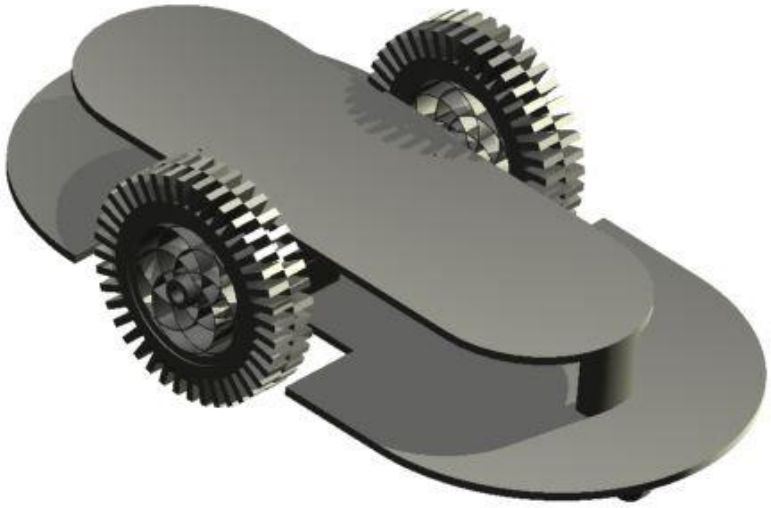
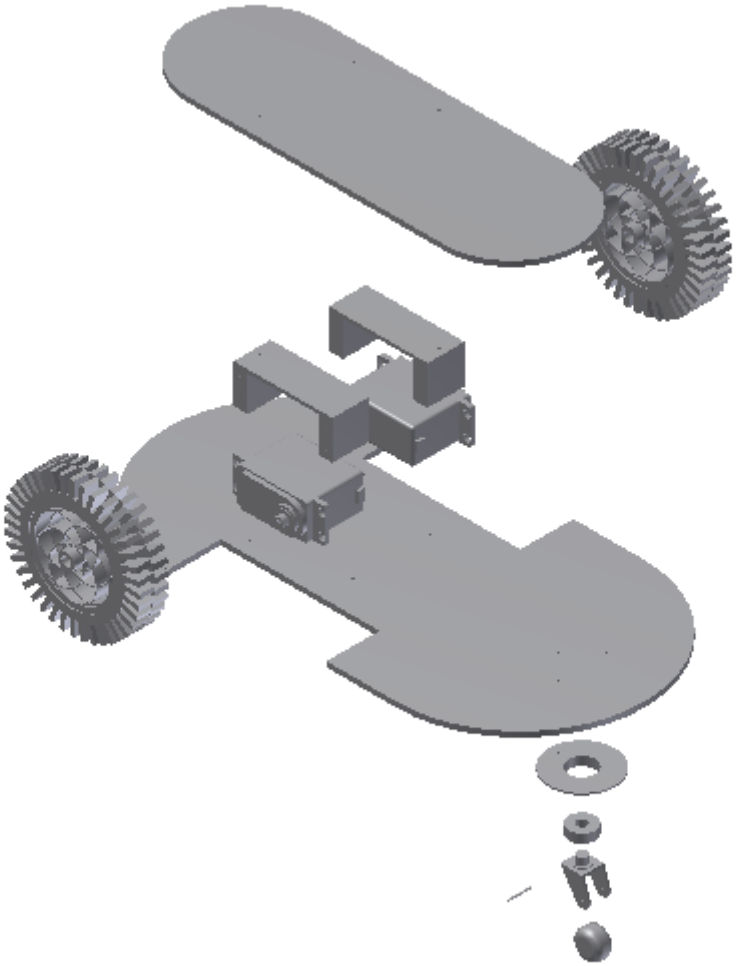
Robot mobilny do celów inspekcyjnych

1. Projekt Konstrukcji

Konstrukcja została zaprojektowana przy użyciu oprogramowanie Autodesk Inventor Professional 2011 w wersji studenckiej. Zaprojektowana konstrukcja składa się z płyt dolnej i górnej wykonanych ze szkła akrylowego (PMMA). Pozostałe elementy zostały zaprojektowane, a następnie wykonane metodą druku 3D GATE firmy 3NOVATICA z materiału PLA. Zaprojektowaną ramę robota przedstawiono na rysunku 1.1 oraz 1.2.



Rys.1.1. Model podwozia robota mobilnego.



Rys.1.2. Model z rozsuniętymi częściami.

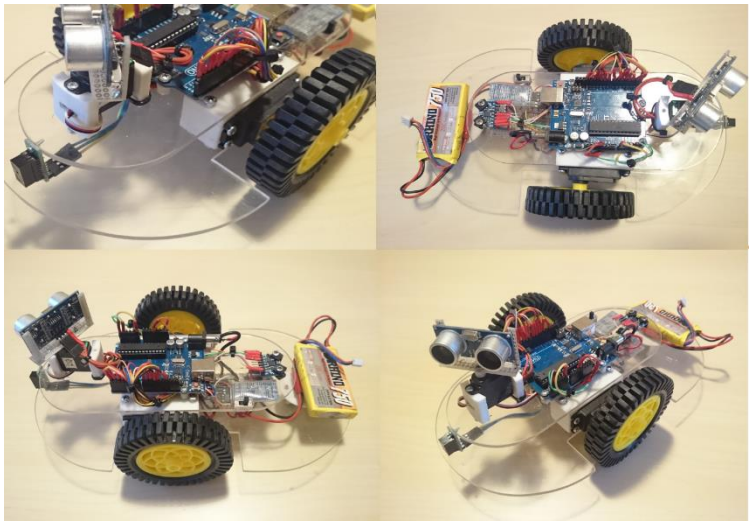
Tab.1.1. Wykaz elementów.

Nazwa	Materiał	Ilość sztuk
Płyta dolna 250x140mm	PMMA	1
Płyta górna 190x80mm	PMMA	1
Mocowanie serwa	PLA	2
Obsada łożyska	PLA	1
Mocowanie kółka	PLA	1
łożysko	-	1
Koło 80x18mm	ABS/guma	2
Koło 16x8mm	-	1
Wkręt 2x10mm	Stal	26

1.2. Konstrukcja robota

Po zaprojektowaniu podwozia robota, rysunki części które miały zostać wydrukowane na drukarce 3D zostały wyeksportowane do formatu .stl dzięki czemu po konwersji programem

Repiter Host, zostały wydrukowane części. Części z PMMA zostały wycięte za pomocą wcześniej przygotowanych szablonów, zostały wycięte ręcznie za pomocą piły brzeszczotowej. Następnie wszystkie otwory zarówno w częściach z PMMA jak i tych wydrukowanych zostały wykonane na wiertarce stołowej wiertłem ϕ2mm. Następnie wszystko zostało poskładane i poskręcane za pomocą wkrętów. W zakupionych kołach o średnicy 80mm należało pogłębić otwory na śruby. Pogłębione zostały wiertłem ϕ5mm na głębokość 4 mm. Łożysko w obsadzie zostało obsadzone za pomocą kleju cyjano-akrylowego, tak samo jak mocowanie kółka w łożysku. Złożonego robota przedstawiają rysunek 1.3.



Rys.1.3 Zmontowany układ mechaniczny oraz elektroniczny robota.

1.3. Napęd robota

Do napędu robota została użyty zespół przekładni z silnikiem DC zintegrowany w jednej obudowie powstały poprzez przerobienie serwomechanizmu modelarskiego. Do przeróbki użyto serwa firmy HITEC HS-325HB typu standard o następujących parametrach:

Tab.1.2. Dane techniczne serwomechanizmu HITEC HS-325HB

Parametr	**Wartość**
Wymiary	40x20x37mm
Masa	43g
Napięcie zasilania	4,8 – 6,0V
Pobór prądu	160mA ruch bez obciążenia dla 4,8V 180mA ruch bez obciążenia dla 6,0V
Moment	0,29Nm dla 4,8V 0,36Nm dla 6,0V
Prędkość	0,19s./60 stopni dla 4,8V 0,15s./60 stopni dla 6.0V

Serwo zostało przerobione następująco: usunięty został ogranicznik, ograniczający ruch serwomechanizmu tylko do 180^0, dzięki temu możliwy jest pełny obrót serwa. Następnie została usunięta cała elektronika (potencjometr oraz moduł kontrolera), wyprowadzone zostały tylko przewody do zasilania silnika DC. Do serw przymocowane zostały koła o średnicy opony 78 mm i szerokości 17 mm. pasujące na wielowypust serwomechanizmu.

Rys.1.4. Koła robota przystosowane do montażu z serwomechanizmami

1.4 Sterownik silników DC

Do sterowania silnikami DC został użyty układ L293D. Bardzo popularny układ scalony bazujący na tranzystorach bipolarnych. Posiada on cztery wejścia sterujące dwoma silnikami, sterowanie odbywa się poprzez odpowiednie ustawienie stanów logicznych na wieściach układu. Dodatkowo zastosowano dwa wejścia ENABLE1 oraz ENABLE2, które służą do sterowania silnikami za pomocą sygnału PWM. Jest to bardzo przydatne do regulacji prędkości.

1.5 Układ zasilania

Układ zasilania składa się z akumulatora litowo-polimerowego oraz układu stabilizatora napięcia. Pakiet akumulatora składa się z dwóch ogniw o napięciu 3,7V i pojemności 750mAh. Połączenie szeregowe powoduje uzyskanie pakietu o napięciu 7,4V. Akumulatory litowo-polimerowe cechują się dużą wydajnością prądową, którą można obliczyć na podstawie wzoru:

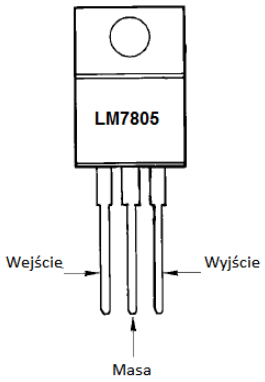
$$I_p = C_p * x \tag{1.1}$$

Gdzie I_p to maksymalna wydajność prądowa pakietu, C_p to pojemność pakietu, a x to liczba, która na pakiecie oznaczona jest literą C. Dla naszego pakietu I_p wynosi na podstawie (1.1):

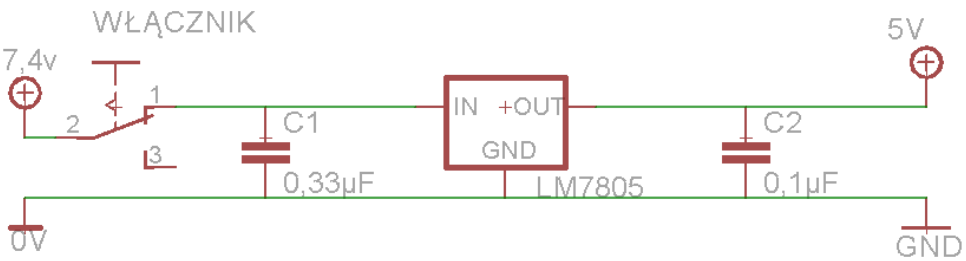
$$I_p = 0,75Ah * 20 = 15A \tag{1.2}$$

Maksymalna wartość napięcia na jednym ogniwie to 4,2V, a minimalna to 3V. Ogniwa litowo-polimerowe są mało odporne na przeładowania dlatego do ładowania używa się ładowarki przystosowane specjalnie do pracy z akumulatorami typu Li-Po. O ile jest taka możliwość zaleca się stosowanie balancera, który zapewnia równomierne naładowanie wszystkich cel akumulatora. Niedopuszczalne jest ładowanie akumulatora powyżej napięcia 4,25V na celę. Za bezpieczną wartość napięcia przyjmuje się 4,2V na celę.

Akumulator podłączony jest poprzez przełącznik suwakowy, który pełni rolę włącznika zasilania układu stabilizatora napięcia. Rolę stabilizatora napięcia pełni układ LM7805. Celem układu stabilizacji napięcia jest uzyskanie stabilnego napięcia stałego, którego wartość nie zmienia się niezależnie od temperatury, czasu, pobieranego prądu, umiejscowienia. Zarówno baterie, akumulatory jaki i zasilacze sieciowe dostarczają napięcie, które jest określone tylko w pewnych granicach. Z kolei, układy elektroniczne powinny pracować w niezmiennych warunkach, a do takich zalicza się, przede wszystkim, zasilanie. Układ LM7805 jest układem stabilizatora liniowego, na jego wejściu podajemy napięcie ze źródła 7-35V, a na wyjściu otrzymujemy ustabilizowane napięcie 5±0,25V, o maksymalnym natężeniu 1,5A. Stabilizatory liniowe cechują się niską ceną oraz są proste w użyciu. Mają one jednak zasadnicze wady, wydzielają ciepło proporcjonalne do różnicy napięć (wej/wyj) i pobieranego prądu oraz nie potrafią wytworzyć napięcia wyższego niż napięcie na wejściu.



Rys.1.6. Układ LM7805



Rys.4.7. Schemat podłączenia stabilizatora LM7805

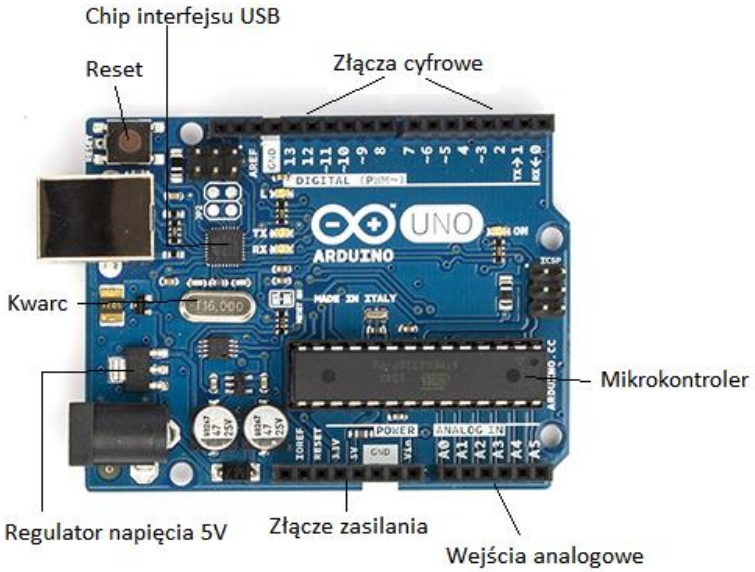
Układ LM7805 został podłączony tak jak powyższym schemacie. Kondensatory C1 oraz C2 zostały zastosowane w celu gromadzenia i uwalnianie elektryczności, co skutkuje wygładzeniem szumu, przepięć i spadków. Bez użycia kondensatorów układ LM7805 nadal będzie dawał napięcie 5V, ale nie będzie tak szybko reagował na zmiany w poborze i dostarczaniu prądu, więc nie zapewni odpowiednio czystego stabilizowanego napięcia.

1.6 Główny kontroler robota- Arduino.

Jako główny kontroler została użyta płytka Arduino, które jest mikrokontrolerem mającym postać płytki wyposażonej w złącze uniwersalnej magistrali szeregowej USB oraz inne wyprowadzenia mogące obsługiwać urządzenia zewnętrzne. Dużą zaletą platformy jest własne, darmowe środowisko z bardzo dużą ilością bibliotek, dodatkowo do programowania nie wymaga żadnego zewnętrznego programatora przez co osoba zaczynająca pracę z mikrokontrolerami może zaoszczędzić dużo czasu na konfiguracji, sprawdzeniu podłączenia oraz instalacji sterowników niezbędnych do działania programatora.

1.6.1 Arduino Uno – opis techniczny

Arduino Uno jest jednym z najprostszych płytek pracujących w platformie Arduino. Zbudowana jest na bazie 8-bitowego mikrokontrolera AVR ATmega328P. Posiada 14 cyfrowych wejść/wyjść, z których 6 może być użyte do sterowaniem sygnałem PWM, 6 wejść analogowych, kwarc 16MHz, port USB, złącze zasilania, przycisk reset, UART, I2C/TWI oraz SPI. Zasilanie może być poprzez USB lub zewnętrzne źródło (akumulator).



Rys.1.8. Płytka Arduino Uno.

Tab. 1.3 Dane techniczne płytki Arduino Uno

Mikrokontroler	ATmega328P
Napięcie zasilania (zalecane)	7-12V
Wyjścia zasilania	3,3V, 5V, Vin, GND
Cyfrowe wejścia/wyjścia	14
Cyfrowe wejścia/wyjścia PWM	6
Wejścia analogowe	6
Prąd wejść/wyjść	20mA
Prąd dla wyjścia	50mA
SRAM	2KB
Pamięć FLASH	32KB
EEPROM	1KB
Częstotliwość taktowania	16MHz
Waga	25g

1.6.2 Arduino – wyprowadzenia

Oprócz standardowych wejść/wyjść cyfrowych wyprowadzenia w Arduino posiadają także pewne funkcję specjalne. Wśród nich możemy wyróżnić:

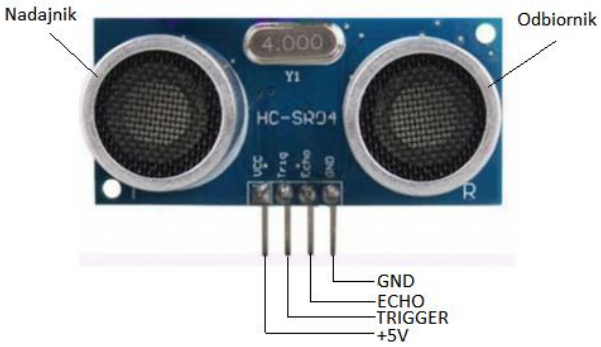
- Interfejs szeregowy UART używany do transmisji poprzez dwie linie RX i TX. Pozwala na wymianę danych z komputerem lub np.: urządzeniem Bluetooth, tak jak tym projekcie. Piny są oznaczone na płytce numerami 0 oraz 1, odpowiednio RX i TX.
- Wyjścia PWM, które dzięki zastosowaniu modulacji szerokości impulsu pozwalają sterować np.: prędkością silnika DC poprzez sygnał o zadanym wypełnieniu. Z projekcie używany jest między innymi do sterowania prędkością silników poprzez układ L293D. Wyjścia zdolne do obsługi PWM oznaczone są symbolem „~”.
- Obsługa przerwań zewnętrznych, umożliwia natychmiastowe przerwanie działającego programu w celu wykonania fragmentu kodu związanego z obsługą przerwania.
- TWI – I2C – interfejs komunikacji synchronicznej, wykorzystywany często do obsługi wyświetlaczy. W tym projekcie wykorzystywany do komunikacji z modułem żyroskopu i akcelerometru GY-521.
- Wejścia analogowe umożliwiają operacje na sygnale analogowym, posiadają wbudowany przetwornik analogowo-cyfrowy o rozdzielczości 10-bitów.

1.7 Wyposażenie robota

Roboty inspekcyjne posiadają przeważnie wiele dodatkowych czujników i rozszerzeń, pozwalających im na pracę w danym środowisku. Istotne są również moduły komunikacyjne, dzięki którym robot jest wstanie wykonywać polecenia operatora oraz dostarczać w czasie rzeczywistym dane na temat otoczenia.

1.7.1 Moduł skanera 2D

Jako czujnik odległości robota od przeszkody został zbudowany prosty skaner 2D, który swoje działanie opiera na ultradźwiękowym czujniku odległości. Czujnik ultradźwiękowy umieszczony jest na serwomechanizmie cyfrowym E-sky EK2-0508, dzięki czemu można zmieniać jego kierunek działania w zakresie 180^0. Jako czujnik ultradźwiękowy został użyty gotowy popularny moduł HC-SR04. Czujnik mierzy odległość od przeszkody w zasięgu 2-200cm. Czujnik zasilany jest napięciem 5V, pobór prądu przez czujnik w trakcie pomiaru to około 15 mA, natomiast w stanie spoczynku czujnik pobiera około 2 mA.

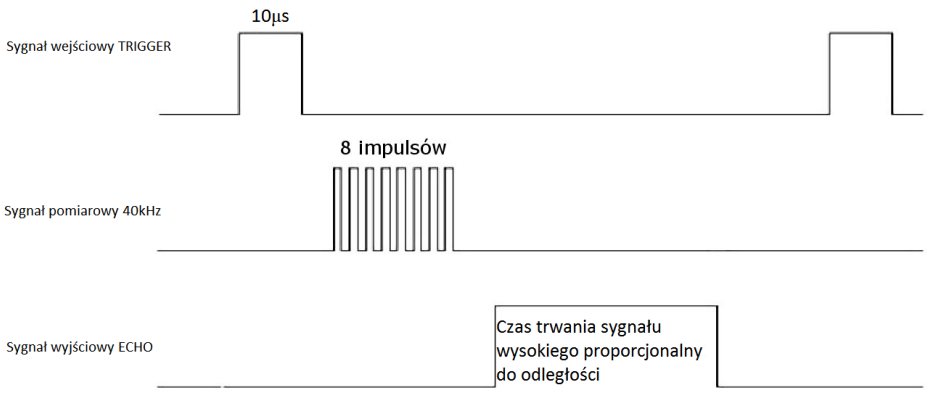


Rys.1.9 Czujnik ultradźwiękowy HC-SR04

Pomiaru dokonuje się podając na pin TRIGGER stan wysoki przez 10µs. Pomiar odległości dokonywany jest przy pomocy fali dźwiękowej o częstotliwości 40kHz. Na wyjściu ECHO podawany jest sygnał, w którym odległość zależy od czasu trwania stanu wysokiego i obliczana jest ze wzoru:

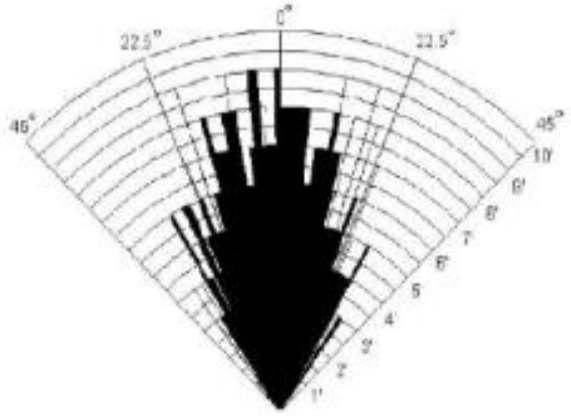
$$\textbf{odległość}[\text{cm}] = \text{czas\_ECHO}[\mu s] / 58 \tag{1.3}$$

Zasada działania została pokazana na poniższym rysunku.



Rys.1.10. Zasada działania czujnika HC-SR04

Czujnik pracuje skutecznie, gdy kąt mierzenia jest równy $\pm 15^0$. Wraz ze wzrostem kąta zmniejsza się zasięg czujnika (rysunek 4.11).

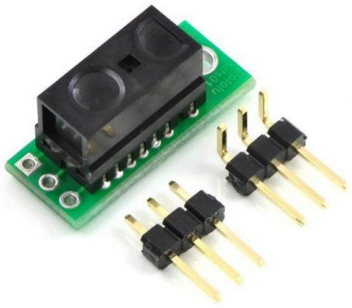


Rys.1.11 Charakterystyka czujnika HC-SR04

Pomiar z obrotem polega na skanowaniu przestrzeni. Czujnik obraca się na serwomechanizmie od 0^0 do 180^0. Pobierane są dane na temat odległości i zapisywane do trzech zmiennych: prawo, lewo, prosto. Następnie zmienne są porównywane, dzięki czemu robot może znaleźć najkorzystniejszą drogę ruchu.

1.7.2 Cyfrowy czujnik odległości

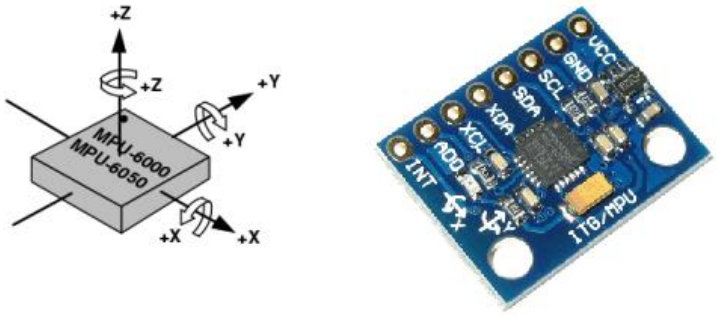
Robot został wyposażony w cyfrowy czujnik odległości Sharp GP2Y0D805Z0F wykrywający obiekty w odległości do 10 cm. Wyjściem jest sygnał cyfrowy, stan wysoki (logiczna jedynka) wskazuje na brak obiektów w polu widzenia czujnika. Stan niski (logiczne zero) pojawia się, gdy obiekt zostanie wykryty. Sensor posiada przylutowaną podstawkę PCB ułatwiająca montaż i korzystanie z czujnika. Płytka posiada niezbędne elementy pasywne do pracy sensora oraz diodę LED sygnalizującą wykrycie obiektu. Najważniejszymi parametrami są: napięcie zasilania od 2,7 V do 6,2V, średni pobór prądu na poziomie 5 mA, zakres pomiarowy od 2 cm do 10 cm oraz czas odpowiedzi 2,55ms. Czujnik został zastosowany w celu zapobiegania kolizji, gdy czujnik ultradźwiękowy nie spełnił swojego zadania detekcji.



Rys.1.12 Cyfrowy czujnik odległości GP2Y0D805Z0F wraz z podstawką

1.7.3 Moduł żyroskopu i akcelerometru.

Moduł żyroskopu i akcelerometru służy do orientowania położenia i ruchu robota. Do tego celu zastosowano moduł GY-521, który jest podstawką pod moduł MPU6050. Jest to układ który łączy w sobie 3-osiowy żyroskop, 3-osiowy akcelerometr oraz cyfrowy termometr. Szczególną cechą jest wbudowany sprzętowo DMP (Digital Motion Processor), który w dużym stopniu ułatwia przeliczanie danych z wszystkich trzech czujników, dzięki czemu odciąża pracę mikrokontrolera i upraszcza kod. Natomiast firma InvenSense produkująca moduł MPU6050 nie upowszechnia informacji na temat kodu oraz algorytmu działania DMP. Żyroskop umożliwia pracę w zakresach pomiarowych $\pm 250^0$/s , $\pm 500^0$/s, $\pm 1000^0$/s oraz $\pm 2000^0$/s. Dla akcelerometru zakresy pomiarowe wynoszą ±2g, ±4g, ±8g oraz ±16g. Za przetwarzanie danych odpowiada 16 bitowy przetwornik analogowo cyfrowy dla każdego kanału. Dodatkowo posiada programowalny filtr dolnoprzepustowy, dodatkową szynę do komunikacji z innymi układami oraz tryb oszczędzania energii. Maksymalny pobór prądu jest mniejszy niż 5,5mA. Układ MPU6050 pracuje w zakresie napięć 2,375-3,46V, dlatego w projekcie została wykorzystana płytka GY-521, która posiada wbudowane konwertery napięć oraz regulator napięcia, przez co konieczne jest zasilanie napięciem 5V. Dane przesyłane są poprzez interfejs I2C, dzięki czemu do komunikacji używa się dwóch złączy SDA i SCL, gdzie MPU6050 jest w trybie slave. Wyjścia XDA oraz XCL służą do kontrolowania zewnętrznym czujnikiem np.: magnetometrem w trybie master. Dodatkowo używane jest wyjście INT, które służy do sygnalizacji mikrokontrolerowi gotowości danych w buforze FIFO do odczytu.[9,10]



Rys.1.13 Po lewej orientacja osi w czujniku MPU6050, z prawej płytka GY-521

1.7.4 Moduł komunikacji bezprzewodowej Bluetooth.

Do komunikacji bezprzewodowej został wykorzystany moduł HC-05 znajdujący się na płytce CZ-HC-05. Moduł bazuje na chipie Cambridge Silicon Radio BC417 wykorzystujący częstotliwość 2,4GHz. Chip korzysta z zewnętrznej pamięci flash o pojemności 8MB. HC-05 jest modułem klasy 2 - maksymalna moc nadajnika +4dBm. Zasięg do około 10m, praca w standardzie Bluetooth 2.0 oraz komunikacja poprzez UART (RX, TX). Standardowo moduł HC-05 zasilany jest napięciem 3,3V oraz wykorzystuje sygnały logicznie z poziomu 3,3V. Płytka CZ-HC-05 wyposażona jest w konwertery napięć poziomów logicznych, stabilizator napięcia obniżający napięcie z 5V do 3,3V, rezystory zabezpieczające linie interfejsu UART, kondensatory filtrujące zasilanie oraz diodę led sygnalizującą stan urządzenia. Istnieje możliwość konfiguracji urządzenia poprzez komendy AT. Konfigurację odbywa się poprzez podanie na wyprowadzenie KEY napięcia 5V, a

następnie poprzez wysyłanie komend za pomocą terminala. Komendy użyte podczas konfiguracji urządzenia:

- AT+NAME:ROBOT INSPEKCYJNY – została zmieniona nazwa urządzenia z domyślnej „HC-05” na „ROBOT INSPEKCYJNY”.
- AT+UART:115200,0,0 - zmienia konfiguracje interfejsu UART prędkość transmisji zmieniona z domyślnej „9600” na „115200”.
- AT+PSWD:0000 – zmiana hasła potrzebnego do parowania urządzenia z domyślnego „1234” na „0000”.
- AT+RMAAD – usunięcie wszystkich wcześniej sparowanych urządzeń.
- AT+ROLE:0 – Ustawienie trybu 0 - Slave, 1 - Master.
- AT+RESET – reset urządzenia po wykonanej konfiguracji.

Rys.1.14. Moduł CZ-HC-05

1.8. Opis podłączenia

Połączenie odpowiednich modułów z płytą Arduino zostało wykonane za pomocą przewodów o przekroju około 0,25mm^2 . Przewody zostały przylutowane do listwy Goldpin oraz zaizolowane koszulkami termokurczliwymi. Moduły zostały połączone z płytką Arduino według tabeli 4.4.

Tab.1.4 Opis połączeń

Nazwa modułu	Nazwa pinu w module	Nr pinu Arduino	Opis
Czujnik ultradźwiękowy HC-SR04	ECHO	10	Sygnał wyjściowy czujnika
	TRIG	9	Sygnał wejściowy czujnika
Sterownik silników L293D	ENABLE1	3	Sygnał PWM dla pierwszego silnika
	ENABLE2	5	Sygnał PWM dla drugiego silnika
	INPUT1	8	Kierunek pierwszego silnika
	INPUT2	7	

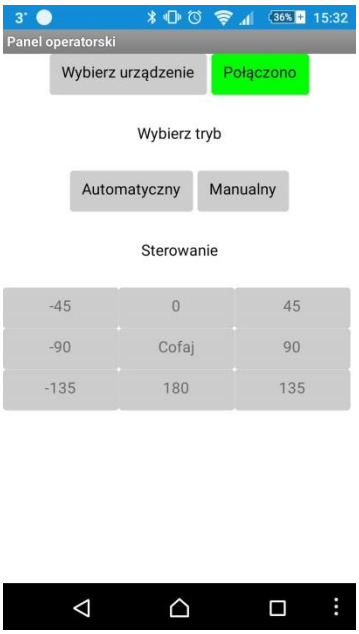
	INPUT3	12	Kierunek drugiego silnika
	INPUT4	4	
Moduł Bluetooth CZ-HC-05	RXD	1 (TX)	Komunikacja UART
	TXD	0 (RX)	
	KEY	11	Używany do konfiguracji modułu
Moduł żyroskopu i akcelerometru GY-521	SCL	A5	Komunikacja I2C
	SDA	A4	
	INT	2	Przerwanie zewnętrzne generowane przez MPU6050
Czujnik odległości Sharp GP2Y0D805Z0F	OUT	A0	Wyjście czujnika odległości
Serwomechanizm E-sky	Pomarańczowy przewód	6	Sygnał PWM serwomechanizmu

2. Algorytm działania programu

Program działania robota działa w dwóch wariantach- sterowanie manualne oraz tryb automatyczny.

2.1 Tryb manualny

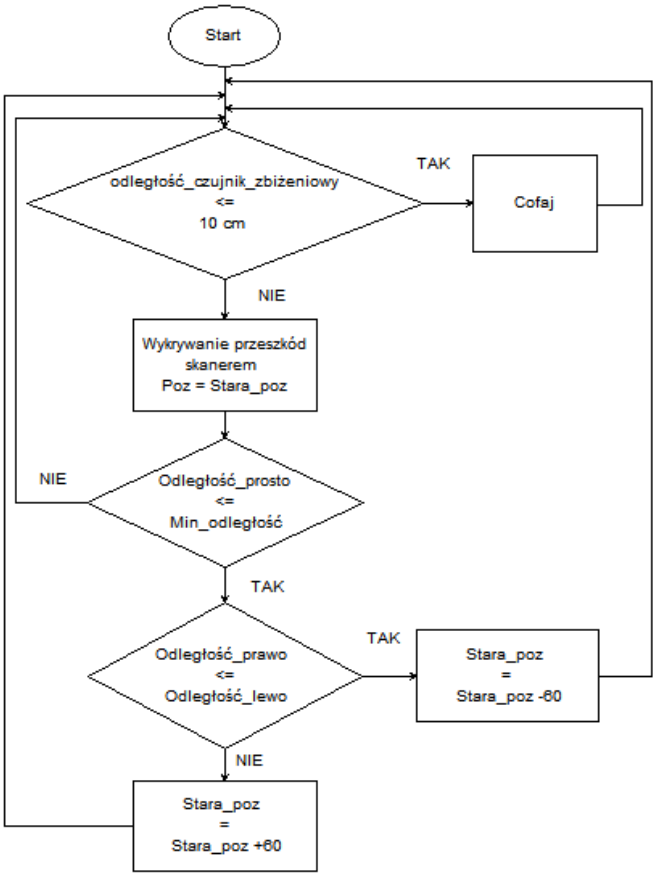
Podczas sterowania manualnego robot porusza się sterowany poprzez użytkownika z wykorzystaniem urządzenia pracującego na platformie Android, wyposażonego w moduł Bluetooth. Do testów użyty został telefon Sony Xperia Z2 z systemem Android w wersji 5.1.1. Aplikacja „Panel operatorski" włączana jest bezpośrednio z pulpitu urządzenia, następnie należy włączyć Bluetooth w urządzeniu oraz włączyć zasilanie robota. Następnie przyciskiem „Wybierz urządzenie" wybieramy urządzenie „ROBOT INSPEKCYJNY". Gdy połączenie nastąpi, pole „Brak połączenia „ zmieni nazwę na pole „Połączono" podświetlone na kolor zielony, dodatkowo zostaną aktywowane przyciski wyboru trybu. Połączenie w robocie sygnalizowane jest poprzez diodę w module CZ-HC-05, urządzenie nie połączone jest sygnalizowane intensywnym miganiem niebieskiej diody co około 0,5s, natomiast gdy połączenie nastąpi niebieska dioda miga co około 2s. Poprzez wybór trybu manualnego aktywowane są przyciski od zadawania orientacji. Sterowanie odbywa się poprzez naciśnięcie wybranej orientacji robota w stopniach, ruch odbywa się w czasie trzymania przycisku. Robot najpierw obraca się do wybranej orientacji a następnie porusza się prosto, wykorzystując odczyt z żyroskopu oraz regulator PID. Po wybraniu trybu „Automatyczny" robot porusza się według zaprogramowanego algorytmu.



Rys 2.1 Zrzut ekranu przedstawiający aplikację „Panel Operatorski”

2.2 Tryb automatyczny.

W trybie automatycznym robot porusza się autonomicznie w środowisku, omijając przeszkody dzięki ultradźwiękowemu skanerowi 2D oraz czujnikowi zbliżeniowemu. Robot skanuje przestrzeń za pomocą czujnika ultradźwiękowego, gdy wykryje przeszkodę na swojej drodze, porównuje odległości z prawej oraz z lewej strony robota, a następnie wybiera drogę, która gwarantuje większą odległość od przeszkody. Dodatkowym zabezpieczeniem przed zderzeniem jest cyfrowy czujnik zbliżeniowy, który po wykryciu przeszkody w zasięgu 10cm od czujnika powoduje cofanie się robota z jednoczesnym skanowaniem otoczenia. Robot po wykryciu przeszkody skręca o zadaną w programie liczbę stopni, a następnie porusza się linią prostą. Zarówno skręt jak i poruszanie się torem prostoliniowym realizowane jest za pomocą odczytu z żyroskopu, który podczas pracy programu mierzy obrót robota z dokładnością do $0{,}01^{o}$. Regulacja prędkości obrotowej kół odbywa się poprzez regulator PID, dla którego wartością wejściową jest zadana orientacja, a sygnałem sprzężenia zwrotnego sygnał z żyroskopu. Sygnałem wyjściowym jest sygnał sterowania PWM do sterowania kołami robota.



Rys.2.2 Schemat blokowy działania programu w trybie automatycznym

2.3 Biblioteki użyte w programie

W programie zostały użyte następujące biblioteki:

- PID_v1.h – biblioteka zawierająca regulator PID
- NewPing.h – biblioteka do obsługi ultradźwiękowego czujnika odległości HC-SR-04.
- Servo.h – biblioteka wykorzystywana do sterowania ruchem serwomechanizmu.
- I2Cdev.h – biblioteka do obsługi interfejsu I2C wykorzystywana przez MPU6050
- MPU6050_6Axis_MotionApps20.h – biblioteka pozwalająca na obsługę modułu MPU6050 za pomocą algorytmu DMP.
- Wire.h – biblioteka pozwala na komunikacje z urządzeniami I2C.

2.4. Główne elementy programu

Najważniejszymi elementami programu tworzonego w środowisku Arduino IDE są funkcje, które są konieczne do działania programu:

- *void setup()* – ustawiane są w niej parametry początkowe systemu. Uruchamiana raz, po resecie lub podłączeniu zasilania
- *void loop()* – pętla główna programu, wykonywana stale.

Wymienione wyżej funkcje są niezbędne w kodzie programu, nawet jeśli nie zawierają żadnego kodu muszą być zdeklarowane. Na rysunku 2.3 została przedstawiona funkcja *jedz_prosto()*, która służy do utrzymywania stałej pozycji oraz sterowania silnikami. Zawiera regulator PID, który korzysta z odczytu żyroskopu, a następnie dostarcza sygnał sterujący PWM, który reguluje prędkość kół robota.

```
void jedz_prosto(){                              //Funkcja do utrzymywania prostoliniowego toru
  gyro();                                        //Użycie funckji gyro() do odczytów żyroskopu
  if (ypr[0]*180/M_PI < Setpoint){
    myPID.SetControllerDirection(DIRECT);        //Zmiana kierunków działania regulatora PID
  }
  else myPID.SetControllerDirection(REVERSE);    //Setpoint - wartość zadana dla PID
  Input =ypr[0]*180/M_PI;                        //Input - wartość wejściowa regulatora - kąt Yaw w stopniach
  myPID.Compute();                               //Wywołanie funkcji liczącej PID
  if (ypr[0]*180/M_PI < Setpoint){               //Sterowanie kołami
    spl= pr_lewa+Output;                         //Output- sygnał sterujący z regulatora PID
    spr= pr_prawa-Output;
  }
  else if (ypr[0]*180/M_PI >= Setpoint){
    spl=pr_lewa-Output;
    spr=pr_prawa+Output;
  }
  if( spr <0 ) spr=0;                            //Ograniczenie min i max sygnałów sterujących
  if( spl <0) spl=0;                             //PWM 0-255
  if( spr >255 ) spr=255;
  if( spl >255) spl=255;
  digitalWrite(8, LOW);                         //Zapis stanów logicznych dla L293d
  digitalWrite(7, HIGH);
  digitalWrite(12, LOW);
  digitalWrite(4, HIGH);
  analogWrite(5, spr);                          //Regulacja prędkośći za pomocą PWM
  analogWrite(3, spl);
}
```

Rys.2.3 Funkcja *jedz_prosto()*

Funkcja *gyro()* korzysta z gotowego algorytmu DMP, najważniejszym elementem tej funkcji jest polecenie *mpu.dmpGetYawPitchRoll(ypr, &q, &gravity)*, które zwraca pozycję za pomocą kątów RPY, dla nas najbardziej intersujący jest kąt YAW, który zapisany jest jako pierwszy element tablicy ypr – *ypr[0]*.

Kolejną funkcja jest *pobierz_odległość()*, która obsługuje czujnik ultradźwiękowy przy pomoc biblioteki NewPing.h

```
void pobierz_odleglosc(){
  runEvery(40){                          //pętla wywoływana co 40us
    uS = sonar.ping();                   //pobranie sygnału z czujnika
    distance = uS / US_ROUNDTRIP_CM;     //wyliczanie odległośći w cm
    if (uS == NO_ECHO){                  // jeśli sensor nie odbierze sygnalu
      distance = MAX_DISTANCE;           //odległość musi byc rowna maksymalnej
    }
    // Serial.print("Ping: ");           //wyświetlanie wartości w serial monitorze
    // Serial.print(distance);
    // Serial.println("cm");
  }
}
```

Rys.2.4. Funkcja *pobierz_odległość()*

Dzięki funkcji *obrót_serwa()* (rys.2.5.), uzyskujemy dane, które wykorzystujemy do funkcji *sterowanie(),* która w zależności od położenia przeszkód dodaje lub odejmuje wartości od pozycji zadanej *Setpoint*. Powoduje to zmianę kierunku jazdy robota a przy tym omijanie przeszkód.

```
void obrot_serwa(){
  runEvery(servoDelay){  //pętla wykonywana co określony przez servoDelay czas
    if(pos < 165 && kierunek_servo == 0){    // 165 = serwo do lewej
      pos = pos + 5;
    }
    if(pos > 15 && kierunek_servo == 1){    // 15 = serwo do prawej
      pos = pos - 5;
    }
  }
  if (pos == 165 ){
    kierunek_servo = 1;  //zmiana kierunku
  }
  if (pos == 15 ){
    kierunek_servo = 0;  //zmiana kierunku
  }
  myservo.write(pos);  //zapis pozycji do serwa

}
```

Rys.2.5 Funkcja *obrót_serwa()*.

2.5. Regulator PID

Regulator PID oblicza wartość uchybu jako różnicę między wartością zadaną Setpoint, a wartością odczytu z żyroskopu – Input. Jego zadaniem jest redukcja uchybu i poprawa właściwości dynamicznych układu poprzez generowanie sygnału wejściowego do obiektu regulacji, sygnał Output.

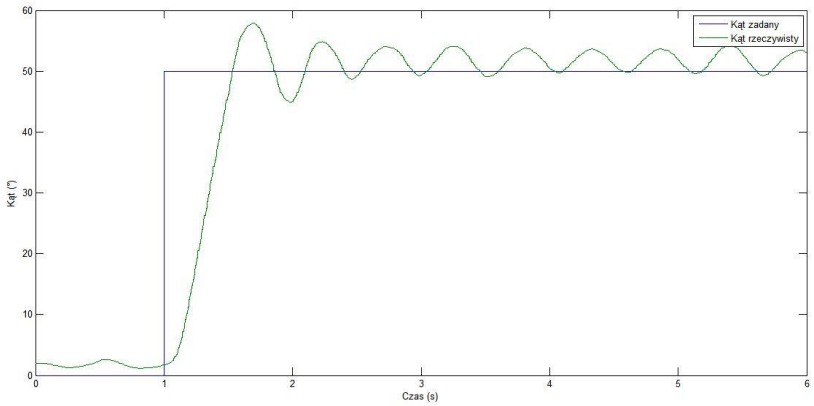
$$Output = K_P e(t) + K_I \int e(t)dt + K_D \frac{d}{dt} e(t) \tag{2.1}$$

Gdzie:

$$e = Setpoint - Input \tag{2.2}$$

W programie został wykorzystany gotowy regulator PID, wraz z biblioteką. Zadaniem do wykonania było dostrojenie parametrów K_P, K_D oraz K_I. Do badania uchybu regulacji oraz właściwości układu wykorzystałem program CoolTerm oraz Matlab 7.12.0. Zmodyfikowano program pracy robota, który miał poruszać się w orientacji 0^o , następnie wykonać obrót do wartości 50^o i poruszać się dalej w tej orientacji. W czasie pracy robota rejestrowane były następujące dane: czas, Setpoint, oraz Input czyli odczyt z żyroskopu pozycji aktualnej. Dane były pobierane do komputera za pomocą kabla USB podłączonego do płyty Arduino i rejestrowane za pomocą programu CoolTerm, który zapisywał dane do pliku. Następnie dane zostały przetworzone w programie Matlab, w którym wykonał wykresy wartości zadanej oraz

odpowiedzi układu w funkcji czasu. Nastawy regulatora PID próbowano dobrać metodą Zieglera-Nicholsa. Jest to tzw. metoda drgań krytycznych. Polega ona na ustawieniu regulatora na działanie proporcjonalne i stopniowym zwiększaniu współczynnika wzmocnienia dochodząc do granicy stabilności. W czasie wystąpienia oscylacji (rysunek 2.6) zmierzono ich okres Tosc=0,52s oraz współczynnik wzmocnienia Kkr=11, przy jakim one wystąpiły.

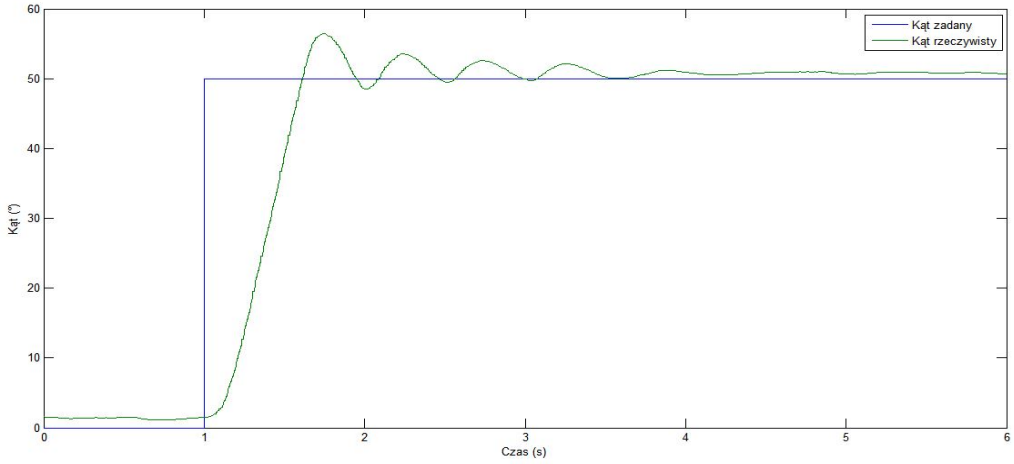


Rys. 2.6. Wykres przedstawiający oscylację układu.

Nastawy zostały obliczone według parametrów dla regulatora PID:

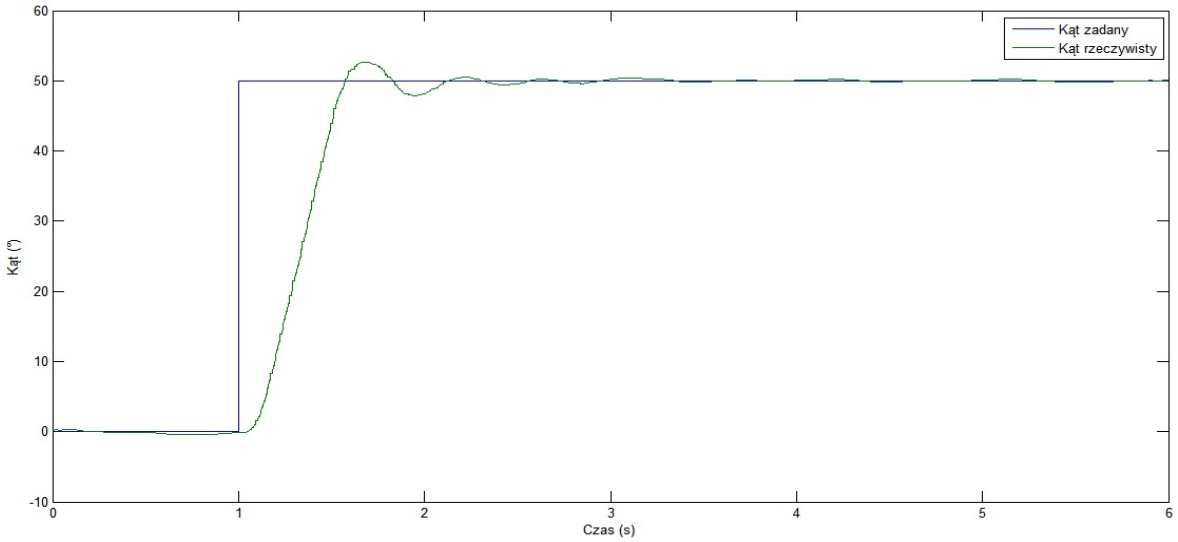
$$K_P = 0,6 \cdot K_{kr} = 6,6$$
$$T_I = 0,5 \cdot T_{osc} = 0,26$$
$$T_D = 0,125 \cdot T_{osc} = 0,065 \qquad (2.3)$$

Po zastosowaniu nastaw otrzymano przebieg jak na rysunku 2.7. Kryterium Ziglera-Nicholsa nie zapewnia odpowiedniej jakości regulacji.



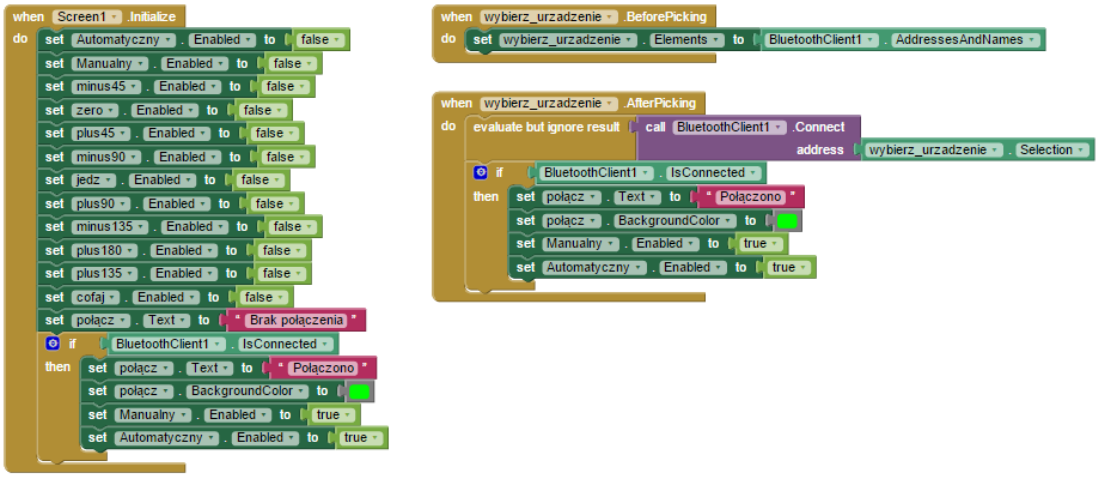
Rys. 2.7. Wykres przedstawiający przebieg po zastosowaniu nastaw na podstawie kryterium Ziglera-Nicholsa

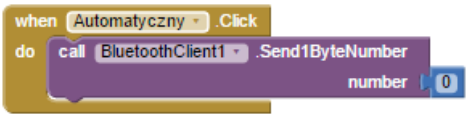
W celu uzyskania odpowiedniej jakości regulacji przeprowadzono próby dostrojenia regulatora ręcznie. Przebieg pokazany na rysunku 2.8 jest efektem końcowym strojenia. Uzyskane nastawy zapewniają przebieg o pożądanych własnościach regulacji, małe przeregulowanie oraz krótki czas regulacji.

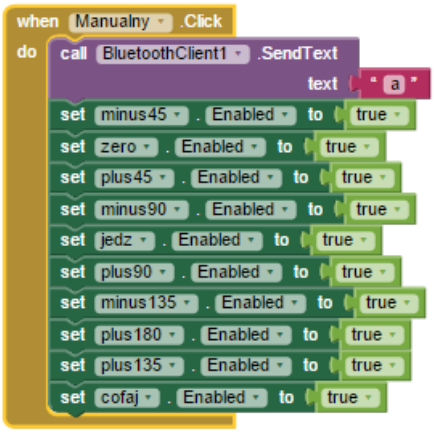


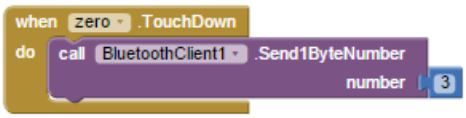
Rys.2.8. Wykres przedstawiający przebieg po dostrajeniu regulatora

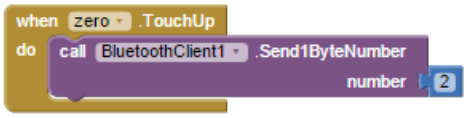
2.6. Kod programu App Inventor 2

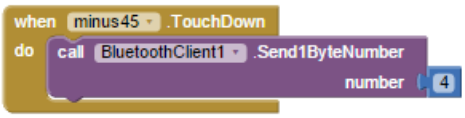


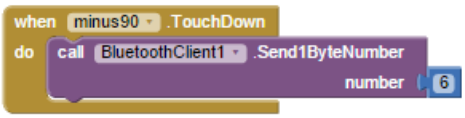

when Automatyczny .Click
do call BluetoothClient1 .Send1ByteNumber
number 0

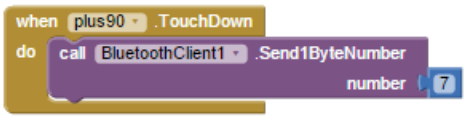

when Manualny .Click
do call BluetoothClient1 .SendText
text " a "
set minus45 . Enabled to true
set zero . Enabled to true
set plus45 . Enabled to true
set minus90 . Enabled to true
set jedz . Enabled to true
set plus90 . Enabled to true
set minus135 . Enabled to true
set plus180 . Enabled to true
set plus135 . Enabled to true
set cofaj . Enabled to true

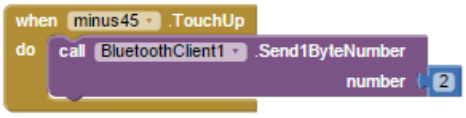

when zero .TouchDown
do call BluetoothClient1 .Send1ByteNumber
number 3

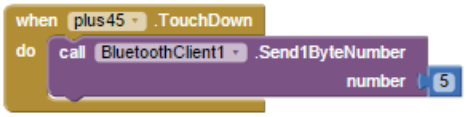

when zero .TouchUp
do call BluetoothClient1 .Send1ByteNumber
number 2

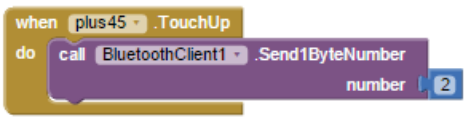

when minus45 .TouchDown
do call BluetoothClient1 .Send1ByteNumber
number 4

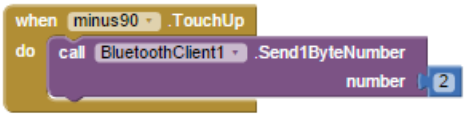

when minus90 .TouchDown
do call BluetoothClient1 .Send1ByteNumber
number 6

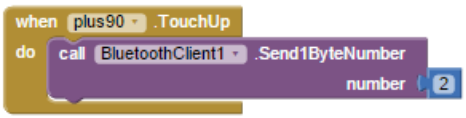

when plus90 .TouchDown
do call BluetoothClient1 .Send1ByteNumber
number 7

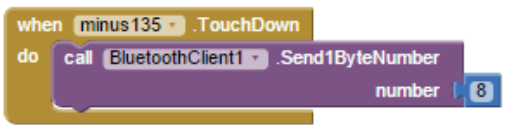

when minus45 .TouchUp
do call BluetoothClient1 .Send1ByteNumber
number 2

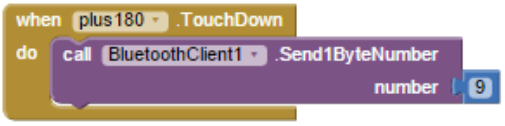

when plus45 .TouchDown
do call BluetoothClient1 .Send1ByteNumber
number 5

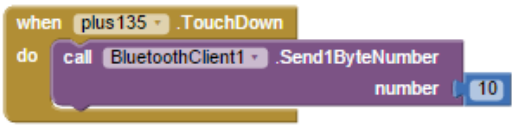

when plus45 .TouchUp
do call BluetoothClient1 .Send1ByteNumber
number 2

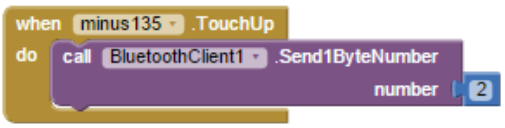

when minus90 .TouchUp
do call BluetoothClient1 .Send1ByteNumber
number 2

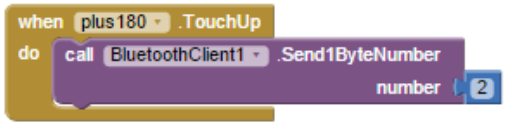

when plus90 .TouchUp
do call BluetoothClient1 .Send1ByteNumber
number 2

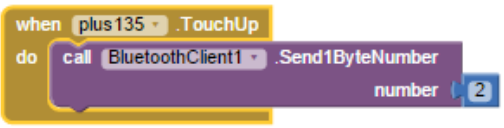

when minus135 .TouchDown
do call BluetoothClient1 .Send1ByteNumber
number 8


when plus180 .TouchDown
do call BluetoothClient1 .Send1ByteNumber
number 9


when plus135 .TouchDown
do call BluetoothClient1 .Send1ByteNumber
number 10


when minus135 .TouchUp
do call BluetoothClient1 .Send1ByteNumber
number 2


when plus180 .TouchUp
do call BluetoothClient1 .Send1ByteNumber
number 2


when plus135 .TouchUp
do call BluetoothClient1 .Send1ByteNumber
number 2