Systemy Wbudowane - Projekt

Temat : System rozproszonego zbierania danych o temperaturze

Autorzy: Arkadiusz Dudzik i Piotr Zmilczak

Projekt składa się z dwóch części, centrali opartej o Raspberry Pi 3 i programu napisanego języku C++ oraz stacji nadającej opartej o Arduino Mini Pro i oprogramowania napisanego w języku Arduino.

Link do źródeł: https://github.com/marwin1991/TemperatureCentralSoft

Konwencja wysyłania i odczytywania wiadomości

Aby zawrzeć wszystkie potrzebne informacje w jednej wiadomości wykorzystaliśmy postać dziesiętną liczby 4 bajtowej (unsigned long) do zakodowania w niej wszystkich potrzebnych informacji.
Maksymalna wartość dla typu unsigned long to: 4.294.967.295

Wyjaśnienie kodowania wiadomości idąc od lewej:
Na żółto - typ wiadomości, na potrzeby projektu korzystamy z 3 typów:

- “1” - oznacza poprawną wiadomość z temperaturą
- “2” - błąd(np nie udało się odczytać temperatury)
- “4” - wiadomości specjalne :
  - 4200000001 - wiadomość “hello” od Arduino do centrali
  - 420000XXXX - wiadomość od centrali do Arduino z przydzielony ID, gdzie XXXX to ID

Na niebiesko - ID:
Każda wiadomość od Arduino do centrali zawiera ID czujnika przydzielone w procesie rejestracji, pozwoli to użytkownikowi rozpoznać z którego czujnika przyszła wiadomość, czyli w którym miejscu jest taka temperatura. ID zostają przydzielone z puli 1000-9999.

Na zielono - stan baterii:
Zakładamy, że czujniki pracują na własnym źródle zasilania. Stan baterii określamy od 0 do 9, gdzie 0 to pusta bateria, a 9 to bateria pełna.

Na czerwono(i fioletowo) - temperatura
Temperaturę kodujemy na 4 cyfrach, w tym dokładność dwóch cyfr po przecinku. W celu zakodowania liczby ujemnej rozpoznajemy ostatnią cyfrę (na fioletowo):

- cyfra parzysta - temperatura dodatnia
- cyfra nieparzysta - temperatura ujemna

Ma to bardzo mały wpływ na dokładność przesyłanej temperatury.

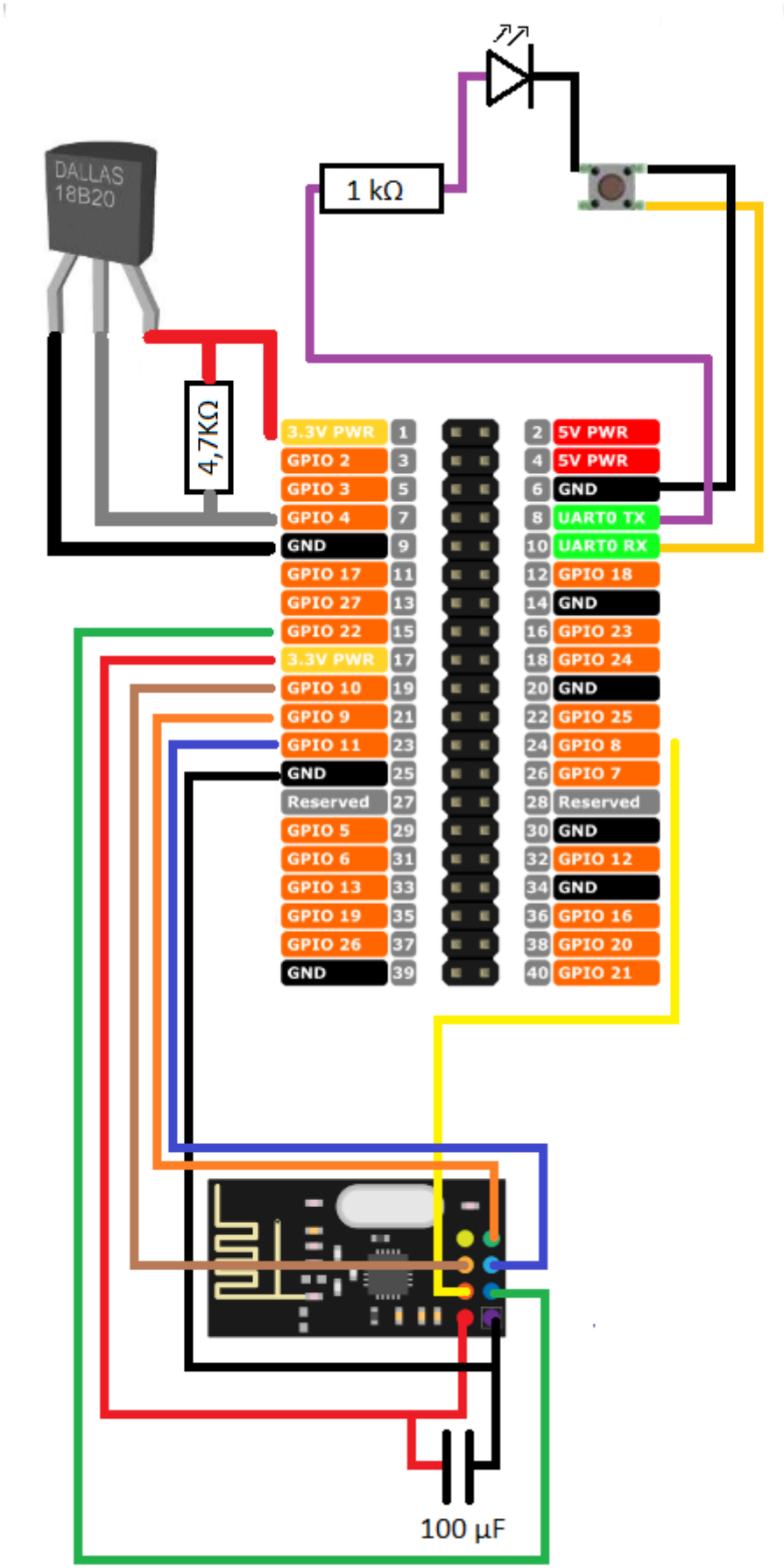
Centrala:

Komponenty:

- Raspberry Pi 3 model B
- Moduł nRF24l01+ - komponent odpowiedzialny za komunikacje radiową
- Kondensator 100 µF
- DS18B20 - czujnik odpowiedzialny za mierzenie temperatury
- Rezystor 4.7 k Ohm
- Przycisk
- Dioda LED

- Rezystor 1 k Ohm

Schemat:



Głównym zadaniem stacji centralnej zaimplementowanej na Raspberry Pi jest odczyt komunikatów z temperaturą za pomocą modułu nRF oraz wypisywanie tych wartości użytkownikowi.

Pomiar temperatury przy Raspberry Pi

Postanowiliśmy rozszerzyć funkcjonalności oferowane przez stację o możliwość pomiaru temperatury w miejscu, w którym się znajduje. Podobnie jak w przypadku Arduino wykorzystaliśmy czujnik temperatury DS18B20. Aby odczytać temperaturę należy zinterpretować odpowiednio wpis umieszczony przez urządzenie w pliku */sys/bus/w1/devices/28-0317233eb7ff/w1_slave.* Realizujemy to w osobnym wątku naszego oprogramowania.

Obsługa komunikacji przez NRF24L01+

Stacja odbiera od urządzeń typu Arduino komunikaty zawierające informacje o wartości temperatury oraz powitalną wiadomość, która jest prośbą o zarejestrowanie. Raspberry Pi jest nadawcą tylko wtedy gdy odpowiada na to “powitanie” - wysyłając wygenerowane ID.
Aby umożliwić poprawne funkcjonowanie tj. odbieranie i wysyłanie wiadomości musieliśmy umieścić kondensator o pojemności 100 μF. Zapewnia on stabilność zasilania oraz eliminuje zakłócenia napięcia, a więc działa jak filtr.
Układ NRF24L01+ komunikuje się z zarządzającym nim kontrolerem za pomocą szeregowego interfejsu SPI (piny MISO, MOSI, CLK). Transmisja ma charakter pół-dupleksu (ang. *half-duplex*) – na raz dane przesyła tylko jedno z urządzeń.

Dodatkowe komponenty

Stację centralną dopełnia przycisk, który sygnalizuje Raspberry Pi, że ma przejść w stan “oczekiwania na rejestrację”. W tym stanie, stacja rejestruje podłączające się urządzenia i nadaje im ID. Po upływie 30 sekund stacja opuszcza stan i żadne urządzenie nadawcze nie zostanie podłączone.

Dodatkowe zalecenia przy rozwijaniu projektu

Przenieść wypisywanie informacji o temperaturze do pliku typu JSON, a następnie stworzyć GUI, które umożliw użytkownikowi łatwiejszy przegląd danych o temperaturze.

Użytkowanie

Centrala jest wyposażona w oprogramowanie sterujące wszystkimi komponentami
Program startuje po włączeniu wpisaniu komendy do terminala:
sudo /home/pi/Desktop/TemeratureCentralSoft/PiSoft/piSoft
Następnie w konsoli wyświetlona informacja o inicjalizacji komponentów, a później zostaje wyświetlana o temperaturze.
Aby zarejestrować stacje nadającą, należy przytrzymać przycisk przez 5 sekund.

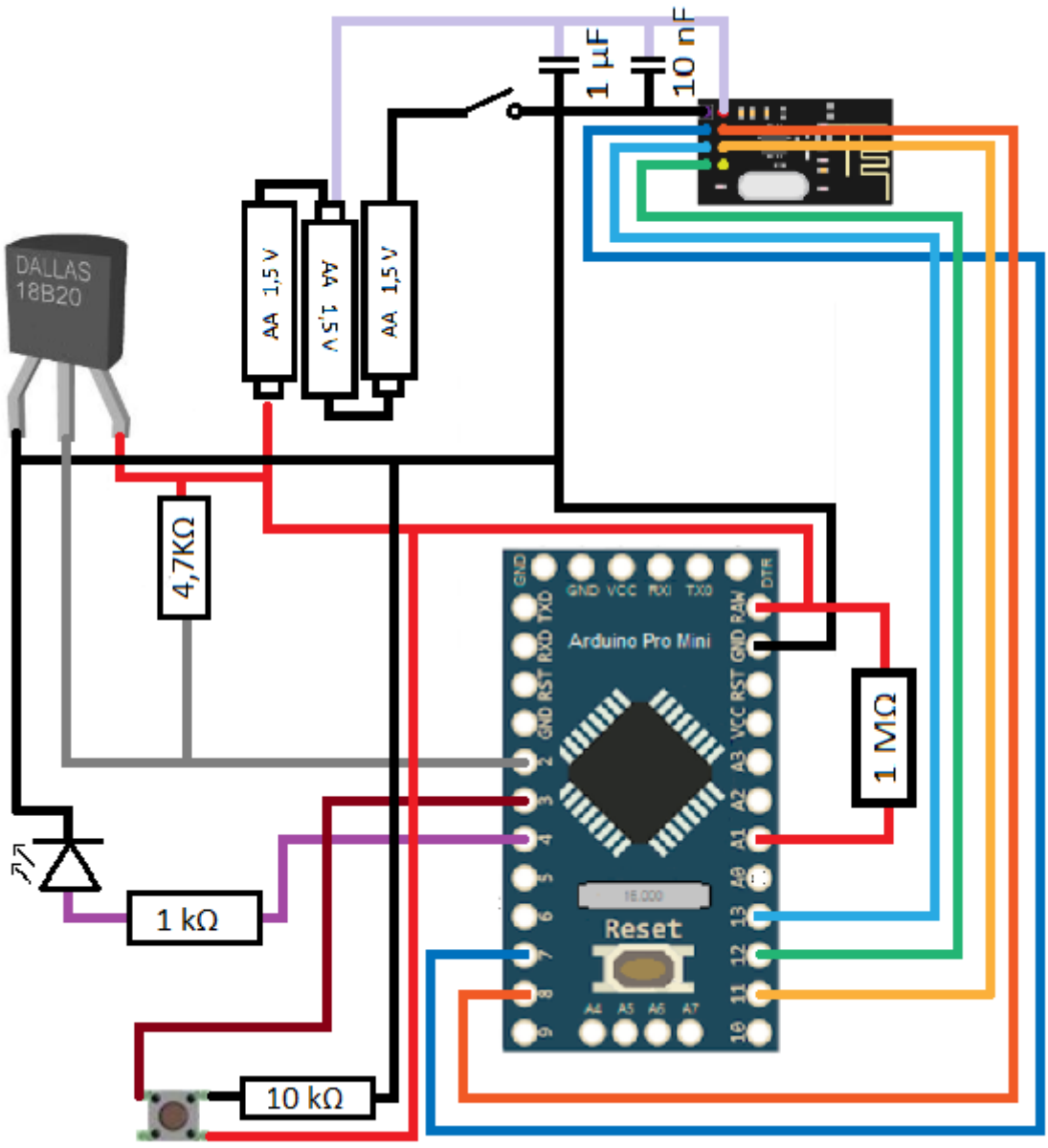
Stacja nadająca:

Komponenty:

- Arduino Mini Pro 5V 16 Mhz

- Moduł nRF24l01+ - komponent odpowiedzialny za komunikacje radiową
- Kondensator 1 µF
- Kondensator 10 nF
- DS18B20 - czujnik odpowiedzialny za mierzenie temperatury
- Przycisk
- Dioda LED
- Rezystor 10 k Ohm
- Rezystor 1 M Ohm
- Rezystor 1 k Ohm
- Rezystor 4,7 k Ohm
- 3 x Bateria AA
- (konwerter USB to TTL do programowania Arduino)

Schemat:



Arduino Mini Pro 5V/16Mhz - 3V/8Mhz

Na potrzeby projektu korzystaliśmy z wersji 5V/16Mhz, ponieważ nie posiadaliśmy wersji o obniżonym napięciu. W tym projekcie jeśli bardzo zależy nam rozmiarach stacji nadającej oraz bardzo niskim zużyciu energii, to należy właśnie skorzystać z Arduino Mini Pro 3V/8Mhz.

Wybór rodzaju zasilania do stacji nadającej

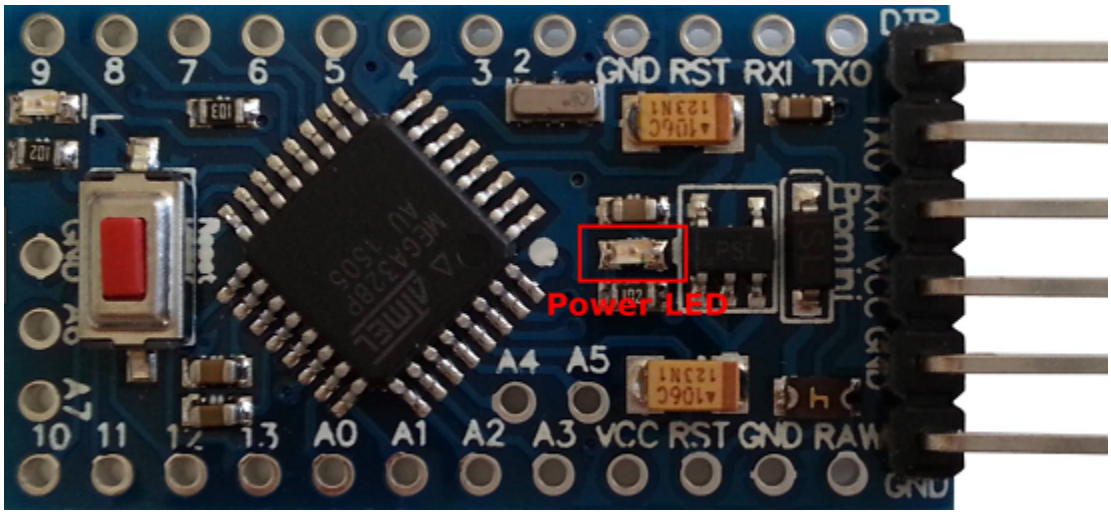
Nasz projekt zakłada, że stacja nadająca powinna być zaopatrzona we własne źródło zasilania. Oczywiście można użyć jej wraz z odpowiednim zasilaczem, który po podłączeniu do 230V zapewni odpowiednie napięcie. Takie rozwiązanie jest proste, nie wymaga dodatkowego rezystora 1 M Ohmów oraz kodu odpowiedzialnego za odczyt stanu baterii. Jednak ma też wady, ponieważ możemy użyć naszej stacji tylko tam gdzie mamy bezpośredni dostęp do gniazdka.

Rozważaliśmy kilka opcji zasilania, które przedstawiamy poniżej:

- **Bateria 9V** - był to nasz pierwszy wybór - dość długo jej używaliśmy, jednak wymagało to regulatora napięcia z 5V na 3.3V, gdyż Arduino Mini Pro 5V/16Mhz nie posiada takiego, a takie napięcie 3.3V jest wymagane przez moduł radiowy nrf24. Dodatkowo, regulator zużywa około 3mA, co przy baterii cynkowo-węglowej(9V -6F22) o pojemności około 400 mAh, drastycznie zmniejszało czas działania do zaledwie 80 godzin.
- **3 baterie AAA** - nasz drugi wybór, pozwalają bezpośrednio dostarczyć zasilanie do modułu radiowego z dwóch baterii, co pozwala na usuniecie regulatora napięciowego. Pojemność trzech baterii AAA to około 3000 mAh, co jest ponad siedem razy lepszym wynikiem od baterii 9V. Jeśli zależy nam małych rozmiarach nadajnika, to należy wybrać ten wariant.
- **3 baterie AA** - nasz ostateczny wybór, ponieważ posiadają dwa razy większa pojemność niż baterie AAA oraz mogliśmy wykorzystać gotowy koszyk na 3 baterie AA.

Usunięcie diody LED z płytki Arduino Mini Pro

W celu zwiększenia żywotności należy koniecznie usunąć diodę LED z płytki, ponieważ zwiększa to zużycie z 0.14mA do 3 mA co jest kolosalną stratą i sprawia, że czas pracy na bateriach maleje dwukrotnie.



Zaznaczoną diodę należy usunąć. Dodatkowo można także rozważyć usunięcie regulatora napięcia, jednak nie przynosi już to aż tak dużych korzyści.

Długość życia stacji nadającej na 3 bateriach AA:

Przybliżona pojemność: 6000 mAh.

Podczas próby rejestracji zużycie wynosi około 20-27mA. Jednak zakładamy, że uda się urządzeniu uzyskać dostęp i swoje ID w dość krótkim czasie, co sprawia nie uwzględniamy tego podczas obliczeń.

Dodatkowo należy uwzględnić czas w jakim bateria sama się rozładowuje.

Pobór podczas stanu bezczynności: 0.11mA czas: 32s.

Pobór podczas wysyłania wiadomości 3mA czas: 0.33s (można zmniejszyć tą wartość likwidując diodę LED, która sygnalizuje wysyłanie).

Średni pobór to 0,14 mA, daje nam to około 4 lata i 9 miesięcy. Jednak musimy uwzględnić, że bateria sama się rozładowuje oraz, że czujniki mogą pracować w rożnych warunkach atmosferycznych, co może przyczynić się do szybszego rozładowania się.

Podsumowując, jeśli chcemy uzyskać maksymalny czas pracy na bateriach, to należy wykorzystać Arduino Mini Pro 3V 8 Mhz, usunąć diodę sygnalizująca wysyłanie wiadomości, zwiększyć czas bezczynności i wysyłać temperaturę rzadziej, zwiększyć maksymalnie rezystancje oporników. Wykonując wszystkie te czynności, oraz pomijając że bateria sama się rozładowuje prawdopodobnie można uzyskać czas działa w okolicach 6-8 lat na trzech bateriach AA.

Wykorzystanie kondensatorów, jako prostego filtra zakłóceń napięcia

W naszym projekcie na schematach widać, że korzystaliśmy z kondensatorów, które służą jako filtry zakłóceń napięcia. Jest to spowodowane tym, że w pierwszych fazach projektu korzystaliśmy z Arduino Nano, które pochodziło z Chin, zatem regulator napięcia zastosowany w nim nie dostarczał stałego napięcia. Sprawiało to, że moduł radiowy nie mógł odbierać wiadomości. Więc jeśli zapewnimy stałe napięcie na poziomie 3.3V to możemy zrezygnować z kondensatorów.

Opis działania stacji nadającej

Oprogramowanie zostało podzielone na 3 części zgodnie z dostarczaną funkcjonalność.

1. Inicjalizacja:
    - Sprawdzenie czujnika temperatury DS18B20 oraz pobranie jego adresu.
    - Inicjalizacja pinu przycisku oraz pinu diody LED.
    - Przyłączenie przerwania mikroprocesorowego do pinu przycisku.
    - Inicjalizacja moduły radiowego nrf24.
    - Próba przywrócenia ID z pamięci, jeśli 0 to rozpoczynamy część odpowiedzialną za rejestracje, w przeciwnym razie rozpoczynamy wysłanie temperatury.

2. Rejestracja:
    - Zapalenie diody LED.
    - Wysyłanie wiadomości na adres centrali, do czasu uzyskania odpowiedzi.
    - Po uzyskaniu odpowiedzi, wyciągamy z niej nadane ID, oraz zapisujemy je do pamieć trwałej.
    - Wyłączamy diodę LED przechodzimy do wysyłania temperatury.

3. Wysyłanie temperatury:
    - Jeśli nasze ID nie jest zero to zapalamy diodę LED, w przeciwnym razie przechodzimy do części drugiej.
    - Odczytujemy temperaturę, stan baterii, tworzymy wiadomość, wysyłamy.
    - Gasimy diodę LED.
    - Przechodzimy w stan uśpienia na 32 sekundy, a potem powtarzamy wszystko.

*. Występuje także przerwanie, po naciśnięciu przycisku, odpowiada za wyzerowanie ID oraz ID przetrzymywanego w pamieć trwałej.

Wykorzystanie biblioteki EEPROM.h do dostępu do pamięci trwałej mikrokontrolera i przechowywaniu tam ID.

Aby uniknąć sytuacji, że chwilowy brak zasilania spowodowany, na przykład wymianą baterii, wymuszał na użytkowniku ponowne przeprowadzenie procesu rejestracji, zastosowaliśmy rozwiązanie, które korzysta z EEPROM. Jest to obszar pamieć trwałej, do której można dostać się dzięki bibliotece EEPROM.h . Pozwala to na efektywne przetrzymywanie ID stacji nadającej.

Kod odpowiedzialny za zapisanie ID w EEPROM:

```
        int id_mem_addr0 = 0;
        int id_mem_addr1 = 1;
        byte ret0;
        byte ret1;
        ----------------------------------------
        unsigned short tempID = nanoID;
        ret0 = (byte)(tempID & 0xff);
        ret1 = (byte)((tempID >> 8) & 0xff);
        EEPROM.write(id_mem_addr0,ret0);
        EEPROM.write(id_mem_addr1,ret1);
```

Kod odpowiedzialny za odczytanie ID z EEPROM:

```
miniProID = (unsigned short)
                    ((EEPROM.read(id_mem_addr1)<<8) | (EEPROM.read(id_mem
_addr0)));
```

NRF24L01+

Konfiguracja nrf24

Kod odpowiedzialny za konfiguracje:

```
RF24 radio(7,8);

#define R_PI  {10,10,10,10,10,10}
#define ME {21,22,23,24,25,26}
```

```
byte addresses[][6] = {R_PI,ME};

//-----------------------------------------------

radio.begin();
radio.setDataRate(RF24_250KBPS);
radio.setPALevel(RF24_PA_MAX);
radio.openWritingPipe(addresses[0]);
radio.openReadingPipe(1,addresses[1]);
radio.printDetails();
```

Główna koncepcja komunikacji radiowej opiera się o dwa kanały, jeden wspólny dla wszystkich nadajników i jeden dla centrali. Dlatego w instrukcji obsługi trzeba zalecić użytkownikowi rejestrowanie nadajników pojedynczo. Dodatkowo fakt, że nadajnik nadaje na wszystkie centrale sprawia, że z jednej strony nie jesteśmy w stanie wyselekcjonować konkretnych nadajników, dla konkretnej stacji. Można osiągnąć ten efekt software'owo sprawdzając nadchodzące wiadomości i porównując ich ID z zawartością rejestru nadanych ID. Potrzebujemy wtedy pliku na przetrzymanie nadanych ID i należało by wprowadzić lepszy model HandShake (komunikacja dotycząca nadania ID).

Wybór mocy nadawania
Na potrzeby naszego projektu korzystaliśmy z poziomu: RF24_PA_MAX
z dostępnych:
RF24_PA_MIN = -18dBm,
RF24_PA_LOW = -12dBm,
RF24_PA_HIGH = -6dBm,
RF24_PA_MAX = 0dBm
Jako że interesuje nas jak najmniejsze zużycie energii, nie chcemy jej tracić na niepotrzebne wysyłanie z zbyt duża mocą.
Dlatego można zaimplementować rozwiązanie ze zmienną mocą, czyli jak nie uda nam się wysłać wiadomości to należy zwiększyć poziom mocy.
Jednak okres w jakim wysyłamy jest na tyle krótki (1 wiadomość co 32 sekund), że nie jest to aż tak istotne, bo czas ten jest znikomy. Bardziej powinniśmy się skupić, aby obniżyć zużycie podczas bezczynności mikrokontrolera.

DS18B20+

Jako czujnik temperatury wybraliśmy zaproponowany w temacie, cyfrowy czujnik DS18D20+, który jest łatwy w obsłudze i nie powoduje problemów przy konfiguracji, tak jak czujniki analogowe.
Biblioteki dla Arduino potrzebne do pracy DS18D20 to:

- SPI
- OneWire
- DallasTemperature

Kod odpowiedzialny za konfiguracje i odczyt temperatury:

```
#define DS18B20_PIN 2
```

```
#define TEMP_RESOLUTION 9
OneWire oneWireDS18B20(DS18B20_PIN);
DallasTemperature temperatureSensors(&oneWireDS18B20);
DeviceAddress thermometerAdress;

//-------------------- in setup:
  temperatureSensors.begin();
  if (!temperatureSensors.getAddress(thermometerAdress, 0))
    Serial.println("Unable to find address of thermometer");
  temperatureSensors.setResolution(thermometerAdress, TEMP_RESOLUTION);
  Serial.println(getTemp());
//----------------------

float getTemp(){
  temperatureSensors.requestTemperatures();
  return temperatureSensors.getTempC(thermometerAdress);
}
```

Zalecenia dla użytkownika:

Stacja nadająca to bardzo proste urządzenie, które po włączeniu czeka na możliwość rejestracji na centrali, sygnalizuje to włączoną diodą LED, następnie, gdy uzyska dostęp do stacji nadające, wysyła temperaturę co 32 sekundy, każdą próbę wysłania sygnalizuje mrugnięciem diody LED.
Istnieje możliwość resetowania nadanego ID poprzez naciśniecie przycisku.
Uwagi:

- rejestrować sensory pojedynczo
- po naciśnięciu przycisku (reset pamięci sensora) odczekać 8 sekund do zapalenia LED lub ponowić czynność lub wyłączyć i włączyć sensor