Robot do teleprezencji sterowany poprzez Wi-Fi

Teleprezencja – zwana „zdalną obecnością" to w robotyce określenie na ogół systemów, pozwalających przenieść naszą osobę w zdalne miejsce. Zazwyczaj działa to w dwie strony, szerzej niż w przypadku np. wideokonferencji, gdyż nie tylko widzimy i jesteśmy widziani, to możemy także oddziaływać z swoim otoczeniem i często poruszać się.

Prezentowany w artykule prosty robot służy do realizowania zdalnej obecności – pozwala na częściowe przeniesienie się w oddalony od nas obszar. Wystarczy, że zapewnione będzie połączenie robota z siecią Internet poprzez Wi-Fi.

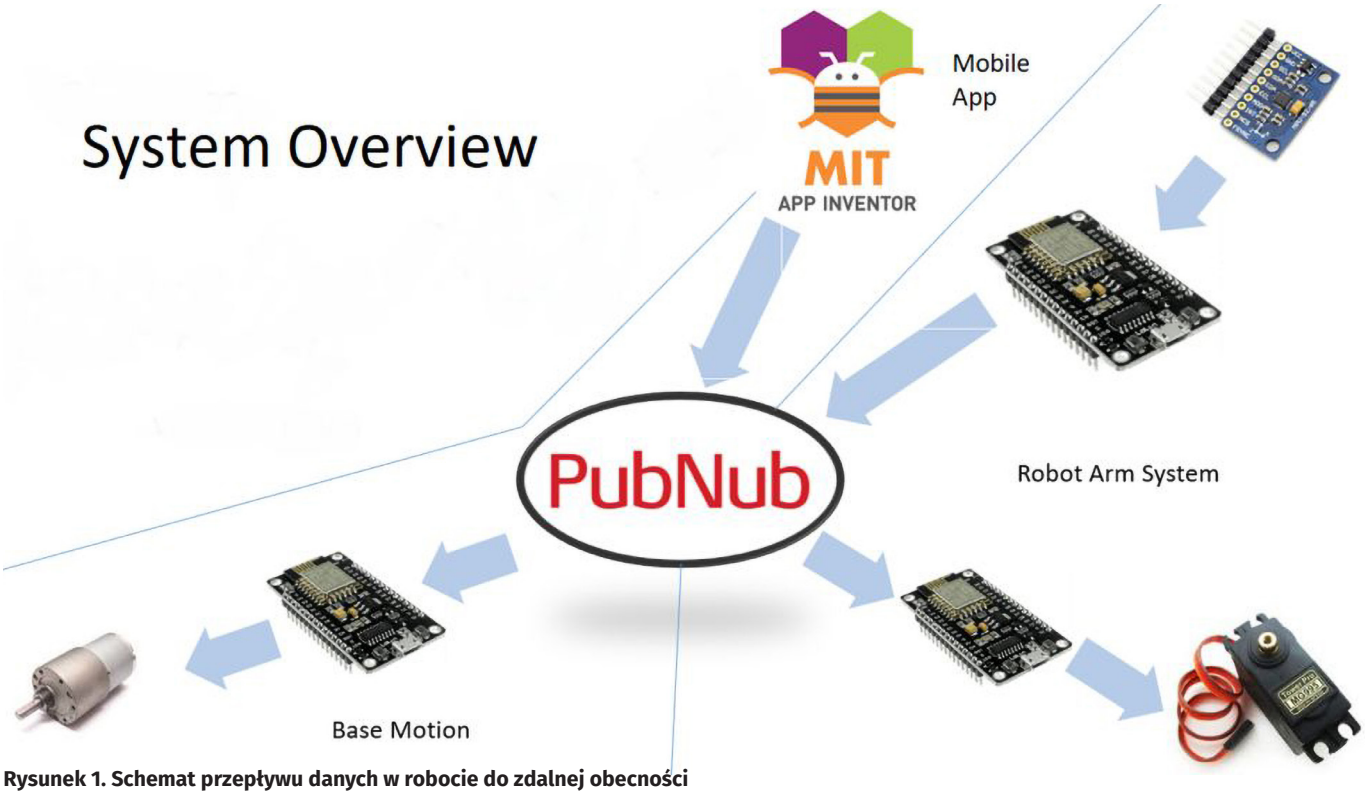
Koncepcja teleprezencji może wydawać się abstrakcyjna, jednakże, to, co do niedawna było, co najwyżej pomysłem z książek science-fiction, obecnie jest proste do zrealizowania nawet we własnym domu. Jeszcze niedawno wideorozmowy pojawiały się tylko filmach SF, a teraz każdy z nas może komunikować się w ten sposób z dowolną osobą na całym świecie. Teleprezencja jest rozszerzeniem idei telekonferencji. Aby mówić o zdalnej obecności, oprócz transmitowania (w obie strony) dźwięku i obrazu, musimy także mieć możliwość, chociażby podstawowych, interakcji z otoczeniem po drugiej stronie łącza.

Opisany projekt to sposób budowy prostego robota do zdalnej obecności, który może być kontrolowany poprzez sieć bezprzewodową Wi-Fi. Potrzebuje Internetu do komunikacji, ale może być sterowany z pomocą prostej aplikacji z dowolnego miejsca na świecie, dzięki wykorzystaniu sieci PubNub do przesyłania danych. Robot taki to nie tylko narzędzie do zaawansowanej komunikacji. Przede wszystkim system ten przydać może się osobom niepełnosprawnym. Dzięki możliwości poruszania się i interakcji z otoczeniem, może zrobić wiele czynności, chociażby pozwolić na samodzielne podanie sobie czegoś do łóżka, z którego chorzy nie są w stanie wstać.

Alternatywnym wykorzystaniem systemów teleprezencji, jest praca w miejscu, gdzie ludzie nie mogą się dostać, albo warunki tam panujące mogłyby być dla nich szkodliwe. Sztandarowym przykładem mogą tutaj być zdalnie sterowane sondy kosmiczne – marsjański łazik nie jest niczym innym, niż bardzo zaawansowanym systemem do zdalnej obecności człowieka na Marsie.

Potrzebne elementy

Robot składa się z kilku niezależnych systemów, które współpracują ze sobą, by umożliwić robotowi pełne działanie. Do jego budowy potrzebować będziemy, oprócz zestawu narzędzi czy komputera, następujące elementy i moduły elektroniczne i mechaniczne:



Rysunek 1. Schemat przepływu danych w robocie do zdalnej obecności

- trzy moduły NodeMCU,
- sterownik silnika DC z L298N,
- zasilacz 12 V,
- moduł ze stabilizatorem napięcia LM2596,
- dwa moduły z sensorami inercjalnymi (IMU) MPU9250,
- cztery serwomotory (moment 1...2 Nm),
- silnik elektryczny prądu stałego 12 V (moment, co najmniej 2,5 Nm),
- elementy konstrukcyjne do robota – może być fragment ok. 1×1 m sklejki,
- pręty gwintowane do skręcenia konstrukcji (dwa kawałki po 1 m),
- elementy z druku 3D (opisane w tekście),
- koła dla robota.

Komunikacja

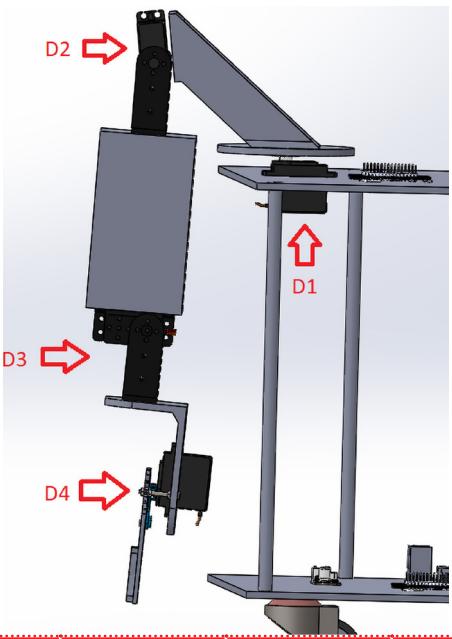
Na **rysunku 1** zaprezentowano schemat przepływu danych, który pozwala zrozumieć, w jaki sposób poszczególne komponenty robota komunikują się ze sobą. W systemie wykorzystana jest sieć do przesyłu danych o nazwie PubNub. Jest to platforma IoT, która może wysyłać wiadomości w czasie rzeczywistym z opóźnieniami na poziomie zaledwie 0,5 sekundy. To najszybszy czas reakcji, jaki możemy uzyskać, wykorzystując łączność przez sieć Internet. Jest to bardzo ważne, ponieważ sterowanie ramieniem robota musi działać w czasie rzeczywistym.

Wszystkie NodeMCU w systemie służą do wysyłania i odbierania danych. W urządzeniu znajdują się dwa osobne systemy, korzystające niezależnie od siebie z sieci PubNub. Moduł NodeMCU na ramieniu robota wysyła dane z czujnika ruchu do PubNub, które są następnie odbierane przez drugie NodeMCU w ramieniu. Drugi system steruje ruchem podstawy robota – gdy aplikacja mobilna wysyła dane z joysticka, jako współrzędne X i Y, są one odbierane przez NodeMCU w podstawie, która może sterować silnikiem poprzez dedykowany sterownik.

Projekt mechaniczny

Na **fotografii tytułowej** pokazano wizualizację projektu robota – daje ona dobre wyobrażenie o tym, jak ma wyglądać całe urządzenie. Wszystkie pliki CAD można pobrać z portali GrabCAD. Podstawa robota wsparta jest na trzech kołach, z których dwa podłączone są do silników prądu stałego z tyłu. Trzecie, przednie koło obraca się swobodnie. Taka konstrukcja może powodować niestabilność ruchu, z uwagi na obciążenie ramieniem z przodu. W takiej sytuacji pojedyncze przednie koło można zastąpić dwoma.

Dolne i górne 'piętra' konstrukcji robota wykonane są ze sklejki. Są połączone ze sobą z pomocą gwintowanych prętów i kontrujących nakrętek. Montując system należy upewnić się, że używa się



Lokacja w robocie	Wykorzystany serwomotor	Początkowy kąt	Zakres ruchów serwa
D1	JX servo	90	140-30
D2	Kuman	10	10-110
D3	Kuman	90	0-180
D4	MG995	90	0-180

Rysunek 2. Schemat ustawień poszczególnych serwomotorów ramienia podczas montażu

Listing 1. Prosty skrypt Arduino pozwalający ustawić serwomotor do zadanego kąta

```
#include <Servo.h>
Servo servo;
// wpisz zamiast X numer pinu,
//do którego podłączone jest serwo
int pin =  X;

void setup() {
    servo.attach(pin);
}

void loop() {
// wpisz zamiast Y kąt
//jaki należy ustawić
    int angle = Y;
    servo.write(angle);
}
```

nakrętek zabezpieczających, dzięki temu robot będzie pewnie pracował przez długi czas.

Cały projekt robota pobrać można tutaj – ***https://bit.ly/2U5htui***. Wiele z elementów urządzenia jest wykonanych w technologii druku 3D. Autor udostępnia wszystkie potrzebne do druku pliki na stronie projektu.

Ramię robota do teleprezencji ma prostą konstrukcję. Elementy zostały zoptymalizowane pod kątem druku 3D przy minimalnej ilości potrzebnego filamentu. Długość ramienia wynosi około 40 cm – w przybliżeniu tyle, ile wynosi ludzkie ramie. Długość ramienia robota zależy od momentu obrotowego generowanego przez serwosilniki. Dokładne obliczenie momentu obrotowego wymaganego w tej konstrukcji znaleźć można na stronie z projektem na portalu instructables.com. Dzięki Temu można dostosować projekt do własnych potrzeb. Należy jednak unikać wykorzystywania maksymalnego momentu obrotowego danego serwomotoru, ponieważ w dłuższej perspektywie spowoduje to jego uszkodzenie.

Montaż robota jest bardzo prosty i ogranicza się do kilku kroków:

1. Przytnij gwintowane pręty na odpowiadającą Tobie długość.
2. Wytnij 2 kawałki sklejki o wymiarach 40×30 cm każdy.
3. Wywierć niezbędne otwory w górnej i dolnej podstawie, jak na rysunku.
4. Zamontuj silniki prądu stałego i kółka na dolnej podstawie.
5. Aby wykonać prostokątny otwór w górnej podstawie, najpierw wykonaj wiertarką okrągły otwór, a następnie wyrzynarką dotnij go wzdłuż krawędzi, aby utworzyć prostokąt.

Rozmieszczenie otworów do montażu gwintowanych prętów należy dobrać do miejsca montażu ramienia tak, aby zachować najlepszą równowagę masy na robocie.

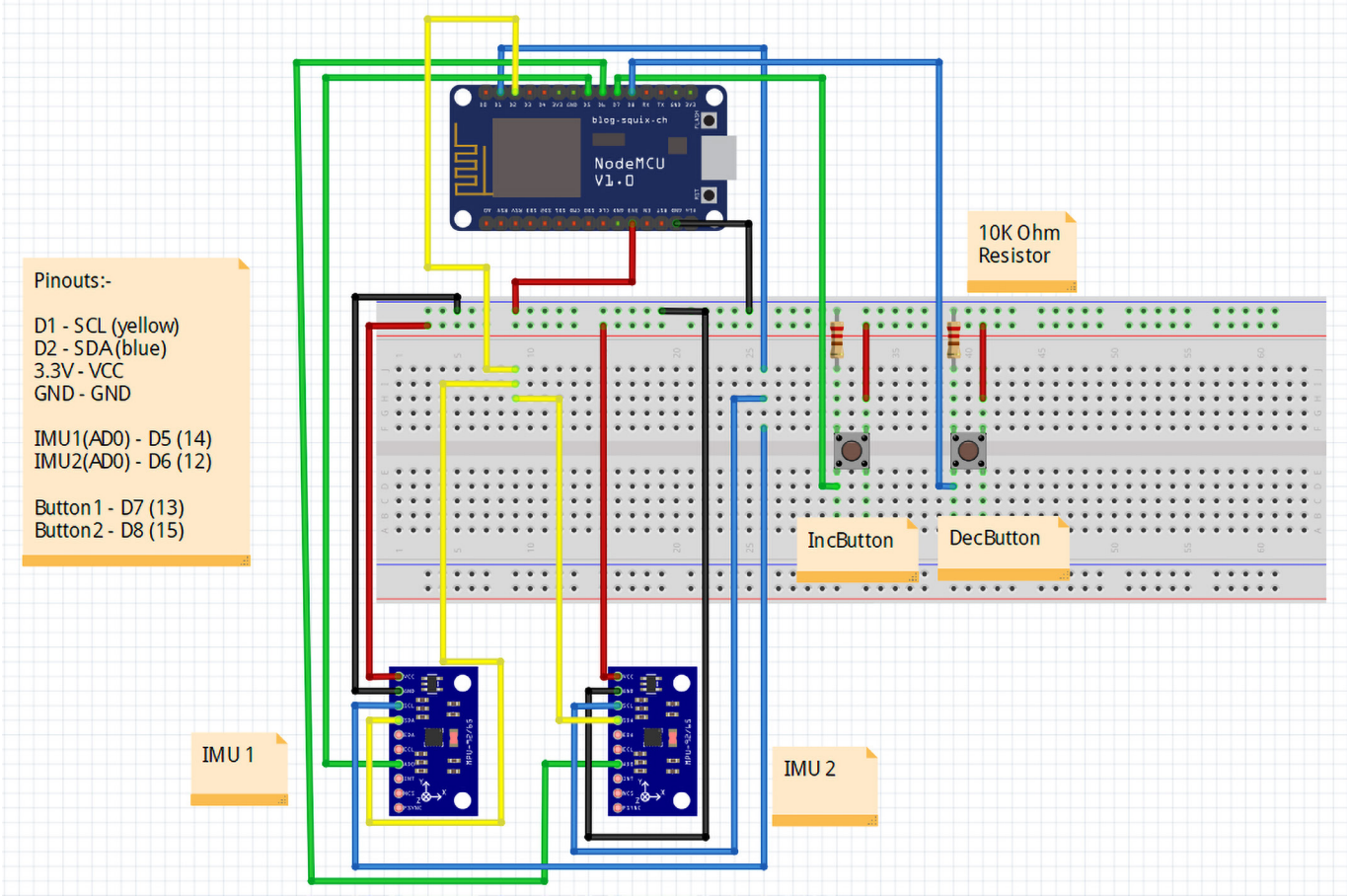
Montaż ramienia robota wymaga szczególnej uwagi. Poza montażem mechanicznym należy upewnić się, że serwa są ustawione pod odpowiednim kątem podczas montażu – zgodnie ze schematem pokazanym na **rysunku 2**. Do ustawienia kąta silników można skorzystać z kodu z **listingu 1**, i modułu Arduino lub NodeMCU. Dużo informacji na temat połączenia serwomechanizmów z modułami Arduino można znaleźć w Internecie, więc nie ma sensu szczegółowe omawianie tego prostego skryptu i sposobu podłączenia serwa.

Układ kontroli ramienia

Układ elektroniczny sterujący ramieniem robota jest dosyć prosty. Składa się z dwóch modułów inercyjnych, które służą do pomiaru położenia ręki operatora. Jej ruch jest następnie tłumaczony, na ruch zdalnego ramienia. Schemat tego kontrolera pokazano na **rysunku 3**.

Kontroler wykonano w oparciu o moduł NodeMCU oraz moduły z sensorami IMU, które wszyte zostały w elastyczny, tekstylny rękaw. Montując sensory na rękawie, koniecznie należy upewnić się, że sensory inercyjne wszyte są w rękaw w poprawnej orientacji.

Finalnie, można podłączyć moduł z mikrokontrolerem do komputera i załadować do niego kod programu poprzez Arduino IDE. Skrypt, który należy załadować do modułu NodeMCU w sterowniku ramienia znajduje się na **listingu 2**. W kodzie programu należy uzupełnić nazwę sieci Wi-Fi, z jaką ma się łączyć moduł i inne dane do logowania, a także dane konta PubNub (o nim w dalszej części artykułu).



Rysunek 3. Schemat kontrolera ramienia robota

Listing 2. Szkic Arduino dla sterownika ramienia

```
void setup() {
    // Wpisz prędkość portu szeregowego do debugowania
    Serial.begin(115200);
    pinMode(imu1, OUTPUT);
    pinMode(imu2, OUTPUT);
    pinMode(IncButton, INPUT);
    pinMode(DecButton, INPUT);

        Serial.println(„Connecting to „);
        Serial.println(ssid);

        WiFi.begin(ssid, pass);
        while (WiFi.status() != WL_CONNECTED){
            delay(500);
            Serial.print(„.”);
        }
        Serial.println(„”);
        Serial.println(„WiFi connected”);
    // Uruchomienie biblioteki PubNub z zadanymi kluczami
    PubNub.begin(pubkey, subkey);
    // Uruchomienie komunikacji z IMU
    IMU.begin();
}


void loop() {
    digitalWrite(imu1, LOW);
    digitalWrite(imu2, HIGH);
    IMU.readSensor();
    int shoulder_yaw = map(IMU.getMagY_uT(), -15, 60, 30, 140);
    int shoulder_pitch = map(IMU.getAccelX_mss(), -10, 10, 180, 0);
    digitalWrite(imu1, HIGH);
    digitalWrite(imu2, LOW);
    IMU.readSensor();
    int arm_pitch = map(IMU.getAccelX_mss(), -8, 8, 0, 180);

    IncGripper = digitalRead(IncButton);
    DecGripper = digitalRead(DecButton);
    Serial.print(IncGripper);
    Serial.print(DecGripper);

    if (IncGripper == HIGH) {
        pos += 5;
    }
    if (DecGripper == HIGH) {
        pos -= 5;
    }
    if (pos < 0) pos=0;
    if (pos > 90) pos=90;

    Serial.print(„shoulder yaw: „);
    Serial.print(shoulder_yaw);
    Serial.print(„shoulder pitch: „);
    Serial.print(shoulder_pitch);
    Serial.print(„arm pitch: „);
    Serial.println(arm_pitch);
    Serial.print(„Gripper Position: „);
    Serial.println(pos);

    char buf[500] = {};
    const int capacity = JSON_OBJECT_SIZE(100);
    StaticJsonDocument<capacity> doc;

    doc[„servo1”] = shoulder_yaw;
    doc[„servo2”] = shoulder_pitch;
    doc[„servo3”] = arm_pitch;
    doc[„grip”] = pos;

    serializeJson(doc, buf);
    PubNonSubClient *pclient = PubNub.publish(pubchannel, buf);
    if (!pclient) return;
    PublishCracker cheez;
    cheez.read_and_parse(pclient);

    Serial.print(„Outcome: „); Serial.print(cheez.outcome());
    Serial.print(‘ ‘); Serial.println(cheez.to_str(cheez.outcome()));
    Serial.print(„description: „); Serial.println(cheez.description());
    Serial.print(„timestamp: „); Serial.println(cheez.timestamp());
    Serial.print(„state: „); Serial.print(cheez.state());
    Serial.print(‘ ‘); Serial.println(cheez.to_str(cheez.state()));

    pclient->stop();
}
```

Listing 3. Szkic Arduino kontrolera ramienia

```
void setup() {
...

    myservo1.attach(D1);
    myservo1.write(90);
    myservo2.attach(D2);
    myservo2.write(60);
    myservo3.attach(D5);
    myservo3.write(0);
    myservo4.attach(D6);
    myservo4.write(90);
}


void loop() {
    PubSubClient *sclient = PubNub.subscribe(subchannel);
    if (!sclient) return; // błąd
    String msg;
    SubscribeCracker ritz(sclient);
    while (!ritz.finished()) {
        ritz.get(msg);
        if (msg.length() > 0) {
            Serial.print(„Received: „);
            Serial.println(msg);

            DynamicJsonDocument json(200);
            deserializeJson(json, msg);

            int angle1 = json[„servo1”];
            int angle2 = json[„servo2”];
            int angle3 = json[„servo3”];
            int angle4 = json[„grip”];

            Serial.println(„Servo angles:- „);

            Serial.println(angle1);
            Serial.println(angle2);
            Serial.println(angle3);
            Serial.println(angle4);

            myservo1.write(angle1);
            myservo2.write(angle2);
            myservo3.write(angle3);
            myservo4.write(angle4);
        }
    }
    sclient->stop();
}
```

Elektronika sterująca

Pozostałe dwa moduły NodeMCU znajdują się na pokładzie robota i służą do sterowania jego silnikami. Jeden moduł odpowiada za sterowanie silnikiem poruszającym całą platformą, a drugi kontroluje serwomotory ramienia. Schemat połączeń obu modułów ze sterownikami silników pokazano na **rysunku 4**. Postępuj zgodnie z tym schematem, aby połączyć w odpowiedni sposób poszczególne moduły składowe robota.

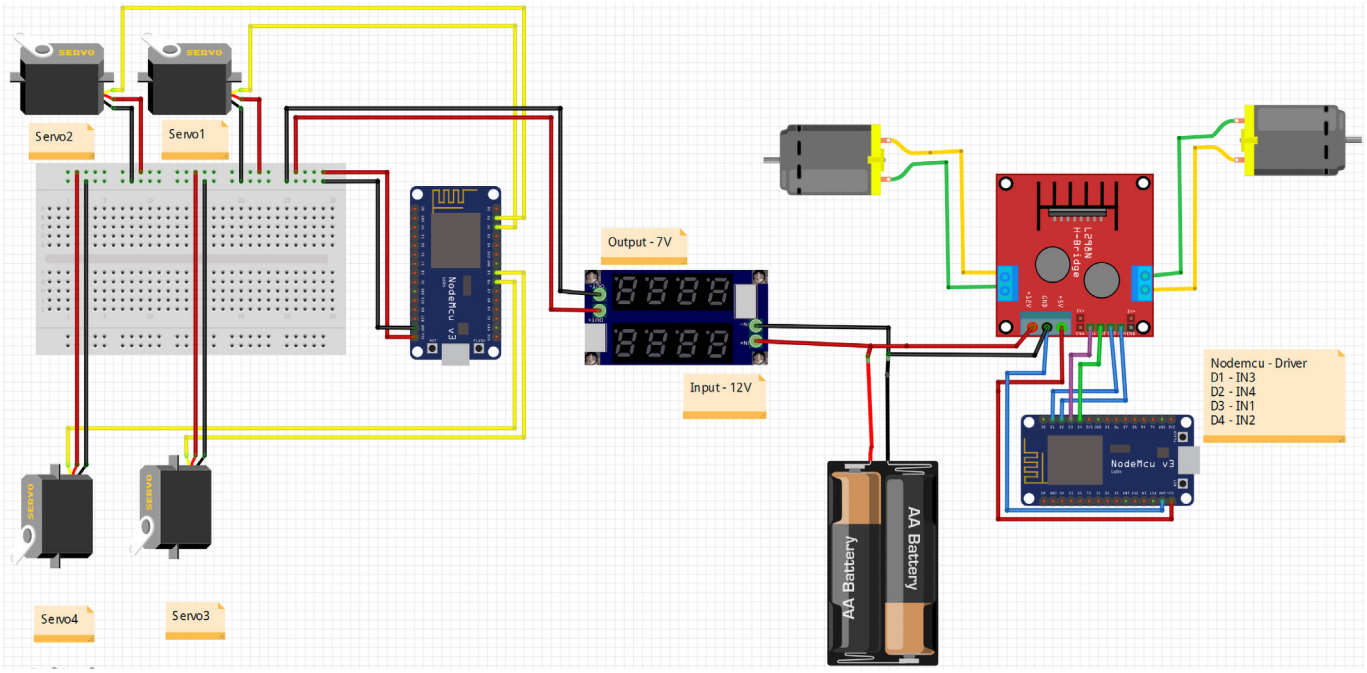
Wszystkie połączenia można wykonać z pomocą zworek nasadzanych na goldpiny modułów. Jedynym miejscem, gdzie należy wykonać połączenie lutowane są zaciski silników prądu stałego. Pobiera on sporo prądu i należy zagwarantować niską rezystancję styku. Po wykonaniu wszystkich połączeń, przed pierwszym uruchomieniem systemu, należy sprawdzić czy nigdzie nie ma zwarć i ustawić napięcie wyjściowe przetwornicy buck na 7 V. Będzie ona zasilała serwosilniki.

Po wykonaniu połączeń elektrycznych i uruchomieniu systemu można załadować firmware do poszczególnych modułów NodeMCU. Do układu sterującego ramieniem należy wgrać szkic z **listingu 3**, a do drugiego z modułów, który steruje silnikiem podstawy, kod z **listingu 4**. W obu szkicach należy uzupełnić wartości zmiennych, związane z danymi sieci Wi-Fi (SSID oraz hasło) oraz sieci PubNub.

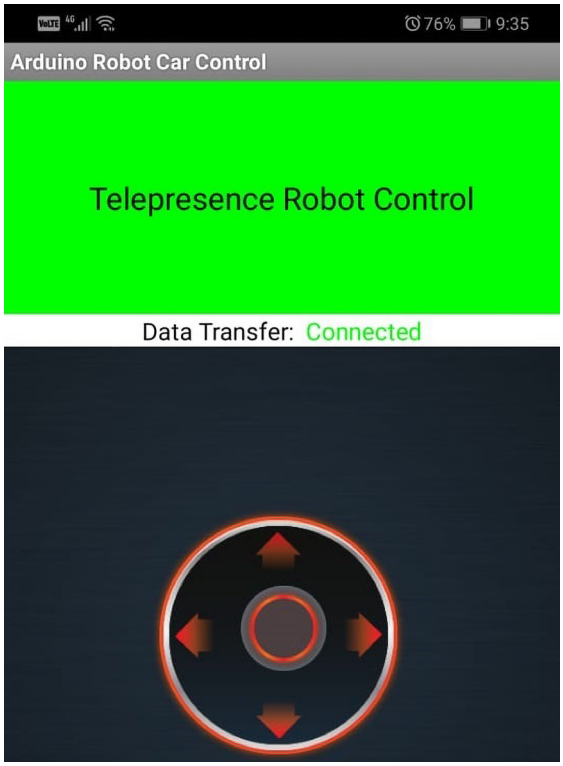
Aplikacja na telefon komórkowy

Do sterowania ruchem robota w osiach X oraz Y. Na **rysunku 5** pokazano widok aplikacji, uruchomionej na smartfonie. Do sterowania pozycją robota służy znajdujący się na środku joystick. Kiedy użytkownik porusza joystickiem, aplikacja wysyła współrzędne X, Y do PubNub, skąd dane są pobierane przez moduł NodeMCU w robocie. Następnie współrzędna są konwertowane na kąt i na tej podstawie robot ustala, w którym kierunku ma się poruszyć. Ruch odbywa się poprzez włączenie/wyłączenie i zmianę kierunku obrotów dwóch silników prądu stałego. Jeśli na przykład polecenie nakazuje ruch do przodu, to oba silniki poruszają się do przodu z pełną prędkością, jeśli w lewo, lewy silnik będzie się cofał, a prawy silnik poruszał do przodu i tak dalej.

Na stronie projektu można pobrać plik źródłowy aplikacji w formacie *.aia, który można edytować za pomocą narzędzia do tworzenia aplikacji MIT. W aplikacji należy wykonać pewne kroki konfiguracyjne, zanim będzie można z niej w pełni korzystać.



Rysunek 4. Schemat połączeń elektroniki robota do teleprezencji



Rysunek 5. Widok aplikacji, uruchomionej na smartfonie

Sieć PubNub

Ostatni krok to skonfigurowanie platformy IoT. PubNub jest tutaj bardzo dobrym wyborem, ponieważ jak pisano powyżej, oferuje bardzo niskie opóźnienia – na poziomie zaledwie 500 ms. Dodatkowo, jest darmowy (do miliona punktów danych miesięcznie), więc nie zwiększa budżetu projektu ani nie wprowadza stałych kosztów eksploatacji systemu.

Zakładanie konta

W pierwszej kolejności należy utworzyć na portalu PubNub (www.pubnub.com) nowe konto użytkownika, jeśli jeszcze takiego nie posiadamy. Następnie, przechodzimy do menu aplikacji (po lewej stronie) i klikamy przycisk podpisany „+Create New App" (utwórz nową aplikację) po prawej stronie. Po nadaniu nazwy aplikacji otworzy się menu, tak jak pokazano na **rysunku 6**, gdzie znaleźć możemy obraz klucza wydawcy i subskrybenta. Tego będziemy używać do łączenia urządzeń z siecią.

Konfiguracja konta w oprogramowaniu

Do skonfigurowania firmware potrzebujemy 4 rzeczy, aby urządzenia mogły się ze sobą komunikować: – klucz publiczny (pubkey), podklucz (subkey), kanał (channel) i dane Wi-Fi (SSID oraz hasło). Klucz i podklucz pozostaną takie same we wszystkich aplikacjach NodeMCU i aplikacjach mobilnych. Dwa urządzenia, które się komunikują, powinny mieć tę samą nazwę kanału. Ponieważ aplikacja mobilna i baza komunikują się ze sobą i muszą

Rysunek 6. Okno z kluczami na stronie PubNub

Listing 4. Szkic Arduino kontrolera silników podstawy

```
void loop() {
    PubSubClient *sclient = PubNub.subscribe(subchannel);
    if (sclient) {
    String msg;
    SubscribeCracker ritz(sclient);
    if (!ritz.finished()) {
        ritz.get(msg);
        if (msg.length() == 0) {
            Serial.println(„motor stop”);
            return;
        } else {
            Serial.print(„Received: „); Serial.println(msg);

            String X = getStringPartByNr(msg, ‘:’, 0);
            String Y = getStringPartByNr(msg, ‘:’, 1);

            X.remove(0,1);

            int Xcord = X.toInt();
            int Ycord = Y.toInt();

            Serial.print(„X: „); Serial.println(Xcord);
            Serial.print(„Y: „); Serial.println(Ycord);

            int angle;
            int throttle = (Xcord + Ycord)/2;

            if ((Xcord > 100) && (Ycord > 100)) {
                angle = round(atan2(Ycord-100,Xcord-100)*(180/3.14));
            }
            else if ((Xcord < 100) && (Ycord > 100)) {
                angle = 180 - round(atan2(Ycord-100,100-Xcord)*(180/3.14));
            }
            else if ((Xcord < 100) && (Ycord < 100)) {
                angle = 180 + round(atan2(100-Ycord,100-Xcord)*(180/3.14));
            }
            else if ((Xcord > 100) && (Ycord < 100)) {
                angle = 270 + round(atan2(Xcord-100,100-Ycord)*(180/3.14));
            }

            Serial.print(„Angle: „); Serial.println(angle);

            if ((45 <= angle) && (angle < 135)) {
                motorR.forward();
                motorL.forward();
                Serial.println(„forward”);
            }
            else if ((135 <= angle) && (angle < 225)) {
                motorR.forward();
                motorL.backward();
                Serial.println(„Left”);
            }
            else if ((225 <= angle) && (angle < 315)) {
                motorR.backward();
                motorL.backward();
                Serial.println(„Back”);
            }
            else if ((315 <= angle) || (angle < 45)) {
                motorR.backward();
                motorL.forward();
                Serial.println(„Right”);
            }
        }
    }
    sclient->stop();
    }
}

String getStringPartByNr(String data, char separator, int index){
    // zmienna do zliczania numerów partii danych
    int stringData = 0;
    // zmienna do zwracania tekstu z funkcji
    String dataPart = „”;
    // Parsowanie tekstu po literze
    for(int i = 0; i<data.length()-1; i++) {

        if(data[i]==separator) {
        // Zlicza ile razy w tekście napotka separator
            stringData++;

        } else if(stringData==index) {
            // Skopiuj tekst, gdy separator jest odpowiedni
            dataPart.concat(data[i]);

        }else if(stringData>index) {
            // Zwróć tekst, gdy pojawi się kolejny separator
            // oszczędza czas CPU
            return dataPart;
            break;
        }
    }
    // Zwraca tekst
    return dataPart;
}
```

mieć tą samą nazwę kanału dla aplikacji na smartfona i robota. Tak jak wspominaliśmy powyżej, firmware należy także uzupełnić o dane Wi-Fi – hasło i nazwę sieci.

W załączonych na listingach szkicach Arduino kanały są już domyślnie nazwane, więc jedyne linijki, które należy uzupełnić to Wi-Fi oraz klucz i podklucz. W każdym skrypcie wyglądają one tak samo:

```
const char *ssid =  „”;
// Wpisz swój SSID i klucz sieci Wi-Fi
const char *pass =  „”;
const char *pubkey = „”;
// Wpisz swoje klucze PubNub
const char *subkey = „”;
```

Uwaga: NodeMCU może łączyć się tylko z Wi-Fi, do którego można uzyskać dostęp bezpośrednio, bez np. strony do logowania się.

Podsumowanie

Kompletny robot pozwala poruszać się po całym terenie, znajdującym się w zasięgu sieci Wi-Fi i manipulować znajdującymi się tam przedmiotami. Konstrukcja jest cały czas rozwijana przez jej autora, ale obecnie ma pewne ograniczenia. Ruchy ramienia, z uwagi na opóźnienie, nie są tak płynne, jak można by wymagać – należy powoli ruszać ręką, sterującą ramieniem, by uzyskać płynny ruch. Podobnie ma się sprawa z sterowaniem ruchami platformy – zalecana jest ostrożność, tym bardziej, że robot nie posiada żadnych sensorów, które mogłyby zapobiegać zderzeniom z przeszkodami.

Robot nie został wyposażony w żaden kanał wideo – nie jest możliwe transmitowanie obrazu do i z robota, jednakże w aplikacji, do jakiej pomyślana jest konstrukcja – pomaganie osobom chorym i starszym nie jest to niezbędne. Można oczywiście do urządzenia dodać kamerę, która transmitować będzie, również przez Wi-Fi, obraz z punktu widzenia robota.

Nikodem Czechowski, EP

Źródło: *https://bit.ly/2WvcDrN*