



**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. Jacek Loska

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	TODO Urządzenie wraz z aplikacją	6
4.1	TODO Określenie problemu	6
4.2	TODO Analiza rozwiązań	6
4.3	TODO Zaproponowane rozwiązanie	6
4.4	TODO Wykonanie	6
4.5	TODO Problemy w trakcie tworzenia sprzętu i aplikacji (nie- potrzebne usunąć!)	7
5	TODO Podsumowanie	8
6	WIP Literatura	9
7	TODO Załączniki	10
8	Kod	11
8.1	Kod mikrokontrolera	11
8.2	Kod aplikacji	16
	Bluetooth	16
	Przyciski	17
	Slidery	18
	Komendy	19

1 **DONE?** Wstęp

Podczas studiowania na kierunku Automatyka i Robotyka można zauważyć 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 pomoc 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 chwytakiem, 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 łatwo dostępnych 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, weryfikacji i przetestowanie sprzętu i 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, poprawki mechaniczne, usprawnienia
4. Doskonalenie projektu — debugowanie, poprawki mechaniczne, usprawnienia
5. Doskonalenie projektu — debugowanie, poprawki mechaniczne, usprawnienia

3 **DONE?** Kosztorys

Lp.	Typ	Producent	Ilość	Cena	Wartość
1.	Mikrokontroler Wemos D1 mini	Wemos	1	9,48 zł	9,48 zł
2.	Moduł Bluetooth HC-05	SZYTF	1	9,40 zł	9,40 zł
3.	Micro Servo 9g SG90	HWAYEH	3	3,97 zł	11,91 zł
4.	Servo Mg996r	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.	Płytki 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 = 40					

4 **TODO** Urządzenie wraz z aplikacją

Należy podać krótki opis celu który chcemy osiągnąć wykonując projekt. Pozostałe punkty mają ułatwić napisanie projektu, ale nie są bezwzględnie wymagane. Powinno się je elastycznie dopasować do zrealizowanego projektu – SZCZEGÓLNICIE nazwy podrozdziałów. Ta część powinna zwierać od 20 do 25 stron.

4.1 **TODO** Określenie problemu

Dokładne opisanie problemu. Wykonać schematy ideowe i blokowe, dołączyć ilustrujące problem rysunki itp. Wykonać założenia potrzebne do rozwiązania postawionego problemu.

Rysunek 1. To jest rysunek podstawowego bloku systemu.

Należy zawsze odwoływać się do tego co jest na Rysunek 1.

4.2 **TODO** Analiza rozwiązań

W tym punkcie powinno się przedstawić jakie są możliwości rozwiązania problemu (urządzenia, technologie i produkty), wraz z określeniem kryteriów wyboru rozwiązania.

4.3 **TODO** Zaproponowane rozwiązanie

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

4.4 **TODO** Wykonanie

W tym punkcie opisać jak zostało wykonane urządzenie oraz zaprogramowana aplikacja, jakie zastosowano technologie, narzędzia do pisania, weryfikowania i testowania aplikacji. Nie należy umieszczać kodu programu, jedynie wybrane fragmenty pokazujące rozwiązanie niestandardowego problemu. Pełny kod programu należy przesłać emailem ew. dodatkowo umieścić w załączniku. Jest to najważniejszy punkt w projekcie – proszę mu poświęcić jak najwięcej uwagi minimum 15-18 stron!!!

4.5 **TODO** Problemy w trakcie tworzenia sprzętu i aplikacji (niepotrzebne usunąć!)

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

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.

7 **TODO** 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.

8 Kod

8.1 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  // Ataching 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         );
83         int PosServo = dataInServoPos.toInt();

```

```

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
posiotion
125                 {
126                     for (size_t i = 0; i < servoNum; i++)

```

```

127         {
128             if (servoPos[i] > servoSPos[i][j])
129             {
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     }
172 }

```

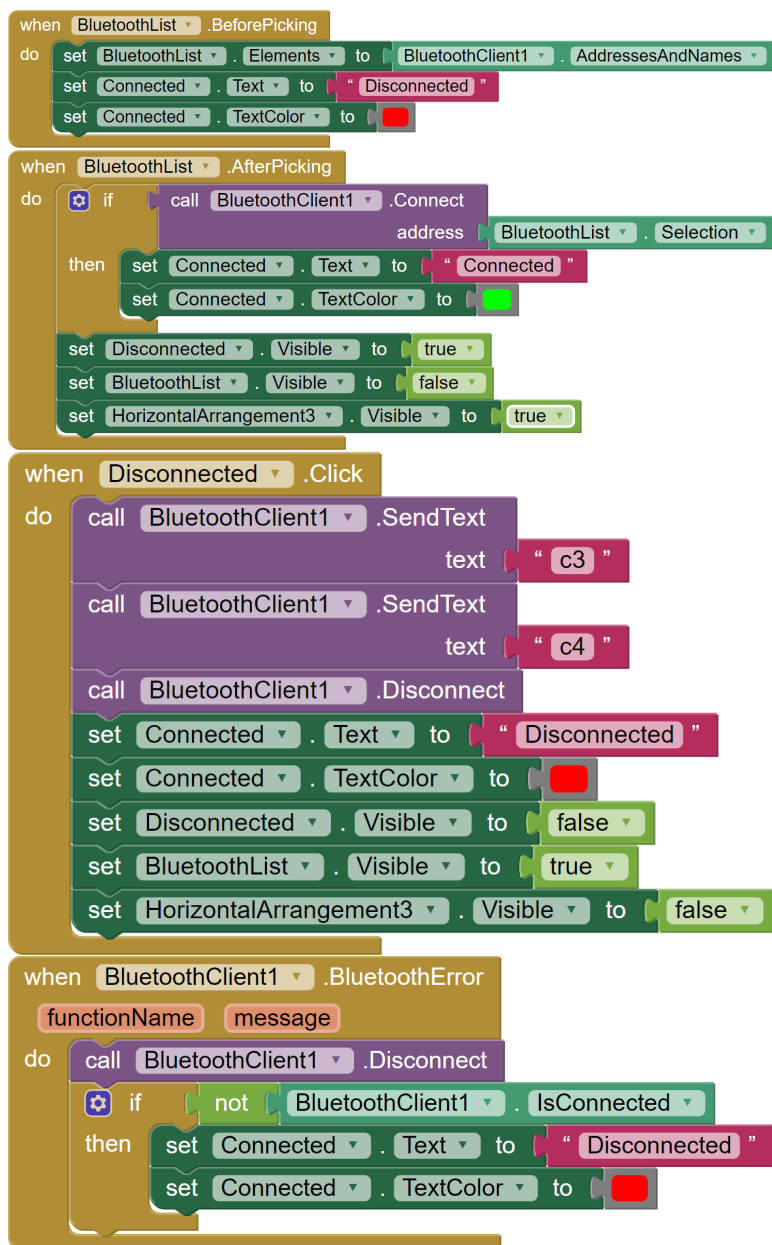
```

170     servo[whichServo].write(servoPos[whichServo]);
171 }
172 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 }

```


8.2 Kod aplikacji

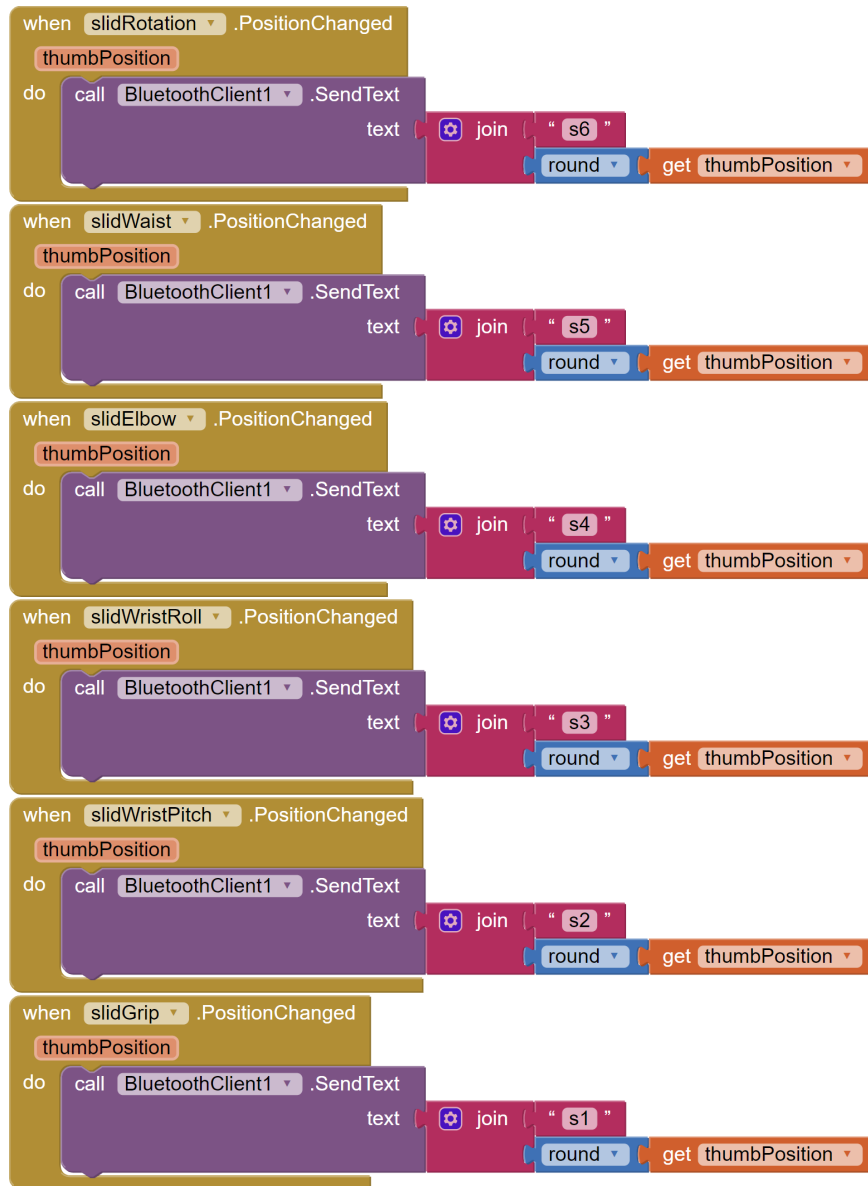
Bluetooth



Przyciski



Slidery



Komendy

