Komunikacja radiowa Arduino - Raspberry Pi z wykorzystaniem modułu Bluetooh HC-06

Uruchomienie komunikacji radiowej pomiędzy Arduino a komputerem Rapsberry Pi, z wykorzystaniem modułu łącza radiowego Bluetooth HC06, wymaga wykonania następujących kroków:

1. Podłączyć modułu HC-06 do Arduno.
    a. W przypadku pracy z płytką edukacyjną etap ten można pominąć. Moduł Bluetooth jest już podłączony.
    b. W przypadku własnego montażu należy:
        i. Pin VCC modułu HC06 podłączyć do pinu 3.3V w Arduino Due
        ii. Pin GND modułu HC06 podłączyć do pinu GND w Arduino Due
        iii. Pin RXD modułu HC06 podłączyć do pinu 19 (RX1) w Arduino Due
        iv. Pin TXD modułu HC06 podłączyć do pinu 18 (TX1) w Arduino Due

        Od tego momentu moduł HC06 jest podłączony do portu szeregowego nr **1** w Arduino Due, a komunikacja z nim odbywać się będzie za pomocą obiektu `Serial1`.
2. Ustawić nazwę modułu HC-06 na unikalną w danej okolicy. W przypadku zajęć proponowana nazwa to *stanXX*, gdzie *XX* to numer stanowiska.
3. Sprawdzić, czy Raspberry Pi widzi moduł po zmianie nazwy.
4. Sparować Raspberry PI z modułem HC-06 i nawiązać połączenie.
5. Przetestować, czy Raspberry Pi może odbierać dane od Arduino. W tym celu potrzeba prostego programu, pracującego w module Arduino i wysyłającego teksty do modułu HC-06.
6. **GOTOWE**. Teraz można już swobodnie przesyłać dane po łączu radiowym.

Kroki te opisano szczegółowo poniżej:

Ustawienie nazwy przyjaznej (identyfikatora modułu Bluetooth HC-06)

Nazwa modułu jest nazwą “przyjazną”, wyświetlaną przez wszelkiego rodzaju urządzenia do komunikacji protokołem Bluetooth. Alternatywą dla nazwy przyjaznej jest numer MAC (np. F0:D2:2E:59:92:F3). Nazwa przyjazna musi być unikalna, np. **stanXX**, gdzie **XX** to numer stanowiska.
Aby ustawić nazwę modułowi HC-06, można skorzystać z następującego programu:

```
#include <ISADefinitions.h>

void setup() {
  pinMode(LED8, OUTPUT);
  Serial1.begin(9600);
  Serial.begin(9600);

  Serial.flush();
  Serial1.flush();
```

```
  Serial1.print("AT+NAMEstan01");
  delay(1000);
  String str = Serial1.readString();
  Serial.print("Odpowiedz: [");
  Serial.print(str);
  Serial.println("]");
}

void loop() {
}
```

Uwaga 1: Nazwa modułu HC-06 zapisywana jest w pamięci nieulotnej Flash. Pamięć ta ma określoną liczbę cykli zapisu, stąd nie wolno zapisywać nazwy w funkcji `loop()`. Uszkodzenie pamięci, odpowiedzialnej za nazwę urządzenia, uniemożliwia jej późniejszą zmianę.
Uwaga 2: Nazwę modułu HC-06 można zmieniać jedynie w dwóch trybach: a) **niesparowany** oraz b) **sparowany/niepołączony**. Jeśli polecenie AT+NAME zostanie wysłane do modułu Bluetooth w trybie **połączony**, zostanie ono potraktowane jako zwykłe dane i przekazana do odbiornika (np. komputera, telefonu komórkowego czy Raspberry Pi).
Uwaga 3: Moduł HC-06, po załączeniu zasilania, jest zawsze w trybie **niesparowany**. Parowanie wykonywane jest manualnie, przez użytkownika urządzenia zdalnego (tutaj Raspberry PI). Wyjątkiem jest sytuacja, gdy urządzenie zdalne jest skonfigurowane do automatycznego parowania z wykrytym modułem HC-06 (lub dowolnym innym).
Uwaga 4: Moduł HC-06 podłączono do portu szeregowego **Serial1**, pod piny: **RX1** (pin 19) oraz **TX1** (pin 18).

Sprawdzenie obecności urządzenia HC-06 po zmianie nazwy

Aby sprawdzić, jakie są dostępne urządzenia Bluetooth w okolicy, należy skorzystać z odpowiednich poleceń systemu Linux. W tym celu należy skorzystać z konsoli - wcisnąć kombinację **Lewy Ctrl+Lewy Alt+T** lub wybrać ikonę Terminal z paska narzędzi systemowych.

Wejście w tryb superużytkownika

W tym celu należy przełączyć się na konto roota:

```
pi@localhost:~ $ sudo su
root@localhost:/home/pi#
```

Sprawdzenie dostępnych urządzeń Bluetooth

A następnie wydać polecenie **hcitool scan**, które wyszukuje urządzenia w okolicy.

```
root@localhost:/home/pi# hcitool scan
Scanning ...
     AC:B5:7D:AB:66:7E   LAPEK
```

```
20:16:12:15:63:50    stan01
```

W uzyskanym wyniku interesuje nas wiersz podświetlony na niebiesko. Pierwsza wartość to adres MAC (nazwa sprzętowa) urządzenia Bluetooth o nazwie przyjaznej **stan01**. Adres MAC **20:16:12:15:63:50** jest unikalny dla każdego modułu i należy go wykorzystywać przy dalszej konfiguracji. Dobrze jest zamontować go na kartce lub w pliku tekstowym.

Podłączenie

Po pomyślnej zmianie nazwy i odnalezieniu swojego urządzenia, należy sparować go w systemie Raspbian. Można to zrobić za pomocą

a) narzędzia dostępnego w pod ikonką w pasku narzędzi systemowych, lub
b) za pomocą następujących poleceń:

Wyłączenie trybu samolotowego (brak komunikacji RF):

W celu aktywacji komunikacji radiowej należy skorzystać z polecenia **rfkill**:

```
root@localhost:/home/pi# rfkill list
0: phy0: Wireless LAN
     Soft blocked: no
     Hard blocked: no
1: hci0: Bluetooth
     Soft blocked: yes
     Hard blocked: no
root@localhost:/home/pi# rfkill unblock 1
root@localhost:/home/pi# rfkill list
0: phy0: Wireless LAN
     Soft blocked: no
     Hard blocked: no
1: hci0: Bluetooth
     Soft blocked: no
     Hard blocked: no
```

W tym przykładzie wyświetlono listę urządzeń, w której moduł Bluetooth z Raspberry Pi ma identyfikator **1**. Po ustaleniu identyfikatora użyto polecenia **unblock**.

Parowanie modułu HC-06 i Raspberry Pi

Aby sparować dowolne urządzenie z Raspberry Pi, należy skorzystać z następujących poleceń narzędzia **bluetoothctl**:

```
root@localhost:/home/pi# bluetoothctl -a
[NEW] Controller B8:27:EB:A7:0C:73 localhost [default]
Agent registered
[bluetooth]# power on
Changing power on succeeded
[bluetooth]# scan on
Discovery started
[CHG] Controller B8:27:EB:A7:0C:73 Discovering: yes
[NEW] Device 20:16:12:15:63:50 nazwa
[NEW] Device 2C:D0:5A:50:DC:8D DESKTOP-4GF1VDR
```

```
[NEW] Device AC:B5:7D:AB:66:7E LAPEK
[bluetooth]# trust 20:16:12:15:63:50
[CHG] Device 20:16:12:15:63:50 Trusted: yes
Changing 20:16:12:15:63:50 trust succeeded
[CHG] Device 2C:D0:5A:50:DC:8D RSSI: -77
[CHG] Device 2C:D0:5A:50:DC:8D RSSI: -65
[bluetooth]# pair 20:16:12:15:63:50
Attempting to pair with 20:16:12:15:63:50
[CHG] Device 20:16:12:15:63:50 Connected: yes
Request PIN code
[agent] Enter PIN code: 1234
[CHG] Device 20:16:12:15:63:50 UUIDs:
      00001101-0000-1000-8000-00805f9b34fb
[CHG] Device 20:16:12:15:63:50 Paired: yes
Pairing successful
[CHG] Device 20:16:12:15:63:50 Connected: no
[bluetooth]# quit
Agent unregistered
[DEL] Controller B8:27:EB:A7:0C:73 localhost [default]
```

Po wykonaniu polecenia **pair** w narzędziu **bluetoothctl** system operacyjny wyświetli monit o treści: **Device 'xxxx' has requested a pairing. Do you accept the request?** Należy odpowiedzieć **OK**. Możliwy jest również następujący komunikat: **GDBus.Error:org.bluez.Error.NotAvailable: Operation currently not available. Try connect manually.** Należy go zignorować (wcisnąć **OK**).

Po wyjściu z narzędzia **bluetoothctl** komputer Raspberry Pi oraz moduł HC-06 o numerze MAC `20:16:12:15:63:50` są sparowane.

Nawiązywanie połączenia pomiędzy sparowanymi urządzeniami

W tej chwili moduł HC-06 znajduje się w trybie **sparowany/niepołączony**. Aby nawiązać połączenie, należy skorzystać z następującego polecenia:

```
root@localhost:/home/pi# rfcomm connect hci0 20:16:12:15:63:50
Connected /dev/rfcomm0 to 20:16:12:15:63:50 on channel 1
Press CTRL-C for hangup
```

Po wydaniu polecenia **rfcomm** w systemie zainstalowane zostanie urządzenie **`/dev/rfcomm0`**, które komunikuje się bezpośrednio z modułem Bluetooth HC-06. **Uwaga!** Wciśnięcie CTRL+C spowoduje zatrzymanie i wyłączenie urządzenia. Jeśli użytkownik ma potrzebę wydawania kolejnych poleceń w terminalu, należy utworzyć nowy terminal.

Nawiązywanie połączenia pomiędzy sparowanymi urządzeniami

Wydanie następującego polecenia:

```
pi@localhost:~ $ cat /dev/rfcomm0
ta
Ala ma kota
Ala ma ko^C
```

spowoduje wyświetlenie w terminalu danych "Ala ma kota\n", otrzymywanych od Arduino. Program wysyłający te dane może mieć następującą postać:

```
#include <ISADefinitions.h>

void setup() {
  pinMode(LED8, OUTPUT);
  Serial1.begin(9600);
}

bool state = false;
const char* tekst = "Ala ma kota\n";
int counter = 0;

void loop() {
  char ch = tekst[counter];
  counter = (counter + 1) % strlen(tekst);

  digitalWrite(LED8, state=!state);
  Serial1.print(ch);
  delay(150);
}
```

Komunikacja - programowanie Raspberry Pi

Nowy projekt w środowisku Qt:
QtCreator -> New Project -> Non-Qt Project -> Pain C/C++ Project (CMake Build).

Uruchamianie w oddzielnym terminalu, ze względu na niepoprawną obsługę strumieni WE/WY w oknie **Qt: Project->Run settings, zaznaczyć "Run in Terminal"**

Kod odbierający dane, przesyłane przez Arduino do Raspberry
Dane są liniami tekstu, zakończonego znakiem \n.

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
   FILE* f = fopen("/dev/rfcomm0", "rw");
   if (f == NULL) {
       perror("fopen");
       exit(1);
   }

   // odbierz linię tekstu; linia MUSI kończyć się symbolem \n

   int cnt = 0;
```

```
    while(1) {
        char line[100];
        fgets(line, sizeof(line), f);

        printf("line %d=%s", cnt++, line);
        fflush(stdout); // zapewnij wyświetlanie wyniku
    }
    fclose(f);
    return 0;
}
```

Kod Arduino odbierający dane od Raspberry PI

Dane są liniami tekstu, zakończonego znakiem \n. W ramach tekstu podawane są polecenia **SET x** oraz **RESET x**, gdzie x jest numerem diody LED zakresu 0-7. Polecenie **SET** włącza diodę, polecenie **RESET** wyłącza.

```
#include <ISADefinitions.h>

void setup() {
  for (int i = 0; i < 8; i++)
    pinMode(LEDS[i], OUTPUT);
  Serial1.begin(9600);
  Serial.begin(9600);
  Serial.println("start");
}

void loop() {
  String command = Serial1.readStringUntil('\n');

  command.trim();
  command.toLowerCase();
  Serial.print(".");

  if (command == "")
    return;

  if (command.startsWith("set")) {
    command.remove(0, 3);
    command.trim();
    int id = command.toInt();
    if (id >= 0 && id <= 7)
      digitalWrite(LEDS[id], true);
  }
  if (command.startsWith("reset")) {
    command.remove(0, 5);
    command.trim();
    int id = command.toInt();
    if (id >= 0 && id <= 7)
```

```
      digitalWrite(LEDS[id], false);
  }
}
```

Kod wysyłający polecenia SET/RESET do Arduino

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
   FILE* f = fopen("/dev/rfcomm0", "w");
   if (f == NULL) {
       perror("fopen");
       exit(1);
   }

   while(1)
   {
       printf("Wpisz polecenie i naciśnij ENTER:\n");
       fflush(stdout);

       char command[100];
       //fgets(stdin, sizeof(command), f);
       gets(command);
       printf("> %s (%d)\n", command, strlen(command));

       int a = fprintf(f, "%s\n", command);
       printf("fprintf: wyslano %d bajtow\n", a);

       if (a==-1) {
           perror("printf");
           exit(1);
       }
   }
   fclose(f);
   return 0;
}
```

Kod realizujący dwukierunkową transmisję Raspberry Pi - Arduino

Uwaga! Operacja `read` jest blokująca - w buforze odbiorczym musi być co najmniej jeden bajt, aby funkcja ta mogła się zakończyć. Proszę skorzystać z funkcji `select`.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
int main(void)
```

```
{
   int fd = open("/dev/rfcomm0", O_RDWR);
   printf("fd=%d\n", fd);

   char buffer[100];
   write(fd, "123\n", 4);

   int b = read(fd, buffer, 100);
   printf("b=%d\n", b);

   close(fd);

  return 0;
}
```

Polecenia:
apt-get remove --purge qtcreator
apt-get install qt5-default
apt-get install qtcreator