

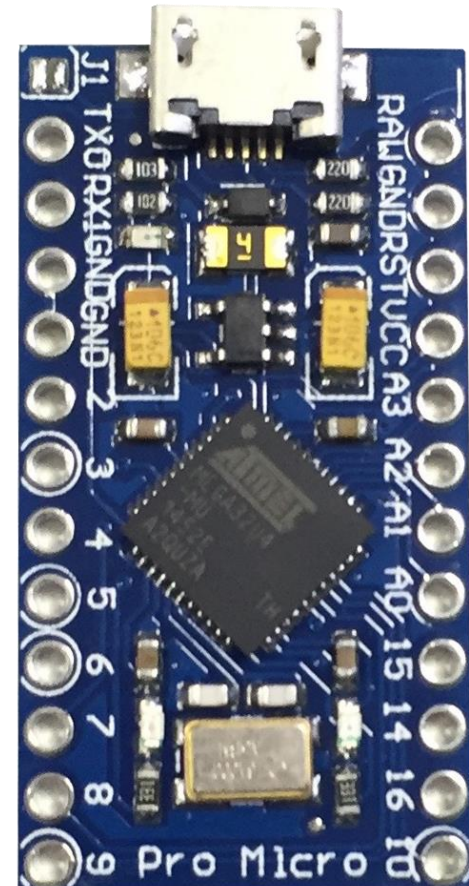
Komputer pokładowy

I sesja technologiczna



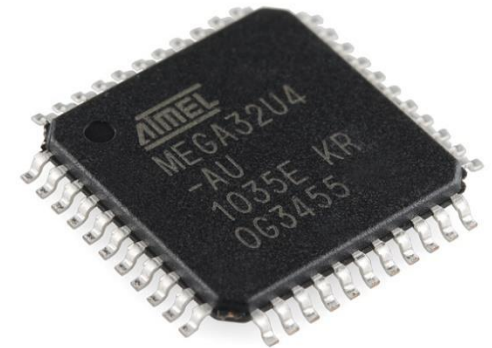
Komputer pokładowy Arduino Pro Micro

- **mikrokontroler ATmega32U4**
- **bootloader Arduino**
- **port microUSB:** (programowanie + serial port + zasilanie)
- **wyprowadzone piny procesora** (m.in. GPIO, ADC, SPI, I2C, UART) + **zasilanie**
- <https://www.sparkfun.com/products/12640>



Mózg CanSata – ATMega 32U4

- **programowalny mikrokontroler**
- **AVR, 8-bit**
- **32K pamięci FLASH programu**
- **2.5K pamięci SRAM**
- **1K pamięci EEPROM**
- **Układy peryferyjne:**
 - **26 x GPIO**
 - **12 x 10-bit ADC**
 - **UART**
 - **2x SPI, I2C**
 - **USB**

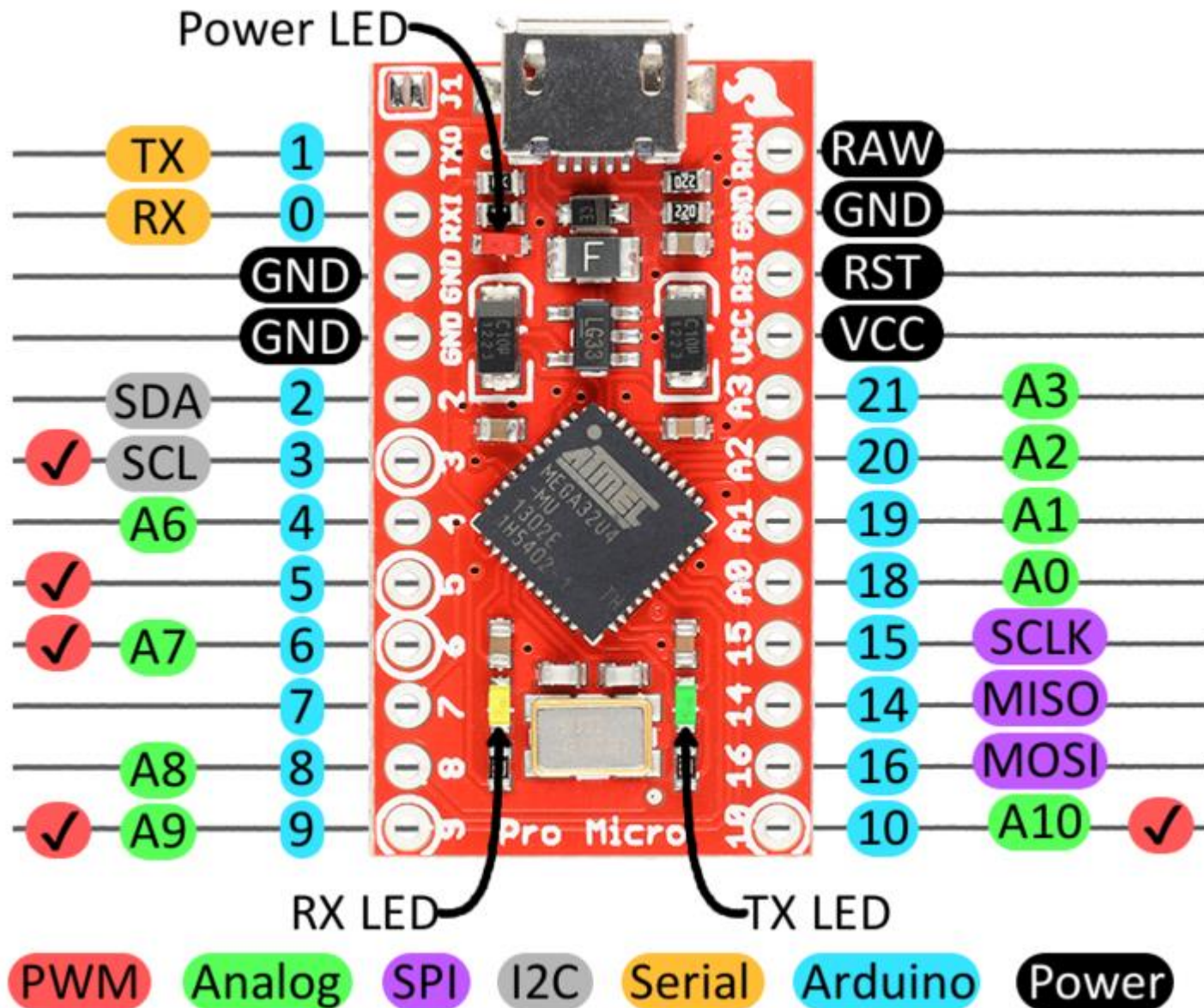


procesor

pamięć

układy peryferyjne

Złącza komputera pokładowego



Programowanie



- **C / C++**
- **Arduino IDE, Atmel Studio...**
- zgodność z platformą **Arduino**
- liczne biblioteki
- mnogość przykładów i tutoriali dotyczących programowania

A screenshot of the Arduino IDE interface. The menu bar at the top includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checking, running, saving, and uploading. The main text area shows a C++ sketch titled "Sensors_LM35". The code includes comments in Polish and uses the `Serial` and `Serial1` objects for communication. The `loop()` function reads the LM35 sensor value, converts it to temperature, and prints it to the serial monitor. The status bar at the bottom right indicates "TMinus1 on COM12".

```
File Edit Sketch Tools Help

Sensors_LM35
Serial1.begin(9600); // inicjalizacja Serial1@9600 bps
Serial1.begin(9600); // inicjalizacja Serial1@9600 bps
}

void loop() {
  // zmierz napięcie z LM35:
  LM35_val = analogRead(LM35_input);

  // przelicz na temperature w stopniach Celsjusza
  LM35_temp = 0.488*LM35_val;

  // przeslij przez serial
  Serial.print("LM35:");
  Serial.println(LM35_temp);

  Serial1.print("LM35:");
  Serial1.println(LM35_temp);

  // poczekaj 1 sekunde
  delay(1000);
}
```

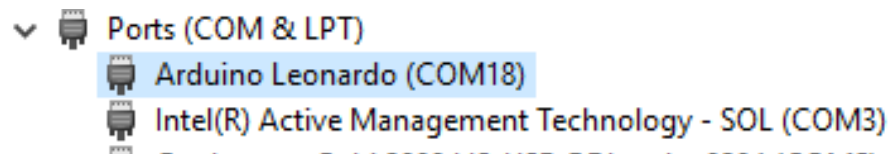
TMinus1 on COM12

Uruchomienie płytki

1. Instalacja sterowników do płytki komputera pokładowego

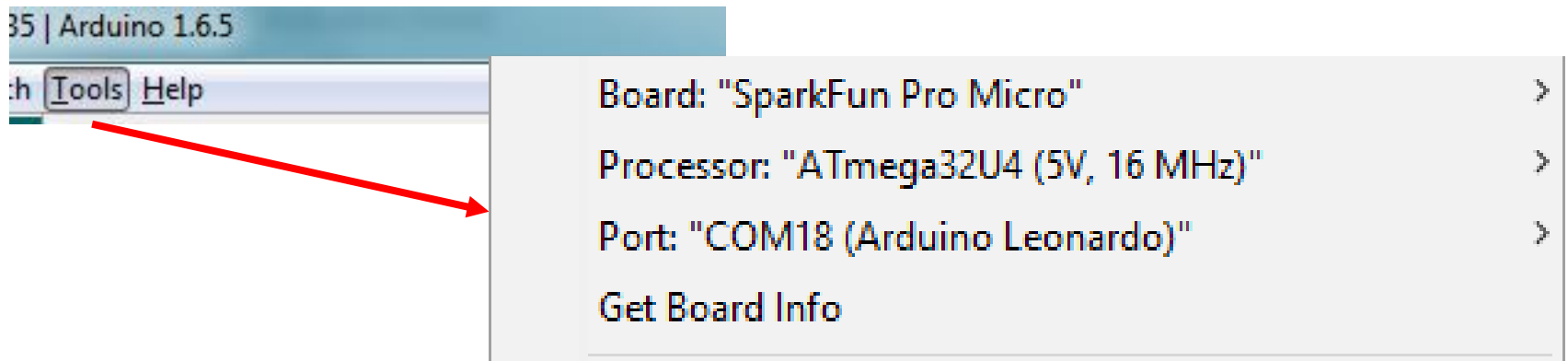
<https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide/installing-windows>

2. Płytką widoczną w **Menedżerze urządzeń**



3. Instalacja Arduino IDE (wersja 1.6.4 lub wyższa)

4. Instalacja płytki komputera pokładowego w Arduino IDE



Uruchomienie płytki

4. Instalacja płytki komputera pokładowego w **Arduino IDE**:

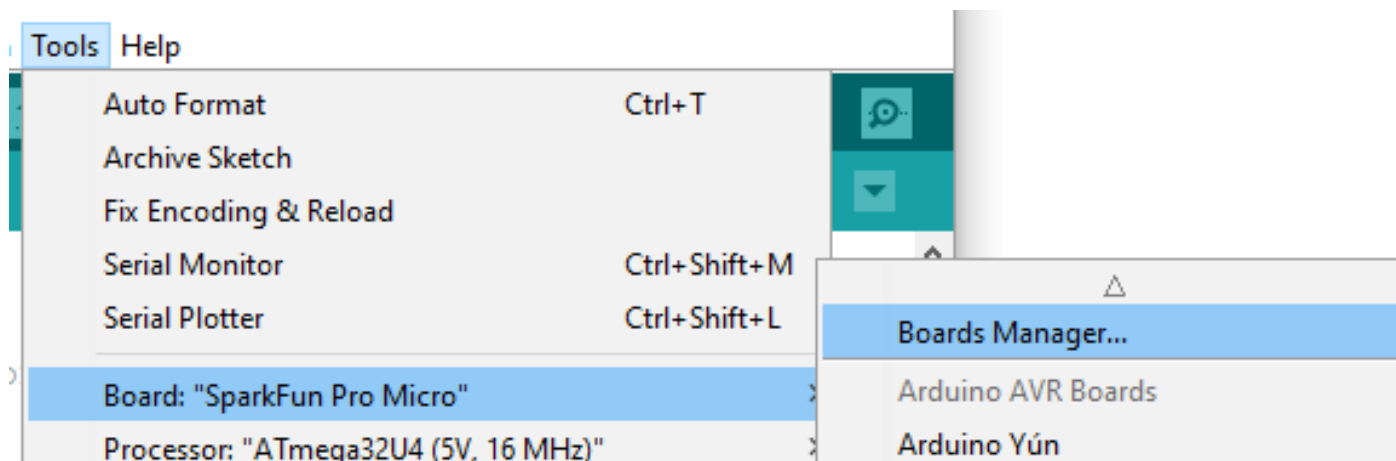
- otwieramy okno **File > Preferences**
- w polu “Additional Board Manager URLs” wpisujemy:

```
https://raw.githubusercontent.com/sparkfun/Arduino_Boards/master/IDE_Board_Manager/package_sparkfun_index.json
```

Uruchomienie płytki

4. Instalacja płytki komputera pokładowego w **Arduino IDE**:

- otwieramy Tools > Board: „xxx” > Board Manager



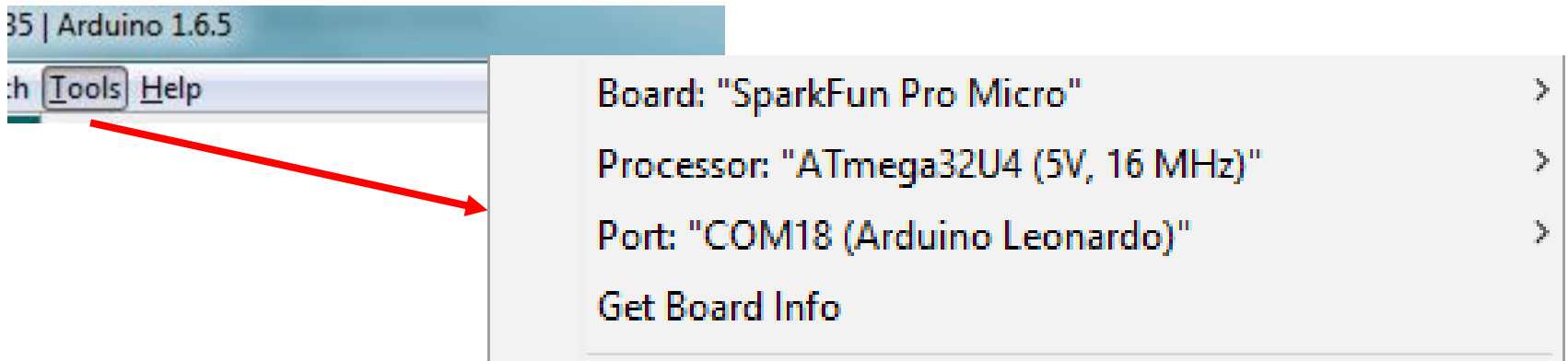
- szukamy SparkFun AVR Boards
- klikamy Install

SparkFun AVR Boards by SparkFun Electronics version **1.1.5** **INSTALLED**
Boards included in this package:

Uruchomienie płytki


4. Instalacja płytki komputera pokładowego w **Arduino IDE**:

- wybieramy płytkę **Tools > Board > „SparkFun Pro Micro”**
- wybieramy procesor **ATMega32U4 (5V, 16 MHz)**
- wybieramy port **wg wskazań Menedżera urządzeń**



Płytką jest gotowa do pracy!

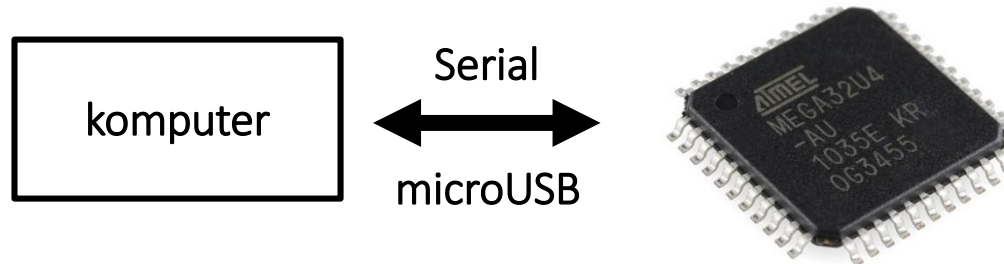
Wgrywanie pierwszego programu

- otwórz plik „**HelloCanSat.ino**” z katalogu „CanSat 2017” na Pulpicie
- **skompiluj** i **wgraj** do procesora: 
- **obserwuj działanie!**

Gdzie jesteśmy?

- ✓ zainstalowane sterowniki do płytki
- ✓ skonfigurowane środowisko programistyczne
- ✓ działająca, przetestowana płytka komputera pokładowego

Komunikacja przez Serial port (UART)



- szybki, dwukierunkowy interfejs komunikacji
- inicjalizacja:

```
Serial.begin(9600);
```

- wysyłanie danych:

```
Serial.print(„Hello world”);
```

- odbieranie danych:

```
Serial.read();
```

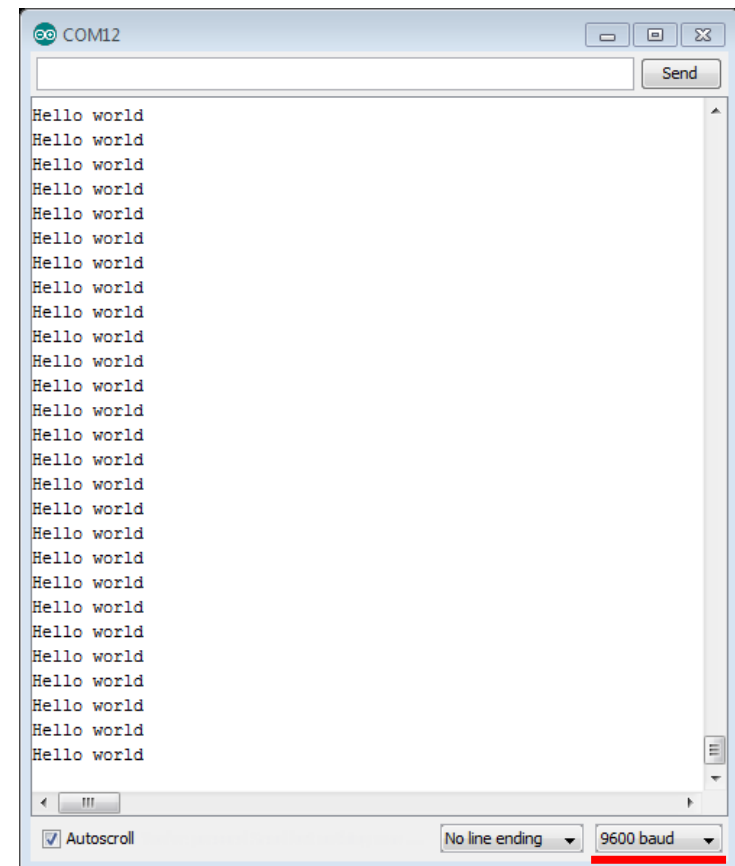
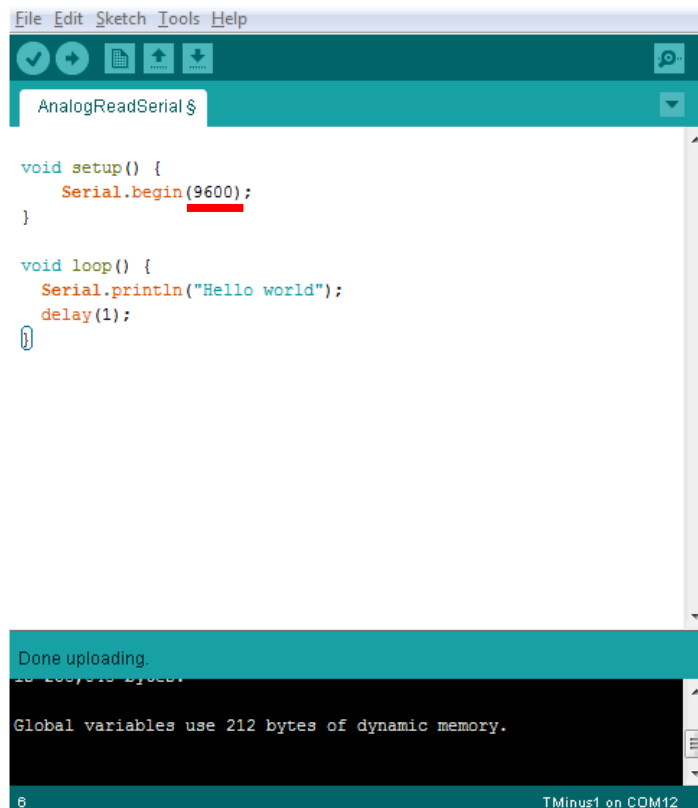
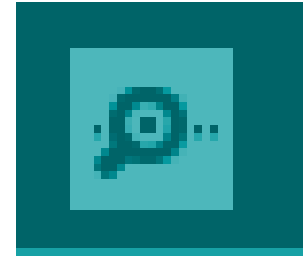
Komunikacja przez Serial port (UART)

```
void setup() {  
    Serial.begin(9600); // inicjalizacja z prędkością 9600bps  
}  
  
void loop() {  
    Serial.print("Warsztaty CanSat "); // wypisz tekst  
    Serial.println(2017);              // wypisz liczbę  
  
    delay(1000);                       // poczekaj 1 s  
}
```



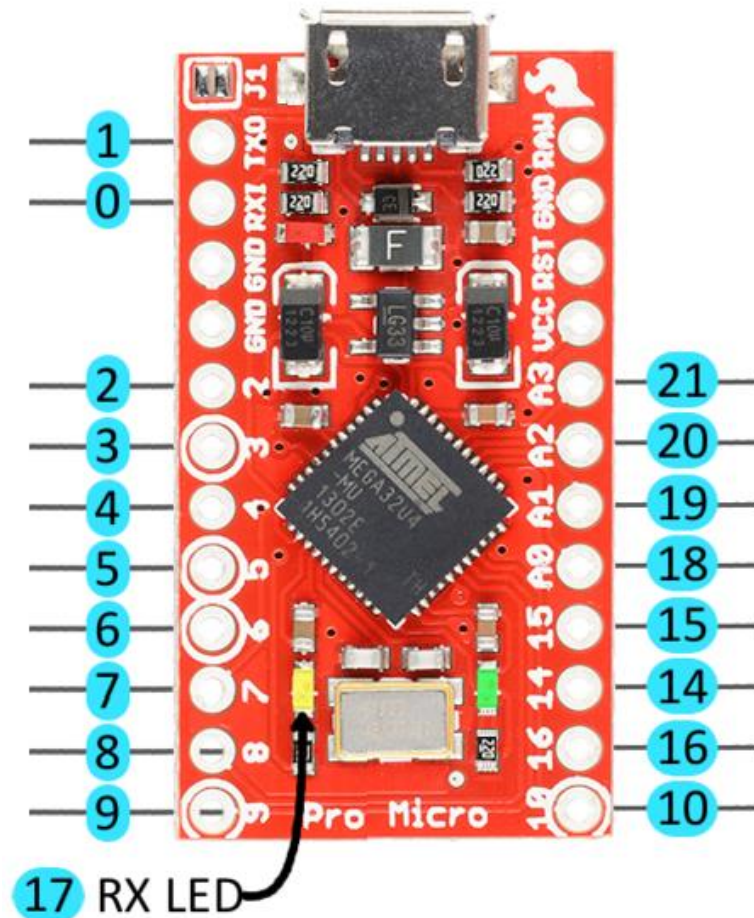
Komunikacja przez Serial port (UART)

- Serial monitor do wyświetlania danych
- baud rate** - nadawania i odbierania – takie same



GPIO – wyjścia cyfrowe

- **18 pinów**
- **dwa stany:**
 - **HIGH** – logiczne „1” – napięcie 5 V
 - **LOW** – logiczne „0” – napięcie 0 V
- **użycie:**
 - inicjalizacja: `pinMode(numer, OUTPUT);`
 - zmiana stanu: `digitalWrite(numer, stan);`



GPIO – wyjścia cyfrowe – miganie diodą

- dioda LED (RX), podłączona do pinu **17**

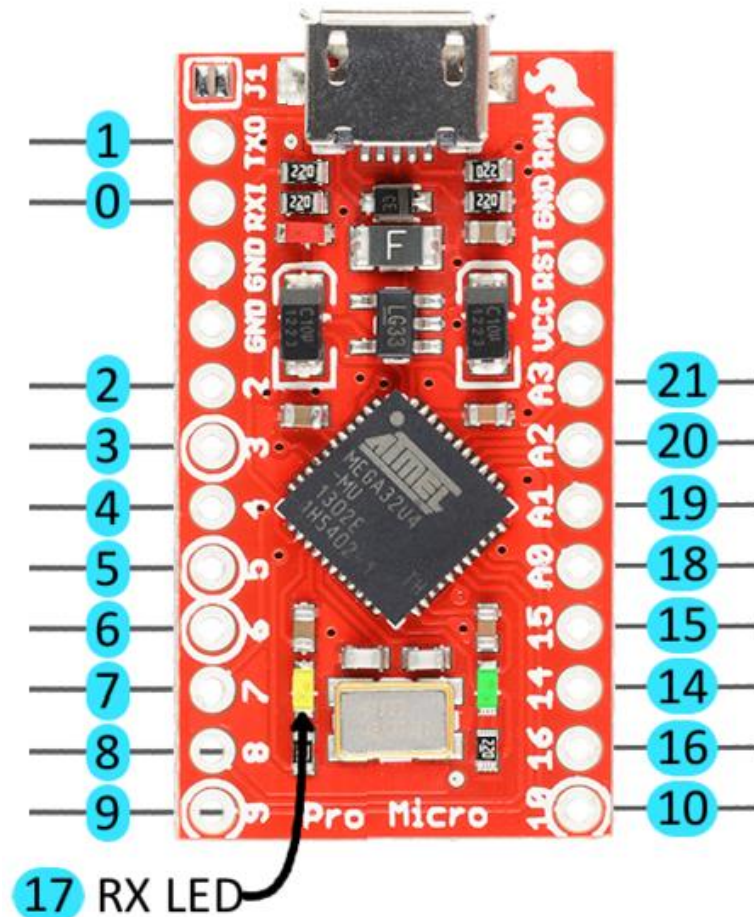
```
void setup() {  
    pinMode(17, OUTPUT);    // GPIO w trybie wyjścia  
}
```

```
void loop() {  
    digitalWrite(17, HIGH); // LED świeci  
    delay(1000);            // poczekaj 1 s  
    digitalWrite(17, LOW);  // LED nie świeci  
    delay(1000);            // poczekaj 1 s  
}
```



GPIO – wejścia cyfrowe

- 18 pinów
- **wykrywają** dwa stany:
 - **HIGH** – logiczne „1” – napięcie 5 V
 - **LOW** – logiczne „0” – napięcie 0 V
- **użycie:**
 - inicjalizacja: `pinMode(numer, INPUT);`
 - odczytanie stanu:
`bool stan = digitalRead(numer);`



GPIO – wejścia cyfrowe + Serial

- wysyłanie stanu wejścia cyfrowego na port Serial
- sprawdź działanie łącząc wejście raz z GND (0 V), następnie z 5 V

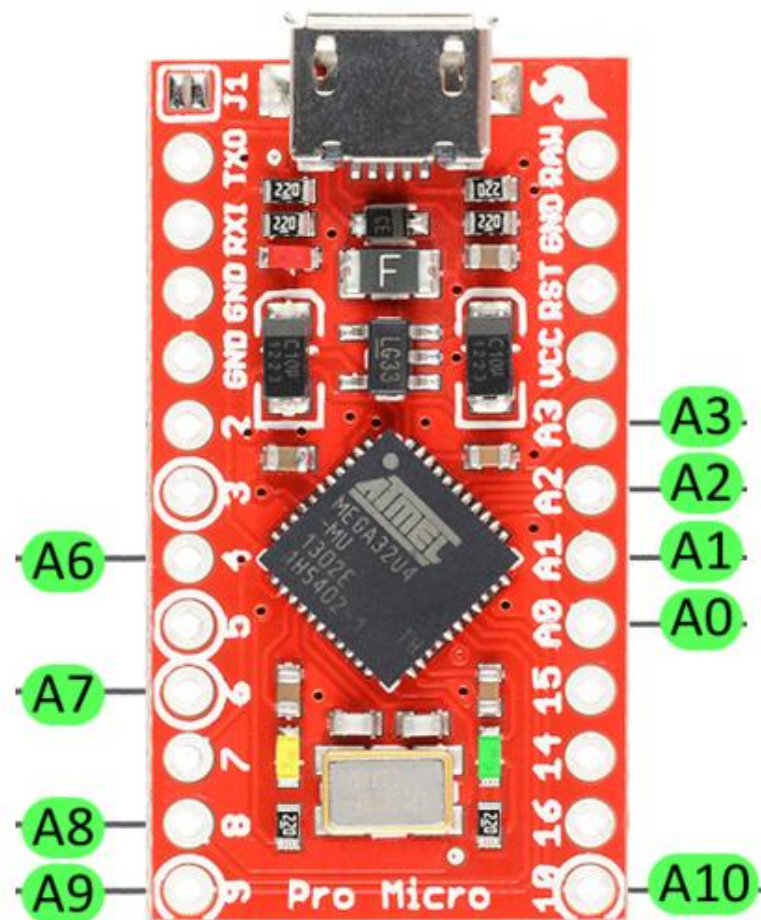
```
void setup() {  
    pinMode(0, INPUT);           // GPIO w trybie wejścia  
    Serial.begin(9600);          // inicjalizacja Serial  
}
```

```
void loop() {  
    bool stan = digitalRead(0);  // odczytaj stan  
    Serial.println(stan);        //wyślij stan  
    delay(1000);                // poczekaj 1 s  
}
```



Wejścia analogowe (ADC)

- mierzą napięcie z przedziału 0 – 5 V
- rozdzielczość 10 bit
- **LSB** = 4,88 mV
- **9 pinów**, oznaczone jako **Ax**
- **użycie:**
 - odczytanie wartości:
`int value = analogRead(Ax);`
(zwracana wartość: 0 – 1023 LSB)
 - przeliczenie LSB na napięcie:
 $\text{napięcie_mV} = \text{LSB} * 4,88$



Wejścia analogowe (ADC) + Serial

- wysyłanie stanu wejścia analogowego na port Serial
- sprawdź działanie łącząc wejście z GND (**0 V**), **1.5 V**, **5 V**.

```
void setup() {  
    Serial.begin(9600);           // inicjalizacja Serial  
}  
  
void loop() {  
    int value = analogRead(A0);   // odczytaj stan  
    float miliwolt = value*4.88;  // przelicz na mV  
  
    Serial.print("LSB = ");  
    Serial.println(value);  
  
    Serial.print("mV = ");  
    Serial.println(miliwolt);  
    delay(1000);                 // poczekaj 1 s  
}
```

Biblioteki qbcan

Instalacja bibliotek do obsługi czujników i modułu radiowego:

1. W Arduino IDE: **Sketch** > **Include Library** > **Add .ZIP Library...**
2. Wybieramy plik **Qbcan.zip**

Biblioteka pozwala na obsługę czujnika ciśnienia BMP180 oraz modułu radiowego RFM69HW.

3. Bibliotekę dołączamy za pomocą dyrektywy `#include`:
`#include <qbcan.h>`

Podsumowanie

Po tej sesji mamy:

- przetestowany i działający komputer pokładowy
- uruchomioną komunikację z komputerem PC
- opanowane podstawowe urządzenia peryferyjne
- zainstalowane biblioteki do obsługi czujników i radia

Q&A