 16 bitów danych żądania które mogą zostać wykorzystane do przesłania bardziej skomplikowanej konfiguracji wykonania polecenia. Kolejne osiem bitów stanowi tzw. podstawową konfigurację polecenia, która może służyć jako identyfikator przy uruchamianiu odpowiedniego wariantu polecenia lub jako nośnik danych potrzebnych do zrealizowania przesłanej komendy. W przypadku gdyby rozmiar bufora konfiguracyjnego nie pozwalał na przechowanie wszystkich wymaganych danych programista może wykorzystać dodatkowe 2 bajty konfiguracji rozszerzonej.

Drugi rodzaj ramek stanowią powiadomienia informujące użytkownika o rezultacie wykonania przesłanej akcji lub aktualnym stanie robota. Podobnie jak w przypadku ramki z żądaniem pierwsze 6 bitów zarezerwowane jest na unikalny identyfikator polecenia dla którego przesyłana jest odpowiedź. Kolejny bit zawiera flagę informującą o rezultacie

zakończenia akcji. Prawidłowe zakończenie wykonania polecenia sygnalizowane jest ustaleniem zera na wspomnianym bicie. Wystąpienie błędu podczas realizacji polecenia lub przesłanie komendy w nieprawidłowym formacie spowoduje ustawienie 1 na bicie STATE. W przypadku gdy ilość danych do przesłania przekracza maksymalny dopuszczalny rozmiar pojedynczej ramki możliwe jest podzielenie danych na fragmenty i przesłanie ich za pomocą sekwencji komunikatów. Koniec sekwencji sygnalizowany jest za pomocą bitu EOT². Dostępność tego rodzaju trybu pozwala na szybką transmisję większych ilości danych bez konieczności wysyłania żądań dla poszczególnych fragmentów.



CMD – unikalny identyfikator polecenia [0 – 63]

STATE – rezultat zakończenia wykonania akcji, 0 – sukces, 1 – porażka

EOT – flaga konieca transmisji, 1 oznacza ostatni pakiet w sekwencji

SIZE – liczba bajtów danych

DATA – dane przesyłane w ramach pakietu

Rysunek 6.5. Schemat funkcjonalny ramki komunikacyjnej z odpowiedzią

Kolejne 8 bitów nagłówka odpowiedzi stanowi informacja o ilości bajtów przesyłanych w sekcji danych ramki. Ogranicza to maksymalny rozmiar ramki przez co transmisja danych nie powoduje licznych retransmisji które bardzo często pojawiają się w przypadku transmisji większych ilości danych. Informacja o szerokości pola danych ramki może również służyć jako suma kontrolna pozwalająca na sprawdzenie czy wszystkie zadeklarowane dane zostały prawidłowo odebrane. Pozwala to programiście tworzącemu oprogramowanie klienta na monitorowanie transmisji i ewentualne reagowanie w przypadku pojawiania się problemów z płynnością transmisji.

² End of Transmission

6.2. Algorytm rekonstrukcji ścieżki powrotnej

6.3. Modernizacja sposobu pobierania obrazu z kamery

6.4. Lokalizacja twarzy na obrazie

W ciągu ostatnich lat obserwuje się bardzo dynamiczny rozwój biometrii. Aplikacje analizujące biometryczne cechy użytkowników znalazły swoje zastosowanie miedzy innymi w systemach zabezpieczeń i monitoringu, robotyce, komputerach przenośnych, aparatach i kamerach cyfrowych czy nawet w nowoczesnych telefonach komórkowych. Pośród szeregu cech które mogą zostać poddane analizie szczególnym zainteresowaniem cieszy się analiza ludzkiej twarzy. Systemy posiadające możliwość zlokalizowania na obrazie twarzy od pewnego czasu towarzyszą nam w życiu codziennym. Niemniej jednak większość istniejących algorytmów pozwalających na rozwiązywanie tego problemu jest dosyć skomplikowana i ma bardzo konkretne wymagania zarówno co do platformy sprzętowej jak i jakości obrazu który będzie poddawany analizie. Dlatego też stworzenie implementacji dla platformy o bardzo ograniczonych zasobach obliczeniowych i pamięciowych jest dużym wyzwaniem. Celem niniejszego rozdziału jest przedstawienie podstawowych rodzajów algorytmów lokalizacji twarzy oraz wskazanie ich wad i zalet. W dalszej części znajduje się szczegółowy opis stworzonej implementacji algorytmu zastosowanego w realizowanej pracy magisterskiej.

6.4.1. Algorytmy

W ciągu ostatnich kilku lat rozwijanych było wiele różnych podejść poprawiających wydajność procesu lokalizacji ludzkiej twarzy. Wśród nich można wyróżnić kilka głównych kategorii. Pierwszą z nich jest metoda opierająca się o bazę wiedzy. Metoda ta pozwala na odszukiwanie niezmiennych cech twarzy nawet w bardzo skomplikowanym otoczeniu, a przez to pozwala na ustalenie pozycji twarzy. Tego typu podejście do problemu eliminuje problemy z uwzględnianiem ułożenia i pozycji twarzy na obrazie. Statystyczny opis zależności pomiędzy poszczególnymi cechami twarzy pozwala na stworzenie modelu który może posłużyć do zbudowania szablonu twarzy dzięki któremu będziemy w stanie jednoznacznie stwierdzić czy na obrazie widoczna jest twarz czy nie. Z takiego ujęcia problemu wywodzi się kolejna metoda nazywana metodą porównywania szablonów (ang. template matching). Do stworzenia wspomnianego szablonu twarzy mogą posłużyć takie parametry jak na przykład kolor skóry, faktura twarzy czy też budowa twarzy. Wśród metod bazujących na porównywaniu szablonów można wyróżnić dwa podejścia. Pierwsze z nich oparte jest bezpośrednio na opisie cech twarzy i korzysta z opisu zależności pomiędzy parametrami twarzy jako szablonu odszukując najbardziej pasujący do szablonu fragment obrazu. Drugim podejściem jest stosowanie globalnego szablonu bazującego na metrykach będących sumą poszczególnych cech za pomocą których jest opisany szablon. Pierwsze podejście wymaga obrazu o stosunkowo dużej rozdzielczości ale ponieważ nie bazuje

ono na wszystkich punktach obrazu będzie obliczeniowo bardziej wydajne niż podejście oparte o szablon globalny które może wymagać przeszukania potencjalnie dużo większej grupy punktów obrazu wejściowego. Główną wadą przedstawionych do tej pory rozwiązań jest są ich dosyć wysokie wymagania. Każdy z omówionych do tej pory algorytmów wymagał wykonywania kosztownych obliczeń mających na celu poprzez dokonywanie przekształceń dopasowanie modelu szablonu do zawartości obrazu wejściowego. Tak więc wykorzystanie tego typu algorytmów wiąże się z koniecznością użycia stosunkowo mocnych jednostek obliczeniowych co w przypadku platformy embeded mogłyby być problematyczne.

Zupełnie innym podejściem do problemu lokalizacji twarzy jest wykorzystywanie koloru skóry do odszukania fragmentów obrazu które potencjalnie mogłyby zwierać twarz. Rozwiązania oparte o analizę kolorów można podzielić na dwie grupy w pierwszej znajdują się rozwiązania bazujące na jednolitym kolorze tła w drugim natomiast obraz wejściowy może zawierać tło o dowolnym stopniu złożoności. Pierwsza z przedstawionych grup w znakomity sposób upraszcza problem odnalezienia twarzy gdyż często usunięcie koloru tła z obrazu gwarantuje nam odnalezienie wszystkich twarzy, zakładając, że obraz zawiera jedynie twarze. Jednakże jest to podejście nakładające duże ograniczenia na warunki w jakich odbywa się akwizycja obrazu. Dlatego też znacznie częściej porusza się problem lokalizacji twarzy w oparciu o analizę kolorów z użyciem obrazów zawierających niejednokrotnie tło o bardzo wysokim stopniu złożoności. Niemniej jednak wszystkie rozwiązania bazujące jedynie o segmentację kolorów są w bardzo wysokim stopniu zależne od warunków w jakim pobierany jest obraz wejściowy. Ze względu na zmieniające się warunki otoczenia może okazać się, że nasz algorytm nie uwzględnia wszystkich kolorów skóry. Dodatkowym utrudnieniem jest pojawianie się grup pikseli które zostały zakwalifikowane jako kolor skóry pomimo, że z twarzą nie mają nic wspólnego. Problem ten jest szczególnie uciążliwy w przypadku obrazów o małych rozmiarach gdyż trudno jest wtedy odróżnić twarz od szumów powstałych wskutek warunków otoczenia w który obraz był pobierany. Pomimo licznych swoich licznych wad segmentacja kolorów jest jedną z najszybszych metod klasyfikacji obrazów pod kątem obecności twarzy na obrazie. Co więcej narzut obliczeniowy na zastosowanie klasyfikatora bazującego o model kolorów skóry jest proporcjonalny do rozmiarów pobranego obrazu. Niestety jak się okazuje w praktyce sama segmentacja kolorów jest niewystarczająca i wymaga użycia dodatkowych metod w celu potwierdzenia odnalezienia twarzy.

Kolejnym ciekawym rozwiązaniem pozwalającym na odszukanie twarzy na ruchomym obrazie jest analiza ruchów twarzy. W tego typu rozwiązańach obecność twarzy w

sekwencji wideo wykrywana jest na podstawie detekcji ruchów charakterystycznych dla ludzkiej twarzy takich jak na przykład ruchy powiek. Mruganie jest ruchem na tyle specyficzny, że wykrycie go pozwala jednoznacznie wnioskować o lokalizacji twarzy na obrazie. Niestety przetwarzanie kilkudziesięciu klatek na sekundę o rozmiarach pozwalających na zarejestrowanie ruchu powiek stawia dosyć wygórowane wymagania związane nie tylko z algorytmem ale również z ilością danych jakie klasyfikator musi przetworzyć w ciągu każdej sekundy nagrania. Istnieje jeszcze wiele innych rozwiązań bazujących na bardzo wyszukanych metodach jak na przykład użycie sieci neuronowych, jednakże ze względu na ich wygórowane wymagania wykorzystanie jednego z nich dla platformy embeded z ograniczonymi zasobami obliczeniowymi i pamięciowymi jest niezwykle trudne, a w niektórych przypadkach niemal całkowicie niemożliwe.

Algorytm implementowany w ramach pracy magisterskiej bazuje na podejściu opartym o segmentację kolorów. Aby wyeliminować niedoskonałości płynące z tego podejścia algorytm w ostatniej fazie wykorzystuje informację o najprostszych cechach twarzy w celu potwierdzenia obecności twarzy w obszarach zaklasyfikowanych jako kolor skóry. Zastosowanie tego typu połączenia dwóch technik zaowocowało postaniem algorytmu nie tylko szybkiego ale zarazem dającego stosunkowo dobre wyniki analizy.

6.4.2. Implementacja

Implementacja modułu do lokalizacji twarzy na obrazie pobranym z kamery robota została w całości oparta o algorytm zaproponowany przez Yao-Jiunn Chen, Yen-Chun Lin w artykule zatytułowanym 'Simple Face-detection Algorithm Based on Minimum Facial Features'. Autorzy do rozwiązania problemu używają obrazów o małej rozdzielcości opisanych za pomocą modelu kolorów RGB.

Zgodnie ze wskazówkami znalezionymi we wspomnianej publikacji, pierwszym krokiem na drodze do zlokalizowania twarzy na obrazie jest akwizycja danych z kamery. Przesłany z kamery obraz automatycznie jest poddawany podstawowej korekcji mającej na celu dopasowanie balansu bieli oraz dopasowanie ostrości. Tak przygotowany obraz poddawany jest binaryzacji. Celem wspomnianego procesu jest odnalezienie na obrazie obszarów będących w kolorze skóry oraz włosów. Zanim jednak detekcja interesujących nas obszarów będzie możliwa konieczne jest przeprowadzenie normalizacji modelu RGB. Zabieg tego typu pozwoli na usunięcie zakłóceń wywołanych przez światła i cienie na obrazie pobranym z kamery robota. Opisany proces normalizacji przeprowadzany jest poprzez pobranie wartości poszczególnych kanałów RGB danego piksela obrazu, a następnie

podzielenie tak otrzymanej liczby przez sumę wartości wszystkich kanałów danego punktu. Korzystając z takiego przepisu otrzymujemy następujący zestaw równań opisujący normalizację każdego kanału z palety RGB.

$$r = \frac{R}{R + G + B} \quad (6.1)$$

$$g = \frac{G}{R + G + B} \quad (6.2)$$

$$b = \frac{B}{R + G + B} \quad (6.3)$$

Na tak przygotowanym obrazie można już podjąć próbę odszukania punktów w kolorze skóry i włosów. Do tego celu użyte zostały równania zaproponowane przez autorów pracy źródłowej. Określają one rozkład kolorów ludzkiej skóry i włosów wyznaczonych, jak twierdzą autorzy, w sposób eksperymentalny. Bazując na takich założeniach ustalony został górny $F_1(r)$ i dolny $F_2(r)$ zakres kolorów w jakich może znaleźć się ludzka skóra.

$$F_1(r) = -1.376r^2 + 1.0743r + 0.2 \quad (6.4)$$

$$F_2(r) = -0.776r^2 + 0.5601r + 0.18 \quad (6.5)$$

Należy jednak zauważyć, że w zdefiniowanych w ten sposób równaniach mieści się również kolor biały $r = 0.33ig = 0.33$ dlatego też należy dodać następujący warunek usuwający niechciany kolor z obszaru zainteresowania.

$$w = (r - 0.33)^2 + (g - 0.33)^2 > 0.001 \quad (6.6)$$

Po połączeniu obydwu równań otrzymujemy następujący układ równań opisujący zakres kolorów jakie powinny reprezentować ludzką skórę.

$$S = \begin{cases} 1 & \text{dla } g < F_1(r) \cap g > F_2(r) \cap w > 0.001 \\ 0 & \text{dla pozostałych} \end{cases} \quad (6.7)$$

Praktyczne doświadczenia z wykrywaniem obszarów skóry z zastosowaniem równania 6.7 wykazały iż niemal wszystkie obszary o kolorze skóry została prawidłowo zaklasyfikowana. Znakomita większość pikseli nie stanowiących koloru skóry została zastąpiona kolorem czarnym. Niemniej jednak wiele pikseli o kolorze niebieski, żółtym i szarym również została błędnie sklasyfikowana. Aby wyeliminować tego rodzaju efekt należy wykorzystać kryteria klasyfikacji oparte o przestrzeń kolorów HSI. Poniżej przedstawiono zależność pomiędzy

modelem RGB a HSI.

$$\phi = \cos^{-1} \left\{ \frac{0.5[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}} \right\}$$

$$\begin{cases} H = \phi & \text{dla } B \leq G \\ H = 360^\circ - \phi & \text{dla } B > G \end{cases} \quad (6.8)$$

Po skorzystaniu z przestrzeni HSI otrzymujemy następujący przepis na kolor skóry.

$$S = \begin{cases} 1 & \text{dla } g < F_1(r) \cap g > F_2(r) \cap w > 0.001 \cap (H > 240 \cup H \leq 20) \\ 0 & \text{dla pozostałych} \end{cases} \quad (6.9)$$

Przy tak zdefiniowanym klasyfikatorze dokonywana jest binaryzacja całego obrazu. Po przeprowadzeniu binaryzacji przeprowadzana jest kwantyzacja obrazu za pomocą próbki o rozmiarze i wysokości pięciu pikseli. Pozwala to zminiejszyć rozdzielncość obrazu, a co za tym idzie wyeliminować pewne rodzaje szumu oraz znacznie uprościć dalsze obliczenia. Wspomniany proces kwantyzacji polega na przeliczeniu liczby pikseli w kolorze skóry i kolorze czarnym w ramach 25 pikseli pobranych jako próbka obrazu. A następnie na podstawie wartości progowej podjęcie decyzji czy cała próbka zostanie oznaczona kolorem skóry czy kolorem czarnym. W ramach implementacji wartość progu została ustalona na poziomie 12 pikseli, oznacza to, że wszystkie próbki zawierające więcej niż 12 pikseli czarnych są w całości oznaczane jako obszar niezawierający skóry i na odwrót.

Podsumowanie

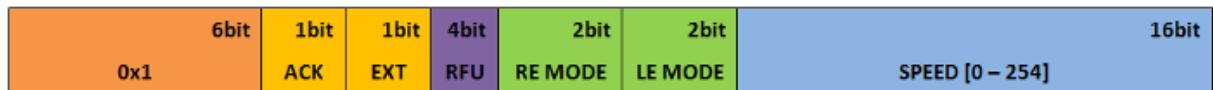
Dodatek A

Specyfikacja poleceń protokołu komunikacji

W ramach niniejszego dodatku zostanie przedstawiona szczegółowa dokumentacja protokołu komunikacyjnego zaprojektowanego na potrzeby sterowania robotem Dark Explorer przy użyciu technologii bluetooth. Zasady przesyłania poleceń wraz z ogólnym zarysem sposobu działania komunikatów zostało szczegółowo omówiony w ramach rozdziału 6.1. W tej części zaprezentowane zostaną wszystkie dostępne polecenia sterujące wraz z przykładami formatów odpowiedzi w przypadku prawidłowego wykonania polecenia.

A.1. Komunikaty sterujące

A.1.1. Sterowanie silnikami



Rysunek A.1. Schemat ramki odpowiedzialnej za sterowanie silnikami

ACK	Wartość 0x1 jeśli komunikat wymaga potwierdzenia
EXT	Wartość 0x1 jeśli komunikat zawiera dane o mocy silników
RFU	Bity zarezerwowane do przyszłego użycia
RE MODE	Tryb pracy silników prawych (0x0 - stop, 0x1 - wprost, 0x2 - wstecz)
LE MODE	Tryb pracy silników lewych (0x0 - stop, 0x1 - wprost, 0x2 - wstecz)
SPEED	Moc silników. Wartość z zakresu [0 - 254]

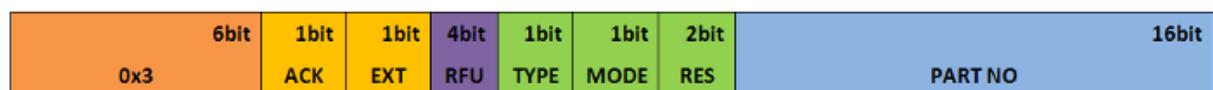
A.1.2. Sterowanie serwomechanizmem



Rysunek A.2. Schemat komunikatu odpowiedzialnego za sterowanie serwomechanizmem

ACK	Wartość 0x1 jeśli komunikat wymaga potwierdzenia
EXT	Wartość 0x1 jeśli komunikat zawiera dane o położeniu
RFU	Bity zarezerwowane do przyszłego użycia
LED	Wartość 0x1 jeśli status diody LED ma zostać zmieniony na przeciwny
SERVO	Wartość 0x1 jeśli stan serwomechanizm ma zostać zmieniony na przeciwny
POSITION	Pozycja (kąt) pod jakim ma zostać ustaliona wieżyczka z kamerą.

A.1.3. Akwizycja obrazu z kamery



Rysunek A.3. Schemat komunikatu odpowiedzialnego za akwizycję obrazu z kamery

ACK	Wartość 0x1 jeśli komunikat wymaga potwierdzenia
EXT	Wartość 0x1 jeśli komunikat zawiera dane o położeniu

RFU	Bity zarezerwowane do przyszłego użycia
TYPE	Typ operacji (0x0 - akwizycja obrazu, 0x1 transmisja pobranego obrazu)
MODE	Tryb pracy kamery (0x0 - odcienie szarości, 0x1 - kolor)
RES	Identyfikator rozmiaru obrazu według następującej specyfikacji:
0x0	- 160x120
0x1	- 320x240
0x2	- 640x480

A.1.4. Sterowanie czujnikami



Rysunek A.4. Schemat komunikatu odpowiedzialnego za odpytywanie czujników

ACK	Wartość 0x1 jeśli komunikat wymaga potwierdzenia
EXT	Wartość 0x1 jeśli komunikat zawiera dane rozszerzone
RFU	Bity zarezerwowane do przyszłego użycia
CAL	Ustawienie wartości 0x1 spowoduje uruchomienie procedury kalibrującej dla sensora podanego w sekcji SENSOR
SENSOR	Unikalny identyfikator typu pomiaru (czujnika) który ma zostać przeprowadzony, według następującej specyfikacji:
0x0	- pomiar napięcia na baterii
0x1	- pomiar temperatury
0x2	- pomiar przyspieszenia statycznego
0x3	- informacje o odległości od najbliższej przeszkody
0x4	- informacje o położeniu na podstawie magnetometru

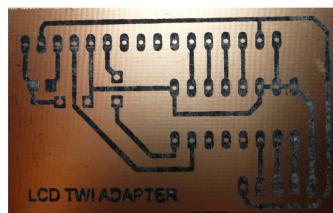
A.2. Komunikaty specjalne

Dodatek B

Wytrawianie płytEK układów elektronicznych

Wszystkie płytki stworzone na rzecz tej pracy magisterskiej zostały wykonane przez autorów pracy w warunkach domowych. W tym dodatku został opisany sposób wytwarzania płytEK.

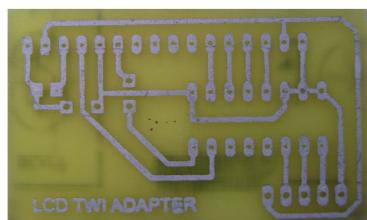
Do wykonania płytEK PCB musimy posiadać maskę ścieżek które umieścimy na laminacie. Autorzy tej pracy magisterskiej użyli w tym celu programu Eagle w wersji edukacyjnej z której można korzystać za darmo. Po zaprojektowaniu schematu oraz layoutu płytki maska ścieżek była drukowana przy pomocy drukarki laserowej na papierze kredowym. Istotne jest dobre przygotowanie layoutu przed drukiem. Jeżeli ścieżki mają znajdować się na górnej stronie płytki, czyli po tej samej stronie po której będą umieszczone elementy elektroniczne, należy stworzyć odbicie lustrzane layoutu. Natomiast w przypadku ścieżek umiejscowionych na warstwie dolnej nie wykonujemy tego kroku. Bardzo przydatnym w tym momencie okazuje się jakiś znacznik na przykład tekst, który na wydruku powinien zawsze być odbiciem lustrzanym. Tak przygotowany layout drukujemy przy użyciu czarno–białej drukarki laserowej na papierze kredowym.



Rysunek B.1. Płytki z naniesionym tonerem

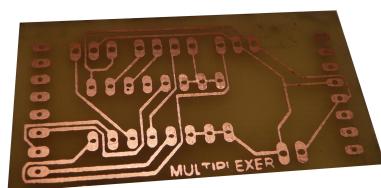
Po wydrukowaniu maski konieczne jest odpowiednie przygotowanie laminatu. W pierwszym kroku używamy papieru ściernego w celu zmatowienia laminatu. Następnie czyścimy laminat płynem do mycia naczyń w celu odtłuszczenia oraz osuszamy go. Po

tych czynnościami możemy przystąpić do nakładania maski na laminat. Wycinamy jeden z wydrukowanych layoutów i przykładamy go zadrukowaną stroną do laminatu. Po tych przygotowaniach prasujemy całość żelazkiem rozgrzanym do temperatury 150°C około 5 minut. Operacja ta powinna spowodować przeniesienie tonera z papieru na laminat. W celu usunięcia papieru kredowego wrzucamy płytę do gorącej wody z dodatkiem proszku do prania. Papier powinien z łatwością się oderwać. Ewentualne przerwania w ścieżkach można poprawić markerem wodoodpornym. Jeżeli uzyskany efekt nie jest satysfakcyjny powtarzamy całą operację usuwając wcześniej toner przy pomocy zmywacza do paznokci.



Rysunek B.2. Wytrawiona płytka

Istnieje wiele środków trawiących miedź. Autorzy tej pracy stosowali B327 czyli nadsiarczan sodowy. Ten środek trawiący działa najlepiej w temperaturze 50°C . W celu osiągnięcia optymalnych warunków do trawienia B327 należy rozcieńczyć w wodzie i umieścić w jakimś naczyniu. Następnie całość wkładamy do drugiego naczynia z gorącą wodą. W takich warunkach trawienie płytki nie powinno trwać dłużej niż 30 minut.



Rysunek B.3. Gotowa płytka

Po zakończeniu procesu trawienia należy usunąć toner przy pomocy zmywacza do paznokci i wywiercić otwory montażowe. Dzięki tej metodzie możemy tworzyć płytki z

bardzo cienkimi ścieżkami wymagającymi precyzji, której nie da się osiągnąć przy ręcznym rysowaniu maski na laminacie. Kolejnymi atutami tej metody jest praktycznie zerowy koszt produkcji płytki oraz krótki czas realizacji.

Dodatek C

Plik konfiguracyjny programatora: triton.cfg

```
#define our ports
telnet_port 4444
gdb_port 3333

#commands specific

source [find interface/turtelizer2.cfg]
source [find board/atmel_at91sam7s-ek.cfg]

ft2232_device_desc "Triton JTAG A"
```

Kod źródłowy

Spis rysunków

1.1.	Prawa robotyki zdefiniowane przez Issaca Assimov'a	10
1.2.	Od lewej: Shakey (Stanford), Genghis (MIT)	12
1.3.	Kolejno: Pathfinder (NASA), Asimo (Honda)	12
1.4.	Klasyfikacja robotów ze względu na ich generację	16
1.5.	Podział robotów ze względu na obszar ich zastosowania	17
2.1.	Struktura platformy robota mobilnego zrealizowanej w ramach poprzedniej pracy magisterskiej [12]	18
2.2.	Obraz wykonany przy pomocy kamery zamontowanej w robocie. 160 x 100 pikseli w kolorze. [12]	20
2.3.	Obraz wykonany przy pomocy kamery zamontowanej w robocie. 320 x 200 pikseli w odcieniach szarości [12]	20
2.4.	Zdjęcie modułu bluetooth zamontowanego na płycie głównej robota.	21
2.5.	Widok na płytę główną Dark Explorer'a	21
2.6.	Koło napędowe Dark Explorer'a	22
2.7.	Okno aplikacji zarządzającej Dark Explorera	24
3.1.	Struktura platformy robota mobilnego po zakończeniu prac. Kolorem oznaczono zakres prac opisanych w bierzącym rozdziale.	26
3.2.	Okno główne IDE Eclipse z dodanym projektem	31
3.3.	Okno dialogowe C++ Project	32
3.4.	Okno dialogowe Properties	32
3.5.	Okno pokazujące odpowiedź prawidłowo zainstalowanego kompilatora	34
3.6.	Poprawnie uruchomiony openocd	35
4.1.	Struktura platformy robota mobilnego po zakończeniu prac. Kolorem oznaczono zakres prac opisanych w bierzącym rozdziale.	37
4.2.	Elementy składowe środowiska rozwojowego Windows Mobile	43
4.3.	Modele tworzenia aplikacji dla Windows Mobile.	45
4.4.	Lista dostępnych na platformie Windows Mobile API komunikacyjnych	47

5.1.	Struktura platformy robota mobilnego po zakończeniu prac. Kolorem oznaczono zakres prac opisanych w bierzącym rozdziale.	51
5.2.	INS francuskiego IRBM S3	52
5.3.	Żyrokompas samolotowy	53
5.4.	IMU stworzone przy pomocy układów MEMS	53
5.5.	Pokrywanie się osi ekranu z kierunkiem wektora grawitacji pozwala na precyzyjne wyznaczenia orientacji urządzenia [10]	56
5.6.	Schemat ideowy akcelerometru pojemnościowego dokonującego pomiaru wzdłuż jednej osi [15]	57
5.7.	Moduł akcelerometru trójosiowego z czujnikiem przyspieszenia MMA7260	59
5.8.	Wartości przyspieszenia statycznego dla układu MMA7260 dostarczone w ramach specyfikacji technicznej urządzenia [16]	60
5.9.	Wartości przyspieszenia odczytywane na wyjściach akcelerometru przed przeprowadzeniem kalibracji	60
5.10.	Wartości odczytywane na wyjściach akcelerometru po przeprowadzeniu kalibracji .	61
5.11.	Siła Coriolis'a	62
5.12.	Budowa elementu pomiarowego żyroskopu [11]	63
5.13.	Dwie oscylujące masy odchylane przez siłę Coriolis'a	63
5.14.	Całkowanie przy pomocy metody trapezów	65
5.15.	Schemat płytki PCB modułu żyroskopu	67
5.16.	Płytki PCB modułu żyroskopu trójosiowego	68
5.17.	Gotowy moduł żyroskopu	68
5.18.	Zakresy czujników pola magnetycznego różnego typu [4]	69
5.19.	Schemat modułu kompasu	72
5.20.	Projekt płytki kompasu elektronicznego	72
5.21.	Gotowy moduł kompasu	72
5.22.	Busola w położeniu początkowym	73
5.23.	Busola przesunięta równolegle o 25 cm,	73
5.24.	Zasada działania dalmierza wykorzystującego podczerwień	74
5.25.	Sposób montażu czujników podczerwieni na robocie [8]	75
5.26.	Przykładowe sytuacje i definicja reakcji robota	76
5.27.	Budowa czujnika GP2D12 wraz z wyporowadzeniami	78
5.28.	Zasada przeprowadzania pomiaru przez czujnik GP2D12	79
5.29.	Wyświetlacz LCD z komunikatem powitalnym	81
5.30.	Płyta rozszerzeń z zaznaczonymi elementami odpowiedzialnymi za obsługę wyświetlacza LCD	82
5.31.	Schemat części cyfrowej płyty rozszerzeń	83
5.32.	Layout cyfrowej części płyty rozszerzeń	83
5.33.	Schemat części analogowej płyty rozszerzeń	84
5.34.	Layout analogowej części płyty rozszerzeń	84

5.35. Wygląd robota po zmontowaniu wszystkich elementów	85
5.36. Wygląd robota po zmontowaniu wszystkich elementów	86
6.1. Struktura platformy robota mobilnego po zakończeniu prac. Kolorem oznaczono zakres prac opisanych w bierzącym rozdziale.	87
6.2. Schemat ramki komunikacyjnej w pierwotnej wersji robota	88
6.3. Diagram protokołów bluetooth z podziałem na warstwy	89
6.4. Schemat funkcjonalny ramki komunikacyjnej z żądaniem	90
6.5. Schemat funkcjonalny ramki komunikacyjnej z odpowiedzią	91
A.1. Schemat ramki odpowiedzialnej za sterowanie silnikami	101
A.2. Schemat komunikatu odpowiedzialnego za sterowanie serwomechanizmem	101
A.3. Schemat komunikatu odpowiedzialnego za akwizycję obrazu z kamery	101
A.4. Schemat komunikatu odpowiedzialnego za odpływanie czujników	102
B.1. Płytkę z naniesionym tonerem	104
B.2. Wytrawiona płytka	105
B.3. Gotowa płytka	105

Spis tabel

5.1.	Charakterystyka mechaniczna żyroskopu L3G4200D dla napięcia zasilania 3.0V i temperatury pracy 25°C	67
5.2.	Opis wyprowadzeń płytki modułowej żyroskopu	68
5.3.	Parametry magnetometru MMC2120	70
5.4.	Opis wyprowadzeń modułu kompasu	71
5.5.	Zachowanie robota w zależności od stanu wykrywanego na czujnikach odległości . .	77

Bibliografia

- [1] *GP2D12 Optoelectronic Device Datasheet*. Sharp Corporation, 2006.
- [2] Acroname robotics. <http://www.acroname.com/robotics/info/articles/sharp/sharp.html>, 2010.
- [3] Wortal robotyki. <http://www.asimo.pl/teoria/robotyka.php>, 2010.
- [4] Dora Sumisławska Adrian Ciz, Marek Gulanowski. Wstępny projekt modułu imu. <http://konar.ict.pwr.wroc.pl/uploads/download/raporty/IMU.pdf>, 2011.
- [5] Isaac Assimov. *Runaround*. Astounding Science Fiction, 1942.
- [6] Jacek Augustyn. *Projektowanie systemów wbudowanych na przykładzie rodziny SAM7S z rdzeniem ARM7TDMI*. Wydawnictwo IGSMiE PAN, 2007.
- [7] dr inż. Tomasz Buratowski. Teoria robotyki. http://www.robotyka.com/teoria_spis.php, 2010.
- [8] A. Gacsadi L. Tepelea I. Gavrilut, V. Tiponut. *Obstacles Avoidance Method for an Autonomous Mobile Robot using Two IR Sensors*. University of Oradea, Politehnica University of Timisoara, 2008.
- [9] Hokuriku Electric Industry. Piezoresistive type 3-axis acceleration sensor application note. <http://www.hdk.co.jp>, 2007.
- [10] Monika Jaworowska. Żywroskopy i akcelerometry mems w elektronice użytkowej. <http://elektronikab2b.pl/technika/>, 2010.
- [11] Monika Jaworowska. Żywroskopy i akcelerometry mems w elektronice użytkowej. <http://elektronikab2b.pl/technika/12098-yroskopy-i-akcelerometry-mems-w-elektronice-uytkowej>, 2010.
- [12] Paweł Kmak. *Rozwój systemu sterowania dla robota mobilnego*. Praca magisterska, Wydział Fizyki i Informatyki Stosowanej, 2009.

- [13] Eugene Kordin. Windows mobile application development. <http://www.mono-project.com/HowToSystemIOPorts>, 2010.
- [14] Przemysław Szulim Piotr Kleczyński. Praktyczne zastosowanie mems-ów w platformach bezzałogowych. <http://elektronikab2b.pl/technika/>, 2010.
- [15] Marcin Pomianowski. Kieszonkowy akcelerometr - miernik przyśpieszenia xyz. Elektronika praktyczna nr 2/2010, 2010.
- [16] Freescale Semiconductor. *Technical Data of Three Axis Low-g Micromachined Accelerometer MMA7260QT*. Freescale Semiconductor, 2008.
- [17] STMicroelectronics. Mem motion sensor:ultra-stable three-axis digital output gyroscope datasheet. http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00265057.pdf, 2010.