



Wydział Automatyki, Elektroniki i Informatyki

Projekt z Systemów Mikroprocesorowych
Ramię robota

Mateusz Siedliski i Radosław Tchórzewski
Rok akademicki 2022/2023, semestr 5, grupa 6, sekcja 2

Kierujący pracą: dr inż. Krzysztof Jaskot

Gliwice 2023

Spis treści

1	DONE? Wstęp	2
1.1	DONE? Cel i zakres projektu	2
	DONE? Cel projektu	2
	DONE? Wymagania	3
	DONE? Zakres projektu	3
2	DONE? Harmonogram	4
2.1	DONE Harmonogram zatwierdzony	4
2.2	DONE? Harmonogram wykonany	4
3	DONE? Kosztorys	5
4	WIP Urządzenie wraz z aplikacją	6
4.1	DONE? Określenie problemu	6
4.2	DONE? Analiza rozwiązań	7
4.3	DONE? Zaproponowane rozwiązanie	8
4.4	WIP Wykonanie	9
	DONE? Wybór elementów	9
	DONE? Model 3D	10
	DONE? Konstrukcja mechaniczna	13
	DONE? Schemat elektryczny	15
	DONE? Aplikacja	17
	DONE? Protokół komunikacyjny	20
	DONE? Kod mikrokontrolera	21
	DONE? Kontrola wersji	25
	WIP Dokumentacja	26
4.5	TODo Problemy w trakcie tworzenia sprzętu i aplikacji . . .	27
5	TODo Podsumowanie	28
6	WIP Literatura	29
7	WIP Załączniki	30

1 **DONE? Wstęp**

Podczas studiowania na kierunku Automatyka i Robotyka można zauważać zadziwiający brak fizycznych pomocy naukowych. Ten projekt ma na celu poprawę tej sytuacji choćby w niewielkim stopniu. W tym celu zaproponowano stworzenie ramienia robota (manipulatora). Ma on na celu pomóc studentom z wizualizacją koncepcji teoretycznych w prawdziwym świecie, a nie tylko w książkach, czy na ekranie komputera.

Projekt może posłużyć także do zachęcenia potencjalnych studentów podczas na przykład dni otwartych czy wycieczek szkolnych.

Główną inspiracją projektu był film zamieszczony na platformie YouTube z kanału „How To Mechatronics“ pod tytułem „DIY Arduino Robot Arm with Smartphone Control“ [1].

Nasz projekt korzysta z tych samych technologii, aczkolwiek wszystkie elementy (model 3D, oprogramowanie mikrokontrolera, kod aplikacji, schemat połączeń itd.) zostały przygotowane przez nas.

W ramach projektu stworzono dydaktyczny model 5 osiowego manipulatora z chwytkiem, zrealizowanego w technologii druku 3D, sterowany aplikacją na urządzenia z systemem Android.

1.1 **DONE? Cel i zakres projektu**

DONE? Cel projektu

Celem projektu była realizacja fizycznego modelu manipulatora przeznaczonego do celów dydaktycznych. Może być on pomocny dla studentów w celu wizualizacji koncepcji teoretycznych w prawdziwym świecie, a nie tylko w książkach, czy na ekranie komputera.

Projekt może posłużyć także do zachęcenia potencjalnych studentów podczas na przykład dni otwartych czy wycieczek szkolnych.

DONE? Wymagania

- Niski koszt budowy.
- Niski koszt eksploatacji.
- Stworzony z łatwodostępnych materiałów.
- Łatwość obsługi.
- Niski koszt szkolenia.
- Niska awaryjność.
- Łatwość naprawy.
- Atrakcyjny wygląd.

DONE? Zakres projektu

- Określenie problemu i wykonanie do niego założeń.
- Analiza możliwych rozwiązań.
- Wybór elementów elektronicznych.
- Wybór elementów mechanicznych.
- Wykonanie projektu zgodnie z wcześniejszymi założeniami.
- Uruchomienie, weryfikacja i przetestowanie sprzętu oraz aplikacji.
- Nakreślenie ewentualnych kierunków rozwoju projektu.
- Wnioski końcowe.

2 DONE? Harmonogram

2.1 DONE? Harmonogram zatwierdzony

1. Projektowanie modelu fizycznego robota oraz jego druk w technologii 3D.
2. Montaż mechaniczny oraz elektryczny.
3. Tworzenie oprogramowania na mikrokontroler.
4. Tworzenie aplikacji sterującej.
5. Projektowanie oraz realizacja komunikacji między mikrokontrolerem, a aplikacją sterującą.

2.2 DONE? Harmonogram wykonany

Realizacja działającego prototypu zajęła zdecydowanie mniej czasu, niż początkowo zakładano. Pozwoliło to na poświęcenie większej ilości czasu na uprawnienia i udoskonalanie projektu.

1. Projektowanie modelu fizycznego robota oraz jego druk w technologii 3D. Montaż mechaniczny oraz elektryczny.
2. Tworzenie oprogramowania na mikrokontroler oraz aplikacji sterującej. Opracowanie protokołu komunikacji między mikrokontrolerem, a aplikacją sterującą.
3. Doskonalenie projektu — debugowanie
4. Doskonalenie projektu — poprawki mechaniczne, usprawnienia
5. Tworzenie dokumentacji

3 DONE? Kosztorys

W nawiasach [] podano link do dokumentacji.

Lp.	Typ	Producent	Ilość	Cena	Wartość
1.	Mikrokontroler Wemos D1 mini [6]	Wemos	1	9,48 zł	9,48 zł
2.	Moduł Bluetooth HC-05 [7, 8]	SZYTF	1	9,40 zł	9,40 zł
3.	Micro Servo 9g SG90 [9]	HWAYEH	3	3,97 zł	11,91 zł
4.	Servo Mg996r [10]	WAVGAT	3	12 zł	36 zł
5.	Obudowa ramienia (druk 3D)	n/d	1	40 zł	40 zł
6.	Przewody	n/d	n/d	n/d	10 zł
7.	Płytnica prototypowa	diymore	1	3,15 zł	3,15 zł
8.	Wkręty M3	n/d	4	0,20 zł	0,80 zł
9.	Śruby M3	n/d	8	0,20 zł	1,60 zł
10.	Nakrętki M3	n/d	18	0,20 zł	3,60 zł
11.	Drewniana podstawa	n/d	1	10,00 zł	10,00 zł
12.	Kondensator 1000 μF	Chong	1	0,50 zł	0,50 zł
					Suma = 136,44 zł
					Ilość roboczogodzin = 60

Tabela 1: Kosztorys

4 WIP Urządzenie wraz z aplikacją

Tworząc ten projekt mamy na celu stworzenie przydatnej pomocy naukowej dla osób zainteresowanych tematem automatyki i robotyki, który pozwoli na łatwiejsze przyswojenie teorii oraz zobaczenie jej praktycznego zastosowania.

4.1 DONE? Określenie problemu

Problemem większości studentów jest często brak motywacji do nauki związany z poczuciem braku sensu przerabianego materiału. Ten projekt ma na celu poprawić ten stan rzeczy. Brak wyobrażenia o realnym zastosowaniu zdobytej wiedzy utrudnia pracę. Studenci nie widzą połączenia teorii z praktyką, jak przedstawiono na Rysunku 1.

Rozwiążanie? Stworzenie fizycznego modelu.

$$A_i = \text{Rot}(z, \theta_i) \text{Trans}(0, 0, \lambda_i) \text{Trans}(l_i, 0, 0) \text{Rot}(x, \alpha_i) =$$

$$= \begin{bmatrix} C_i & -S_i C_{\alpha_i} & S_i S_{\alpha_i} & l_i C_i \\ S_i & C_i C_{\alpha_i} & -C_i S_{\alpha_i} & l_i S_i \\ 0 & S_{\alpha_i} & C_{\alpha_i} & \lambda_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[2]

⇓

?

⇓



Rysunek 1: Połączenie teorii z praktyką

4.2 DONE? Analiza rozwiązań

Przykładowe rozwiązania przedstawionego problemu:

- Wycieczki do zakładów przemysłowych
 - + Obserwacja prawdziwych zastosowań w praktyce
 - Zmiana organizacji roku akademickiego
 - Organizacja wyjazdu
- Więcej zajęć praktycznych
 - + Praca na urządzeniach pozwoli lepiej przyswoić wiedzę
 - Zmiana organizacji roku akademickiego
 - Wysoki koszt zakupu urządzeń
 - Wysoki koszt kadrowy
- Dopuszczenie studentów do pracy na prawdziwym sprzęcie już od początku studiowania
 - + Motywacja od wczesnych etapów edukacji
 - Zmiana organizacji roku akademickiego
 - Wysoki koszt zakupu urządzeń
 - Wysoki koszt kadrowy
- Stworzenie modelu prawdziwego urządzenia
 - + Niski koszt
 - + Prostota realizacji
 - Nakład pracy włożony w stworzenie modelu

Kryteria wyboru rozwiązania:

- Niskie koszta finansowe
- Niskie koszta kadrowe
- Niska ingerencja w organizację roku akademickiego
- Wysoka dostępność
- Motywacja studentów

4.3 DONE? Zaproponowane rozwiązanie

W tym punkcie należy przedstawić wybrane rozwiązanie problemu, wraz z uzasadnieniem wyboru na postawie kryteriów z poprzedniego punktu.

Wybrane rozwiązanie: „Stworzenie modelu prawdziwego urządzenia“.

Ta propozycja spełnia wszystkie wymagania projektowe:

- Niskie koszta finansowe
 - Koszt < 150 zł
- Niskie koszta kadrowe
 - Uczelnia już na chwilę obecną posiada wyszkoloną kadrę
- Niska ingerencja w organizację roku akademickiego
 - Niewielka ilość godzin potrzebna na pracę z modelem
- Wysoka dostępność
 - Niski koszt i dostępność materiałów pozwala na stworzenie wielu modeli
- Motywacja studentów
 - Dostęp do fizycznego modelu

4.4 **WIP** Wykonanie

DONE? Wybór elementów

Prace rozpoczęto od doboru elementów elektronicznych i mechanicznych oraz wyboru procesu technologicznego wykonania manipulatora. Poniżej przedstawiono argumentację wyboru najważniejszych elementów.

Elementy:

- Mikrokontroler Wemos D1 mini [6]
- Moduł Bluetooth HC-05 [7, 8]
- HWAYEH Micro Servo 9g SG90 [9]
- WAVGAT Servo Mg996r [10]

wybrano ze względu na:

- Niski koszt
- Duża dostępność
- Zaznajomienie autorów z elementem

Jako proces technologiczny wykorzystany do stworzenia korpusu urządzenia wybrano druk 3D ze względu na niski koszt i powszechną dostępność. Technologia ta również umożliwia optymalizację procesu twórczego poprzez wielokrotne iteracje.

DONE? Model 3D

Następnym krokiem było zaprojektowanie modelu 3D manipulatora przy użyciu programu Autodesk Fusion 360¹.

Program Autodesk Fusion 360 to bardzo przystępna alternatywa dla środowiska Autodesk Inventor. Przy braku poprzedniego doświadczenia autorzy byli w stanie całkowicie od zera zaprojektować i zrealizować w pełni działający model manipulatora. Proces tworzenia korpusu pokazano na Rysunkach 2 i 3², a gotowy model przedstawiono na Rysunku 4.



Rysunek 2: Proces tworzenia modelu 3D - Widok z boku

¹Zintegrowane oprogramowanie CAD, CAM, CAE i PCB

²Wizualizacje stworzono w programie Blender

Ramię robota



Rysunek 3: Proces tworzenia modelu 3D - Widok z góry

Ramię robota



 **AUTODESK** Viewer

 **AUTODESK**

Rysunek 4: Gotowy model w programie [Autodesk Viewer](#)
Narzędzie online do wyświetlania plików 2D i 3D

DONE? Konstrukcja mechaniczna

Model złożono przy użyciu śrub, wkrętów i nakrętek M3, co przedstawiono na Rysunkach 5 i 6.



Rysunek 5: Konstrukcja mechaniczna - chwytak

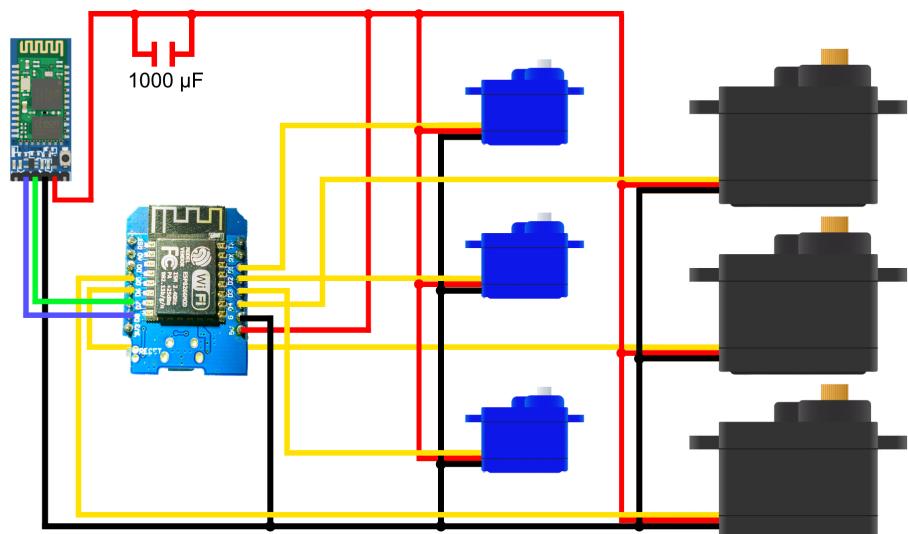
Ramię robota



Rysunek 6: Konstrukcja mechaniczna - podstawa

DONE? Schemat elektryczny

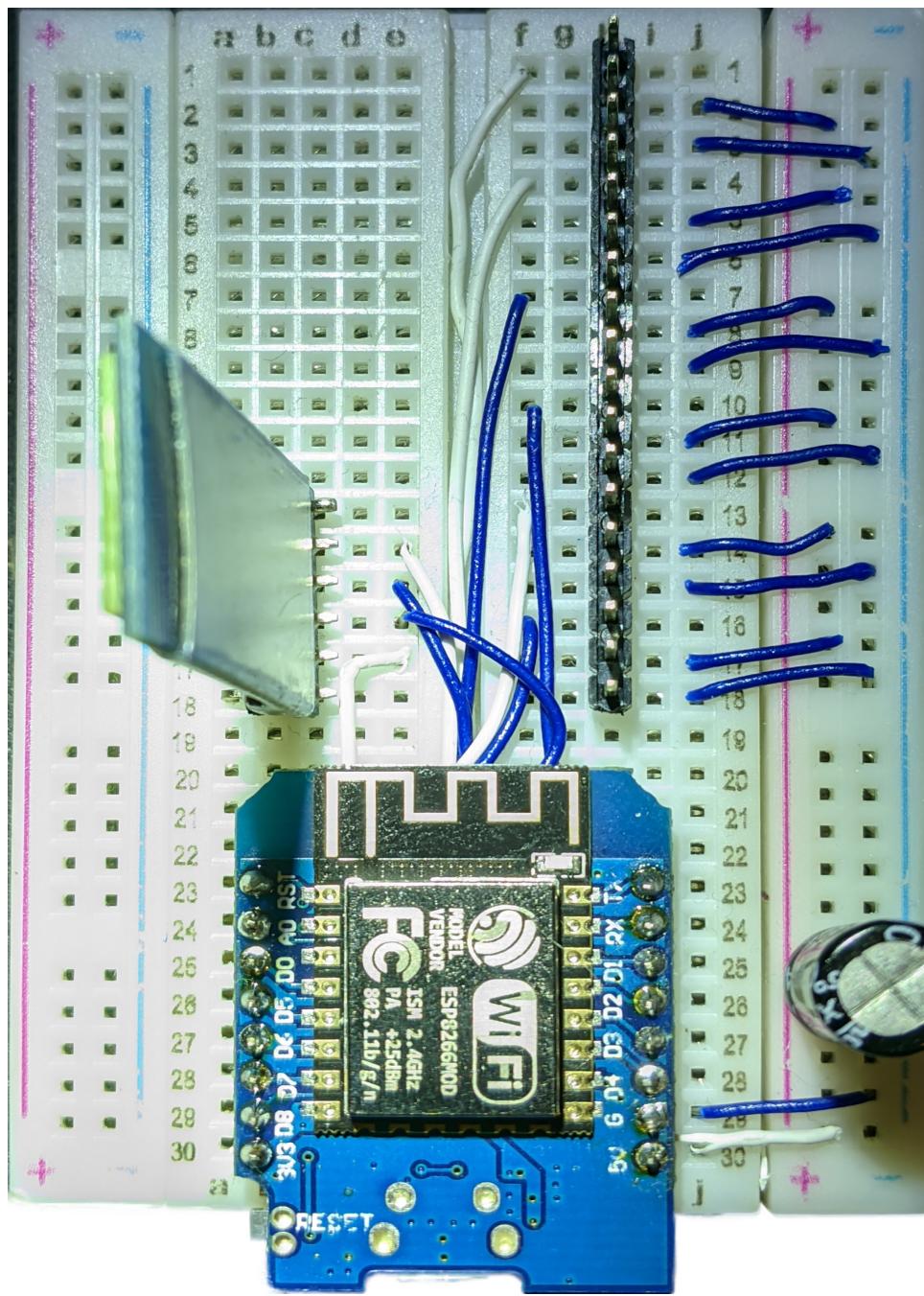
Na Rysunku 7³ przedstawiono połączenie elementów elektronicznych. Natomiast faktyczne połączenia elektryczne pokazano na Rysunku 8.



Rysunek 7: Schemat połączeń elektrycznych

³Stworzono przy użyciu [circuito.io](#)

Ramię robota



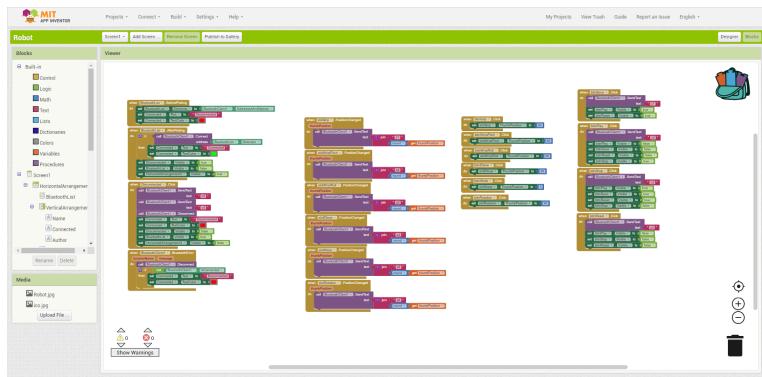
Rysunek 8: Połączenia elektryczne

DONE? Aplikacja

Aplikacja powstała w MIT App Inventor⁴ (Rysunki 9 i 10). Środowisko to oferuje prostotę w realizacji projektów. Nie wymaga żadnego doświadczenia od użytkownika. Programowanie odbywa się we własnym języku graficznym (duże podobieństwo do Scratch⁵). Kod dostarczonej aplikacji pokazano na Załącznikach 2, 3, 4 i 5. Na Rysunku 11 przedstawiono ostateczną szatę graficzną aplikacji, a na Rysunku 12 zademonstrowano logo aplikacji.



Rysunek 9: MIT App Inventor - aplikacja

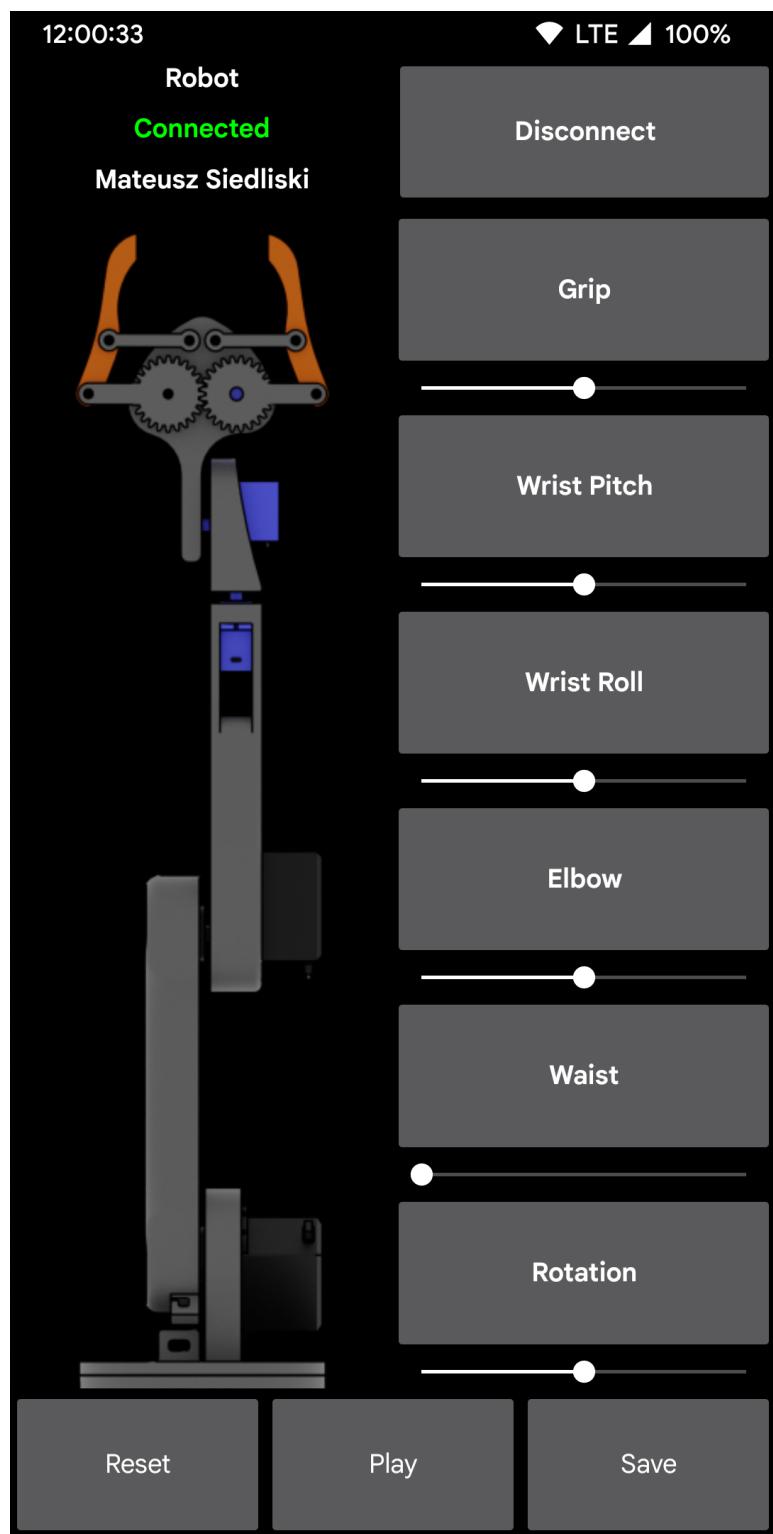


Rysunek 10: MIT App Inventor - kod

⁴Zintegrowane środowisko programistyczne do tworzenia aplikacji mobilnych

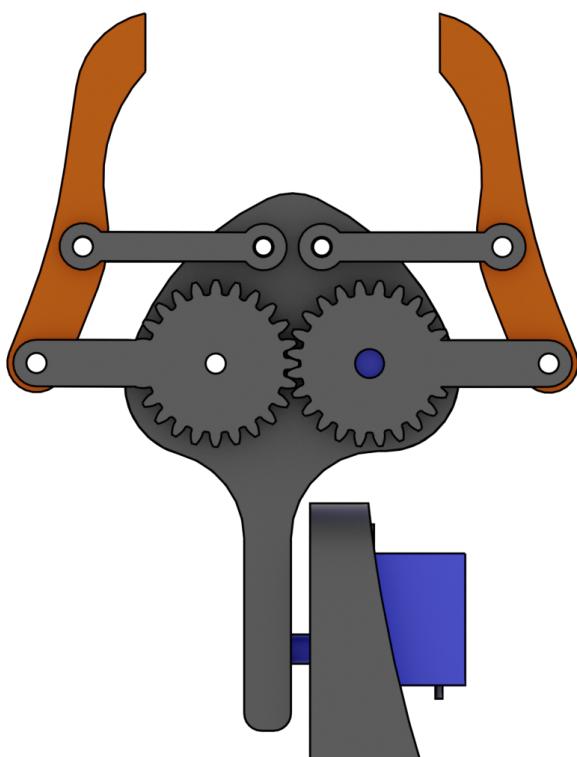
⁵Interpretowany wizualny język programowania

Ramię robota



Rysunek 11: Gotowa aplikacja

Ramię robota



Rysunek 12: Logo aplikacji

DONE? Protokół komunikacyjny

Komunikacja aplikacji mobilnej z mikrokontrolerem odbywa się poprzez protokół Bluetooth przy użyciu modułu HC-05. Każdy komunikat to odpowiednio sformatowany string. Komendy komunikacyjne przedstawiono w Tabeli 2. Przykładowe komunikaty:

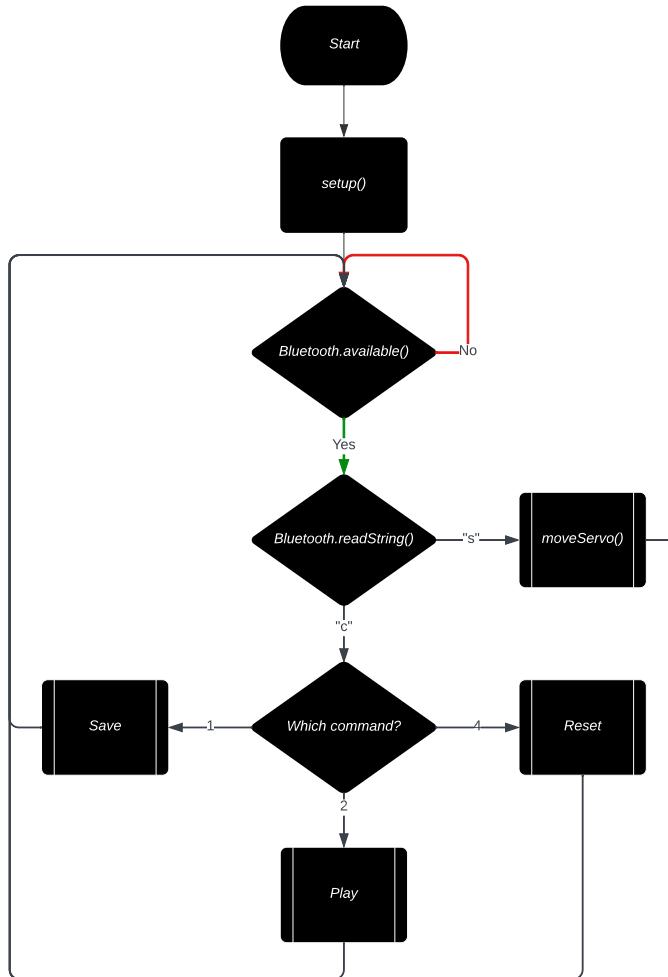
- „s1128\0“ - ustawienie serwomechanizmu 1 na pozycję 128
- „s564\0“ - ustawienie serwomechanizmu 5 na pozycję 64
- „c1\0“ - zapisanie aktualnej pozycji serwomechanizmów w pamięci do późniejszego odtworzenia
- „c4\0“ - wyczyszczenie zapisanej sekwencji ruchowej

Komenda	Numer	Wartość	Funkcja
s	1	x	Ustawienie serwomechanizmu na pozycję x
s	2	x	Ustawienie serwomechanizmu na pozycję x
s	3	x	Ustawienie serwomechanizmu na pozycję x
s	4	x	Ustawienie serwomechanizmu na pozycję x
s	5	x	Ustawienie serwomechanizmu na pozycję x
s	6	x	Ustawienie serwomechanizmu na pozycję x
c	1	-	Save - zapisanie aktualnej pozycji serwomechanizmów w pamięci do późniejszego odtworzenia
c	2	-	Play - odtwarzanie zapisanej sekwencji ruchowej
c	3	-	Stop - koniec odtwarzania zapisanej sekwencji ruchowej
c	4	-	Reset - wyczyszczenie zapisanej sekwencji ruchowej

Tabela 2: Protokół komunikacyjny

DONE? Kod mikrokontrolera

Kod sterujący działaniem mikrokontrolera powstał w Visual Studio Code⁶ przy pomocy PlatformIO⁷. Schemat blokowy kodu przedstawiono na rysunku 13⁸. Całość kodu źródłowego znajduje się w Załączniku 1.



Rysunek 13: Schemat blokowy kodu

⁶Darmowy edytor kodu źródłowego z kolorowaniem składni dla wielu języków, stworzony przez Microsoft o otwartym kodzie źródłowym

⁷ „A user-friendly and extensible integrated development environment with a set of professional development instruments, providing modern and powerful features to speed up yet simplify the creation and delivery of embedded products”[3]

⁸Schemat wykonano w programie Lucidchart

W celu ograniczenia prędkości poruszania się każdego z serwomechanizmów zastosowano funkcję przedstawioną na Rysunku 14. Wykonuje ona swoje zadanie poprzez iteracyjną inkrementację lub dekrementację wartości pozycji.

```
1 void moveServo(int whichServo, int PosServo)
2 {
3     servoPos[whichServo] = PosServo;
4
5     // Movement speed adjustment
6     int offset = 2;
7
8     if (whichServo <= 2)
9     {
10         offset = 30;
11     }
12
13     if (servoPos[whichServo] > servoPPos[whichServo])
14     {
15         for (int i = servoPPos[whichServo]; i <= servoPos[
16             whichServo]; i += offset)
17         {
18             servo[whichServo].write(i);
19         }
20         servo[whichServo].write(servoPos[whichServo]);
21     }
22     else if (servoPos[whichServo] < servoPPos[whichServo])
23     {
24         for (int i = servoPPos[whichServo]; i >= servoPos[
25             whichServo]; i -= offset)
26         {
27             servo[whichServo].write(i);
28         }
29         servo[whichServo].write(servoPos[whichServo]);
30     }
31 }
```

Rysunek 14: void moveServo()

Innym ciekawym rozwiązaniem jest realizacja odtwarzania zapisanej sekwencji ruchów, które pokazano na Rysunku 15. Rozwiążanie polega na iteracyjnej inkrementacji bądź dekrementacji wartości pozycji dla wszystkich serwomechanizmów po kolej.

```
1  case 2: // Play
2
3      // Repeat until Stop command
4      while (!dataIn.startsWith("c3"))
5      {
6          for (int j = 0; j < indexS; j++)
7          {
8
9              dataIn = Bluetooth.readString(); // Receive BT data
10             if (dataIn.startsWith("c3"))           // Stop?
11             {
12                 break;
13             }
14
15             // Move all servos simultaneously
16             while (checkPos(j)) // Check if all servos are in
posioton
17             {
18                 for (size_t i = 0; i < servoNum; i++)
19                 {
20                     if (servoPos[i] > servoSPos[i][j])
21                     {
22                         servo[i].write(--servoPos[i]);
23                     }
24                     else if (servoPos[i] < servoSPos[i][j])
25                     {
26                         servo[i].write(++servoPos[i]);
27                     }
28                 }
29             }
30         }
31     }
32
33     break;
```

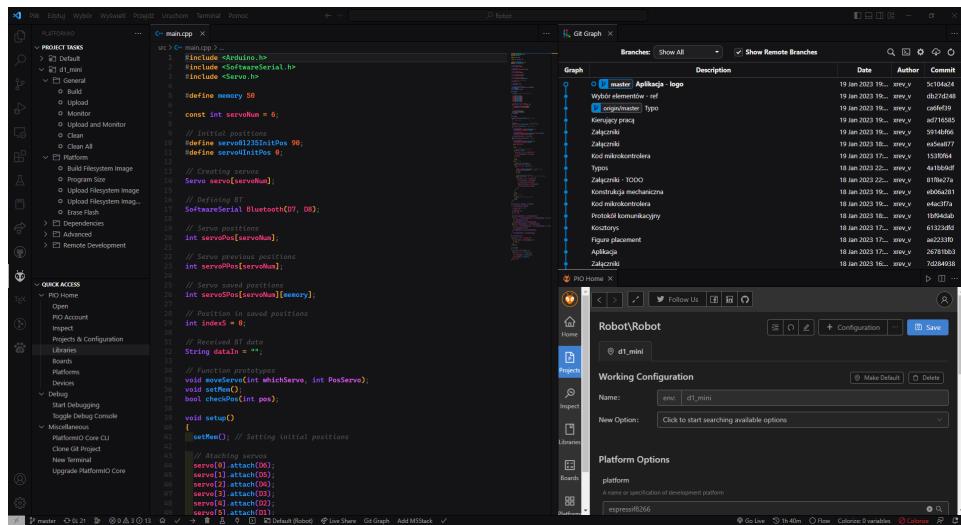
Rysunek 15: case 2: - Play

Ramię robota

Nieocenioną pomocą okazały się biblioteki:

- Arduino.h [4]
- SoftwareSerial.h [5]
- Servo.h [6]

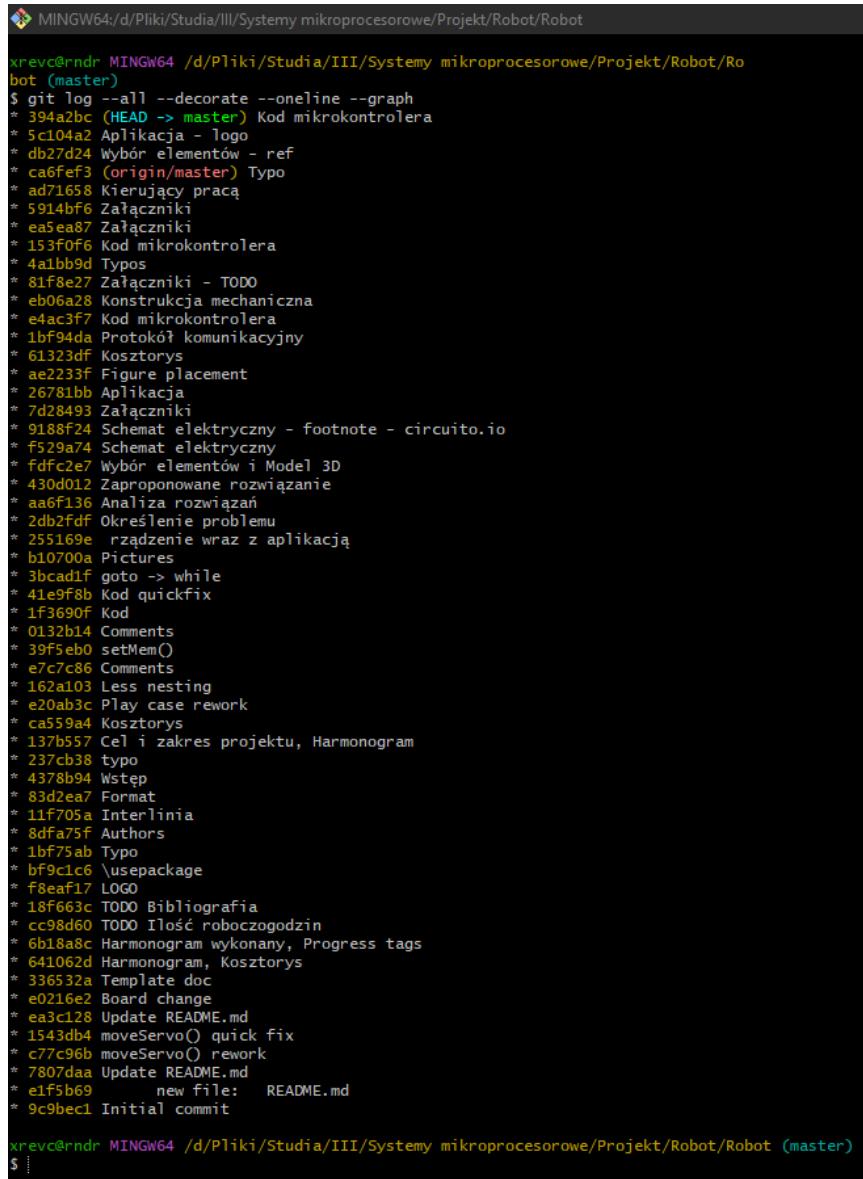
Środowisko Visual Studio Code (Rysunek 16) pozwoliło nam na sprawną pracę.



Rysunek 16: Środowisko Visual Studio Code

DONE? Kontrola wersji

W ramach tworzenia projektu została wykorzystana kontrola wersji Git⁹ (Rysunek 17) w połączeniu z serwisem GitHub¹⁰.



```
xrevc@rndr MINGW64 /d/Pliki/Studia/III/Systemy mikroprocesorowe/Projekt/Robot/Robot
$ git log --all --decorate --oneline --graph
* 394a2bc (HEAD -> master) Kod mikrokontrolera
* 5c104a2 Aplikacja - logo
* db27d24 Wybór elementów - ref
* ca6fef3 (origin/master) Typo
* ad71658 Kierujący pracą
* 5914bf6 Załączniki
* ea5ea87 Załączniki
* 153f0f6 Kod mikrokontrolera
* 4a1bb9d Typos
* 81f8e27 Załączniki - TODO
* eb06a28 Konstrukcja mechaniczna
* e4ac3f7 Kod mikrokontrolera
* 1bf94da Protokół komunikacyjny
* 61323df Kosztorys
* ae2233f Figure placement
* 26781bb Aplikacja
* 7d28493 Załączniki
* 9188f24 Schemat elektryczny - footnote - circuito.io
* f529a74 Schemat elektryczny
* fdfe2e7 Wybór elementów i Model 3D
* 430d012 Zaproponowane rozwiązanie
* aa6f136 Analiza rozwiązań
* 2db2fdf Określenie problemu
* 255169e rzadzenie wraz z aplikacją
* b10700a Pictures
* 3bcdaf1 goto -> while
* 41e9f8b Kod quickfix
* 1f3690f Kod
* 0132b14 Comments
* 39f5eb0 setMem()
* e7c7c86 Comments
* 162a103 Less nesting
* e20ab3c Play case rework
* ca559a4 Kosztorys
* 137b557 Cel i zakres projektu, Harmonogram
* 237cb38 typo
* 4378b94 Wstęp
* 83d2ea7 Format
* 11f705a Interlinia
* 8dfa75f Authors
* 1bf75ab Typo
* bf9c1c6 \usepackage
* f8eaf17 LOGO
* 18F663c TODO Bibliografia
* cc98d60 TODO Ilość roboczogodzin
* 6b18a8c Harmonogram wykonany, Progress tags
* 641062d Harmonogram, Kosztorys
* 336532a Template doc
* e0216e2 Board change
* ea3c128 Update README.md
* 1543db4 moveServo() quick fix
* c77c96b moveServo() rework
* 7807daa Update README.md
* e1f5b69 new file: README.md
* 9c9bec1 Initial commit

xrevc@rndr MINGW64 /d/Pliki/Studia/III/Systemy mikroprocesorowe/Projekt/Robot/Robot (master)
$ :
```

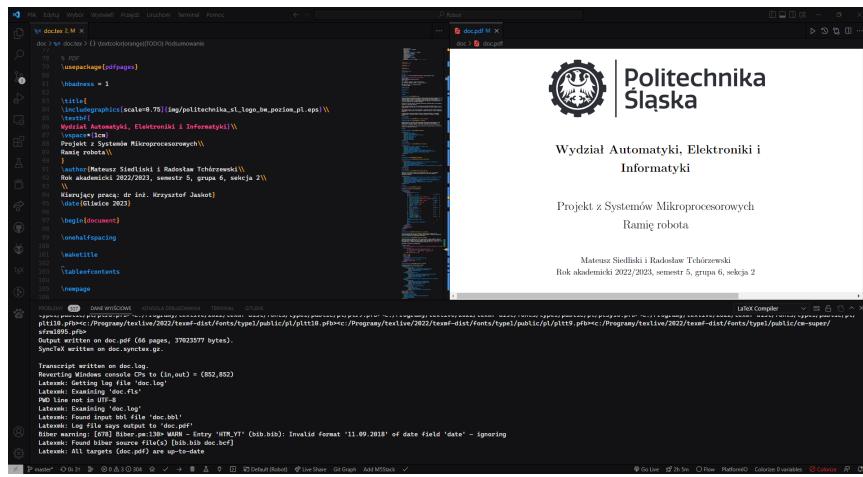
Rysunek 17: git log –all –decorate –oneline –graph

⁹Rozproszony system kontroli wersji

¹⁰Hostingowy serwis internetowy przeznaczony do projektów programistycznych wykorzystujących system kontroli wersji Git

WIP Dokumentacja

Dokumentacja została stworzona z wykorzystaniem LATEX¹¹ (Rysunek 18).



Rysunek 18: Proces tworzenia dokumentacji

¹¹Oprogramowanie do zautomatyzowanego składu tekstu, a także związany z nim język znaczników, służący do formatowania dokumentów tekstowych i tekstowo-graficznych

4.5 **TODO** Problemy w trakcie tworzenia sprzętu i aplikacji

W tym punkcie należy przedstawić jakie były problemy przed którymi stanęli autorzy projektu w trakcie jego realizacji i jak je rozwiązali.

5 **TODO** Podsumowanie

W ostatnim punkcie opisać co zostało wykonane, jakiej części założeń nie wykonano i dlaczego. Co można zrobić, by dany projekt poprawić i w jakim kierunku może pójść dalszy rozwój tego projektu.

6 WIP Literatura

- [1] How To Mechatronics. *DIY Arduino Robot Arm with Smartphone Control*. Data dostępu: 2022. URL: https://www.youtube.com/watch?v=_B3gWd3A_SI.
- [2] Tadeusz Szkodny. *Zbiór zadań z podstaw robotyki*. Gliwice: Wydawnictwo Politechniki Śląskiej, 2013. ISBN: 978-83-7880-162-7.
- [3] *PlatformIO*. Data dostępu: 2022. URL: <https://platformio.org>.
- [4] Suk-Hyun Cho. *framework-arduinoespressif8266 - Arduino.h*. Data dostępu: 2022. URL: <https://github.com/choey/framework-arduinoespressif8266/blob/master/cores/esp8266/Arduino.h>.
- [5] Peter Lerup. *EspSoftwareSerial*. Data dostępu: 2022. URL: <https://github.com/plerup/espsoftwareserial>.
- [6] Suk-Hyun Cho. *framework-arduinoespressif8266 - EspServo*. Data dostępu: 2022. URL: <https://github.com/choey/framework-arduinoespressif8266/tree/master/libraries/Servo>.

7 WIP Załączniki

Na prezentację (obronę) projektu trzeba przygotować prezentację w PowerPoint lub Prezi trwającą ok 12-15 minut Zwykle jest to ok. 20 slajdów. Te slajdy muszą zawierać tytuł projektu, autora, prowadzącego. Następnie plan prezentacji, najważniejsze osiągnięcia , podsumowanie. Slajdy muszą być czytelne, należy umieszczać rysunki i schematy, także nie powinno się na nich umieszczać zbyt dużo tekstu (tekst powinien mieć ok. 20-24pkt). W trakcie lub po prezentacji powinno się zademonstrować wykonany sprzęt i oprogramowanie. Można również zaprezentować w zamian tego film z działania urządzenia.

Przykładowa zawartość załącznika: Kody źródłowe (z komentarzami!!!) na CD spakowane. Wysłane mailem zawartość projektu do prowadzącego. Prezentacja. Wszystkie opisy układów, programów, narzędzi stosowanych w projekcie itp.

Załącznik 1: Kod mikrokontrolera

```
1 #include <Arduino.h>
2 #include <SoftwareSerial.h>
3 #include <Servo.h>
4
5 #define memory 50
6
7 const int servoNum = 6;
8
9 // Initial positions
10#define servo01235InitPos 90;
11#define servo4InitPos 0;
12
13// Creating servos
14 Servo servo[servoNum];
15
16// Defining BT
17 SoftwareSerial Bluetooth(D7, D8);
18
19// Servo positions
20 int servoPos[servoNum];
21
22// Servo previous positions
23 int servoPPos[servoNum];
24
25// Servo saved positions
26 int servoSPos[servoNum][memory];
27
28// Position in saved positions
29 int indexS = 0;
30
31// Received BT data
32 String dataIn = "";
33
34// Function prototypes
35 void moveServo(int whichServo, int PosServo);
36 void setMem();
37 bool checkPos(int pos);
38
39 void setup()
40 {
41     setMem(); // Setting initial positions
42
43     // Attaching servos
```

```
44  servo[0].attach(D6);
45  servo[1].attach(D5);
46  servo[2].attach(D4);
47  servo[3].attach(D3);
48  servo[4].attach(D2);
49  servo[5].attach(D1);
50
51 // Setting initial positions
52 for (size_t i = 0; i < servoNum; i++)
53 {
54     servo[i].write(servoSPos[i][0]);
55     servoPPos[i] = servoSPos[i][0];
56 }
57
58 // Initializing BT
59 Bluetooth.begin(38400);
60 Bluetooth.setTimeout(1);
61 delay(100);
62 }
63
64 void loop()
65 {
66     while (Bluetooth.available() <= 0) // Check BT
67     {
68         // Nothing to do
69     }
70
71     delay(1); // Wait for BT data
72     dataIn = Bluetooth.readString(); // Receive BT data
73
74     if (dataIn.startsWith("s")) // Servo position?
75     {
76         // Which servo?
77         String dataInServo = dataIn.substring(1, 2);
78         int SelectServo = dataInServo.toInt();
79
80         // Which position?
81         String dataInServoPos = dataIn.substring(2, dataIn.length());
82         int PosServo = dataInServoPos.toInt();
83
84         moveServo(SelectServo - 1, PosServo);
85     }
86     else if (dataIn.startsWith("c")) // Command?
```

```
87  {
88  // Which command?
89  String dataInFunc = dataIn.substring(1, 2);
90  int SelectFunc = dataInFunc.toInt();
91
92  switch (SelectFunc)
93  {
94
95  case 1: // Save
96
97  // Save all current servo positions
98  if (indexS < memory)
99  {
100    for (size_t i = 0; i < servoNum; i++)
101    {
102      servoSPos[i][indexS] = servoPos[i];
103    }
104    indexS++;
105  }
106
107  break;
108
109 case 2: // Play
110
111 // Repeat until Stop command
112 while (!dataIn.startsWith("c3"))
113 {
114   for (int j = 0; j < indexS; j++)
115   {
116
117     dataIn = Bluetooth.readString(); // Receive BT data
118     if (dataIn.startsWith("c3")) // Stop?
119     {
120       break;
121     }
122
123     // Move all servos simultaneously
124     while (checkPos(j)) // Check if all servos are in
125     posioton
126     {
127       for (size_t i = 0; i < servoNum; i++)
128       {
129         if (servoPos[i] > servoSPos[i][j])
130         {
```

```
130         servo[i].write(--servoPos[i]);
131     }
132     else if (servoPos[i] < servoSPos[i][j])
133     {
134         servo[i].write(++servoPos[i]);
135     }
136 }
137 }
138 }
139 }
140
141     break;
142
143 case 4: // Reset
144
145     setMem(); // Setting initial positions
146
147     break;
148 }
149 }
150 }
151
152 void moveServo(int whichServo, int PosServo)
153 {
154     servoPos[whichServo] = PosServo;
155
156     // Movement speed adjustment
157     int offset = 2;
158
159     if (whichServo <= 2)
160     {
161         offset = 30;
162     }
163
164     if (servoPos[whichServo] > servoPPos[whichServo])
165     {
166         for (int i = servoPPos[whichServo]; i <= servoPos[
167             whichServo]; i += offset)
168         {
169             servo[whichServo].write(i);
170         }
171         servo[whichServo].write(servoPos[whichServo]);
172     }
173     else if (servoPos[whichServo] < servoPPos[whichServo])
```

```
173 {  
174     for (int i = servoPPos[whichServo]; i >= servoPos[  
175         whichServo]; i -= offset)  
176     {  
177         servo[whichServo].write(i);  
178     }  
179     servo[whichServo].write(servoPos[whichServo]);  
180 }  
181 servoPPos[whichServo] = servoPos[whichServo];  
182 }  
183  
184 bool checkPos(int pos)  
185 {  
186     for (size_t i = 0; i < servoNum; i++)  
187     {  
188         if (servoPos[i] - servoSPos[i][pos] != 0)  
189         {  
190             return 1;  
191         }  
192     }  
193     return 0;  
194 }  
195  
196 void setMem()  
197 {  
198     // Setting initial positions  
199     for (size_t j = 0; j < memory; j++)  
200     {  
201         for (size_t i = 0; i < servoNum; i++)  
202         {  
203             servoSPos[i][j] = servo01235InitPos;  
204         }  
205         servoSPos[4][j] = servo4InitPos;  
206     }  
207     indexS = 0;  
208 }
```

Ramię robota

```
when BluetoothList .BeforePicking
do set BluetoothList . Elements to BluetoothClient1 . AddressesAndNames
set Connected . Text to " Disconnected "
set Connected . TextColor to red

when BluetoothList .AfterPicking
do if call BluetoothClient1 .Connect
address BluetoothList . Selection
then set Connected . Text to " Connected "
set Connected . TextColor to green
set Disconnected . Visible to true
set BluetoothList . Visible to false
set HorizontalArrangement3 . Visible to true

when Disconnected .Click
do call BluetoothClient1 .SendText
text " c3 "
call BluetoothClient1 .SendText
text " c4 "
call BluetoothClient1 .Disconnect
set Connected . Text to " Disconnected "
set Connected . TextColor to red
set Disconnected . Visible to false
set BluetoothList . Visible to true
set HorizontalArrangement3 . Visible to false

when BluetoothClient1 .BluetoothError
functionName message
do call BluetoothClient1 .Disconnect
if not BluetoothClient1 . IsConnected
then set Connected . Text to " Disconnected "
set Connected . TextColor to red
```

Załącznik 2: Kod aplikacji - Bluetooth

Ramię robota

```
when btnRotation .Click
do set slidRotation .ThumbPosition to 90

when btnWaist .Click
do set slidWaist .ThumbPosition to 0

when btnElbow .Click
do set slidElbow .ThumbPosition to 90

when btnWristRoll .Click
do set slidWristRoll .ThumbPosition to 90

when btnWristPitch .Click
do set slidWristPitch .ThumbPosition to 90

when btnGrip .Click
do set slidGrip .ThumbPosition to 90
```

Załącznik 3: Kod aplikacji - Przyciski

Ramię robota

```
when slidRotation .PositionChanged
  thumbPosition
do call BluetoothClient1 .SendText
  text [ join [ " s6 " ]
    round [ get thumbPosition ] ]
```

```
when slidWaist .PositionChanged
  thumbPosition
do call BluetoothClient1 .SendText
  text [ join [ " s5 " ]
    round [ get thumbPosition ] ]
```

```
when slidElbow .PositionChanged
  thumbPosition
do call BluetoothClient1 .SendText
  text [ join [ " s4 " ]
    round [ get thumbPosition ] ]
```

```
when slidWristRoll .PositionChanged
  thumbPosition
do call BluetoothClient1 .SendText
  text [ join [ " s3 " ]
    round [ get thumbPosition ] ]
```

```
when slidWristPitch .PositionChanged
  thumbPosition
do call BluetoothClient1 .SendText
  text [ join [ " s2 " ]
    round [ get thumbPosition ] ]
```

```
when slidGrip .PositionChanged
  thumbPosition
do call BluetoothClient1 .SendText
  text [ join [ " s1 " ]
    round [ get thumbPosition ] ]
```

Załącznik 4: Kod aplikacji - Slidery

Ramię robota

```
when btnSave .Click
do call BluetoothClient1 .SendText
    text "c1"
    set btnPlay .Visible to true
    set btnReset .Visible to true

when btnPlay .Click
do call BluetoothClient1 .SendText
    text "c2"
    set btnPlay .Visible to false
    set btnSave .Visible to false
    set btnReset .Visible to false
    set btnStop .Visible to true

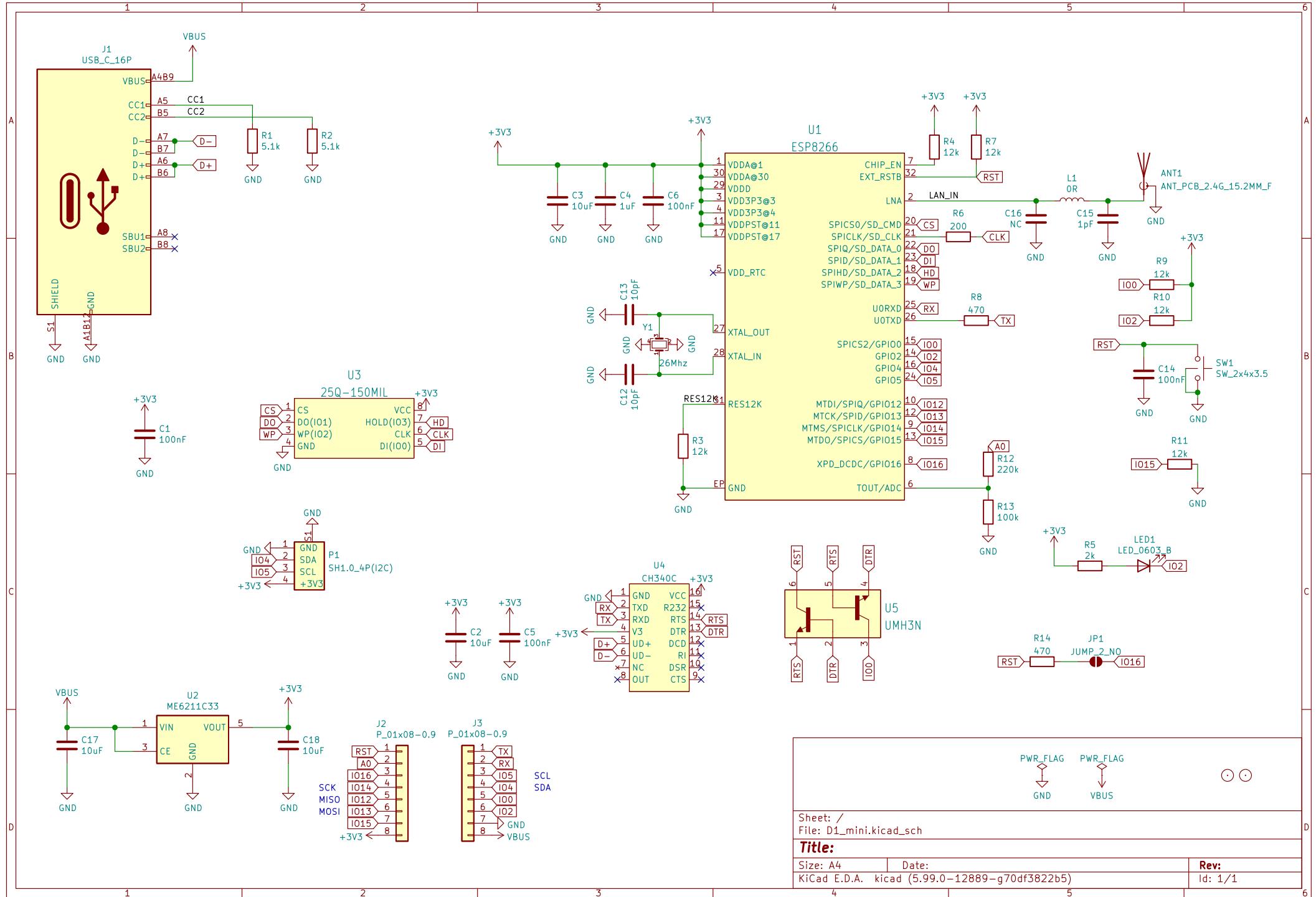
when btnStop .Click
do call BluetoothClient1 .SendText
    text "c3"
    set btnPlay .Visible to true
    set btnSave .Visible to true
    set btnReset .Visible to true
    set btnStop .Visible to false

when btnReset .Click
do call BluetoothClient1 .SendText
    text "c4"
    set btnPlay .Visible to false
    set btnStop .Visible to false
    set btnReset .Visible to false
```

Załącznik 5: Kod aplikacji - Komendy

Ramię robota

Załącznik 6: Wemos D1 mini - datasheet



Ramię robota

Załącznik 7: HC-05 - datasheet

HC Serial Bluetooth Products

User Instructional Manual

1 Introduction

HC serial Bluetooth products consist of Bluetooth serial interface module and Bluetooth adapter, such as:

(1) Bluetooth serial interface module:

Industrial level: HC-03, HC-04(HC-04-M, HC-04-S)

Civil level: HC-05, HC-06(HC-06-M, HC-06-S)

HC-05-D, HC-06-D (with baseboard, for test and evaluation)

(2) Bluetooth adapter:

HC-M4

HC-M6

This document mainly introduces Bluetooth serial module. Bluetooth serial module is used for converting serial port to Bluetooth. These modules have two modes: master and slaver device. The device named after even number is defined to be master or slaver when out of factory and can't be changed to the other mode. But for the device named after odd number, users can set the work mode (master or slaver) of the device by AT commands.

HC-04 specifically includes:

Master device: HC-04-M, M=master

Slave device: HC-04-S, S=slaver

The default situation of HC-04 is slave mode. If you need master mode, please state it clearly or place an order for HC-04-M directly. The naming rule of HC-06 is same.

When HC-03 and HC-05 are out of factory, one part of parameters are set for activating the device. The work mode is not set, since user can set the mode of HC-03, HC-05 as they want.

The main function of Bluetooth serial module is replacing the serial port line, such as:

1. There are two MCUs want to communicate with each other. One connects to Bluetooth master device while the other one connects to slave device. Their connection can be built once the pair is made. This Bluetooth connection is equivalently liked to a serial port line connection including RXD, TXD

signals. And they can use the Bluetooth serial module to communicate with each other.

2. When MCU has Bluetooth slave module, it can communicate with Bluetooth adapter of computers and smart phones. Then there is a virtual communicable serial port line between MCU and computer or smart phone.

3. The Bluetooth devices in the market mostly are slave devices, such as Bluetooth printer, Bluetooth GPS. So, we can use master module to make pair and communicate with them.

Bluetooth Serial module's operation doesn't need drive, and can communicate with the other Bluetooth device who has the serial. But communication between two Bluetooth modules requires at least two conditions:

- (1) The communication must be between master and slave.
- (2) The password must be correct.

However, the two conditions are not sufficient conditions. There are also some other conditions basing on different device model. Detailed information is provided in the following chapters.

In the following chapters, we will repeatedly refer to Linvor's (Formerly known as Guangzhou HC Information Technology Co., Ltd.) material and photos.

2 Selection of the Module

The Bluetooth serial module named even number is compatible with each other; The slave module is also compatible with each other. In other word, the function of HC-04 and HC-06, HC-03 and HC-05 are mutually compatible with each other. HC-04 and HC-06 are former version that user can't reset the work mode (master or slave). And only a few AT commands and functions can be used, like reset the name of Bluetooth (only the slaver), reset the password, reset the baud rate and check the version number. The command set of HC-03 and HC-05 are more flexible than HC-04 and HC-06's. Generally, the Bluetooth of HC-03/HC-05 is recommended for the user.

Here are the main factory parameters of HC-05 and HC-06. Pay attention to the differences:

HC-05	HC-06
Master and slave mode can be switched	Master and slave mode can't be switched
Bluetooth name: HC-05	Bluetooth name: linvor
Password:1234	Password:1234

<p>Master role: have no function to remember the last paired slave device. It can be made paired to any slave device. In other words, just set AT+CMODE=1 when out of factory. If you want HC-05 to remember the last paired slave device address like HC-06, you can set AT+CMODE=0 after paired with the other device. Please refer the command set of HC-05 for the details.</p>	<p>Master role: have paired memory to remember last slave device and only make pair with that device unless KEY (PIN26) is triggered by high level. The default connected PIN26 is low level.</p>
<p>Pairing: The master device can not only make pair with the specified Bluetooth address, like cell-phone, computer adapter, slave device, but also can search and make pair with the slave device automatically.</p> <p>Typical method: On some specific conditions, master device and slave device can make pair with each other automatically. (This is the default method.)</p>	<p>Pairing: Master device search and make pair with the slave device automatically.</p> <p>Typical method: On some specific conditions, master and slave device can make pair with each other automatically.</p>
<p>Multi-device communication: There is only point to point communication for modules, but the adapter can communicate with multi-modules.</p>	<p>Multi-device communication: There is only point to point communication for modules, but the adapter can communicate with multi-modules.</p>
<p>AT Mode 1: After power on, it can enter the AT mode by triggering PIN34 with high level. Then the baud rate for setting AT command is equal to the baud rate in communication, for example: 9600.</p> <p>AT mode 2: First set the PIN34 as high level, or while on powering the module set the PIN34 to be high level, the Baud rate used here is 38400 bps.</p> <p>Notice: All AT commands can be operated only</p>	<p>AT Mode: Before paired, it is at the AT mode. After paired it's at transparent communication.</p>

<p>when the PIN34 is at high level. Only part of the AT commands can be used if PIN34 doesn't keep the high level after entering to the AT mode. Through this kind of designing, set permissions for the module is left to the user's external control circuit, that makes the application of HC-05 is very flexible.</p>	
<p>During the process of communication, the module can enter to AT mode by setting PIN34 to be high level. By releasing PIN34, the module can go back to communication mode in which user can inquire some information dynamically. For example, to inquire the pairing is finished or not.</p>	<p>During the communication mode, the module can't enter to the AT mode.</p>
<p>Default communication baud rate: 9600, 4800-1.3M are settable.</p>	<p>Default communication baud rate: 9600, 1200-1.3M are settable.</p>
<p>KEY: PIN34, for entering to the AT mode.</p>	<p>KEY: PIN26, for master abandons memory.</p>
<p>LED1: PIN31, indicator of Bluetooth mode. Slow flicker (1Hz) represents entering to the AT mode2, while fast flicker(2Hz) represents entering to the AT mode1 or during the communication pairing. Double flicker per second represents pairing is finished, the module is communicable.</p> <p>LED2: PIN32, before pairing is at low level, after the pairing is at high level.</p> <p>The using method of master and slaver's indicator is the same.</p> <p>Notice: The PIN of LED1 and LED2 are connected with LED+.</p>	<p>LED: The flicker frequency of slave device is 102ms. If master device already has the memory of slave device, the flicker frequency during the pairing is 110ms/s. If not, or master has emptied the memory, then the flicker frequency is 750m/s. After pairing, no matter it's a master or slave device, the LED PIN is at high level.</p> <p>Notice: The LED PIN connects to LED+ PIN.</p>
<p>Consumption: During the pairing, the current is</p>	<p>Consumption: During the pairing, the current is</p>

fluctuant in the range of 30-40mA. The mean current is about 25mA. After paring, no matter processing communication or not, the current is 8mA. There is no sleep mode. This parameter is same for all the Bluetooth modules.	fluctuant in the range of 30-40 m. The mean current is about 25mA. After paring, no matter processing communication or not, the current is 8mA. There is no sleep mode. This parameter is same for all the Bluetooth modules.
Reset: PIN11, active if it's input low level. It can be suspended in using.	Reset: PIN11, active if it's input low level. It can be suspended in using.
Level: Civil	Level: Civil

The table above that includes main parameters of two serial modules is a reference for user selection.

HC-03/HC-05 serial product is recommended.

3. Information of Package

The PIN definitions of HC-03, HC-04, HC-05 and HC-06 are kind of different, but the package size is the same: 28mm * 15mm * 2.35mm.

The following figure 1 is a picture of HC-06 and its main PINs. Figure 2 is a picture of HC-05 and its main PINs. Figure 3 is a comparative picture with one coin. Figure 4 is their package size information. When user designs the circuit, you can visit the website of Guangzhou HC Information Technology Co., Ltd. (www.wavesen.com) to download the package library of protle version.

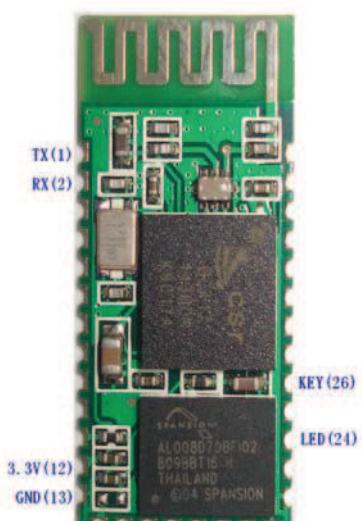


Figure 1 HC-06

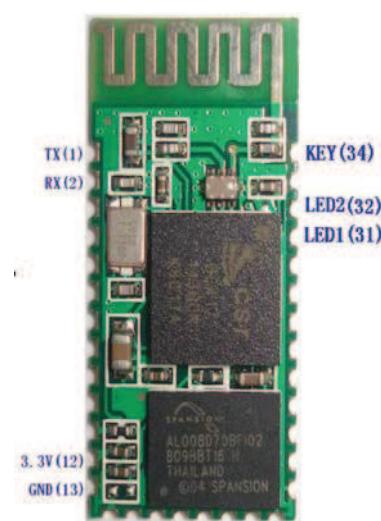


Figure 2 HC-05

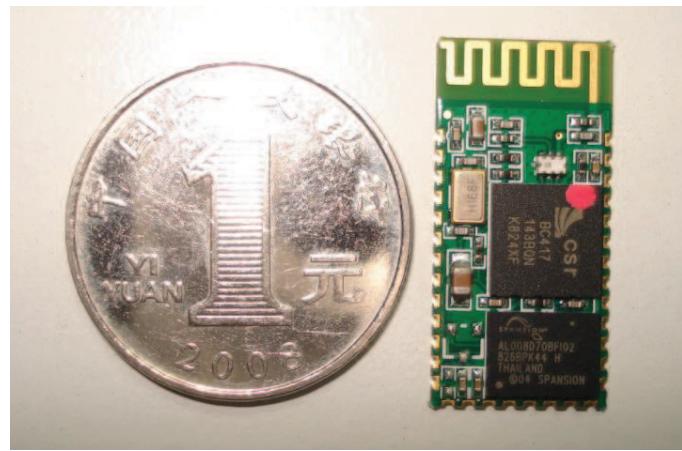
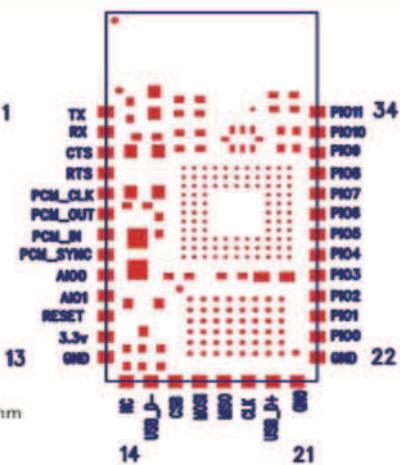
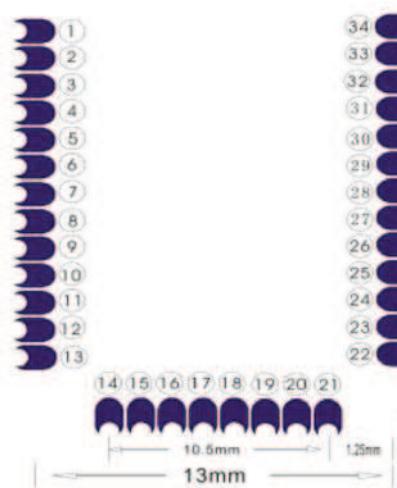


Figure 3 Comparative picture with one coin

LINVOR BLUE T
www.linvor.com

LV-BC-2.0

单位：mm



NO	PIN NAME	NO	PIN NAME
1	TX	20	USB D+
2	RX	21	GND
3	CTS	22	GND
4	RTS	23	PIO0
5	PCM_CLK	24	PIO1
6	PCM_OUT	25	PIO2
7	PCM_IN	26	PIO3
8	PCM_SYNC	27	PIO4
9	AIO0	28	PIO5
10	AIO1	29	PIO6
11	RESET	30	PIO7
12	3.3V	31	PIO8
13	GND	32	PIO9
14	NC	33	PIO10
15	USB D-	34	PIO11
16	CSB		
17	MOSI		
18	MISO		
19	CLK		

PCB Layout 请参考实物

Figure 4 Package size information

4. The Using and Testing Method of HC-06 for the First Time

This chapter will introduce the using method of HC-06 in detail. User can test the module according to this chapter when he or she uses the module at the first time.

PINs description:

PIN1	UART_TXD , TTL/CMOS level, UART Data output
PIN2	UART_RXD, TTL/COMS level, s UART Data input
PIN11	RESET, the reset PIN of module, inputting low level can reset the module, when the module is in using, this PIN can connect to air.
PIN12	VCC, voltage supply for logic, the standard voltage is 3.3V, and can work at 3.0-4.2V
PIN13	GND
PIN22	GND
PIN24	LED, working mode indicator Slave device: Before paired, this PIN outputs the period of 102ms square wave. After paired, this PIN outputs high level. Master device: On the condition of having no memory of pairing with a slave device, this PIN outputs the period of 110ms square wave. On the condition of having the memory of pairing with a slave device, this PIN outputs the period of 750ms square wave. After paired, this PIN outputs high level.
PIN26	For master device, this PIN is used for emptying information about pairing. After emptying, master device will search slaver randomly, then remember the address of the new got slave device. In the next power on, master device will only search this address.

(1) The circuit 1 (connect the module to 3.3V serial port of MCU) is showed by figure 5.

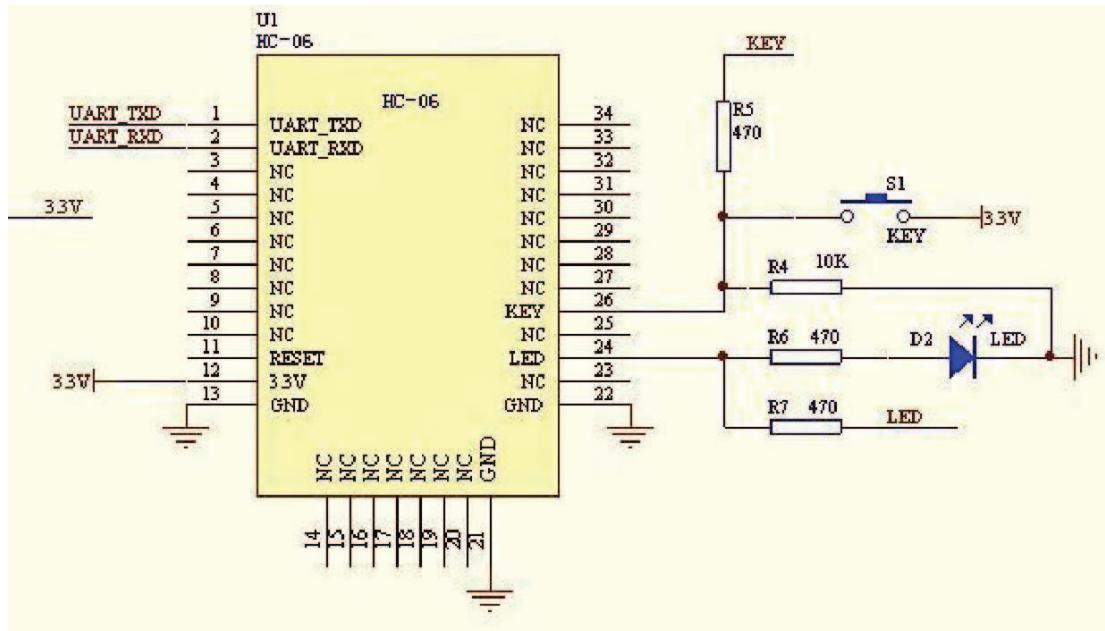


Figure 5 The circuit 1

In principle, HC-06 can work when UART_TXD, UART_RXD, VCC and GND are connected. However, for better testing results, connecting LED and KEY are recommended (when testing the master).

Where, the 3.3V TXD of MCU connects to HC-06's UART_RXD, the 3.3V RXD of MCU connects to HC-06's UART_TXD, and 3.3V power and GND should be connected. Then the minimum system is finished.

Note that, the PIN2:UART_RXD of Bluetooth module has no pull-up resistor. If the MCU TXD doesn't have pull-up function, then user should add a pull-up resistor to the UART_RXD. It may be easy to be ignored.

If there are two MCU which connect to master and slave device respectively, then before paired(LED will flicker) user can send AT commands by serial port when the system is power on. Please refer to HC-04 and HC-06's data sheet for detailed commands. In the last chapter, the command set will be introduced. Please pay attention to that the command of HC-04/HC-06 doesn't have terminator. For example, consider the call command, sending out AT is already enough, need not add the CRLF (carriage return line feed).

If the LED is constant lighting, it indicates the pairing is finished. The two MCUs can communicate with each other by serial port. User can think there is a serial port line between two MCUs.

(2) The circuit 2 (connect the module to 5V serial port of MCU) is showed by figure 6.

Figure 6 is the block diagram of Bluetooth baseboard. This kind of circuit can amplify Bluetooth module's operating voltage to 3.1-6.5V. In this diagram, the J1 port can not only be connected with MCU system of 3.3V and 5V, but also can be connected with computer serial port.

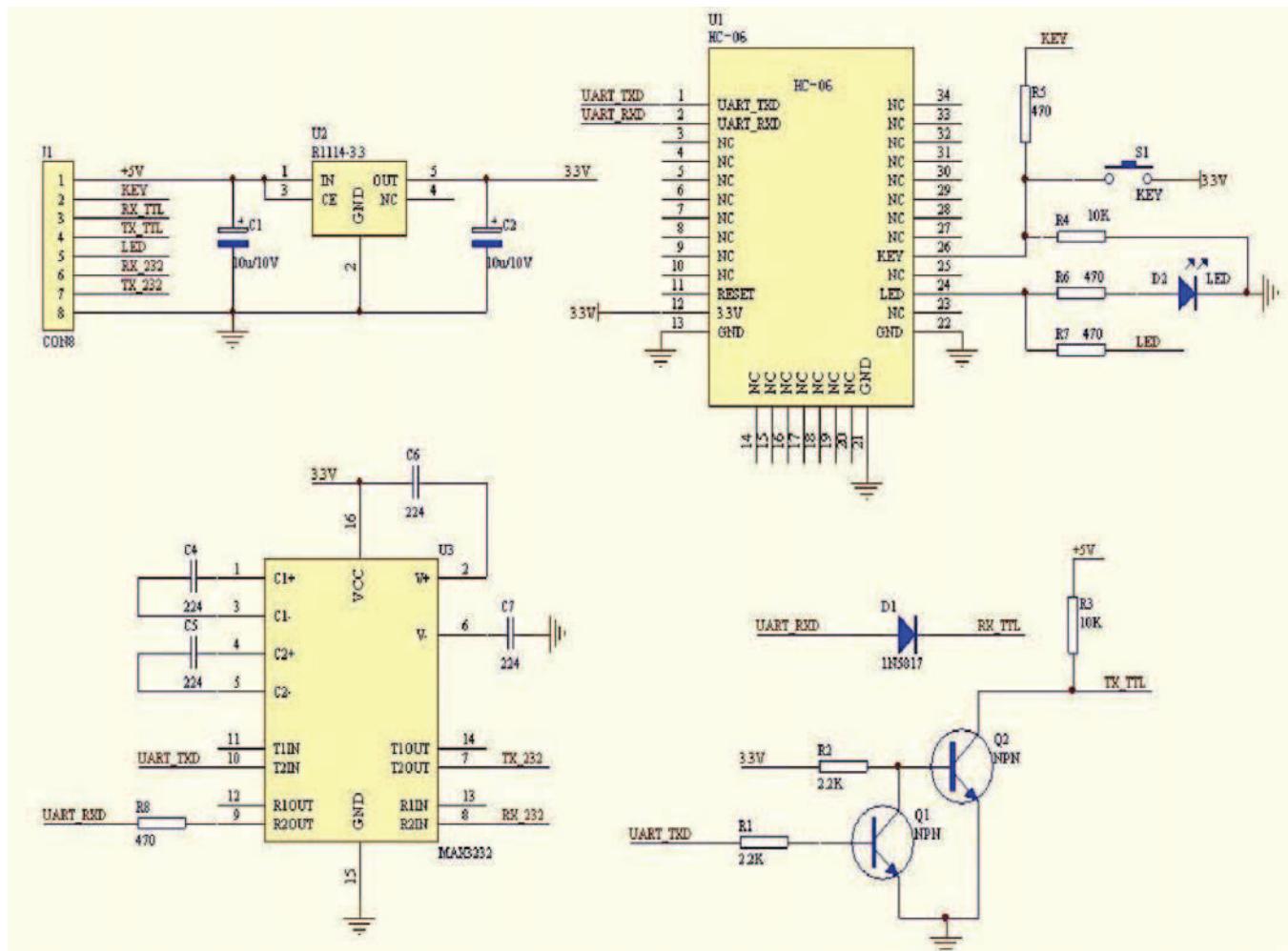


Figure 6 The circuit 2

(3) AT command test

Before paired, the mode of HC-04 and HC-06 are AT mode.

On the condition of 9600N81, OK will be received when user send the two letters AT. Please refer to the last chapter of datasheet for other commands of HC-06. Please pay attention to that sending out AT is already enough, need not add the CRLF (carriage return line feed).

The command set of Version V1.4 doesn't include parity. The version V1.5 and its later version have parity function. Moreover, there are three more commands of V1.5 than V1.4. They are:

No parity (default) AT+PN

Odd parity	AT+PO
Even parity	AT+PE

Do not let the sending frequency of AT command of HC-06 exceed 1Hz, because the command of HC-06 end or not is determined by the time interval.

(4) Pairing with adapter

User can refer to the download center of the company's website for "The Introduction of IVT" that introduces the Bluetooth module makes pair with computer adapter. That document taking HC-06-D for example introduces how the serial module makes pair with the adapter. That method is like to make pair with cell-phone. But the difference is that cell-phone need a third-party communication software to help. It's liked the kind of PC serial helper of and the hyper terminal. A software named "PDA serial helper" provided by our company is suitable for WM system. It has been proven that this serial module is supported by many smart phone systems' Bluetooth, such as, sybian, android, windows mobile and etc.

(5) Pairing introduction

HC-06 master device has no memory before the first use. If the password is correct, the master device will make pair with the slave device automatically in the first use. In the following use, the master device will remember the Bluetooth address of the last paired device and search it. The searching won't stop until the device is found. If master device's PIN26 is input high level, the device will lose the memory. In that occasion, it'll search the proper slave device like the first use. Based on this function, the master device can be set to make pair with the specified address or any address by user.

(6) Reset new password introduction

User can set a new password for the HC-06 through AT+PINxxxx command. But the new password will become active after discharged all the energy of the module. If the module still has any energy, the old one is still active. In the test, for discharging all the system energy and activating the new password, we can connect the power supply PIN with GND about 20 seconds after the power is cut off. Generally, shutting down the device for 30 minutes also can discharge the energy, if there is no peripheral circuit helps discharge energy. User should make the proper way according to the specific situation.

(7) Name introduction

If the device has no name, it's better that user doesn't try to change the master device name. The name should be limited in 20 characters.

Summary: The character of HC-06: 1 not many command 2 easy for application 3 low price. It's good for some specific application. HC-04 is very similar with HC-06. Their only one difference is HC-04 is for industry, HC-06 is for civil. Except this, they don't have difference.

The following reference about HC-04 and HC-06 can be downloaded from company website www.wavesen.com:

HC-06 datasheet.pdf	(the command set introduction is included)
HC-04 datasheet.pdf	(the command set introduction is included)
IVT BlueSoleil-2.6	(IVT Bluetooth drive test version)
Bluetooth FAQ.pdf	
HC-04-D(HD-06-D)datasheet(English).pdf	
HC-06-AT command software (test version)	(some commands in V1.5 is not supported by V1.4)
PCB package of Bluetooth key modules	(PCB package lib in protel)
IVT software manual.pdf	(introduce how to operate the modern and make pair with Bluetooth module)
PDA serial test helper.exe	(serial helper used for WM system)

5 manual for the first use of HC-05

This chapter will introduce how to test and use the HC-05 if it's the first time for user to operate it.

(1) PINs description

PIN1	UART_TXD, Bluetooth serial signal sending PIN, can connect with MCU's RXD PIN
PIN2	UART_RXD, Bluetooth serial signal receiving PIN, can connect with the MCU's TXD PIN, there is no pull-up resistor in this PIN. But It needs to be added an eternal pull-up resistor.
PIN11	RESET, the reset PIN of module, inputting low level can reset the module, when the module is in using, this PIN can connect to air.
PIN12	VCC, voltage supply for logic, the standard voltage is 3.3V, and can work at 3.0-4.2V
PIN13	GND

	<p>LED1, indicator of work mode. Has 3 modes:</p> <p>When the module is supplied power and PIN34 is input high level, PIN31 output 1Hz square wave to make the LED flicker slowly. It indicates that the module is at the AT mode, and the baud rate is 38400;</p> <p>When the module is supplied power and PIN34 is input low level, PIN31 output 2Hz square wave to make the LED flicker quickly. It indicates the module is at the pairable mode. If PIN34 is input high level, then the module will enter to AT mode, but the output of PIN31 is still 2Hz square wave.</p> <p>After the pairing, PIN31 output 2Hz square wave.</p> <p>Note: if PIN34 keep high level, all the commands in the AT command set can be in application. Otherwise, if just excite PIN34 with high level but not keep, only some command can be used. More information has provided at chapter 2.</p>
PIN32	Output terminal. Before paired, it output low level. Once the pair is finished, it output high level.
PIN34	Mode switch input. If it is input low level, the module is at paired or communication mode. If it's input high level, the module will enter to AT mode. Even though the module is at communication, the module can enter to the AT mode if PIN34 is input high level. Then it will go back to the communication mode if PIN34 is input low level again.

(2) Application circuit 1 (connect to the 3.3V system)

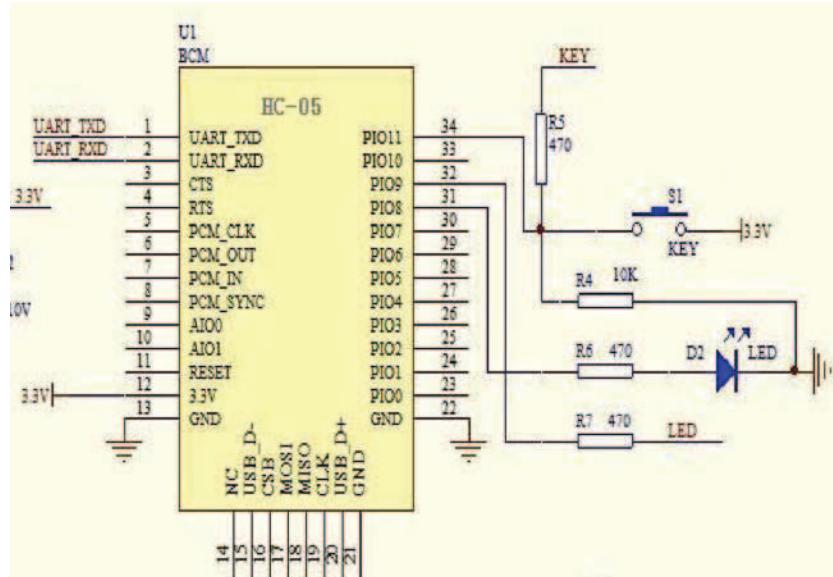


Figure 7 Application 1

(3) Application circuit 2 (connect to 5V serial system or PC serial)

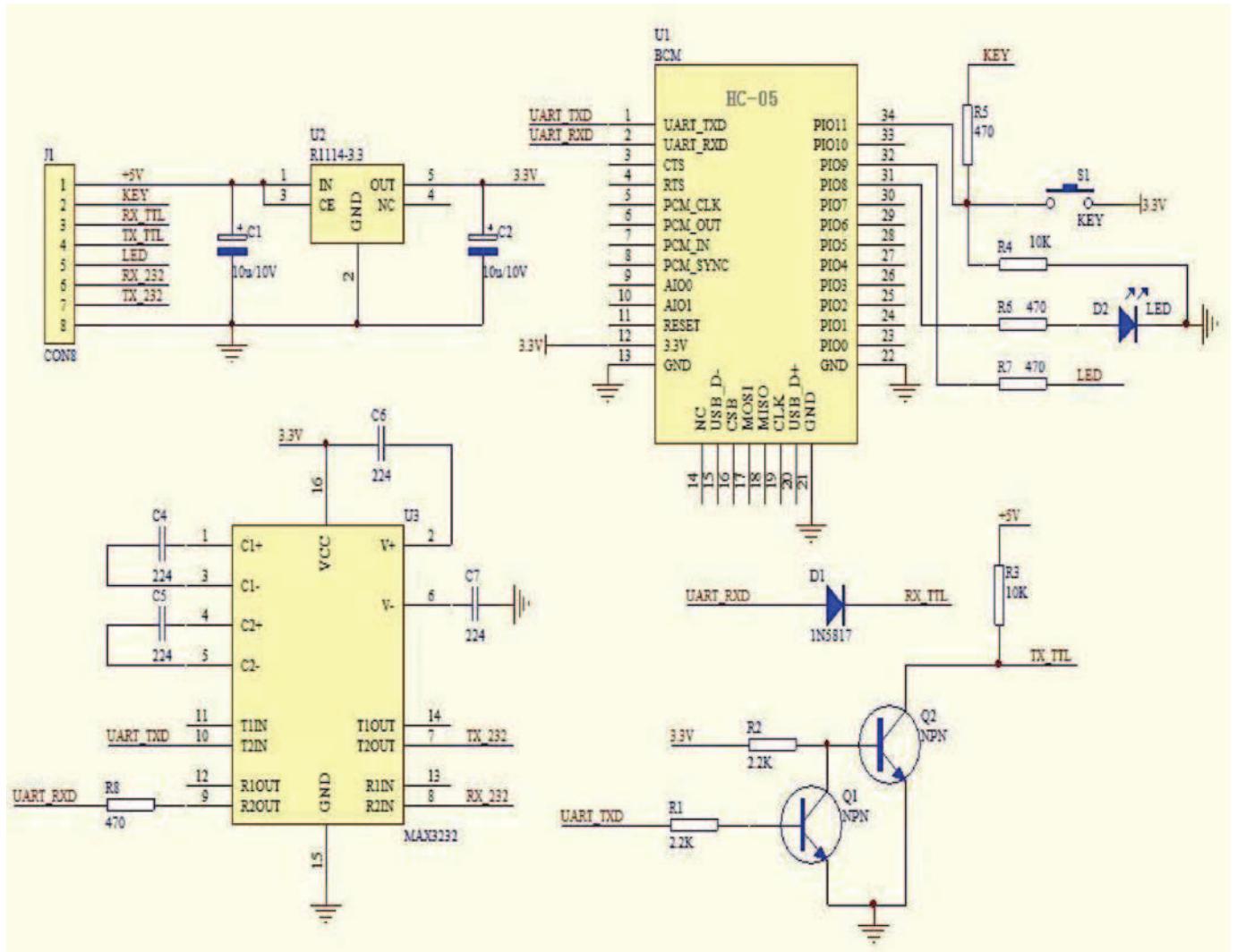


Figure 8 Application circuit 2

(4) AT command test

This chapter introduces some common commands in use. The detail introduction about HC-05 command is in HC-0305 AT command set.

Enter to AT mode:

Way1: Supply power to module and input high level to PIN34 at the same time, the module will enter to AT mode with the baud rate-38400.

Way2: In the first step, supply power to module; In the second step, input high level to PIN34. Then the module will enter to AT mode with the baud rate-9600. Way1 is recommended.

Command structure: all command should end up with “\r\n” (Hex: 0X0D X0A) as the terminator. If

the serial helper is installed, user just need enter “ENTER” key at the end of command.

Reset the master-slave role command:

AT+ROLE=0 ----Set the module to be slave mode. The default mode is slave.

AT+ROLE=1 ----Set the module to be master mode.

Set memory command:

AT+CMODE=1

Set the module to make pair with the other random Bluetooth module (Not specified address). The default is this mode.

AT+CMODE=1

Set the module to make pair with the other Bluetooth module (specified address). If set the module to make pair with random one first, then set the module to make pair with the Bluetooth module has specified address. Then the module will search the last paired module until the module is found.

Reset the password command

AT+PSWD=XXXX

Set the module pair password. The password must be 4-bits.

Reset the baud rate

AT+UART== <Param>,<Param2>,<Param3>.

More information is provided at HC-0305 command set

Example:

AT+UART=9600,0,0 ----set the baud rate to be 9600N81

Reset the Bluetooth name

AT+NAME=XXXXXX

Summary:

HC-05 has many functions and covers all functions of HC-06. The above commands are the most common ones. Besides this, HC-05 leaves lots of space for user. So HC-05 is better than HC-06 and

Ramię robota

Załącznik 8: HC-05 - AT

Default:

Slave, 9600 baud rate, N, 8, 1. Pincode 1234

AT command:

1. Communications Test :

Sent : AT

receive : OK

2. Change baud rate :

Sent : AT+BAUD1

receive : OK1200

Sent : AT+BAUD2

receive : OK2400

1-----1200
2-----2400
3-----4800
4-----9600
5-----19200
6-----38400
7-----57600
8-----115200

Baud rate setting can be save even power down.

3. Change Bluetooth device name:

Sent : AT+NAMEdevicename

receive : OKname

(devicename is the name you want the device to be , and it will be searched with this name)

Name setting can be save even power down.

4. Change Pincode:

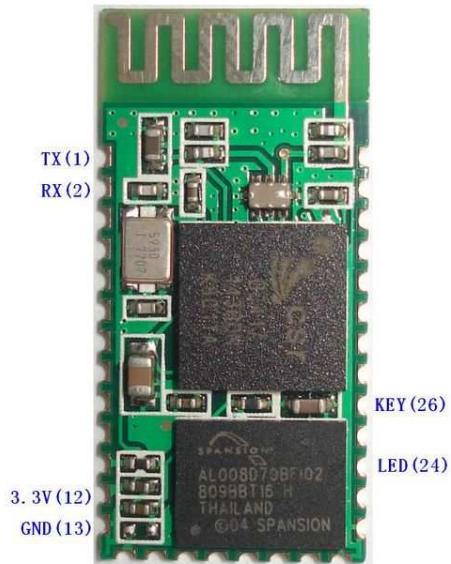
Sent : AT+PINxxxx

receive : OKsetpin

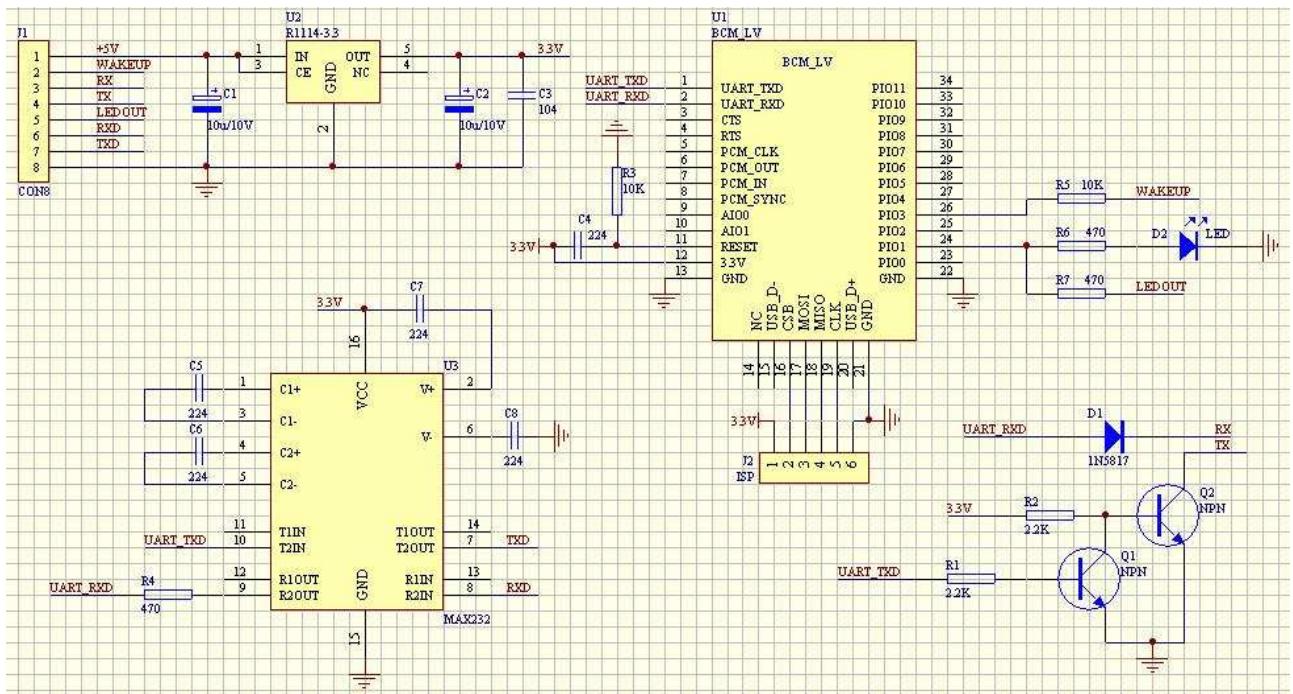
(xxxx is the pin code you set)

Pin code can be save even power down.

Pin Map:

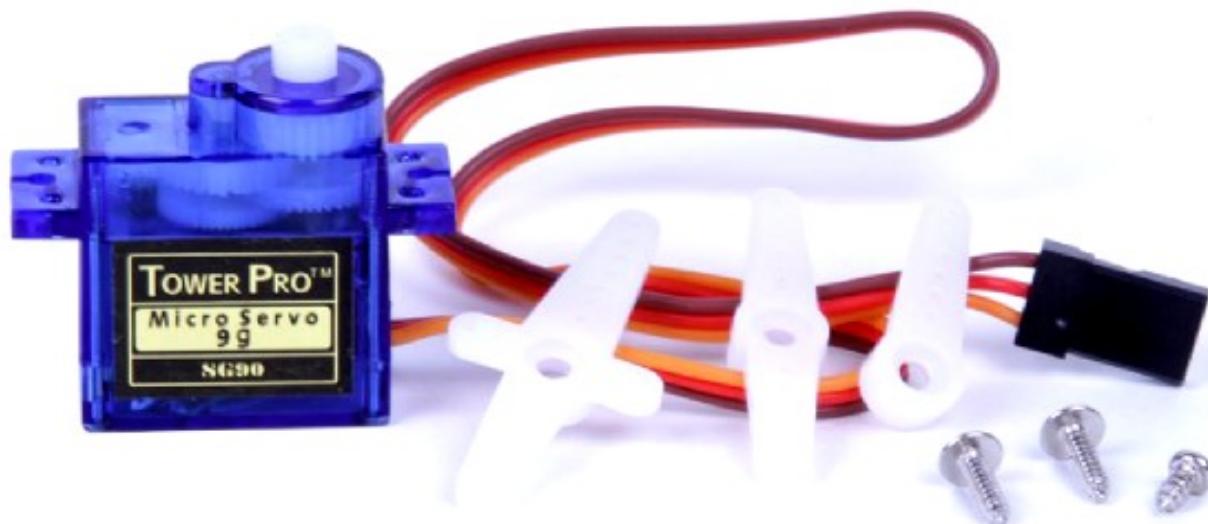


Demonstration Circuit:

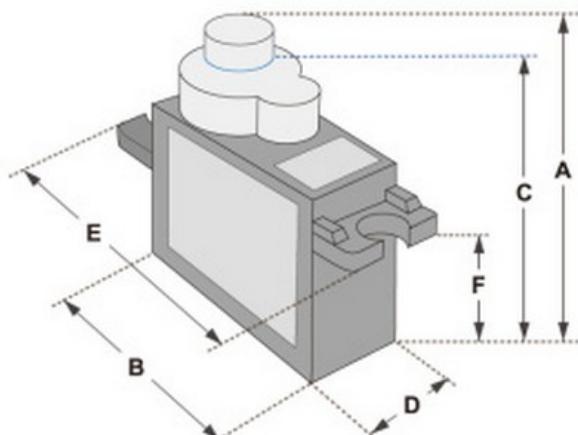


Ramię robota

Załącznik 9: Micro Servo 9g SG90 - datasheet



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

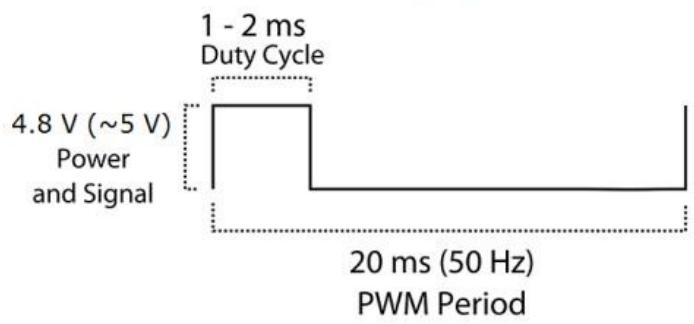


Dimensions & Specifications

A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

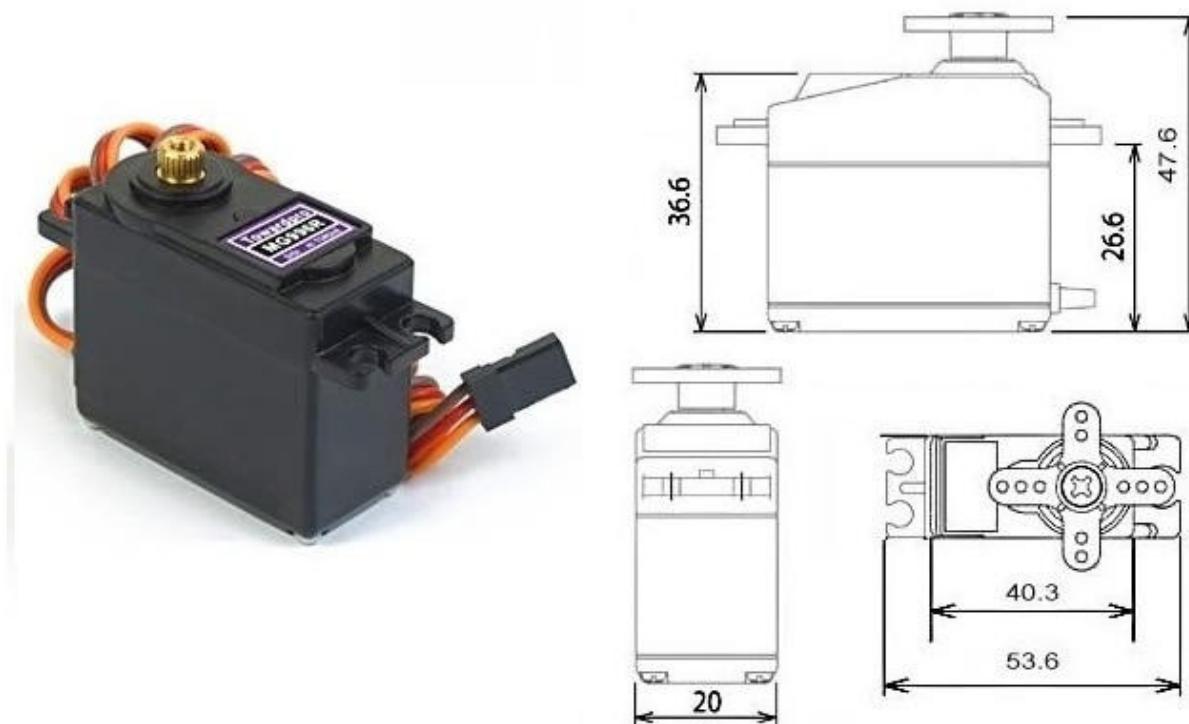
PWM=Orange (脉冲)
Vcc=Red (+)
Ground=Brown (-)



Ramię robota

Załącznik 10: Servo Mg996r - datasheet

MG996R High Torque Metal Gear Dual Ball Bearing Servo



This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60° (4.8 V), 0.14 s/60° (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 μ s
- Stable and shock proof double ball bearing design
- Temperature range: 0 °C – 55 °C

PWM=Orange (⊿⊿)
 Vcc = Red (+)
 Ground=Brown (-)

