Podstawowe funkcje sterowania pinami cyfrowymi

pinMode(8, OUTPUT**)**;//ustawienie końcówki jako wyjście
pinMode(8, INPUT**)**; // ustawienie końcówki jako wejście
pinMode(8, INPUT_PULLUP**)**; // ustawienie końcówki jako wejście
// z rezystorem podciągającym
digitalWrite(8, HIGH**)**; //ustawienie wyjścia w stan wysoki (1)
digitalWrite(8, LOW**)**; //ustawienie wyjścia w stan niski (0)

Uwaga: **dioda na wyjściu 13** świeci gdy na wyjście podamy 0.

Sterowanie diody za pomocą klawisza

Podłączmy klawisz i diodę do Arduino i programowo będziemy sterować zapalaniem diody. W tym przypadku można stworzyć wyłącznik czasowy, który wyłączy diodę po 4 sekundach.

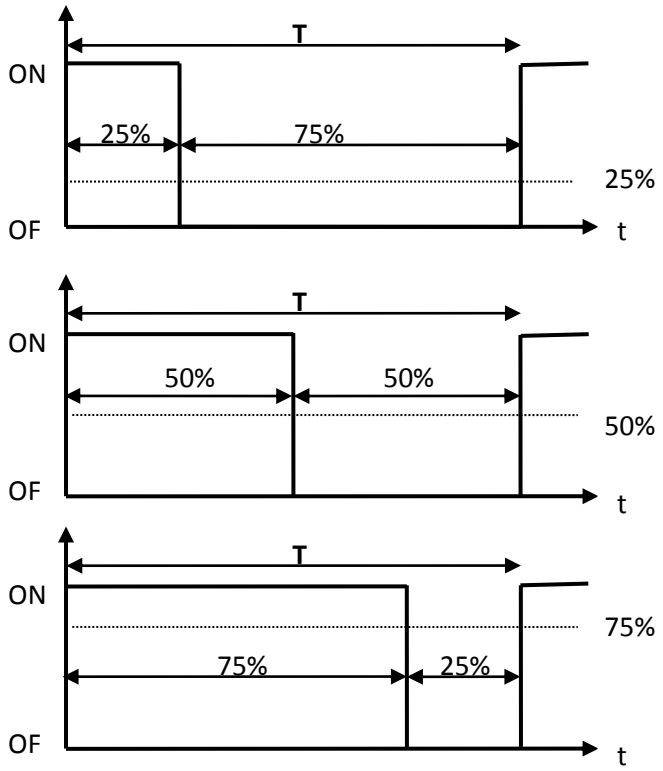
Przykład 1:

```
const  int  readPin = 8;
const  int  writePin = 11;


void  setup()
{
  pinMode(readPin, INPUT_PULLUP);
  pinMode(writePin, OUTPUT);
}

const  int  interval = 50;
const  int  maxLightMS = 4000;      // 4s
int  activated = 0;

void loop()
{
  boolean  active = !digitalRead(readPin);
  if (active)
    activated += interval;
  else
    activated = 0;

 if (active && (activated <= maxLightMS))
 {
   digitalWrite(writePin, HIGH);
 }
 else
 {
  digitalWrite(writePin, LOW);
 }
 delay(interval);
}
```

Sygnały PWM (Pulse Width Modulated)

Arduino nie może wytwarzać sygnałów analogowych lecz może generować sygnały prostokątne o zmienianym programowo współczynniku wypełnienia. Dzięki temu można wytwarzać sygnały wyjściowe o regulowanej wartości średniej napięcia. Może to być wykorzystane np. do regulacji jasności świecenia diody LED lub sterowania prędkością obrotową silnika prądu stałego.

Wpływ współczynnika wypełnienia (T_i/T) na wartość średnią napięcia



Do ustawienia współczynnika wypełnienia używamy funkcji

analogWrite(numerPinu, wspWypełnienia**)**;

gdzie wspWypełnienia może przyjmować wartości od 0 (0%) do 255 (100%), zaś numer pinu określa wyjście analogowe od 0 do 13

Przykłady programów zmieniających jasność świecenia diody korzystające z modulacji PWM

Przykład 2:

```
int  led = 11;
void setup()
{
  pinMode(led, OUTPUT);
}

void loop()
{
   int  value = 0;
   for (value = 0; value <255; value++)
   {
        analogWrite(led, value);
        delay(10);              // opóźnienie 10ms
   }
}
```

Przykład 3:

```
int  led = 11;       // dioda LED podłączona do pinu 11
int poziom = 0;    // określa jasność diody LED
int zmiana = 5;    // szybkość zmiany jasności diody LED

void setup()
{
   pinMode(led, OUTPUT);   // określenie końcówki 11 jako wyjścia
}

void loop()
{
  analogWrite(led, poziom);    // ustawienie jasności świecenia diody LED
  poziom = poziom + zmiana; // zmiana poziomu świecenia przy każdym cyklu

 // zmiana kierunku zmian jasności świecenia przy końcu przedziału:
 if (poziom == 0 || poziom == 255)
    zmiana = - zmiana ;
 delay(30);       // opóźnienie 30ms
}
```

Zadanie 1:

Sprawdzić możliwość sterowania poziomem świecenia diody za pomocą monitora portu szeregowego. Zmierzyć poziom napięcia na wyjściu analogowym układu przy różnych współczynnikach wypełnienia impulsu.

Uwaga: Do wprowadzenia liczby z monitora portu szeregowego wykorzystać funkcję ASCII2long() z następującego przykładu:

Przykład 4:

```
// Wczytywanie liczby w kodzie ASCII z portu szeregowego i zamiana na zmienną typu long
long liczba = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  liczba = ASCII2long();

  Serial.print("Wprowadzono: ");
  Serial.println(liczba);
  Serial.print(liczba);
  Serial.print(" razy 2 to ");
  liczba = liczba * 2;
  Serial.println(liczba);
}

long ASCII2long()
{
  long a;
  long n = 0; // wstępne zerowanie wczytywanej zmiennej
  Serial.flush(); // oczyszczenie bufora z ewentualnych wcześniejszych znaków
  while (Serial.available() == 0)
  {
    // oczekiwanie na wprowadzone znaki;
    // po wprowadzeniu znaków do bufora funkcja Serial.available()
    // zwraca liczbę znaków oczekujących na przetworzenie
  }
  // każdy przetwarzany znak powoduje:
  while (Serial.available() > 0)
  {
    // pomnożenie zawartości n przez 10
    n = n * 10;
    // odczyt kolejnego znaku (cyfry) z bufora, skorygowanie do wartości binarnej cyfry
    a = Serial.read() - '0';
    // i dodanie do zmiennej n
    n = n + a;
    // krótkie oczekiwanie na kolejny znak
    delay(5);
  }
  return n;
}
```

Zadanie 2:

Napisać program, który przesyła do Arduino przez monitor portu szeregowego trzy liczby całkowite i sprawdza czy mogą one być długościami trójkąta prostokątnego.

Wejścia analogowe

Układ Arduino Mega posiada 16 wejść analogowych A0-A15, które mogą służyć do odczytu wartości podawanego na nie napięcia. Wykorzystywany jest w tym celu przetwornik analogowo-cyfrowy 10 bitowy.
Do odczytu wartości napięcia służy funkcja **analogRead(numer pinu);**

Przykład 5:

```
const  int sensorPin = A0;
int  analogValue = analogRead(sensor pin);
Serial.print(”wartość analogowa = ”);
Serial.println(analogValue, DEC);
```

lub w postaci liczby zmiennoprzecinkowej:

```
float voltage = 0.0048 * analogValue;
```

Wartość mierzonych napięć standardowo wynosi 0 – 5V lecz może być zmieniony za pomocą funkcji:

analogReference(typ);

gdzie parametr typ może przyjmować wartości:
INTERNAL – wewnętrzne źródło 1,1 V
EXTERNAL – wartość podawana na końcówkę AREF np. 3,3 V
DEFAULT - napięcie +5 V

Przykład 6:

```
int analogValue = 0;
int  ledPin = 11;
void setup(){
// nie trzeba definiować trybu pracy wejść / wyjść analogowych
}

void loop(){
  analogValue  = analogRead(A0) / 4;
  analogWrite(ledPin, analogValue);
  delay(20);
}
```

Zadanie 3:

Podłączyć potencjometr 10 kΩ do płytki Arduino i sprawdzić działanie powyższego przykładu (regulujemy za pomocą potencjometru poziom świecenia diody).
Wyświetlać w oknie monitora portu szeregowego aktualną wartość napięcia podawanego z potencjometru.

Przykład 7:

Zapoznać się z opisem funkcji map() i constrain()

```
// Wczytanie wartości z portu analogowego  i wysłanie
// po przeskalowaniu do portu szeregowego
// Dioda LED świeci z częstotliwością zależną od ustawienia potencjometru

const int ledPin = 13; // dioda LED podłączona do pinu 13
const int sensorPin = 0; // suwak potencjometru (sensor) do wejścia analogowego A0

// następne dwie linie określają minimalny i maksymalny czas świecenia diody
const int minCzas = 100; // minimum czasu świecenia
const int maxCzas = 1000; // maximum czasu świecenia

void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
    int czas = analogRead(sensorPin); // odczyt napięcia na wejściu analogowym

    czas = map(czas, 200,800,minCzas, maxCzas); // przeskalowanie
    czas = constrain(czas, minCzas, maxCzas); // ograniczenie
    Serial.println(czas);
    digitalWrite(ledPin, HIGH);
    delay(czas); // świeci przez określony czas
    digitalWrite(ledPin, LOW);
    delay(czas); // zgaszona przez określony czas
}
```

Funkcja **map**(x, in_min, in_max, out_min, out_max)

Przeskalowuje wartość do podanego zakresu

```
long map(long x, long in_min, long in_max, long out_min, long out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Funkcja **constrain**(x, a, b)

Ogranicza wartość do podanego zakresu.

Parametry

x: wartość ograniczana, dowolnego typu

a: dolny zakres ograniczania, dowolnego typu

b: górny zakres ograniczania, dowolnego typu

Zwraca

x: jeśli x jest pomiędzy a i b

a: jeśli x jest mniejsze niż a

b: jeśli x jest większe niż b