Prosty pomiar napięcia (004; 12.07.2009; *arduino*)

Jedną z głównych funkcji mikrokontrolera (w Arduino Duemilanove jest nim Atmega328) jest pomiar napięcia (różnicy potencjałów) przy użyciu wejścia analogowego (analog input, AIN). Odbywa się to jednak zupełnie inaczej niż jak przy użyciu miernika uniwersalnego.

Parametry wejścia analogowego Arduino Duemilanove (dostępne 6 takich wejść):

rozdzielczość:	10bitów
liczba rozróżnianych poziomów sygnału:	$2^{10}=1024$
Maksymalna częstotliwość próbkowania sygnału:	~10kHz

Dla napięcia odniesienia (referencyjnego, AREF; standardowo 5V) rozdzielczość napięciowa próbkowania wynosi: $\frac{5V}{1023}\approx 4{,}89\,mV$. Jest to zatem teoretyczna dokładność pomiaru napięcia wykonywanego przez Arduino, jaki jest błąd tego pomiaru – należało by poszukać informacji w dokumenacji mikrokontrolera.

Zakresy napięcia wejściowego i odpowiadające im wartości odczytywane z wejścia analogowego wyglądają następująco:

0	0V – 0,00489V
1	0,00489V – 0,00978V
…	...
1023	4,99511V – 5V

Prosty program do Arduino odczytujący wartość sygnału na wejściu analogowym nr 0 i zapisujący ją do zmiennej:

```
int PoziomSygnalu;
void setup()
{
}
void loop()
{
  PoziomSygnalu = analogRead(0);
  delay(100);
}
```

Wartość wewnątrz nawiasu za poleceniem *delay* oznacza (w uproszczeniu) liczbę milisekund przerwy w wykonywaniu kodu.

Wysyłanie wartości odczytanych z wejścia analogowego do komputera (poprzez port szeregowy USB).

W pierwszej kolejności należy ustanowić parametry połączenia szeregowego między Arduino a komputerem. Są nimi:

- nazwa/adres portu szeregowego – np. COM4 (ustalone przez system podczas instalacji Arduino, a dokładniej to FTDI), możliwość sprawdzenia w ustawieniach systemowych bądź z poziomu aplikacji Arduino.
- Prędkość przesyłu danych – wyrażona w bitach na sekundę, dla komunikacji Arduino z komputerem musi przyjmować jedną z wartości: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200.

Maksymalna prędkość przesyłu danych wynosi zatem:
115 200 bitów na sekundę (b/s , bps),

lub w przeliczeniu:
14 400 bajtów na sekundę (B/s , Bps)

Port szeregowy umożliwia przesył danych w postaci kolejno wysyłanych pakietów wielobitowych, przy czym są one standardowo ustawione na maksymalną wartość 8 bitów (1bajt).

Przesyłanie wartości 8bitowych.

Aby odczytaną z wejścia analogowego wartość napięcia skwantowaną w 10bitach przedstawić w postaci 8 bitowej wystarczy dzielić ją przez 4. Otrzymujemy przy tym rozdzielczość sygnału równą $\frac{5\mathrm{V}}{255} \approx 19{,}6\,mV$

Zastosujemy program dla mikrokontrolera postaci:

```
int PoziomSygnalu;
void setup()
{
 Serial.begin(9600);
}
void loop()
{
  PoziomSygnalu = analogRead(0);
 Serial.print(PoziomSygnalu/4, BYTE);
  delay(100);
}
```

Serial.begin(9600) – konfiguracja połączenia szeregowego na prędkość 9600b/s (jest to wartość standardowo używana równa w przeliczeniu 1200B/s).

Serial.print(PoziomSygnalu/4, BYTE) – wysłanie 8bitowej wartości sygnału poprzez port szeregowy jako jeden bajt.

Po wczytaniu programu do Arduino i uruchomieniu zakładki Serial Monitor kolejno wypisywane będą wartości wysłane na port szeregowy (w zapisie dziesiętnym) z częstotliwością 10 pomiarów na sekundę. Częstotliwość tą można zwiększać zmniejszając wartość delay(). Skasowanie delay'a spowoduje wykonywanie pętli przez Arduino z maksymalną możliwą szybkością, przy czym może nastąpić osiągnięcie granicy przepustowości portu, a nawet przepełnienie jego bufora.

Aby kolejne wartości oddzielane był spacją należy dodać linijkę Serial.print(" ") przed delay. Aby kolejne wartości pojawiały się w kolejnych wierszach możemy w kodzie zmienić:
Serial.println(PoziomSygnalu/4, BYTE).

Serial.println() – polecenie do przesyłanych danych dołącza znak powrotu karetki (carriage return, ASCII 13, '\r') i znak nowej lini (newline, ASCII 10, '\n').

Pamiętać jednak należy, że rozdzielanie danych spacją lub pisanie ich w kolejnych wierszach utrudnić może komunikację, gdy sygnały te odbierać chcemy w innym programie (o czym w poście 005).

Ponieważ nie podłączaliśmy niczego do wejścia analogowego obserwować będziemy różne przypadkowe wartości, połączenie wejścia analogowego 0 z masą (GND) skutkować będzie wskazaniem zerowym.

Uwaga! Nie wolno podłączać bezpośrednio do wejść analogowych źródeł prąd (np. połączenie pinu 5V Arduino z wejściem analogowym, podłączenie baterii pomiędzy wejście analogowe i masę), gdyż może to spowodować uszkodzenie mikrokontrolera. Podłączenie wejścia analogowego musi być odpowiednio przemyślane, tak aby prąd wpływający do wejścia analogowego nie przekroczył wartości dopuszczalnej, stosować należy zatem np. dzielnik napięcia (o czym w poście 005).

dokładna specyfikacja komend przesyłania danych przez port szeregowych:

http://arduino.cc/en/Reference/Serial