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

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 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, 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.	Płytka 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 – SZCZEGÓLNIE 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 postawie 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 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.

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.

7.1 Kod mikrokontrolera

```
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <Servo.h>

#define memory 50

const int servoNum = 6;

// Initial positions
#define servo01235InitPos 90;
#define servo4InitPos 0;

// Creating servos
Servo servo[servoNum];

// Defining BT
SoftwareSerial Bluetooth(D7, D8);

// Servo positions
int servoPos[servoNum];

// Servo previous positions
int servoPPos[servoNum];

// Servo saved positions
int servoSPos[servoNum][memory];

// Position in saved positions
int indexS = 0;

// Received BT data
String dataIn = "";

// Function prototypes
void moveServo(int whichServo, int PosServo);
void setMem();
bool checkPos(int pos);

void setup()
{
  setMem(); // Setting initial positions

```

```
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      int PosServo = dataInServoPos.toInt();
83
84      moveServo(SelectServo - 1, PosServo);
85    }
```

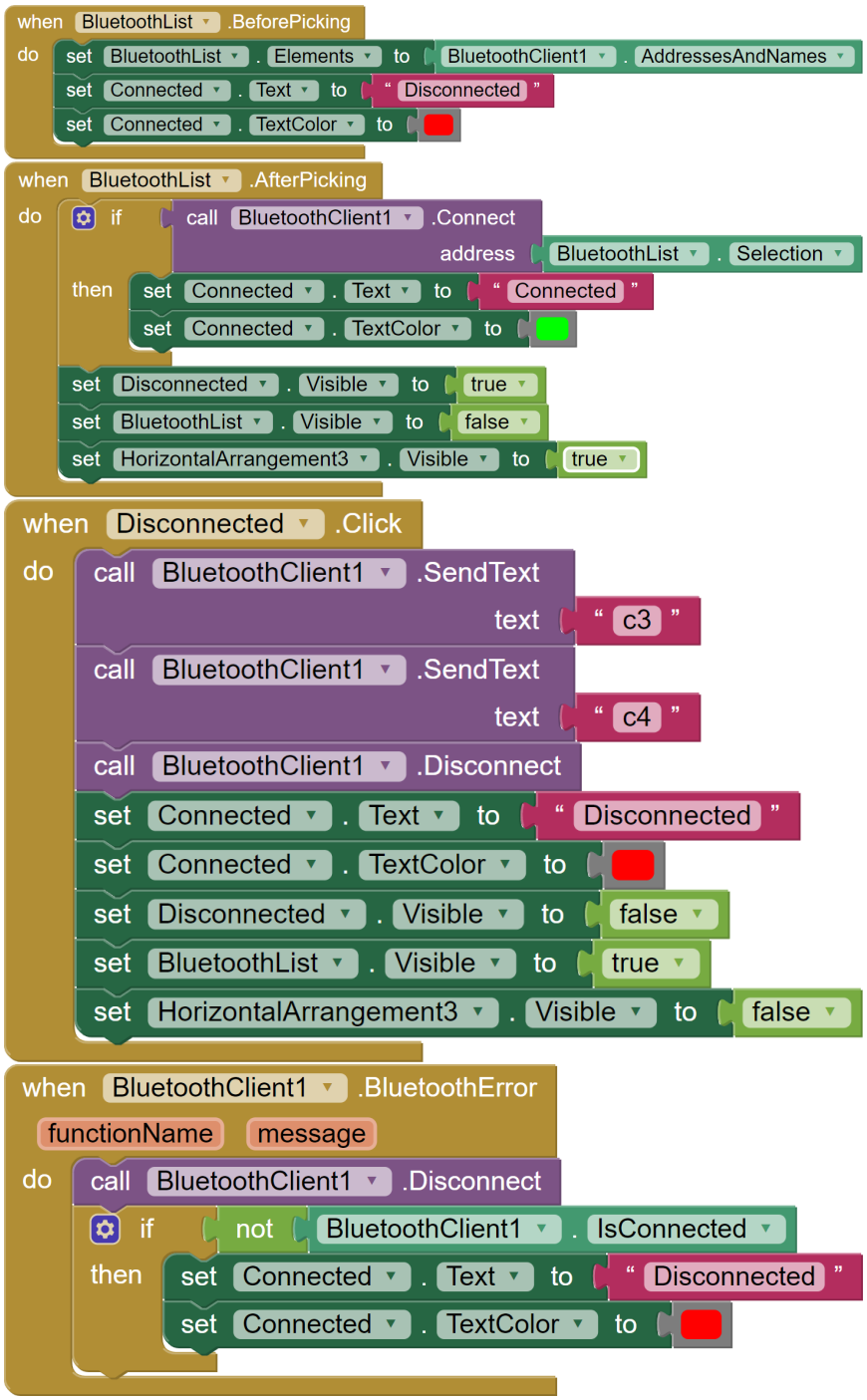
```
86  else if (dataIn.startsWith("c")) // Command?
87  {
88    // Which command?
89    String dataInFunc = dataIn.substring(1, 2);
90    int SelectFunc = dataInFunc.toInt();
91
92    switch (SelectFunc)
93    {
94
95    case 1: // Save
96
97      // Save all current servo positions
98      if (indexS < 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   PLAY:
112
113     for (int j = 0; j < indexS; j++)
114     {
115
116       dataIn = Bluetooth.readString(); // Receive BT data
117       if (dataIn.startsWith("c3"))     // Stop?
118       {
119         break;
120       }
121
122       // Move all servos simultaneously
123       while (checkPos(j)) // Check if all servos are in
  posiotion
124       {
125         for (size_t i = 0; i < servoNum; i++)
126         {
127           if (servoPos[i] > servoSPos[i][j])
128           {
```

```
129               servo[i].write(--servoPos[i]);
130             }
131             else if (servoPos[i] < servoSPos[i][j])
132             {
133               servo[i].write(++servoPos[i]);
134             }
135           }
136         }
137       }
138
139       // Repeat until Stop command
140       if (!dataIn.startsWith("c3"))
141       {
142         goto PLAY;
143       }
144
145       break;
146
147     case 4: // Reset
148
149       setMem(); // Setting initial positions
150
151       break;
152     }
153   }
154 }
155
156 void moveServo(int whichServo, int PosServo)
157 {
158   servoPos[whichServo] = PosServo;
159
160   // Movement speed adjustment
161   int offset = 2;
162
163   if (whichServo <= 2)
164   {
165     offset = 30;
166   }
167
168   if (servoPos[whichServo] > servoPPos[whichServo])
169   {
170     for (int i = servoPPos[whichServo]; i <= servoPos[
      whichServo]; i += offset)
171     {
```

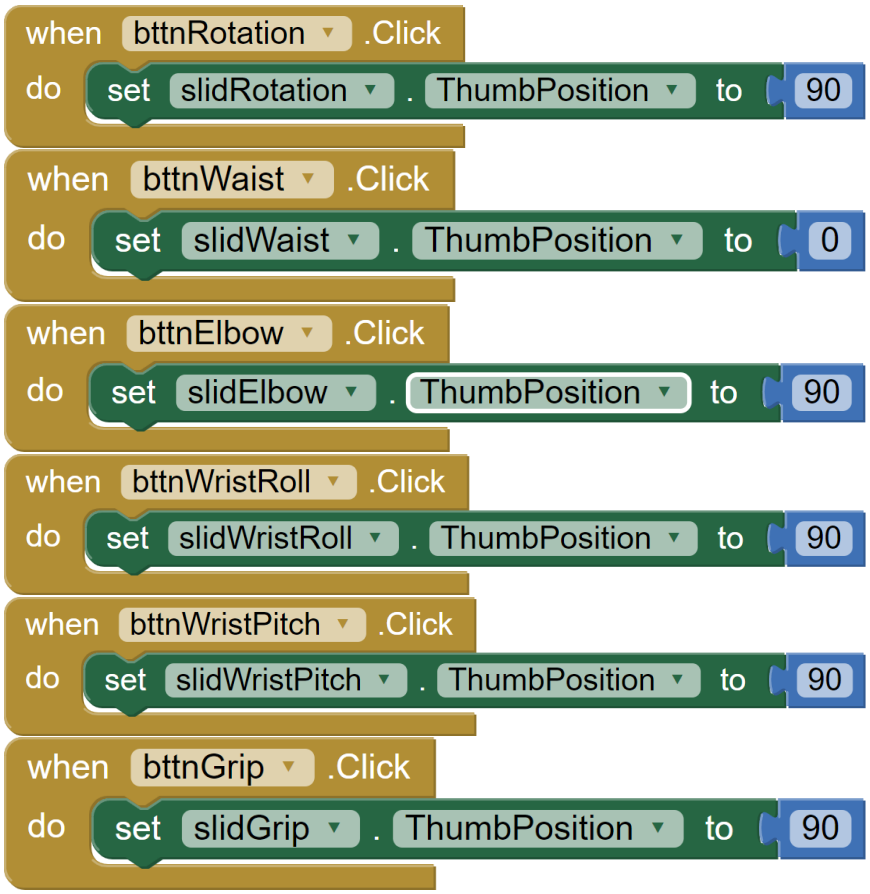
```
172       servo[whichServo].write(i);
173     }
174     servo[whichServo].write(servoPos[whichServo]);
175   }
176   else if (servoPos[whichServo] < servoPPos[whichServo])
177   {
178     for (int i = servoPPos[whichServo]; i >= servoPos[
      whichServo]; i -= offset)
179     {
180       servo[whichServo].write(i);
181     }
182     servo[whichServo].write(servoPos[whichServo]);
183   }
184
185   servoPPos[whichServo] = servoPos[whichServo];
186 }
187
188 bool checkPos(int pos)
189 {
190   for (size_t i = 0; i < servoNum; i++)
191   {
192     if (servoPos[i] - servoSPos[i][pos] != 0)
193     {
194       return 1;
195     }
196   }
197   return 0;
198 }
199
200 void setMem()
201 {
202   // Setting initial positions
203   for (size_t j = 0; j < memory; j++)
204   {
205     for (size_t i = 0; i < servoNum; i++)
206     {
207       servoSPos[i][j] = servo01235InitPos;
208     }
209     servoSPos[4][j] = servo4InitPos;
210   }
211   indexS = 0;
212 }
```

7.2 Kod aplikacji

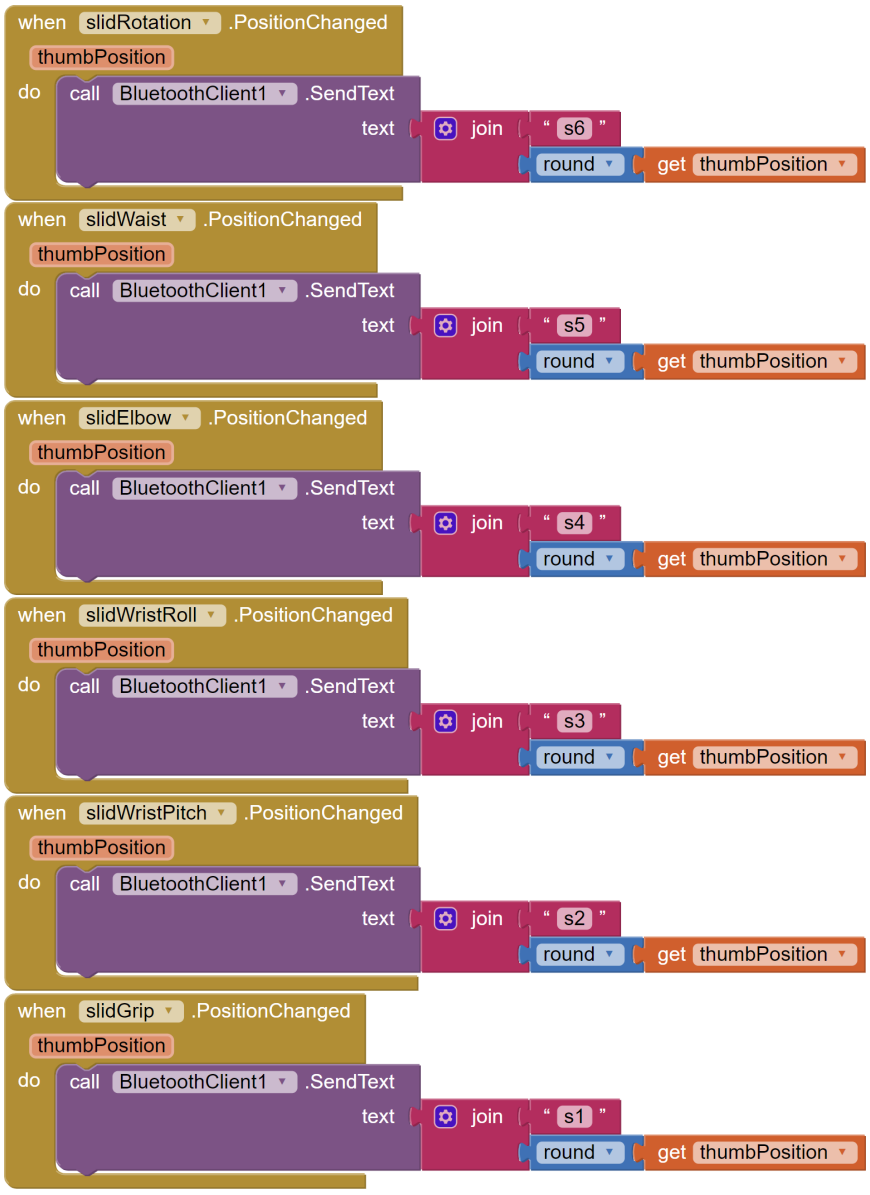
7.3 Bluetooth



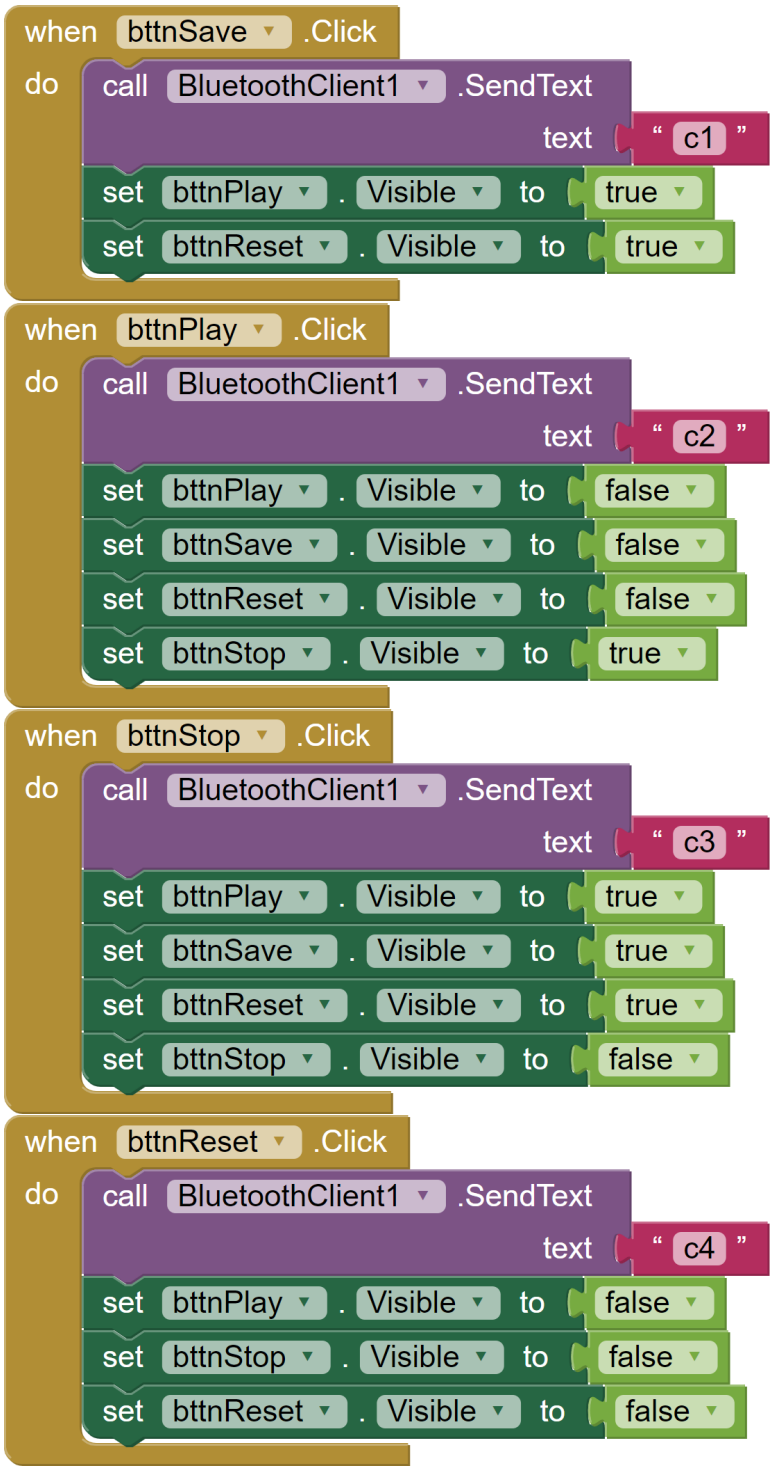
7.4 Przyciski



7.5 Slidery


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

7.6 Komendy


when bttnSave .Click
do call BluetoothClient1 .SendText
text " c1 "
set bttnPlay . Visible to true
set bttnReset . Visible to true
when bttnPlay .Click
do call BluetoothClient1 .SendText
text " c2 "
set bttnPlay . Visible to false
set bttnSave . Visible to false
set bttnReset . Visible to false
set bttnStop . Visible to true
when bttnStop .Click
do call BluetoothClient1 .SendText
text " c3 "
set bttnPlay . Visible to true
set bttnSave . Visible to true
set bttnReset . Visible to true
set bttnStop . Visible to false
when bttnReset .Click
do call BluetoothClient1 .SendText
text " c4 "
set bttnPlay . Visible to false
set bttnStop . Visible to false
set bttnReset . Visible to false