Podstawy Systemów Wbudowanych

Wykład 5: Technologie komunikacji bezprzewodowej

Angelika Tefelska Dariusz Tefelski

Zakład Fizyki Jądrowej, Wydział Fizyki PW

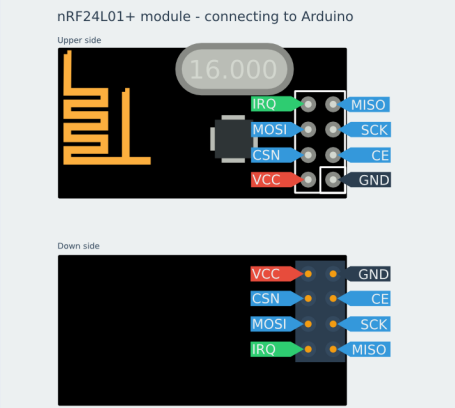
7 kwietnia 2021

O czym będzie wykład...

1. Komunikacja drogą radiową
2. Bluetooth
3. WiFi
4. Sterownie za pomocą podczerwieni
5. RFID
6. Ethernet

Komunikacja drogą radiową

- Komunikacją drogą radiową odbywa się przy wykorzystaniu anten, które emitują i odbierają fale elektromagnetyczne odpowiednio wzmacniane. W zależności od tego wzmocnienia, transmisja taka może mieć różny zasięg (od kilku cm w przypadku kart zbliżeniowych do setek tysięcy kilometrów w przypadku sond kosmicznych).
- Moduły do komunikacji radiowej są tanie i łatwe w obsłudze. Natomiast trzeba pamiętać, że większość z nich wymaga zasilania 3.3V.



Biblioteki do obsługi modułów radiowych

Do obsługi modułów radiowych można skorzystać z bibliotek:

- **RF24** - biblioteka głównie do obsługi modłów NRF24L01 zarówno pod Arduino jak i Raspberry Pi. Jedna z najbardziej popularnych i najprostszych w obsłudze bibliotek.
- MySensors - biblioteka, która służy do integracji automatyki domowej, może być wykorzystana do tworzenia sieci radiowej pomiędzy czujnikami. Korzysta z modyfikowanej wersji biblioteki RF24.
- RadioHead - bardzo rozbudowana biblioteka, która wspiera wiele różnych modułów radiowych.
- Mirf - biblioteka głównie do modułu NRF24L01, natomiast bardziej skomplikowana w użyciu niż RF24.

Wszystkie biblioteki są do pobrania ze strony: http://playground.arduino.cc.

Biblioteka RF24

Cała dokumentacja do biblioteki RF24 znajduje się na stronie: http://tmrh20.github.io/. Najważniejsze funkcje biblioteki:

- Utworzenie obiektu klasy RF24:

  RF24 radio(pin podłączony do CE, pin podłączony do CSN);

- Funkcja do inicjalizacji obiektu:

  radio.begin();

- Funkcja, która rozpoczyna nasłuchiwanie:

  radio.startListening();

- Funkcja, która kończy nasłuchiwanie oraz przełącza moduł na tryb transmisji:

  radio.stopListening();

- Funkcja, która sprawdza czy są bajty do odczytu:

  radio.available();

- Funkcja do odczytu danych:

  radio.read(adres zmiennej do której dane mają zostać zapisane, długość danych do zapisania);

Biblioteka RF24

- Funkcja, która wysyła dane:

  radio.write(adres zmiennej która ma zostać wysłana, długość zmiennej);

- Funkcja, która otwiera łącze do wysyłania danych:

  radio.openWritingPipe(byte array z adresem 5-cio bitowym);

- Funkcja, która otwiera łącze do odczytu danych (maksymalnie na raz może być otwartych 6 łączy do odczytu):

  radio.openReadingPipe(numer łącza, byte array z adresem z którego chcemy odczytywać dane);

- Funkcja, która zwraca blok danych przydatnych do debugging:

  radio.printDetails();

- Funkcja ustawiająca ile razy ma moduł podjąć próbę nawiązania komunikacji zanim zwróci informacje o błędzie:

  radio.setRetries;(czas pomiędzy powtórzeniami, liczba powtórzeń);

- Funkcja, która ustawia moc wyjściową nadajnika:

  radio.setPALevel(RF24_PA_MIN / RF24_PA_LOW / RF24_PA_HIGH / RF24_PA_MAX);

  gdzie: RF24_PA_MIN= -18dBm, RF24_PA_LOW= -12dBm, RF24_PA_HIGH = -6dBm, RF24_PA_MAX = 0 dBm.

Przykład - nadajnik

Przykład: Wysyłanie liczby przy użyciu modułu radiowego.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <printf.h>

RF24 radio(48,53);
const byte rxAddress[6]="00001";
int i=10;
void setup()
{
  printf_begin();
  radio.begin();
  radio.setPALevel(RF24_PA_MAX); //Moc wyjsciowa nadajnika =0dBm
  radio.setRetries(15,15);
  radio.openWritingPipe(rxAddress);
  radio.stopListening();
}
void loop()
{
  radio.write(&i, sizeof(int));
  delay(1000);
}
```

Przykład - odbiornik

Przykład: Odbieranie liczby przy użyciu modułu radiowego.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <printf.h>

RF24 radio(48, 53);
const byte rxAddr[6] = "00001";
int i;

void setup()
{
  printf_begin();
  radio.begin();
  radio.setPALevel(RF24_PA_MIN);
  radio.openReadingPipe(1, rxAddr);
  radio.startListening();
}

void loop()
{
  if(radio.available()) radio.read(&i, sizeof(int));
}
```

Bluetooth

- Bluetooth jest technologią bezprzewodowej komunikacji krótkiego zasięgu. Generalnie jest wykorzystywana do komunikacji między urządzeniami elektronicznymi (słuchawki, myszki, klawiatura, telefon komórkowy itd).
- Nazwa Bluetooth pochodzi od przydomka króla duńskiego Haralda Sinozębego (Blåtand), który przyczynił się do zjednoczenia rywalizujących plemion z Danii i Norwegii. Przez analogię do historii technologia bluetooth ma zjednoczyć różne urządzenia. Sam znak pochodzi od znaków runicznych (haglaz, berkanan), które są odpowiednikami liter H i B.

- Urządzenia korzystające z bluetooth są widoczne jako wirtualne porty szeregowe w systemie operacyjnym. Oprogramowanie urządzeń przy wykorzystaniu arduino odbywa się za pomocą funkcji dla standardowego portu szeregowego.

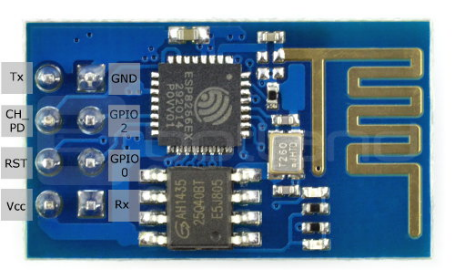
Bluetooth - przykład

Przykład: Odczyt danych korzystając z modułu bluetooth. Gdy odczytana wartość to 1, zapalenie diody LED.

```
#define LED 2

char data = 0;

void setup()
{
    Serial.begin(9600);
    pinMode(LED, OUTPUT);
}

void loop()
{
   if(Serial.available() > 0)
   {
      data = Serial.read();

      if(data == '1') digitalWrite(LED, HIGH);
      else digitalWrite(LED, LOW);
   }
}
```

Moduł ESP8266

- Moduł ESP8266 jest tanim modułem działającym w standardzie WiFi 802.11 na częstotliwości 2.4GHz. Posiada pamięć Flash 512kB oraz antenę PCB.



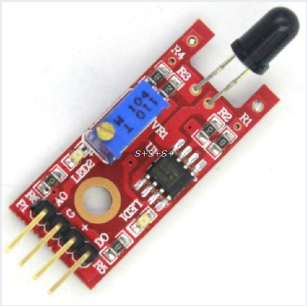
- Do obsługi modułu ESP8266można skorzystać z biblioteki ESP8266wifi, która jest dostępna do pobrania ze strony https://github.com/ekstrand/ESP8266wifi

Biblioteka ESP8266wifi - przykład: odbieranie i wysyłanie danych przez WiFi

```
#define esp8266 Serial1
#define CH_PD 2
#define speed8266 115200

void setup()
{
  esp8266.begin(speed8266);
  Serial.begin(115200);
  reset8266(); // Pin CH_PD musi byc zresetowany przed
     ↪ rozpoczeciem komunikacji
}
void reset8266 ()
{
    pinMode(CH_PD, OUTPUT);
    digitalWrite(CH_PD, LOW);
    delay(300);
    digitalWrite(CH_PD, HIGH);
}
void loop()
{
  while(esp8266.available()) Serial.write(esp8266.read());
  while(Serial.available()) esp8266.write(Serial.read());
}
```

Sterowanie za pomocą podczerwieni

- W projekcie można zastosować sterowanie za pomocą podczerwieni z wykorzystaniem np. pilota.
- Piloty zawierają diodę LED emitującą światło w paśmie podczerwieni. Naciśnięcie przycisku spowoduje wielokrotne włączanie i wyłączanie diody LED według unikatowego wzorca zależnego od wybranego przycisku.
- Pod arduino jest dostępna biblioteka IRremote, która służy do obsługi komunikacji przez podczerwień. Biblioteka jest dostępna na github i można ją pobrać korzystając z adresu: https://github.com/z3t0/Arduino-IRremote.



Sterowanie za pomocą podczerwieni - przykład

Przykład: Zapalanie/gaszenie jednej z 3 diod w zależności od wybranego przycisku na pilocie.

```
#include <IRremote.h>
#define DATA 11 //pin D(Linia danych) odbiornika podczerwieni
#define code1  0xFF807F //Kod przycisku 1
#define code2  0xFFA05F //Kod przycisku 2
#define code3  0xFF906F //Kod przycisku 3
#define LED1 2
#define LED2 3
#define LED3 4

IRrecv irrecv(DATA);
decode_results results;
unsigned int value;
int stan1=LOW, stan2=LOW, stan3=LOW;

void setup()
{
    irrecv.enableIRIn(); //Wlaczenie odbiornika podczerwieni
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
}
```

Sterowanie za pomocą podczerwieni - przykład

```
void loop()
{
    if(irrecv.decode(&results)) //Sprawdzenie czy otrzymano sygnal
    {
        value = results.value;
        switch(value)
        {
            case code1:
                stan1=!stan1;
                digitalWrite(LED1, stan1);
                break;
            case code2:
                stan2=!stan2;
                digitalWrite(LED2, stan2);
                break;
            case code3:
                stan3=!stan3;
                digitalWrite(LED3, stan3);
                break;
        }
    }
    irrecv.resume();
}
```

Etykiety RFID

- RFID (Radio-frequency identification) to technika, która wykorzystuje fale radiowe do przesyłania danych na niewielkie odległości. Każde urządzenie posiada unikatową etykietę RFID składającą się z 14 liczb.
- Moduły RFID znalazły powszechne zastosowania wszędzie tam gdzie niezbędna jest szybka identyfikacja np. przy otwieraniu drzwi, karty zbliżeniowe.
- Odczyt etykiet RFID jest bardzo prosty gdyż moduły te są podłączone bezpośrednio do portu szeregowego.

Etykiety RFID - przykład

Przykład: Porównanie dwóch etykiet: wzorcową i odczytaną z breloczka RFID.

```
int wzor[]={1,2,3,4,5,6,7,8,9,10,11,12,13,14};
int odczyt[14];
int j=0;

void setup()
{
    Serial.flush(); //Czyszczenie bufora
    Serial.begin(9600);
}

void compareRFID(int a[14], int b[14])
{
    bool thesame=true;
    for(unsigned int i=0; i<14; i++)
    {
        if(a[i]!=b[i]) thesame=false;
    }

    return thesame;
}
```

Etykiety RFID - przykład

```
void loop()
{
    while(Serial.available())
    {
        odczyt[j]=Serial.read();
        j++;
    }
    j=0;
    Serial.flush();

    if(compareRFID(wzor,odczyt)) Serial.println("Karta
       ↪ zaakceptowana");
}
```

Ethernet

- Technologia Ethernet - standard budowy sieci komputerowych, umożliwiający urządzeniom wszelkiego rodzaju komunikację między sobą poprzez wysyłanie i odbieranie strumieni danych (pakietów).
- Najczęściej stosuje się Arduino Ethernet Shield, który ułatwia połączenie się przez Ethernet.

Biblioteka Ethernet

W środowisku IDE instalowana jest domyślnie biblioteka Ethernet. Biblioteka ta zawiera kilka klas: Ethernet, Server, Client oraz klasy do ogólnego przeznaczenia, obsługujące protokół UDP (User Datagram Protocol) do rozgłaszania informacji w sieci. Przegląd najważniejszych funkcji:

- Funkcja, do inicjalizacji. Jeśli stosuje się shield to adres IP jest konfigurowany automatycznie. Natomiast można też ręcznie podać adres IP, bramkę (najczęściej adres IP routera) oraz maskę podsieci (domyślnie: 255.255.255.0):
  - **Ethernet.begin(mac);**
  - **Ethernet.begin(mac, ip);**
  - **Ethernet.begin(mac, ip, brama);**
  - **Ethernet.begin(mac, ip, brama, maska);**
- Funkcja do utworzenia serwera nasłuchującego na określonym porcie:

  Server(port);
- Funkcja do uruchomienia serwera:

  Server.begin();
- Funkcja, która zwraca obiekt klienta gdy są dane do odebrania:

  Server.available();

Biblioteka Ethernet

- Funkcja, do wysyłania danych do wszystkich podłączonych klientów:
  - **Server.write(byte lub char);**
  - **Server.write(tablica typu byte, długość tablicy);**
  - **Server.print(char, byte, int, long, lub string);**
  - **Server.print(char, byte, int, long, lub string, BIN/DEC/HEX);**
  - **Server.println(char, byte, int, long, lub string);**
  - **Server.println(char, byte, int, long, lub string, BIN/DEC/HEX);**
- Funkcja, która tworzy obiekt klienta, który może łączyć się z określonym adresem IP i portem:

  Client(ip, port);
- Funkcja, która zwraca informacje, czy klient jest podłączony z serwerem:

  Client.connected();
- Funkcja, która nawiązuje połączenie:

  Client.connect();
- Funkcja, która wysyła dane do serwera:
  - **Client.write(byte lub char);**
  - **Client.write(tablica typu byte, długość tablicy);**
  - **Client.print(char, byte, int, long, lub string);**
  - **Client.print(char, byte, int, long, lub string, BIN/DEC/HEX);**
  - **Client.println(char, byte, int, long, lub string);**
  - **Client.println(char, byte, int, long, lub string, BIN/DEC/HEX);**

Biblioteka Ethernet

- Funkcja, która zwraca liczbę bajtów gotowych do odczytu:

  Client.available();

- Funkcja do odczytu bajtu danych z serwera:

  Client.read();

- Funkcja, która czyści bufor danych do odczytu:

  Client.flush();

- Funkcja, która zamyka połączenie z serwerem:

  Client.stop();

Biblioteka Ethernet - przykład utworzenia serwera na Arduino

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[]={0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; //Adres mac jest
    ↪ nadrukowany na Shieldzie

IPAddress manualIP(192.168.1.120); //Gdy DHCP nie dziala wtedy
    ↪ trzeba recznie adres IP wpisac.

EthernetServer server(80); //Inicjalizacja serwera na porcie 80

boolean dhcpConnected = false;

void setup()
{
    if(!Ethernet.begin(mac))
    {
        Ethernet.begin(mac, manualIP);
    }
    server.begin();
}
```

Biblioteka Ethernet - przykład utworzenia serwera na Arduino

```
void loop()
{
    EthernetClient client=server.available(); //Oczekiwanie na
        ↪ klientow

    if(client)
    {
        while(client.connected())
        {
            if(client.available())
            {
                char c = client.read();
                if(c =='\n') client.println("Odebralem od Ciebie
                    ↪ dane!");
            }
        }

        delay(1); //Opoznienie aby client odebral komunikat
        client.stop();
    }
}
```

Biblioteka Ethernet - przykład utworzenia klienta na Arduino

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress server(74,125,232,128);  //Gdy nie ma DNS, numer IP
    ↪ google
char server[] = "www.google.com";

IPAddress ip(192, 168, 0, 177);
EthernetClient client;

void setup()
{
  Serial.begin(9600);
  if(!Ethernet.begin(mac)) Ethernet.begin(mac, ip);
  delay(1000); //Czas na inicjalizacje

  client.connect(server, 80);
}
```

Biblioteka Ethernet - przykład utworzenia klienta na Arduino

```
void loop()
{
  if (client.available()) //Sprawdzenie czy sa dane do odczytu z
      ↪ serwera
  {
    char c = client.read();
    Serial.print(c);
  }

  if (!client.connected())
  {
    Serial.println("Rozlaczono");
    client.stop();

    while (true);
  }
}
```

Biblioteka EthernetUDP

Do komunikacji przez ethernet można skorzystać z biblioteki EthernetUDP (bez wykorzystania serwera i klienta). Najważniejsze funkcje klasy EthernetUDP:

- Funkcja do inicjalizacji obiektu UDP oraz określenia numeru portu do nasłuchiwania:

  EthernetUDP.begin(port);

- Funkcja do odczytu pakietów UDP z bufora:

  EthernetUDP.read(bufor typu char, maksymalny rozmiar buforu);

- Funkcja do wysyłania komunikatu do innego urządzenia:
  - **EthernetUDP.write(komunikat typu char);**
  - **EthernetUDP.write(tablica typu byte lub char, rozmiar tablicy);**
- Funkcja, która określa adres IP i port urządzenia docelowego:

  EthernetUDP.beginPacket(ip, port);

- Funkcja, która kończy komunikat:

  EthernetUDP.endPacket();

- Funkcja, która sprawdza czy są pakiety UDP do odczytania. Zwraca rozmiar pakietów:

  EthernetUDP.parsePacket();

- Funkcja, która zwraca ilość danych odebranych i gotowych do odczytu:

  EthernetUDP.available();

EthernetUDP - przykład

```
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 1, 177);

unsigned int localPort = 8888;      // Lokalny port do nasluchu

char packetBuffer[UDP_TX_PACKET_MAX_SIZE];  //bufor do
    ↪ przechowywania przychodzacych informacji
char  ReplyBuffer[] = "acknowledged";       //String do odeslania

EthernetUDP Udp;

void setup()
{
  Ethernet.begin(mac, ip);
  Udp.begin(localPort);
  Serial.begin(9600);
}
```

EthernetUDP - przykład

```
void loop() {
  int packetSize = Udp.parsePacket(); //Sprawdzenie czy sa dane do odczytu
  if(packetSize)
  {
    Serial.print("Dane z: ");
    IPAddress remote = Udp.remoteIP();
    for (int i = 0; i < 4; i++)
    {
      Serial.print(remote[i], DEC);
      if (i < 3) {
        Serial.print(".");
      }
    }
    Serial.print(", port ");
    Serial.println(Udp.remotePort());

    Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
    Serial.println("Odczytano:");
    Serial.println(packetBuffer);

    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    Udp.write(ReplyBuffer);
    Udp.endPacket();
  }
  delay(10);
}
```

THAT'S ALL FOR TODAY....
ANY QUESTION??

Back-up

Biblioteka ESP8266wifi - przykład 2

```
#include <ESP8266wifi.h>
#define esp8266_reset_pin 2 // Pin CH_PD modulu esp8266

ESP8266wifi wifi(Serial1, Serial1, esp8266_reset_pin, Serial);

String inputString;
boolean stringComplete = false;
unsigned long nextPing = 0;

void setup()
{
  inputString.reserve(32);
  Serial1.begin(115200);
  Serial.begin(115200);
  Serial.println("Starting wifi");

  wifi.setTransportToTCP(); // wifi.setTransportToUDP();
  wifi.endSendWithNewline(true);
  wifi.begin();
  wifi.connectToAP("nazwa wifi","haslo"); //Access point
  wifi.connectToServer("111.111.11.11", "2222");
  wifi.send(SERVER, "ESP8266 test app started");
}
```

Biblioteka ESP8266wifi - przykład 2

```
void loop()
{
  if (!wifi.isStarted()) wifi.begin();

  static char buf[20];
  if (stringComplete)
  {
    inputString.toCharArray(buf, sizeof buf);
    wifi.send(SERVER, buf);
    inputString = "";
    stringComplete = false;
  }

  if (millis() > nextPing) {
    wifi.send(SERVER, "Ping ping..");
    nextPing = millis() + 10000;
  }
```

Biblioteka ESP8266wifi - przykład 2

```
  WifiMessage in = wifi.listenForIncomingMessage(6000);
  if (in.hasData) {
    if (in.channel == SERVER)
      Serial.println("Message from the server:");
    else
      Serial.println("Message a local client:");
    Serial.println(in.message);
    wifi.send(in.channel, "Echo:", false); //Echo back;
    wifi.send(in.channel, in.message);
    nextPing = millis() + 10000;
  }
  // wifi.disconnectFromServer();
}
void serialEvent() //Odczyt danych z konsoli portu szeregowego
{
  while (Serial.available())
  {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n') stringComplete = true;
  }
}
```

Biblioteka ESP8266wifi - przykład 3

```
#include <ESP8266wifi.h>
#define esp8266_reset_pin 2
#define SERVER_PORT "2121"
#define SSID "nazwa wifi"
#define PASSWORD "haslo"

ESP8266wifi wifi(Serial1, Serial1, esp8266_reset_pin, Serial);
void processCommand(WifiMessage msg);
uint8_t wifi_started = false;
const char RST[] PROGMEM = "RST"; // TCP Commands
const char IDN[] PROGMEM = "*IDN?";
const char CLOSE[] PROGMEM = "CLOSE";

void setup()
{
  Serial.begin(115200);
  Serial1.begin(115200);
  wifi.setTransportToTCP();
  wifi.endSendWithNewline(false);
  wifi_started = wifi.begin();
  if(wifi_started)
  {
    wifi.connectToAP(SSID, PASSWORD);
    wifi.startLocalServer(SERVER_PORT);
  }
}
```

Biblioteka ESP8266wifi - przykład 3

```
void loop()
{
  delay(2);

  static WifiConnection *connections;
  if (wifi_started) wifi.checkConnections(&connections);

  for (int i = 0; i < MAX_CONNECTIONS; i++)
  {
    if (connections[i].connected==1)
    {
      WifiMessage msg = wifi.getIncomingMessage();
      if (msg.hasData) processCommand(msg);
    }
  }
}
```

Biblioteka ESP8266wifi - przykład 3

```
void processCommand(WifiMessage msg) {
  char espBuf[MSG_BUFFER_MAX];
  int set;
  char str[16];

  sscanf(msg.message,"%15s %d",str,&set);

  if ( !strcmp_P(str,IDN) ) wifi.send(msg.channel,"ESP8266wifi Example\n")
      ↪ ;
  else if( !strcmp_P(str,CLOSE) ) wifi.closeChannel(msg.channel);
  // Reset system by temp enable watchdog
  else if ( !strcmp_P(str,RST) )
  {
    wifi.send(msg.channel,"SYSTEM RESET...\n");
    wifi.stopLocalServer();
    asm volatile ("  jmp 0");
  }
  else wifi.send(msg.channel,"ERR\n");
}
```