LABORATORIUM nr1

SYSTEMY WBUDOWANE

Temat: Informacje podstawowe w zakresie mikrokontrolerów. Podstawy Arduino – środowisko, budowa sprzętowa. Pierwsze proste programy w systemie.

Cel: Celem ćwiczenia jest zapoznanie się z środowiskiem programistycznym Arduino oraz wykonanie podstawowych operacji na wyjściu (pin)

Wprowadzenie:

Systemy wbudowane, znane również jako systemy embedded,(specjalny rodzaj systemów komputerowych) to specjalne rodzaje komputerów lub układów elektronicznych, które są zaprojektowane do wykonywania konkretnych zadań lub funkcji, zwykle w ramach większego systemu.

MIKROPROCESOR to układ scalony lub chip który jest centralna jednostką obliczeniową w urządzeniach elektronicznych. Jego głównym zadaniem jest wykonywanie instrukcji programu, przetwarzania danych oraz sterowania operacjami w systemie komputerowym.

MIKROKONTROLER to układ scalony z procesorem oraz wbudowanymi pamięciami EEPROM, FLASH, RAM, ROM, układami wejścia wyjścia, timerami, licznikami, układami do komunikacji, układem kontroli zasilania, przetwornikiem analogowo - cyfrowym i wieloma innymi specjalnymi układami. Główne zastosowanie do wykonywania określonych zadań sterowania lub obliczeniowych w urządzeniach elektronicznych.

Mikrokontrolery dzielimy na:

- mikrokontrolery z pamięcią stałą ROM,
- wielokrotnie programowalne,
- programowalne wielokrotnie w zmontowanym urządzeniu docelowym,
- mikrokontrolery do połączenia z pamięcią zewnętrzną.

Mikrokontrolery z rodziny ATMEL AVR ATmega:

Mikrokontrolery z rodziny ATMEL AVR ATmega to popularna rodzina mikrokontrolerów produkowanych przez firmę Atmel, która jest znana z wysokiej jakości i wszechstronności swoich układów. Rodzina ATmega jest szczególnie popularna w środowisku projektowania elektroniki i mikrokontrolerów ze względu na swoją wydajność, dostępność oraz bogatą gamę funkcji. Cechy charakterystyczne dla mikrokontrolerów z rodziny ATMEL AVR ATmega:

1. Wysoka wydajność: ATmega oferuje różne modele z różnymi poziomami wydajności, od mikrokontrolerów o niskiej mocy do bardziej zaawansowanych, które pozwalają na przetwarzanie bardziej skomplikowanych zadań.

2. Duża liczba pinów GPIO: Mikrokontrolery ATmega mają różną liczbę pinów GPIO (General-Purpose Input/Output), co pozwala na obsługę różnych urządzeń i interfejsów.

3. Pamięć programu i pamięć danych: Posiadają wbudowaną pamięć programu typu Flash, która przechowuje kod źródłowy, oraz pamięć RAM do przechowywania danych.

4. Zintegrowane funkcje: W zależności od modelu, mikrokontrolery ATmega mogą zawierać różne zintegrowane funkcje, takie jak przetworniki analogowo-cyfrowe (ADC), przetworniki cyfrowo-analogowe (DAC), interfejsy komunikacyjne, generatory sygnałów PWM, timery i wiele innych.

5. Obsługa przerwań: ATmega obsługują przerwania, co pozwala na reakcję na zdarzenia w czasie rzeczywistym, co jest szczególnie przydatne w systemach czasu rzeczywistego.

6. Dostępność narzędzi programistycznych: Istnieje wiele środowisk programistycznych, takich jak Arduino IDE, które umożliwiają łatwe programowanie i rozwijanie aplikacji na mikrokontrolery z rodziny ATmega.

Mikrokontrolery z rodziny ATMEL AVR ATmega znajdują zastosowanie w wielu projektach elektronicznych, od prostych układów do bardziej zaawansowanych systemów wbudowanych, prototypowania urządzeń, robotyki, automatyki budynkowej, elektroniki hobby i wielu innych dziedzinach.

Wybór platformy sprzętowej:

Arduino jest platformą typu **Open Hardware. (**otwarta platforma sprzętowa i oprogramowania). Jest szeroko wykorzystywana do projektowania urządzeń elektronicznych i interaktywnych.

Darmowe oprogramowanie Arduino IDE - **1.9 BETA**.

Język programowania bazujący na C/C++

Do ćwiczeń wykorzystamy najpopularniejsza płytkę – **Arduino UNO**:

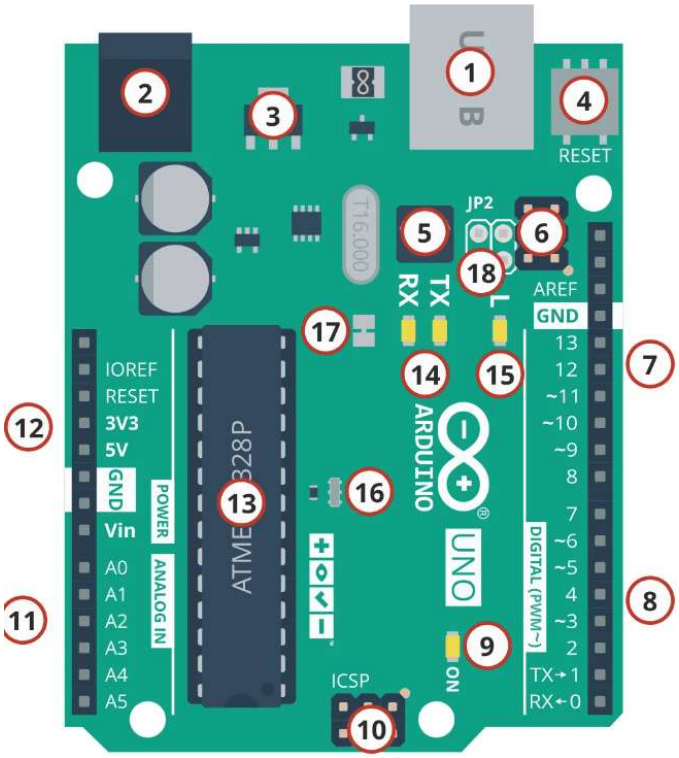
Wyposażenie sprzętowe:

- 8 bitowy mikrokontroler firmy **Atmel, AVR ATmega328** pracujący z częstotliwością **16 MHz**.
- **14 programowalnych cyfrowych wejść/wyjść**. Sześć z nich można używać jako wyjścia **PWM** (np. do sterowania silnikami), a kolejne 6 jako **analogowe wejścia**. Znajdziemy tam również sygnał resetu oraz zasilanie.

Arduino może być zasilane na kilka sposobów. Najpopularniejsze metody to:

1. Zasilanie przez przewód USB
2. Zasilanie przez zasilacz wtyczkowy (optymalnie 7V - 12V) lub baterie

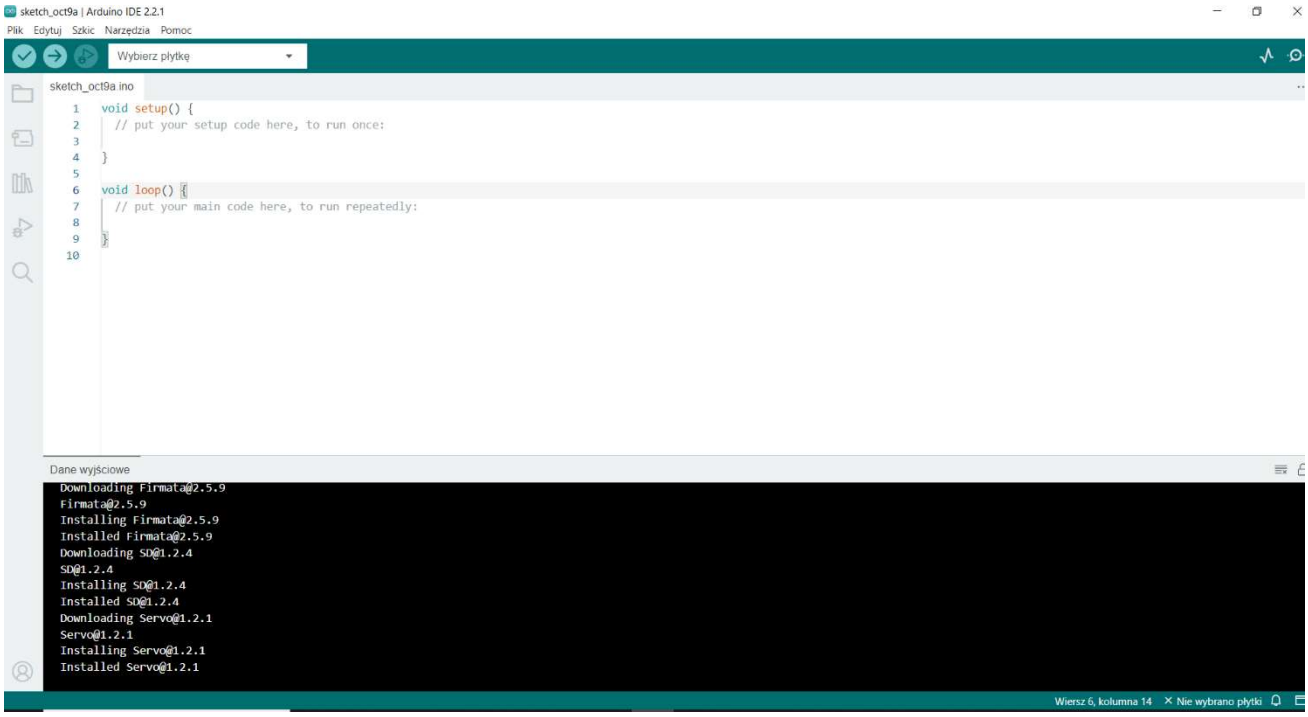
Graficzny układ najważniejszych elementów:



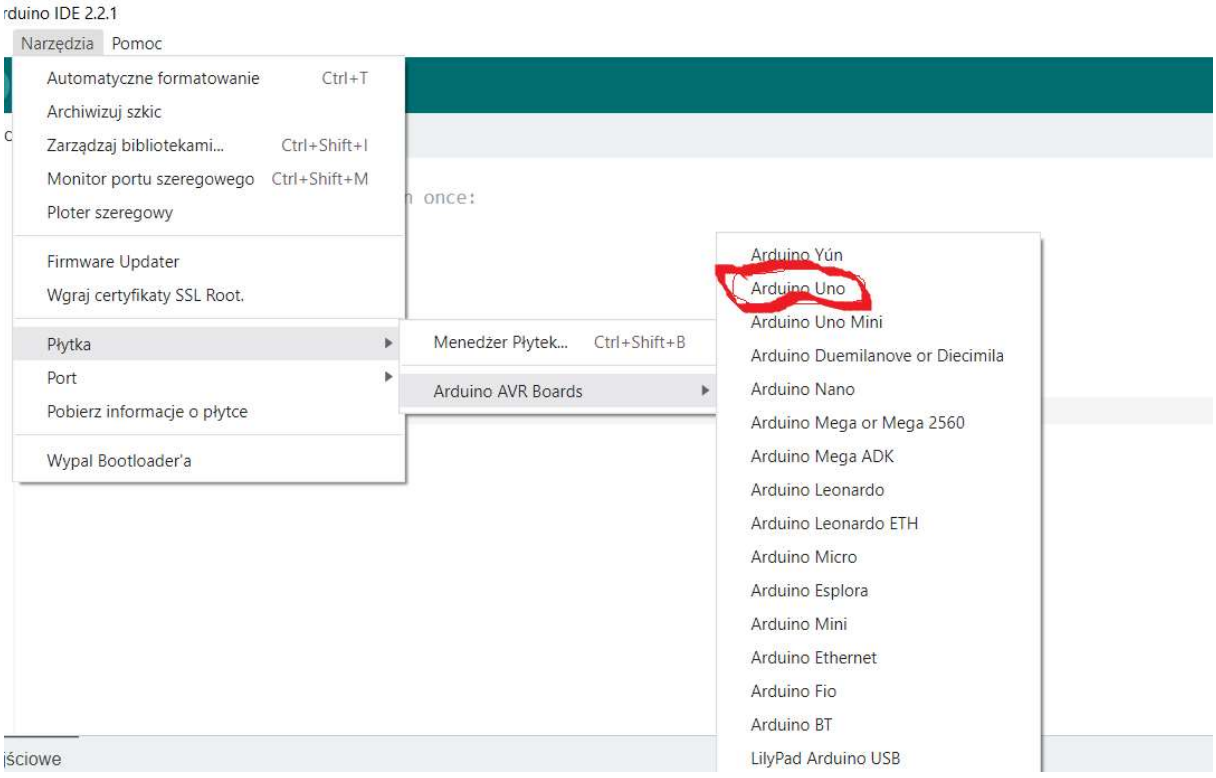
1. **Złącze USB** - wykorzystywane do zasilania, programowania oraz komunikacji z komputerem
2. **Złącze zasilania** (optymalnie 7V - 12V)
3. **Stabilizator napięcia** - napięcie wejściowe ze złącza nr 2 obniżane jest do 5V dzięki temu układowi
4. **Przycisk resetu** - resetuje płytkę Arduino
5. Mikrokontroler odpowiedzialny za komunikację z komputerem przez **USB**
6. Złącze programowania do mikrokontrolera z punktu 5.
7. Złącze sygnałowe*
8. Złącze sygnałowe*
9. **Dioda LED** sygnalizująca podłączenie napięcia do Arduino
10. Wyjście programatora dla mikrokontrolera z punktu 13.
11. Złącze sygnałowe*
12. Złącze zasilania*
13. **Serce Arduino**, główny mikrokontroler AVR ATmega328
14. **Diody LED** sygnalizujące transmisję do/z komputera
15. **Dioda LED** do dyspozycji użytkownika
16. **Rezonator ceramiczny** taktujący mikrokontroler (punkt 13) z częstotliwością 16MHz

17. **Zworka**, której przecięcie wyłącza automatyczne resetowanie Arduino.
18. Pola lutownicze z wyprowadzonymi sygnałami mikrokontrolera z punktu 5, używane ekstremalnie rzadko w bardzo specyficznych i niestandardowych sytuacjach.

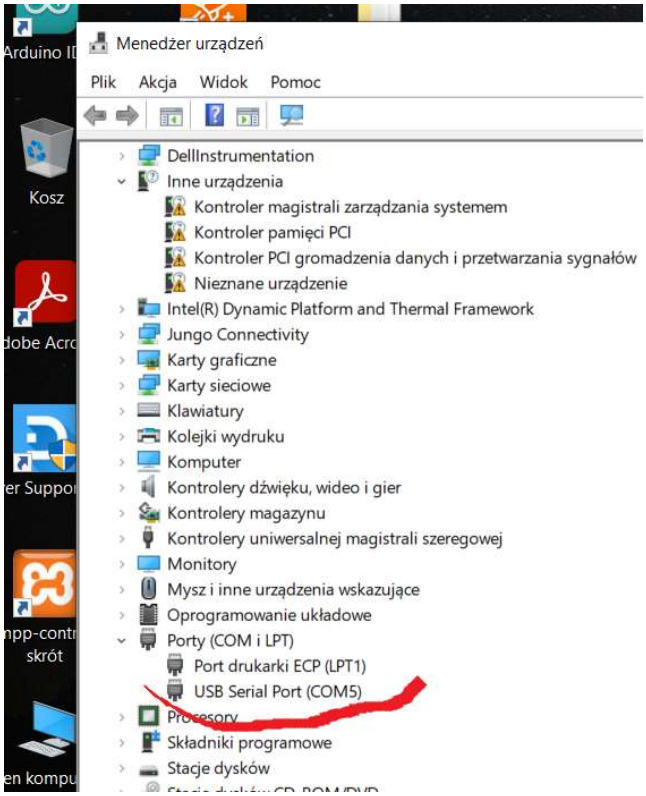
Budowa Arduino IDE

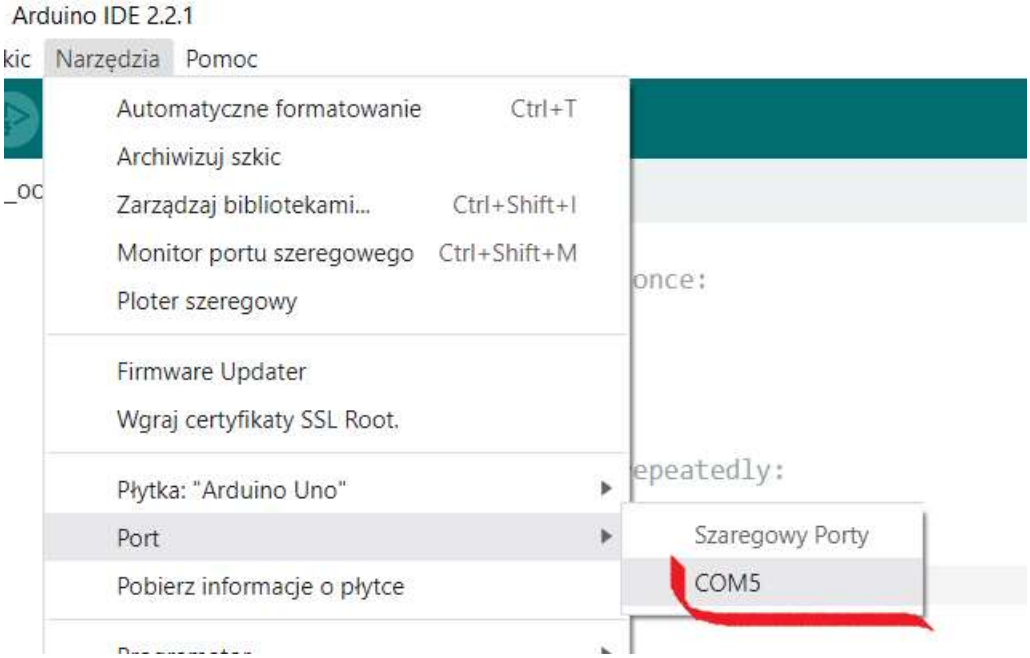


Po uruchomieniu programu należy połączyć płytkę (układ Andino UNO)



Sprawdzenie portu COM





Test sprawdzający działania:

Wgrajmy przykładowy program żeby sprawdzić czy wszystko działa np.

Plik > Przykłady > 01. Basics > Blink

Otworzy się osobne okienko z kodem programu.

Program musimy wgrać, odbywa się to w dwóch fazach:

- Kompilacja – Opcja „ weryfikuj” 

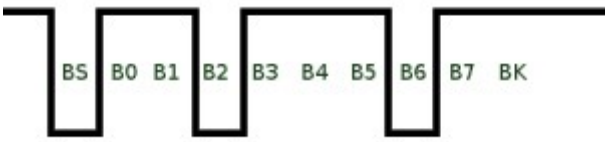
sprawdza i kompiluje napisany program.
- Wysyła program do Arduino UNO 

oraz sprawdza kompiluje.

Powinny zamigać diody opisane jako TX i RX - oznacza, że dana są przesyłane z/do komputera.

Krótko o UART - (Universal Asynchronous Receiver-Transmitter) to standardowy interfejs komunikacyjny do przesyłania danych między urządzeniami mikrokontrolerów, komputerów lub innych systemów cyfrowych.

Jego zasada działania opiera się na szeregowym wysłaniu ciągu bitów, które następnie składane są w informację. Pojedyncza ramka (w uproszczeniu bajt) danych jest nadawany w poniższej postaci:



Przykładowa ramka UART

Transmisja rozpoczyna się od **bitu startu**, zaznaczonego na rysunku jako **BS**. Zawsze jest to bit będący logicznym zerem. Następnie, zależnie od konfiguracji, następuje po sobie **7, 8 lub 9 bitów danych** (tutaj zaznaczone jako B0-B7), które są wysyłaną informacją. **Bit stopu** (zaznaczony tutaj jako bit BK) to bit będący logiczną jedynką - mówi o końcu transmisji.

Wykorzystując UART w Arduino interesują nas dwa piny:

- **Tx** do wysyłania danych (pin 1 w Arduino),
- **Rx** do odbierania danych (pin 0 w Arduino).

Aby transmisja przebiegała prawidłowo, w obu układach musi być ustawiona **ta sama prędkość** przesyłu danych - zwana jako **baud-rate**. Określa ilość transmitowanych bitów na sekundę. Często spotykane wartości to: *9600 oraz 115200*.

Komputer, z którym chcemy nawiązać łączność, musi być również **wyposażony w odpowiedni interfejs np**. portu szeregowego w standardzie RS-232, RS – 422, RS – 485 , USB (najczęściej w dzisiejszych czasach)

USB. Niestety zadanie to jest dość trudne. Dlatego często stosowane są konwertery UART <-> USB, które znacznie ułatwiają sprawę. Korzystając z Arduino nie trzeba się o to martwić. Konwerter taki został już **wbudowany w nasza płytkę.**

Łączność dzięki USB.

W związku z tym nie musimy nawet podłączać nigdzie pinów 0 oraz 1. Wystarczy połączenie komputera z Arduino poprzez kabel USB

Przykład 1:

Program który przesyła do/z Arduino przez port USB.

```
void setup(){

  Serial.begin(9600); //Ustawienie prędkości transmisji
  Serial.println("Witaj  w systemie wbudowanym !"); //Jednorazowe wysłanie tekstu
}

void loop() {

  delay(6000);
  Serial.println("Minelo 6 sekund!"); //Wysyłanie w pętli
}
```

Aby to sprawdzić wejdź : **Narzędzia->*Monitor Portu Szeregowego***

Pierwsze funkcje:

Serial.begin(speed), gdzie **speed** określa prędkość transmisji

Serial.println(val), gdzie **val** to ciąg znaków lub liczba.

delay(wartość w milisekundach), funkcja ta oznacza opóźnienie w programie,

Podstawy programowania:

Szkielet programu

W Arduino kod programu jest zbudowany z dwóch funkcji:

void setup() – umieszczamy tu instrukcje które wykonują się jeden raz, najczęściej zawiera pewne ustawienia pinów wejścia/wyjścia mikrokontrolera (akcje które mają się dziać tylko raz)

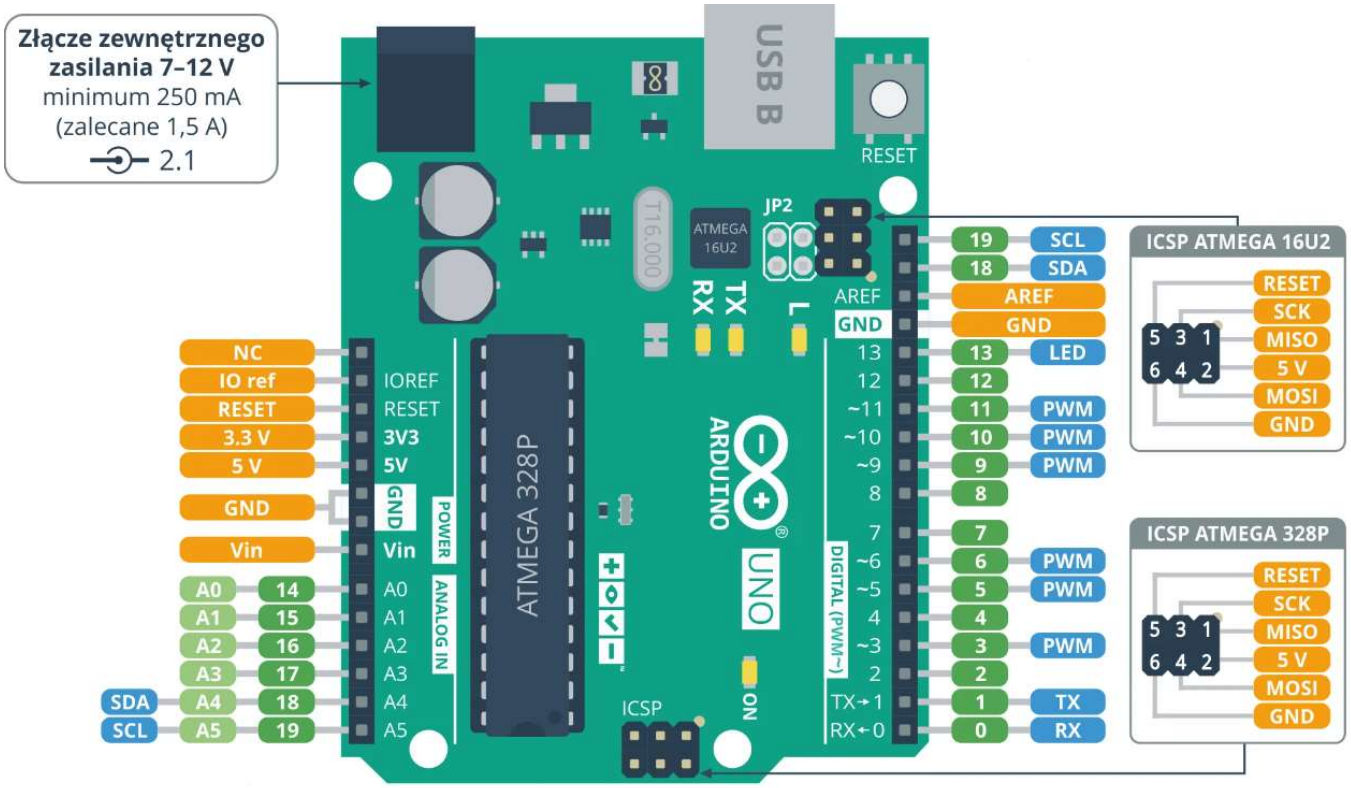
void loop() – umieszczamy właściwy kod który będzie wykonywał się cały czas **(w pętli)**

Są to funkcje dostarczane z bibliotek oczywiście możemy pisać funkcje samodzielnie.

Wyprowadzenia Arduino UNO

Korzystając ze specjalnych złącz, do Arduino można podłączyć elementy zewnętrzne takie jak np.: diody i przyciski. Zanim jednak do tego dojdziemy **musimy poznać opis wyprowadzeń**.

Najważniejsze wyprowadzenia Arduino UNO.



Ciemnozielony (nr od 0 do 19) uniwersalne, cyfrowe piny wejścia/wyjścia (I/O). Gdy będziemy wykorzystywać je w roli wyjść będziemy mogli na nich ustawić 0V (logiczne 0, stan niski) lub

5V (logiczne 1, stan wysoki). Gdy zostaną skonfigurowane w roli wejść będą mogły wykrywać połączenie pniu z 0V lub 5V.

Jasnozielony zaznaczone zostały wejścia analogowe (A0-A5). Są to wyjątkowe piny, które pozwalają na **pomiar napięcia** (w zakresie 0-5V). Jak widać numeracja tych wejść pokrywa się z pinami uniwersalnymi (numery od 14 do 19). Praca w trybie analogowym to dodatkowa funkcja tych pinów.

Niebieskie zostały alternatywne funkcje dla poszczególnych sygnałów. Oznacza to, że oprócz bycia standardowym wejściem lub wyjściem mogą one pełnić bardziej skomplikowane funkcje.:

- **SDA, SCL** - wyprowadzenia magistrali **I²C** to nazwa dwuprzewodowego interfejsu, służącego do przesyłania danych pomiędzy dwoma lub większą liczbą układów cyfrowych. podłączenie np. wyświetlacza LCD
- **TX, RX** - interfejs UART, wykorzystywany głównie do komunikacji z komputerem,
- **PWM** - wyprowadzenia, na których możliwe jest generowanie sygnału prostokątnego o zmiennym wypełnieniu. Bardzo przydatna funkcja np.: przy sterowaniu serwomechanizmami,
- **LED** - dioda świecąca, wbudowana na stałe w Arduino, która połączona jest z pinem nr 13.

Pomarańczowy to wyprowadzenia które nie są programowalne. Odpowiadają one głównie za zasilanie układu.

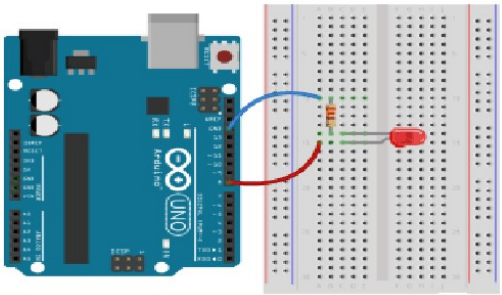
Przykłady 2.

Wyjścia

– zaświecenie diody LED

Program, który wykorzysta dowolny pin I/O -na którym można ustawić jeden z dwóch stanów (niski lub wysoki) 0V lub 5V zaświeci diodę

Patrz rysunek!!



Diodę łączymy szeregowo z rezystorem (330R). Następnie dłuższą nóżkę diody (anodę) łączymy z wyprowadzeniem nr 9. Drugą, przez rezystor z masą, którą znajdziemy w złączu zasilania (opisaną jako GND).

```
void setup() {
  pinMode(9, OUTPUT); //konfiguracja pinu 9 jako wyjście
  digitalWrite(9, HIGH); //
}

void loop() {
}
```

Funkcja **pinMode(Pin, Tryb)** umożliwia wybranie, czy dany pin jest wejściem, czy wyjściem. **Pin** może być liczbą całkowitą z zakresu od 0 do 13, zaś **Tryb** to:

- *INPUT,*
- *OUTPUT,*
- *INPUT_PULLUP*.

Funkcja **digitalWrite(Pin, Stan)**. **Stan** jest stanem logicznym, który może być **HIGH** bądź **LOW** (wysoki bądź niski).

Przykład 3.

- miganie diodą LED

W tym przypadku wykorzystamy funkcje opóźnienia - **delay.** Schemat połączeń jest taki sam jak w przykładzie 2. Zmiana nastąpi w kodzie

```
void setup() {
  pinMode(8, OUTPUT); //Konfiguracja pinu 8 jako wyjście
}

void loop() {
  digitalWrite(8, HIGH); //Włączenie diody
  delay(1000); //Odczekanie 1 sekundy
  digitalWrite(8, LOW); //Wyłączenie diody
}
```

W programie zostały dodane opóźnienia z pomocą funkcji **delay(Czas)** .

Do powyższego przykładu przykładu wprowadź zmienna typu całkowitego o dowolnej nazwie. Zmienna

Zadania:

Zadanie 1
Sprawdź przy jakiej **najmniejszej wartości** opóźnień będziesz w stanie zauważyć miganie diody! Co stanie się, gdy dioda będzie migała zbyt szybko?

Zadanie 2
Wybierz wolny pin i podłącz do niego drugą diodę. Napisz program, który będzie włączał obie diody LED. Następnie napisz program, w który sprawi, że obie diody będą migały **na zmianę.**

Zadanie 3.

Napisz program, w którym zostanie połączone 8 diod LED i uzyskamy na nich efekt przesuwającego się punktu świetlnego z prawej do lewej a następnie z powrotem po 5 sekundowej przerwie.

Wszystkie wyniku, wnioski oraz schematy umieść w sprawozdaniu.