

Termometr Bluetooth.



Potrzebne podzespoły:

- praktycznie dowolna płytki z modulem BTM222 np AVT1635,
- płytki Arduino Uno lub podobna,
- czujnik temperatury typu LM35,

Najważniejsze o BTM222.

Moduł może komunikować się za pośrednictwem kilku różnych interfejsów, ze względu na łatwość obsługi polecany jest UART. Do poprawnej komunikacji za pośrednictwem UART'u nie są niezbędne sygnały RTS i CTS, ale wtedy należy wprowadzić niewielkie opóźnienia czasowe pomiędzy bajtami danych, wysyłanymi do modułu.

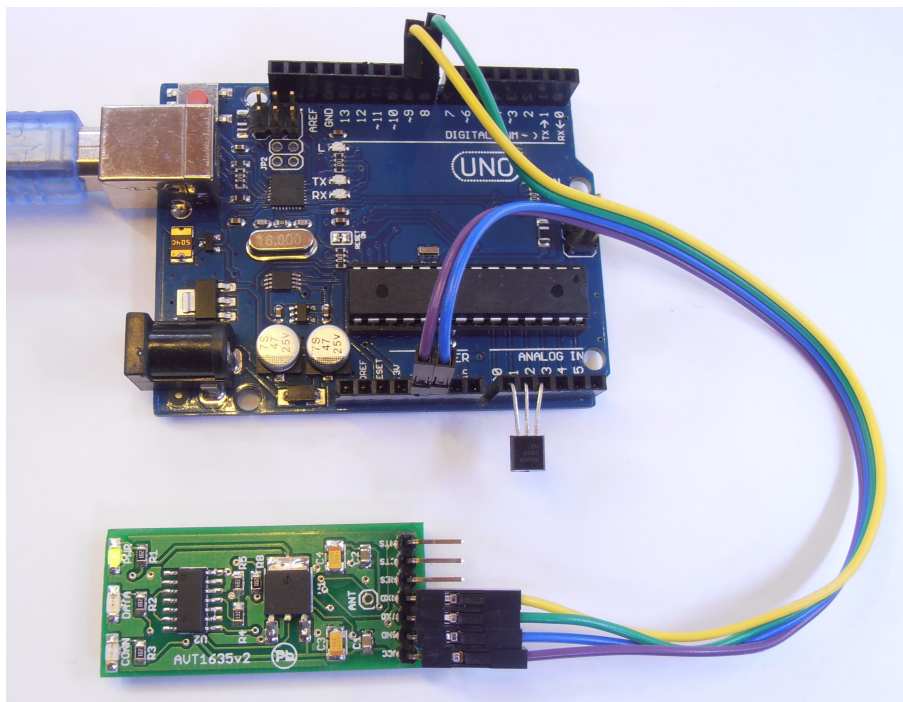
Należy bardzo ostrożnie zmieniać parametry komunikacji interfejsu UART, ponieważ niewłaściwe ustawienie może zablokować dostęp do modułu a nie ma sposobu na przywrócenie ustawień bez poprawnej komunikacji z modulem.

Moduł wymaga zasilania 3,3V i nie toleruje napięcia 5V na wejściach dlatego przy współpracy z Arduino musi być wyposażony w translator napięć.

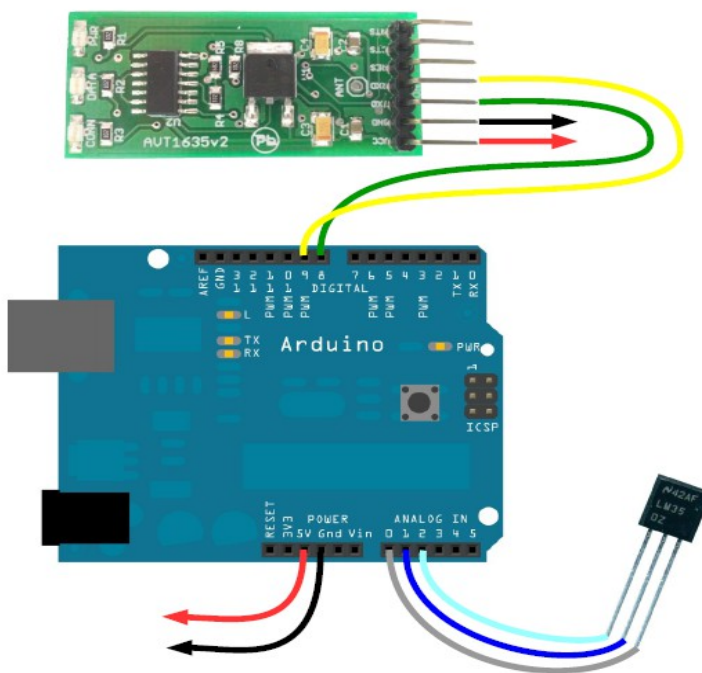
Arduino i BTM222.

Płytkę Arduino posiada sprzętowy interfejs szeregowy UART ale jest on połączony z konwerterem USB i dlatego bardzo się przydaje w czasie uruchamiania programów. Dlatego do komunikacji z modułem Bluetooth zostanie wykorzystany software'owy UART.

Podzespoły należy połączyć jak na fotografii 1 lub według schematu połączeń (rys 1). Po podłączeniu USB, dioda CONN na płytce AVT1635 powinna migać. Już teraz moduł Bluetooth będzie widziany przez inne urządzenia jako Serial Adapter (rys 2), można go sparować np ze smartfonem podając w trakcie tej operacji kod pin 1234.



Fotografia 1. Sposób połączenia.



Rys 1. Schemat połączeń.

Program testowy.

Pierwszy program posłuży do sprawdzenia poprawności połączeń i komunikacji z płytą AVT1635. Na początku dołączana jest biblioteka *SoftwareSerial.h*, deklarowane są zmienne, dwa bufor i dwa znaczniki odebranych komunikatów. Ostatecznie w sekcji *setup()* program uruchamia dwa interfejsy szeregowy, jeden programowy do komunikacji z modułem Bluetooth, drugi sprzętowy do śledzenia przebiegu programu i komunikatów z i do modułu.

Listing 1. Konfiguracja.

```
#include <SoftwareSerial.h>          //dodaje niezbędną bibliotekę

//dodatkowy software'owy port szeregowy dla komunikacji z Bluetooth
SoftwareSerial blueSerial(8, 9);

String baseBuffer = "";              //bufor danych z USB
String blueBuffer = "";              //bufor danych z Bluetooth
int baseComplete = 0;                //znacznik odebranych danych z USB
int blueComplete = 0;                //znacznik odebranych danych z Bluetooth

//-----
void setup() {
    //uruchamia sprzętowy port szeregowy dostępny z USB
    Serial.begin(19200);
    //uruchamia dodatkowy port szeregowy dla Bluetooth
    blueSerial.begin(19200);
}
```

W pętli głównej program sprawdza czy zostały odebrane jakieś dane i umieszcza je w buforach - funkcje *baseReceive()* i *blueReceive()*. W przypadku komunikacji po USB pierwsze wystąpienie znaku nowej linii *\n* traktowane jest jako odebranie komunikatu, co naturalnie pokrywa się z wciśnięciem klawisza ENTER na klawiaturze. Natomiast moduł BTM222 każdy komunikat poprzedza i kończy znakiem nowej linii, dlatego odebranie komunikatu następuje po drugim znaku nowej linii - *blueComplete* ≥ 2 .

Listing 2. Pętla główna oraz odczyt z interfejsów szeregowych.

```
//-----
void loop() {
    //odczytujemy dane z portów szeregowych
    baseReceive();
    blueReceive();

    //-----
    if (baseComplete  $\geq 1$ ) {                //jeśli odebrano z USB
        blueSend(baseBuffer);              //przełącz do Bluetooth
        baseBuffer = "";                   //wyczyść bufor
        baseComplete = 0;                  //zeruj znacznik
    }

    //-----
    //Moduł BTM222 rozpoczyna każdy komunikat znakiem końca linii
    //więc czekamy aż będą 2 znaki końca
    if (blueComplete  $\geq 2$ ) {                //jeśli odebrano z Bluetooth
        Serial.println(blueBuffer);         //podgląd odebranych danych
        blueBuffer = "";                    //wyczyść bufor
        blueComplete = 0;                   //zeruje znacznik odebranych danych
    }

    //-----
    delay(1);
}
```

```

//-----
//odbieranie danych z USB
void baseReceive()
{
    while (Serial.available()) { //jeśli coś odebrane
        char inData = (char)Serial.read(); //odbierz dane
        baseBuffer += inData; //i umieść w buforze
        if (inData == '\n') { //jeśli koniec linii
            baseComplete++; //ustaw znacznik
        }
    }
}

//-----
//odbieranie danych z Bluetooth
void blueReceive()
{
    while (blueSerial.available()) { //jeśli coś odebrane
        char inData = (char)blueSerial.read(); //odbierz dane
        blueBuffer += inData; //i umieść w buforze
        if (inData == '\n') { //jeśli koniec linii
            blueComplete++; //ustaw znacznik
        }
    }
}

```

Ważną funkcją jest *blueSend()*, która odpowiada za wysyłanie komunikatów do modułu. Po wysłaniu każdego znaku wstawiona jest przerwa *delay(50)*. Dodatkowo znak nowej linii *\n* zamieniany jest na znak *\r*; ponieważ moduł BTM222 wymaga aby każda komenda zakończona była najpierw właśnie tym znakiem.

Listing 3. Funkcja wysyłania do BTM222.

```

//-----
//Wysyła dane do Bluetooth
void blueSend (String data) {
    int i;
    int data_length = data.length(); //ilość danych
    //Serial.println(data); //podgląd wysyłanych

    //komunikacja z BTM222 bez kontroli przepływu,
    //tylko sygnały RXD i TXD, bez RTS i CTS,
    //wymaga wstawienia przerwy ok 50ms po każdym znaku
    for (i = 0; i < data_length; i++)
    {
        //komendy dla BTM222 muszą być zakończone znakiem CR
        //dlatego zamieniam '\n' na '\r'
        if (data.charAt(i) == '\n') data.setCharAt(i, '\r');

        blueSerial.print(data.charAt(i)); //wysyła jeden znak
        delay(50); //przerwa 50ms
    }
}

```

Po zaprogramowaniu płytki Arduino należy uruchomić dostępny w pasku narzędzi monitor szeregowy, ustawić prędkość komunikacji na 19200 a obok ustawić „Nowa linia” jak na rys. 2. Teraz w pasku wysłaj należy wpisać AT i przycisnąć klawisz ENTER, układ powinien odpowiedzieć OK. Tak można wysłać każdą z komend modułu BTM222 np. „ATB?”, która zwróci sprzętowy identyfikator układu.



Rys 3. Monitor szeregowy i komendy testowe

Wykrywanie połączenia.

Moduł BTM222 na każdy odebrany komunikat odpowiada OK lub ERROR. Ponadto sam może wysłać jeden z komunikatów CONNECT lub DISCONNECT które sygnalizują nawiązanie i zakończenie połączenia. Teraz program zostanie rozbudowany o funkcje wykrywające te zdarzenia. Na początku jednak zostanie dodana nowa zmienna, która będzie przechowywała informacje o stanie połączenia oraz będzie decydowała o zaświeceniu diody LED znajdującej się na płytce Arduino.



Rys 2. Sparowany moduł BTM222

Listing 4. Zmienna statusu połączenia oraz konfiguracja sterowania diodą LED.

```
int blueStatus = 0;           //status połączenia Bluetooth, aktywne to 1
int ledPin = 13;              //dioda led na płycie Arduino

//-----
void setup() {
    Serial.begin(19200);
    blueSerial.begin(19200);

    //konfiguruje sterowanie diodą led
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW);
}
```

Pętlę główną zmodyfikujemy dodając następujące warunki:

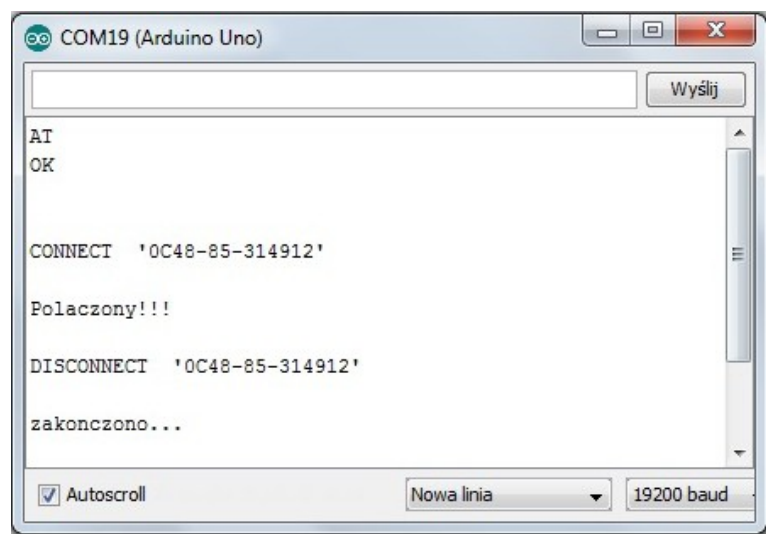
Listing 5. Wykrywanie komunikatów o połączeniu i sterowanie diodą LED.

```
//-----  
if (blueComplete >= 2) {                                //jeśli odebrano z Bluetooth  
    Serial.println(blueBuffer);                          //podgląd odebranych danych  
  
    //odebranie treści "CONNECT" oznacza połączenie np ze smartfonem  
    if (blueBuffer.substring(2, 9) == "CONNECT") {  
        Serial.println("Polaczony!!!"); //wysyła info do USB  
        blueStatus = 1;                //ustawia aktywny status połączenia  
    }  
  
    //odebranie treści "DISCONNECT" oznacza zakończenie połączenia  
    if (blueBuffer.substring(2, 12) == "DISCONNECT") {  
        Serial.println("zakonczone..."); //wysyła info do USB  
        blueStatus = 0;                //zeruje status połączenia  
    }  
  
    blueBuffer = "";                                //czyści bufor  
    blueComplete = 0;                              //zeruje znacznik odebranych danych  
}  
  
//-----  
if (blueStatus == 1)  
    digitalWrite(ledPin, HIGH);                      //aktywne połączenie LED świeci  
else  
    digitalWrite(ledPin, LOW);                        //brak połączenia LED nie świeci  
delay(1);
```

Po skompilowaniu zaprogramowaniu płytki należy uruchomić monitor szeregowy i połączyć się z modulem bluetooth przy pomocy dołączonej aplikacji. Temperatura jeszcze nie jest wskazywana ale dioda LED reaguje prawidłowo – zaświeca się przy nawiązaniu połączenia i gaśnie po zamknięciu aplikacji a urządzenie wysyła komunikaty podobne do tych na rys 3.

Pomiar temperatury.

Do pomiaru temperatury wykorzystany został czujnik LM35, który posiada wyjście analogowe. Napięcie na wyjściu jest proporcjonalne do temperatury i wynosi 10mV/*C. Takie rozwiązanie jest bardzo łatwe w realizacji i daje zakres pomiarowy od ok 2 do 110 stopni. Najpierw zostaną dodane odpowiednie zmienne i skonfigurowane odpowiednie wyprowadzenia ponieważ czujnik będzie zasilany z portów Arduino.



Rys 4. Monitor szeregowy i komunikaty o połączeniu

Listing 6. Konfiguracja czujnika temperatury.

```
//czujnik temperatury to LM35 dołączony do wyprowadzeń A1, A0, A2
int temperature_gnd = A0;           //masa zasilania dla LM35
int temperature_pin = A1;           //wyjście z LM35
int temperature_vdd = A2;           //plus zasilania dla LM35
int temperature_val;                //odczytana wartość temperatury
String temperature_str;             //temperatura w postaci ramki "12,3'C"
int temperature_period = 0;         //odliczanie częstości pomiarów

//-----
void setup() {
    //uruchamia sprzętowy port szeregowy dostępny z USB
    Serial.begin(19200);
    //uruchamia dodatkowy port szeregowy dla Bluetooth
    blueSerial.begin(19200);

    //konfiguruje wyprowadzenia A0 i A1 tak by zasilaly czujnik temperatury
    pinMode(temperature_gnd, OUTPUT);
    pinMode(temperature_vdd, OUTPUT);
    digitalWrite(temperature_gnd, LOW);
    digitalWrite(temperature_vdd, HIGH);

    //odczyt temperatury będzie wykonywany poprzez przetwornik ADC,
    //dla większej dokładności wykorzystam napięcie referencyjne 1,1V
    analogReference(INTERNAL);

    //konfiguruje sterowanie diodą led
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, LOW);
}
```

Teraz w pętli głównej zostanie dodany fragment, który odpowiada za pomiar temperatury oraz jej zapis w postaci ramki: xx,x'C. Informacja ta będzie wysyłana do USB oraz jeśli połączenie będzie aktywne *blueStatus == 1* to ramka będzie wysyłana także do modułu Bluetooth i w efekcie wartość temperatury będzie wyświetlana w oknie aplikacji.

Listing 7. Pomiar temperatury.

```
//-----
temperature_period++;                //odlicza do następnego pomiaru
if (temperature_period >= 2000) {    //jeśli upłynęło ok 2s
    temperature_period = 0;          //zeruje licznik

    //odczytuje temperaturę z czujnika LM35 przy pomocy przetworzika ADC
    temperature_val = analogRead(temperature_pin);
    //LM35 daje na wyjściu napięcie proporcjonalne do temperatury 10mV/*C
    //skłauje wynik i dopasowuje do napięcia referencyjnego,
    //maksymalna temp. 110*C
    temperature_val = map(temperature_val, 0, 1023, 0, 1100);

    //oddziela jednostki od dziesiętnych i tworzy ramkę postaci "12,3'C"
    temperature_str = String(temperature_val / 10) + ",";
    temperature_str += String(temperature_val % 10) + "'C\r";
    Serial.println(temperature_str); //wysyła wynik do USB

    //jeśli aktywne połączenie to wysyła do Bluetooth
    if (blueStatus == 1) blueSend(temperature_str);
}
```

Aplikacja.

Aplikacja została stworzona w środowisku Android Studio na podstawie projektu Arduino-Android-Sensors opisanego na stronie <https://github.com/HarryGoodwin/Arduino-Android-Sensors>.

Po uruchomieniu aplikacji pojawia się okno z wyborem urządzenia bluetooth, należy wybrać Serial Adapter, aplikacja połączy się z modulem i zacznie wyświetlać temperaturę rys 5 i 6.



Rys 5. Okno wyboru urządzenia



Rys 6. Okno wyświetlania temperatury

Pełne źródła wszystkich programów i oraz aplikacji dostępne są w materiałach dodatkowych do artykułu

KS