

Projekt nr 1

Zespół „Dumplings”

Strilchuk Mariia
Haivoronska Alina
Drozd Kateryna
Serozhenko Arkadii
Yeusikau Maksim



„RoboCat”

Prezentujemy Państwu naszego robota!

RoboCat – to jest robot-odkurzacz, który:

- Potrafi posprzątać twoje mieszkanie
- Jest sterowany za pomocy aplikacji mobilnej i wifi
- Będzie słuchać twoje głosowe polecenia

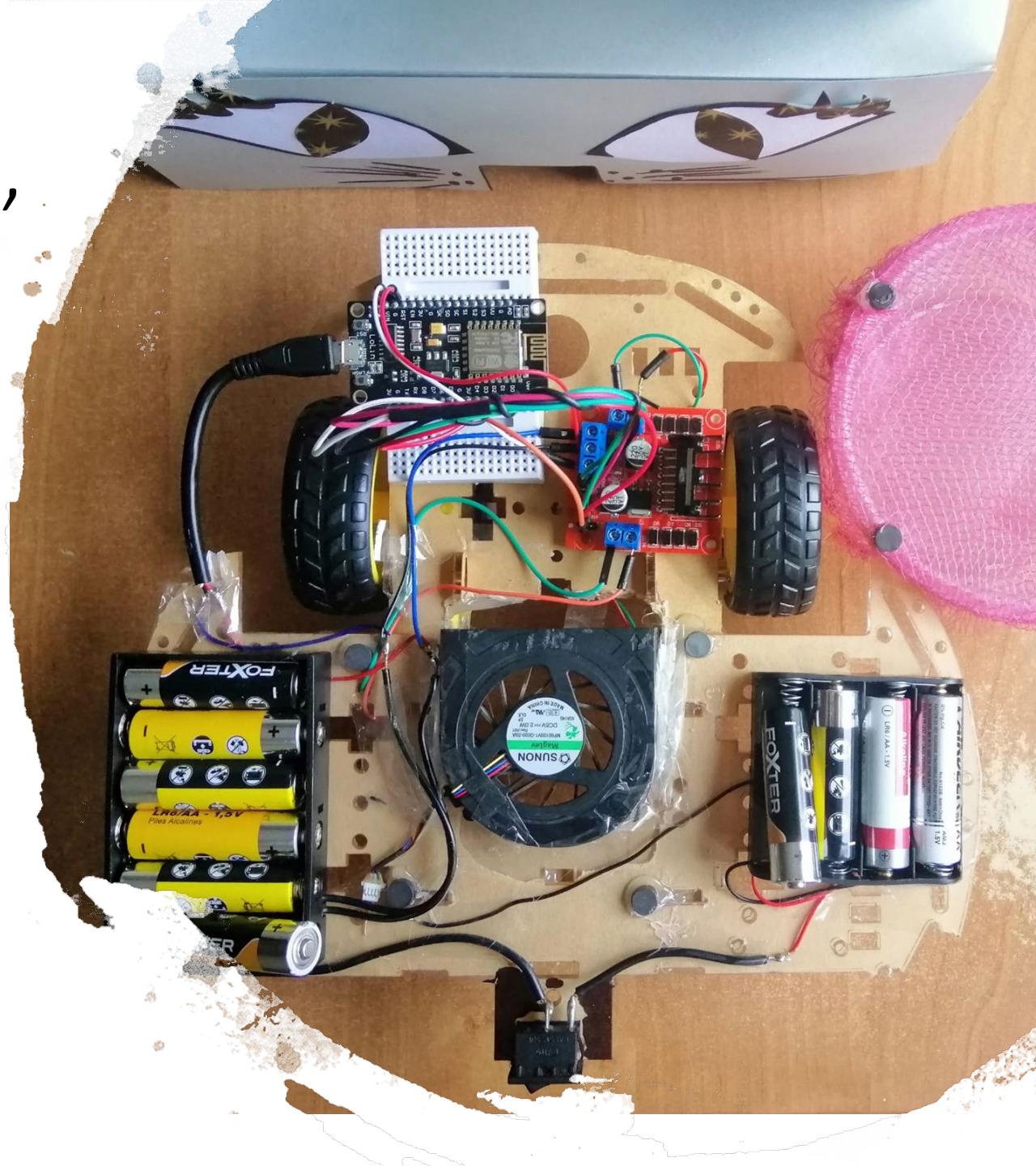
RoboCat jest całkiem naszym pomysłem i
nigdzie nie znajdą Państwo czegoś
podobnego w Internecie! :)



Zaczniemy od początku, czyli

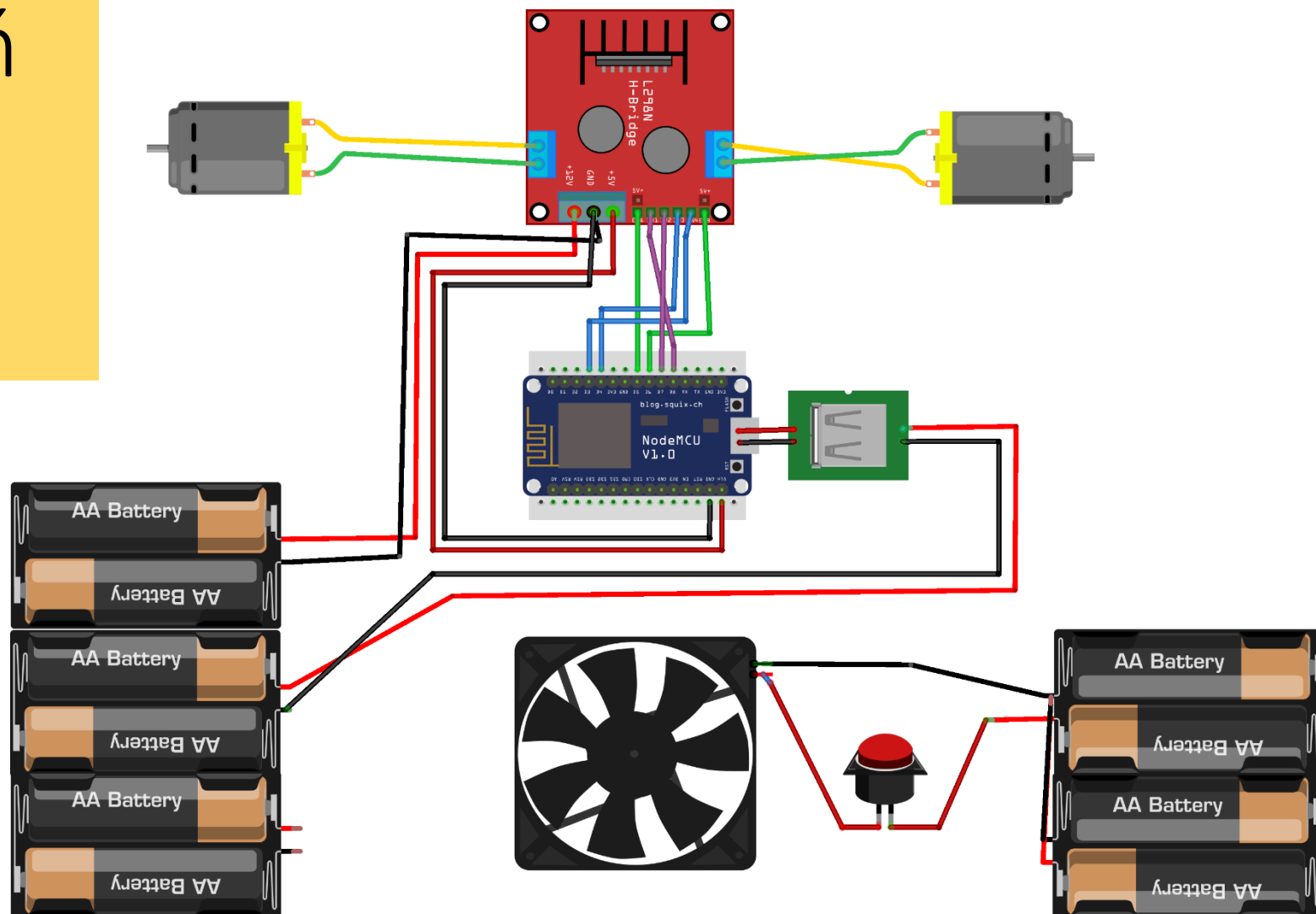
Wykorzystane materiały:

- 1x Moduł NodeMCU
- 1x Moduł sterownika L298N
- 1x Wentylator cooler 5V
- 2x Sterowniki silników DC
- 10x baterie 1.5V
- 1x Przycisk
- 1x Pojemniczek
- 8x Magnezy
- 1x Gąbka do kąpieli (filter)
- 1x Karton
- 100500x Godzin roboty

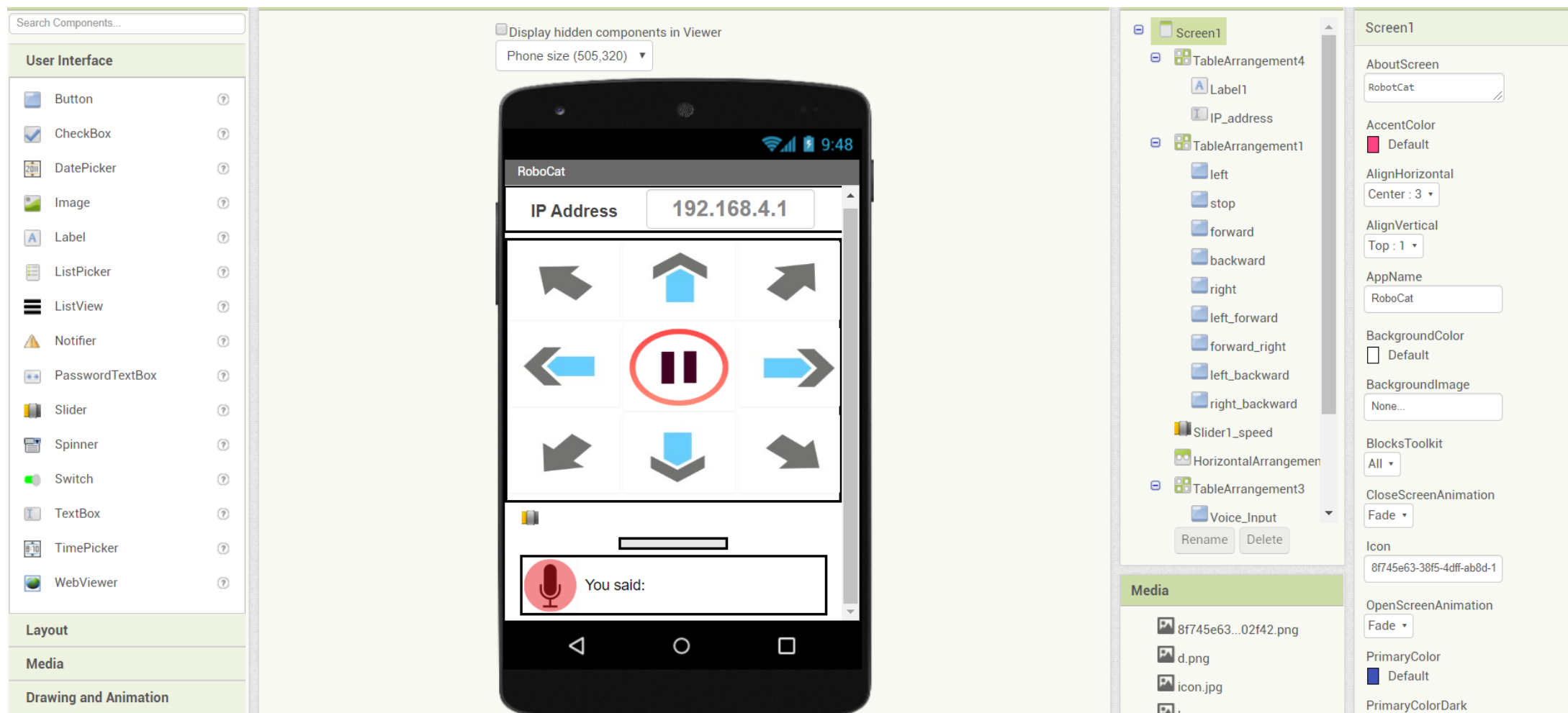


Schemat połączeń

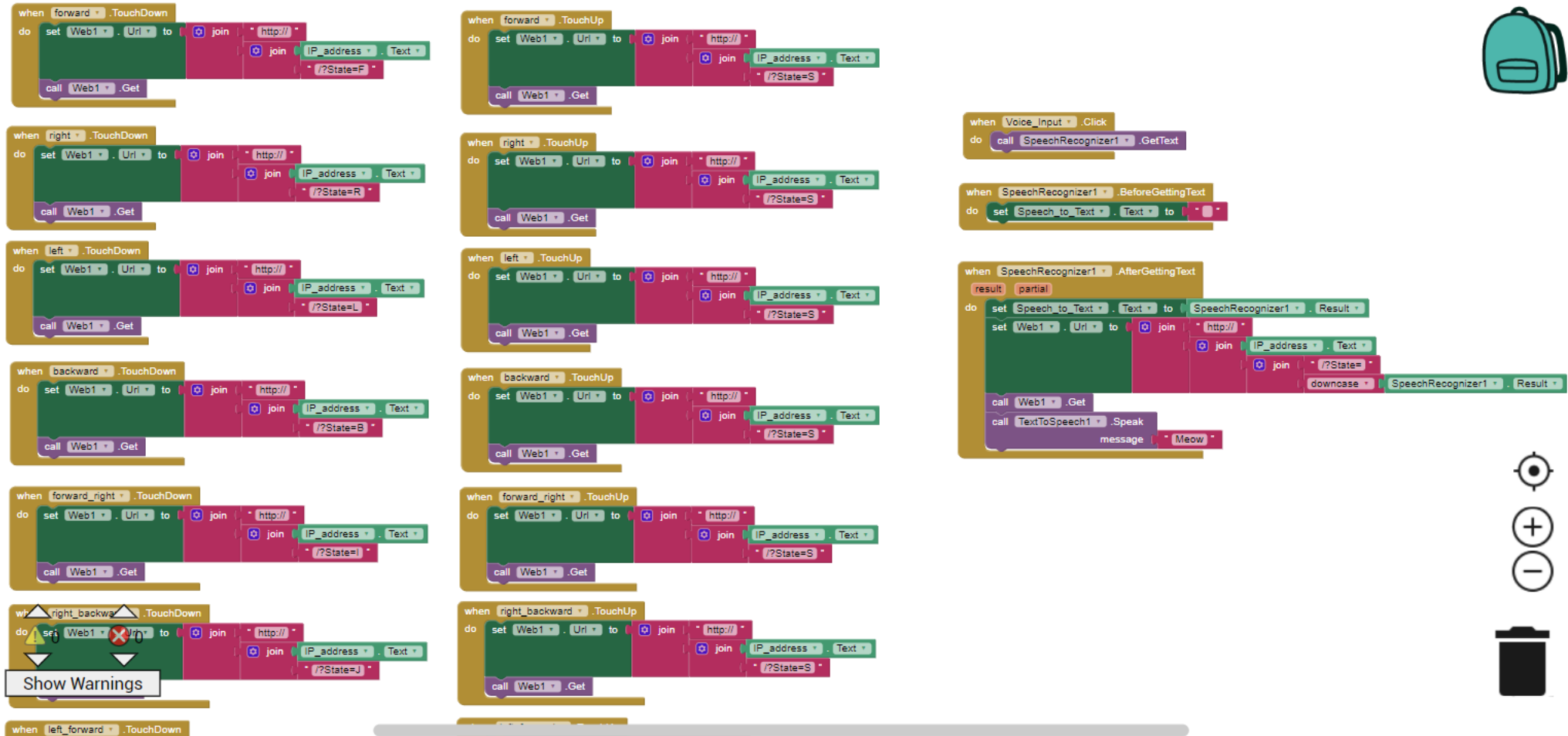
W ten sposób wyglądają połączenia.
Schemat dla łatwości zrozumienia
zaprojektowaliśmy
w programie Fritzing.



Właśnie, zaprojektowaliśmy własną aplikację w środowisku MIT App Inventor



Programowanie w MIT wygląda następująco i nie jest zbyt skomplikowane



The image displays a collection of MIT App Inventor code blocks, organized into two columns. The blocks are color-coded: yellow for 'when' events, green for 'do' loops, and purple for 'call' functions. The code is designed to handle various touch events (forward, right, left, backward, forward_right, right_backward) and voice input. Each touch event block sets a URL, joins it with an IP address and a state parameter, and calls a Web1 .Get function. The voice input block calls SpeechRecognizer1 .GetText. The SpeechRecognizer1 .BeforeGettingText block sets Speech_to_Text .Text to an empty string. The SpeechRecognizer1 .AfterGettingText block sets Speech_to_Text .Text to SpeechRecognizer1 .Result, calls Web1 .Get, and calls TextToSpeech1 .Speak with the message 'Meow'. A 'Show Warnings' button is visible at the bottom left. On the right side, there is a blue backpack icon and a vertical toolbar with a target icon, a plus icon, a minus icon, and a trash can icon.

```
when forward.TouchDown
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when forward.TouchUp
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when right.TouchDown
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when right.TouchUp
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when left.TouchDown
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when left.TouchUp
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when backward.TouchDown
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when backward.TouchUp
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when forward_right.TouchDown
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when forward_right.TouchUp
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when right_backward.TouchDown
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when right_backward.TouchUp
do
  set Web1.Url to join http:// IP_address .Text
  call Web1.Get

when Voice_Input.Click
do
  call SpeechRecognizer1.GetText

when SpeechRecognizer1.BeforeGettingText
do
  set Speech_to_Text.Text to ""

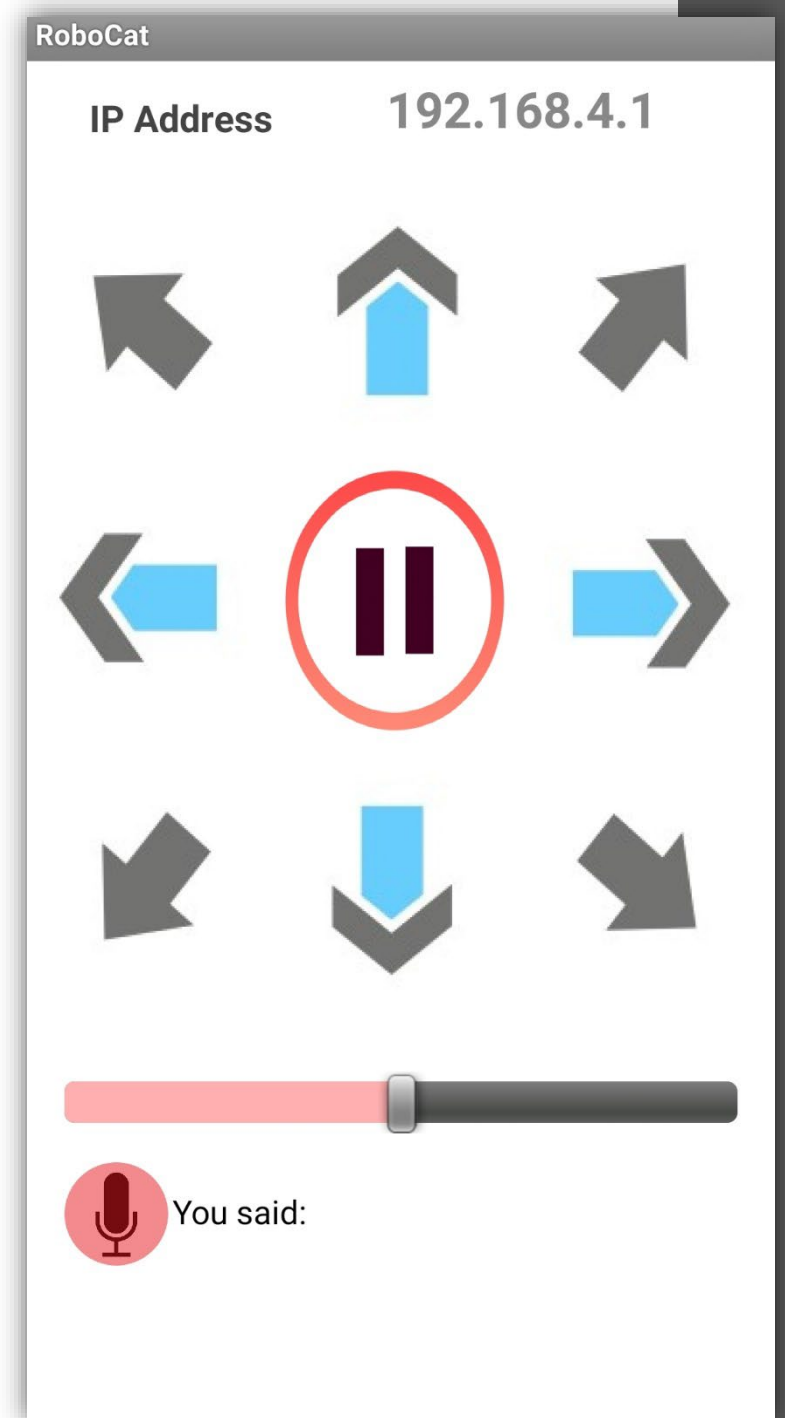
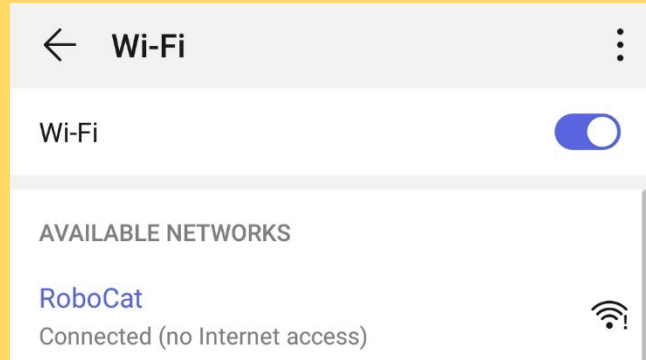
when SpeechRecognizer1.AfterGettingText
do
  set Speech_to_Text.Text to SpeechRecognizer1.Result
  call Web1.Get
  call TextToSpeech1.Speak
  message Meow
```

Aplikacja mobilna

Aplikacja polega na tym, że robot NodeMCU po uruchomieniu tworzy HotSpot Wifi. Podłączając do WIFI wprowadzamy w naszej aplikacji adres IP robota, który w naszym przypadku to *192.168.4.1*

Dalej, klikając na przyciski sterowania, wysyłamy do modułu NodeMCU polecenia, które były poprzednio zaprogramowane w MIT i kodzie Arduino.

W taki sposób robot jest sterowany.

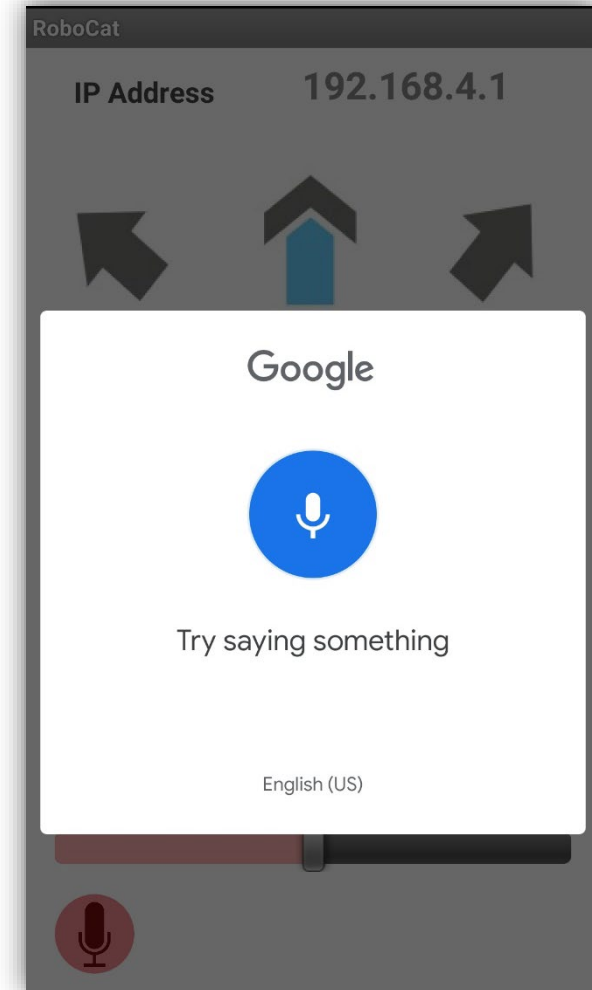
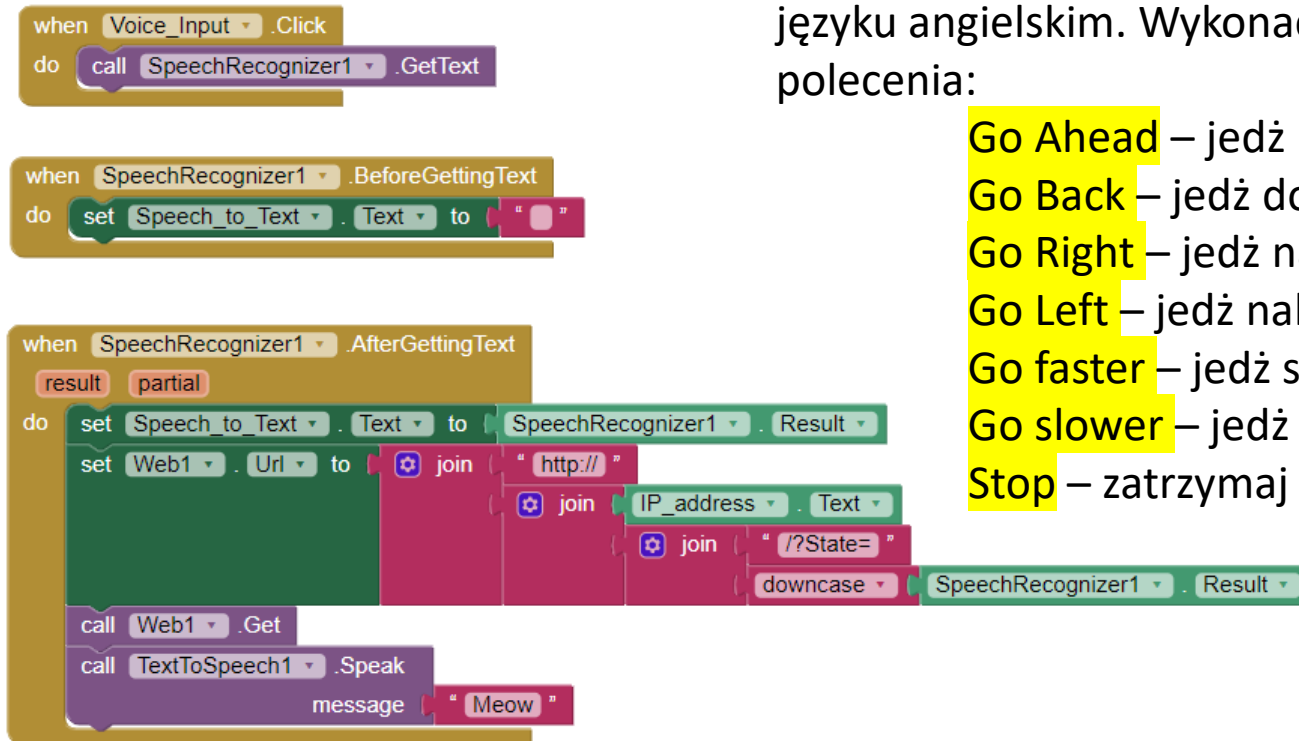


Sterowanie głosem i Google Speech

Sterowanie głosem jest też zaimplementowane w MIT, a w samej aplikacji wywołujemy Google Speech Recognizer do następującego rozpoznania mowy.

Na razie, sterowanie możliwe tylko w języku angielskim. Wykonać można takie polecenia:

Go Ahead – jedź prosto
Go Back – jedź do tyłu
Go Right – jedź naprawo
Go Left – jedź nalewo
Go faster – jedź szybciej
Go slower – jedź wolniej
Stop – zatrzymaj się




```

#define ENA 14 // Enable/speed motors Right GPIO14 (D5)
#define ENB 12 // Enable/speed motors Left GPIO12 (D6)
#define IN_1 15 // L298N in1 motors Right GPIO15 (D8)
#define IN_2 13 // L298N in2 motors Right GPIO13 (D7)
#define IN_3 2 // L298N in3 motors Left GPIO2 (D4)
#define IN_4 0 // L298N in4 motors Left GPIO0 (D3)

```

```

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

String command = ""; //String to store app command state.
int speedCar = 800; // 400 - 1023.
int speed_Coeff = 3;
const char* ssid = "RoboCat";
ESP8266WebServer server(80);

```

```

void setup() {

  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN_1, OUTPUT);
  pinMode(IN_2, OUTPUT);
  pinMode(IN_3, OUTPUT);
  pinMode(IN_4, OUTPUT);

  Serial.begin(115200);

```

Ustawienie HotSpot
robota
i synchronizacja z
aplikacją

Definiowanie podłączonych
pinów do NodeMCU

```

// Connecting WiFi

WiFi.mode(WIFI_AP);
WiFi.softAP(ssid);

IPAddress myIP = WiFi.softAPIP();
Serial.print("AP IP address: ");
Serial.println(myIP);

// Starting WEB-server
server.on ( "/", HTTP_handleRoot );
server.onNotFound ( HTTP_handleRoot );
server.begin();
}

```

Wczytywanie obecnego stanu
pinów (dla poruszania się)

Metoda-polecenie dla
jazdy do przodu

```

void goAhead() {
  digitalWrite(IN_1, HIGH);
  digitalWrite(IN_2, LOW);
  analogWrite(ENA, speedCar);

  digitalWrite(IN_3, LOW);
  digitalWrite(IN_4, HIGH);
  analogWrite(ENB, speedCar);
}

```

Programowanie w Arduino

```
void loop() {
```

```
    server.handleClient();  
    command = server.arg("State");  
    if (command == "F" || command == "go ahead") goAhead();  
    else if (command == "B" || command == "go back") goBack();  
    else if (command == "L" || command == "go left") goLeft();  
    else if (command == "R" || command == "go right") goRight();  
    else if (command == "I" || command == "go ahead right") goAheadRight();  
    else if (command == "G" || command == "go ahead left") goAheadLeft();  
    else if (command == "J" || command == "go back right") goBackRight();  
    else if (command == "H" || command == "go back left") goBackLeft();  
    else if (command == "0") speedCar = 400;  
    else if (command == "1" || command == "go slower") speedCar = 470;  
    else if (command == "2") speedCar = 540;  
    else if (command == "3") speedCar = 610;  
    else if (command == "4") speedCar = 680;  
    else if (command == "5") speedCar = 750;  
    else if (command == "6") speedCar = 820;  
    else if (command == "7") speedCar = 890;  
    else if (command == "8") speedCar = 960;  
    else if (command == "9" || command == "go faster") speedCar = 1023;  
    else if (command == "S" || command == "stop") stopRobot();  
}
```

Polecenia głosowe

Wykonanie metody
„jeżeli”
było takie polecenie

Regulowanie prędkości

```
void HTTP_handleRoot(void) {
```

```
    if( server.hasArg("State") ){  
        Serial.println(server.arg("State"));  
    }  
    server.send ( 200, "text/html", "" );  
    delay(1);  
}
```

Polecenia rozpoznawane po
kliknięciu przycisków w aplikacji

Sposób wczytywania
poleceń z aplikacji i
przekazanie je na „serwer”

No i w sumie to wszystko...

Może się wydawać że był robot zrobiony tak -



Ale szczerze mówiąc wyglądało to następująco:





Dziękujemy za uwagę!

