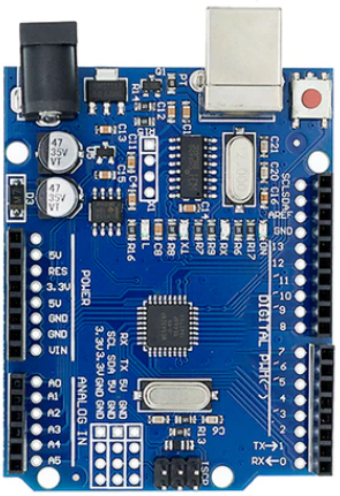
Poradnik programowania arduino dla zestawu pierwszego “Zestaw dla początkujących”

Link do projektów na GITHUB:
https://github.com/EssaTechnology/zestaw-dla-poczatkujacych-1

Piotr Prabucki EssaTech

1.Opis działania elementów zestawu

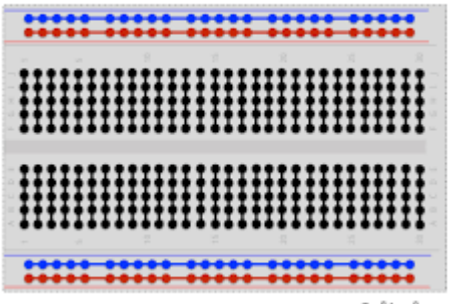


1.1 Arduino Uno R3 - ATmega328P

Trzecia wersja kultowej płytki Arduino Uno wyposażona w mikrokontroler ATmega328P zasilana napięciem od 7V do 12V. Więcej w dziale [Wprowadzenie do arduino]

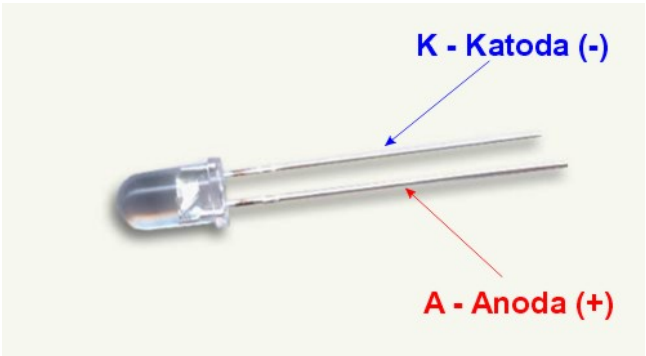
1.2 Płytka stykowa

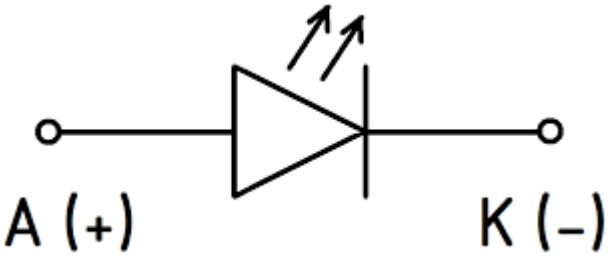
Płytka stykowa nazywana również prototypową umożliwia łatwe łączenie elementów bez konieczności lutowania układów, dzięki temu możliwe jest szybsze testowanie układów i wprowadzanie zmian w przypadku ewentualnych błędów. Poniżej przedstawiono w jaki sposób połączone są wejścia w płytce prototypowej:



1.3 Diody LED

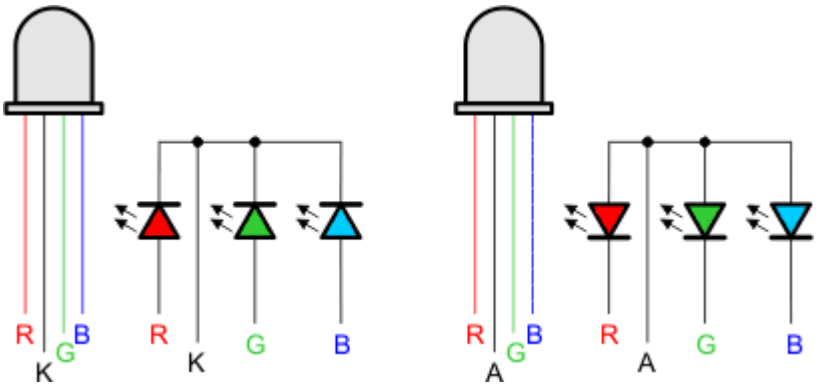
Głównym zadaniem Diod LED jest przetwarzanie energii elektrycznej na światło. Jednak tak samo jak zwykłe diody umożliwiają one przepływ prądu tylko w jedną stronę, dlatego ważne jest sprawdzanie w jaki sposób należy podłączyć diodę led do układu. Aby odróżnić katodę(-) diody od anody(+) diody wystarczy spojrzeć na długość wyprowadzeń, dłuższa “nóżka” to anoda a krótsza katoda. Dodatkowo diody wymagają ograniczenia prądu dopływającego do diody aby uniknąć ich przypalenia, w tym celu stosuje się rezystory o których opowiedziane jest w dalszej części poradnika





1.4 Diody wielokolorowe

Zasada działanie diod wielokolorowych jest taka sama jak zwykłych jednak sprawa komplikuje się jeżeli chodzi o podłączenie takiej diody. Wyróżniamy dwie wersje diod wielokolorowych, te ze wspólną anodą(+) oraz te ze wspólną katodą(-). Aby zrozumieć działanie diody wielokolorowej wystarczy spojrzeć na jej schemat:



jak widać dioda wielokolorowa zawiera w sobie po jednej diodzie z każdego koloru, w naszym zestawie zostały zastosowane diody ze wspólną katodą do oznacza że sterujemy napięciem chodzącym do diody.Tak samo jak przy zwykłych diodach led wymagane jest ograniczenie prądu wchodzącego do diody.

1.5 Rezystory

Zadaniem rezystorów jest zmniejszenie natężenia prądu w układzie elektronicznym wynika to z jednego z podstawowych praw, prawa ohma:

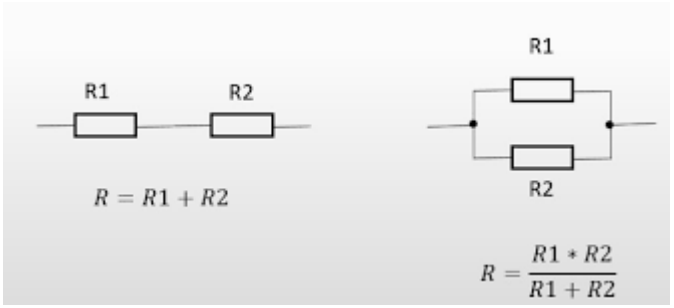
U = R*I

U to napięcie w układzie, w naszym przypadku 5V, z takim napięciem arduino zasila odbiorniki.
R to rezystancja układu zwiększamy ją przy pomocy rezystorów
I to natężenie prądu w układzie
powyższy wzór można przekształcić do postaci:

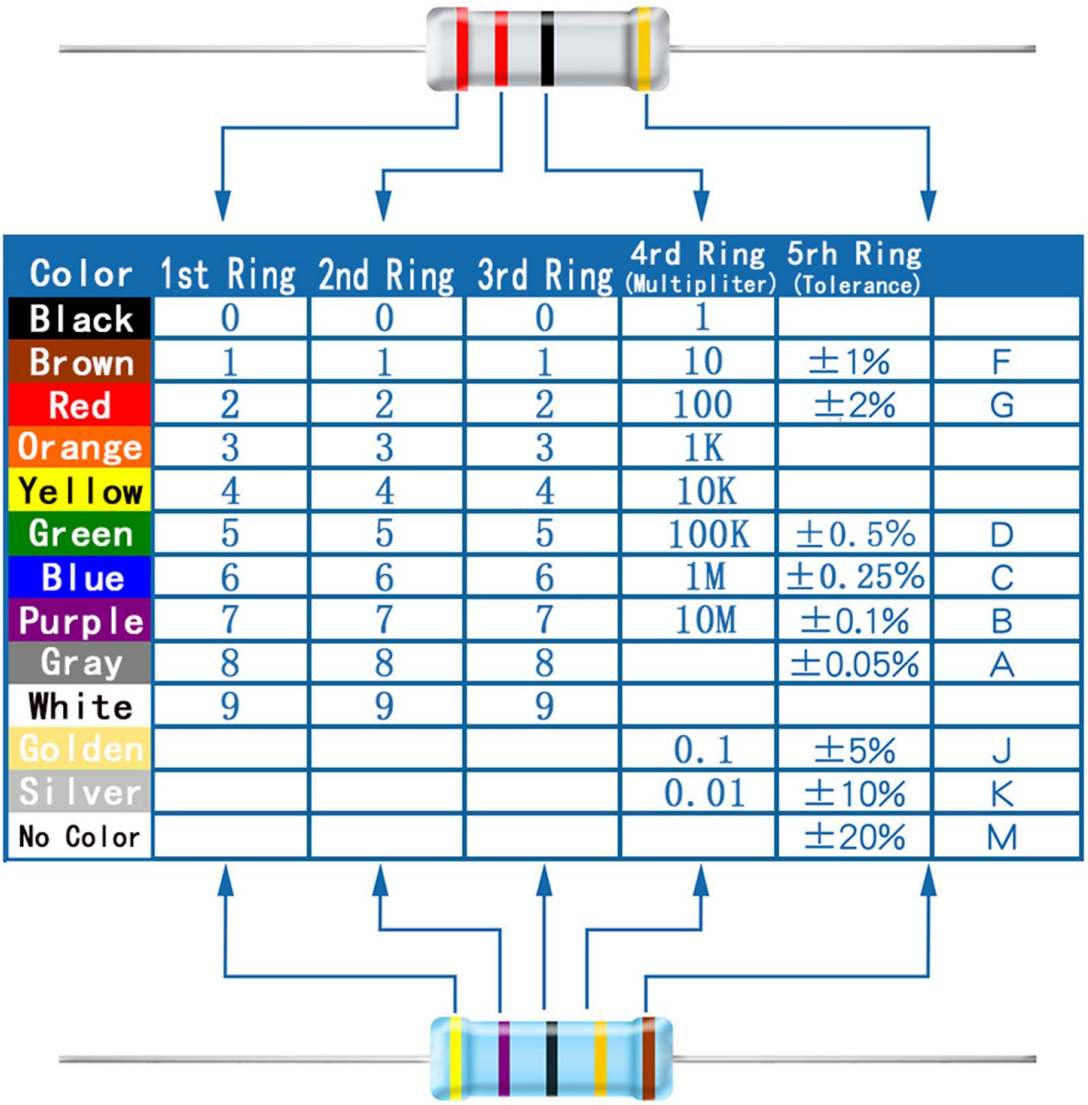
I=U/R lub R=U/I

Dzięki temu możemy obliczyć jakie natężenie będzie panowało w układzie po zwiększeniu rezystancji. A więc zakładając że nasz układ początkowo ma 5Ω wiedząc że napięcie w układzie jest równe 5V możemyobliczyć natężenie w układzie: 5/5=1A. załóżmy jednak że nasza dioda ma maksymalny prąd równy 0.1A, co możemy zrobić w takim przypadku? zwiększyć rezystancję układu poprzez dołożenie rezystora. po dołożeniu rezystora 50Ω natężenie prądu spadnie do 5/50 = 0.1A.

Dodatkowo rezystory można ze sobą łączyć szeregowo lub równolegle. dzięki temu jeżeli nie mamy rezystora 50Ω ale mamy dwa rezystory 25Ω możemy połączyć je szeregowo dzięki czmu uzyskamy łącznie 50Ω. Jeżeli mamy dwa rezystory 100Ω możemy połączyć je równolegle dzięki czemuuzyskamy 50Ω



Aby odczytać rezystancję danego rezystora wystarczy spojrzeć na tabele rezystancji:



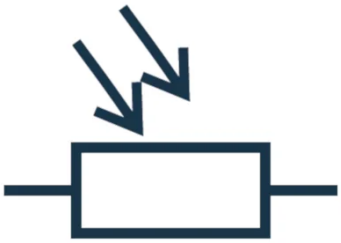
Color	1st Ring	2nd Ring	3rd Ring	4rd Ring (Multipliter)	5rh Ring (Tolerance)	
Black	0	0	0	1		
Brown	1	1	1	10	±1%	F
Red	2	2	2	100	±2%	G
Orange	3	3	3	1K		
Yellow	4	4	4	10K		
Green	5	5	5	100K	±0.5%	D
Blue	6	6	6	1M	±0.25%	C
Purple	7	7	7	10M	±0.1%	B
Gray	8	8	8		±0.05%	A
White	9	9	9			
Golden				0.1	±5%	J
Silver				0.01	±10%	K
No Color					±20%	M

Dla przykładu rezystor o rezystancji 10kΩ(1000Ω) i tolerancji +/-5% będzie miał kolory:

Brązowy -1
czarny-0
pomarańczowy 1K
złoty - +/-5%
10 * 1000 = 10 000 = 10KΩ

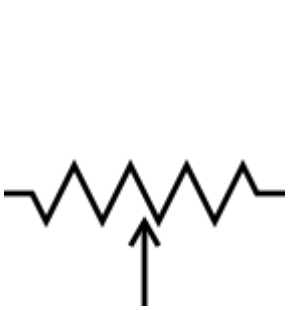
1.6 Fotorezystory

Fotorezystory zmieniają swoją rezystancję w zależności od ilości światła które na nie pada



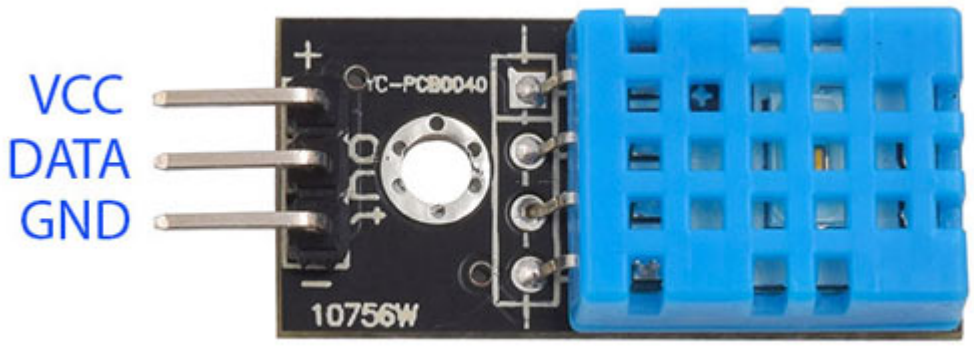
1.7 Potencjometr

Potencjometry zmieniają swoją rezystancję w zależności od położenia pokrętła



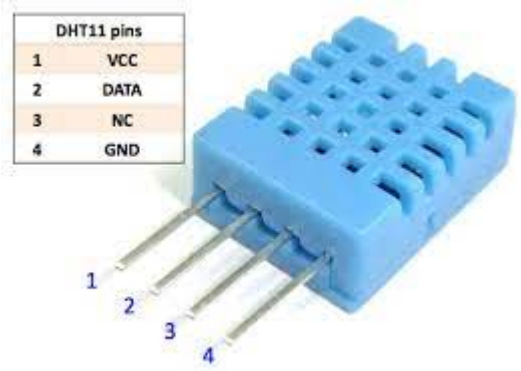
1.8 czujnik DHT11

Czujnik DHT11 umożliwia pomiar temperatury powietrza w zakresie od 0C do 50C oraz wilgotności od 20% do 90% RH.



VCC - inaczej anoda czyli zasilanie
GND - Katoda czyli “-” układu
DATA - złącze którym czujnik przesyła do mikrokontrolera wszystkie informacje

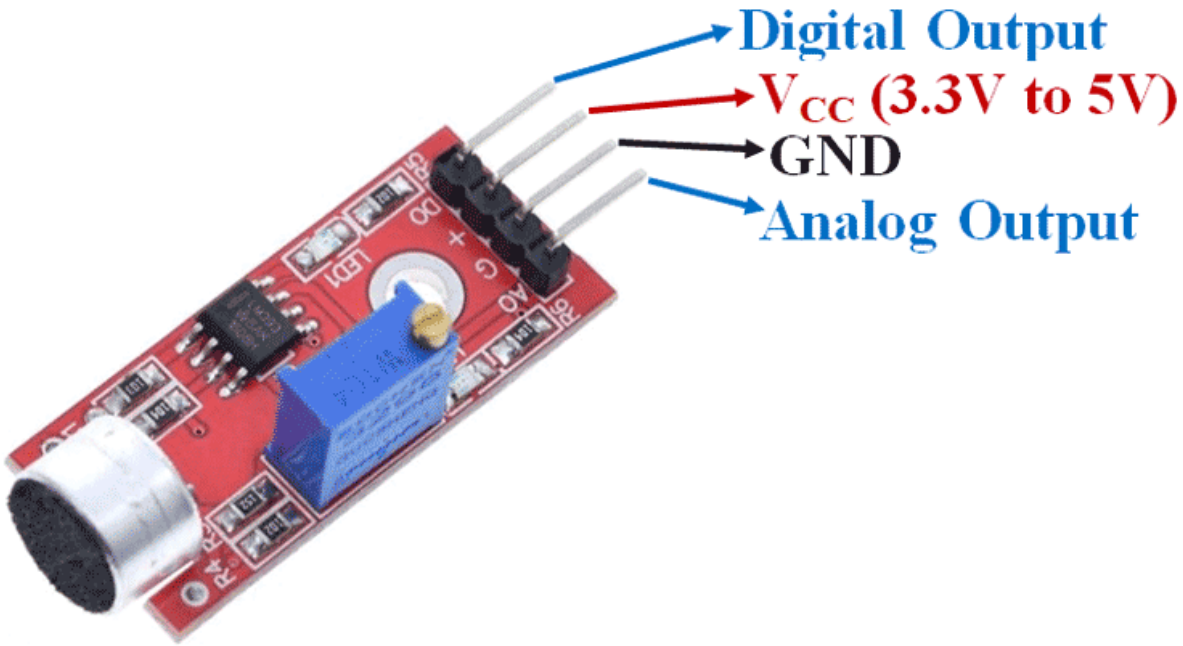
W wersji czujnika bez płytki widoczne są 4 złącza:



złącze 3 jest oznaczone NC z angielskiego Not Connected(nie podłączony) nie pełni ono żadnej funkcji

1.9 Czujnik dźwięki LM393

czujnik pozwala na pomiar natężenia dźwięku w pomieszczeniu



VCC- zasilanie
GND- masa/katoda(-)
Analog Output - wyjście analogowe(więcej w części 2.Wprowadzenie do arduino)

Digital Output - wyjście cyfrowe(więcej w części 2.Wprowadzenie do arduino)

układ posiada potencjometr(niebieskie “pudełko” ze śrubą) który poprzez zmianę swojej rezystancji ustala poziom hałasu który aktywuje czujnik. Układ oparty jest na mikrokontrolerze LM393.

1.10 Tact switch

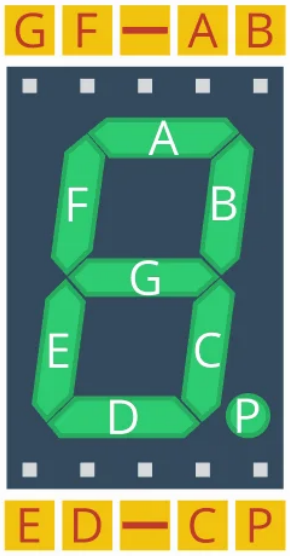
Tact switch czyli zwyczajnie przyciski dostępne są w dwóch odmianach 2 oraz 4 pinowej. W switchach 4 pin piny połączone są parami. Dzięki 4 pinom zapewniona jest większa stabilność po przylutowaniu, jednak 4 pinowe switche są trudniejsze do montażu na płytce prototypowej dlatego w naszym zestawie znajdują się wersje 2-pinowe

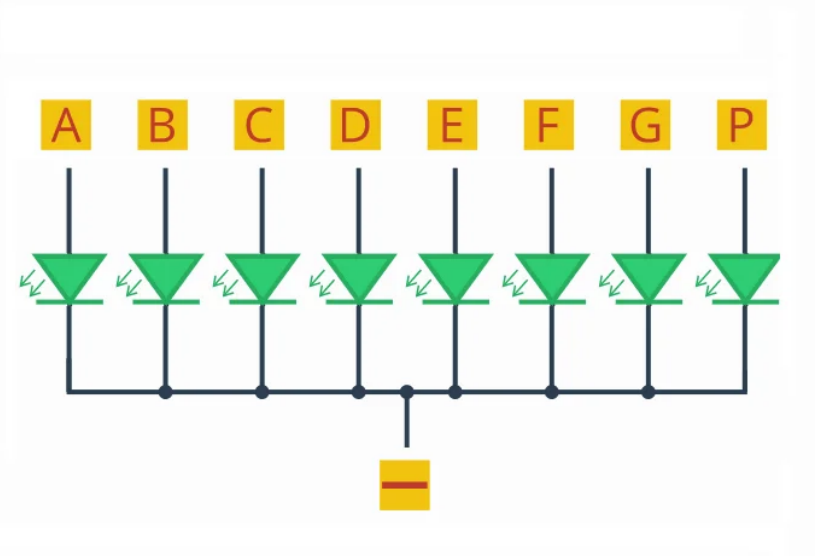
1.11 Buzzer

Jego zadaniem jest zamiana energii elektrycznej na fale dźwiękowe. Odróżniamy dwie wersje buzzerów, z generatorem oraz bez. W wersji z generatorem wystarczy podłączyć napięcie stałe do elementu aby ten zaczął wydawać dźwięk. w wersji bez generatora sprawy się komplikują, należy podawać do niego napięcie z częstotliwością taką jaki dźwięk chcemy uzyskać. Wersja bez generatora umożliwia za to zmianę częstotliwości dźwięku. W wersji z generatorem ważne jest prawidłowe podłączenie elementu, w przeciwnym razie może się on uszkodzić

1.12 Wyświetlacz Led

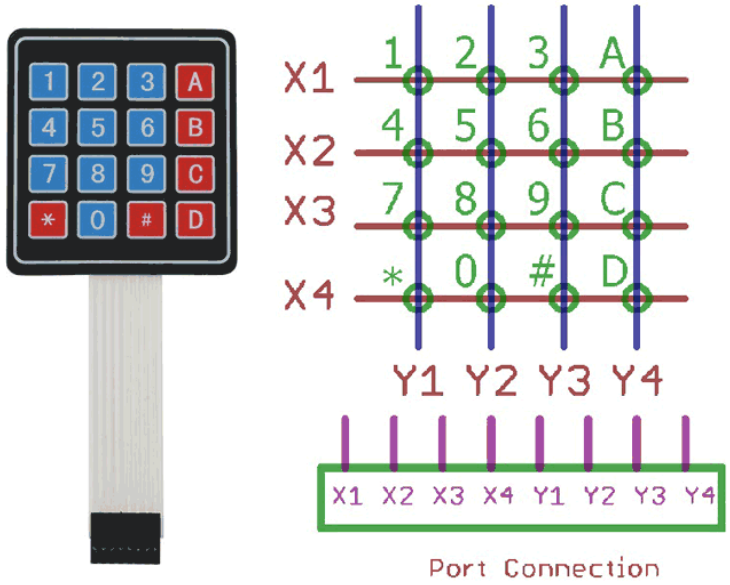
Podobnie jak dioda wielokolorowa zawiera wiele diod led, należy zastosować rezystor na każdej z nich aby uniemożliwić ich uszkodzenie.





1.13 Klawiatura numeryczna 4x4

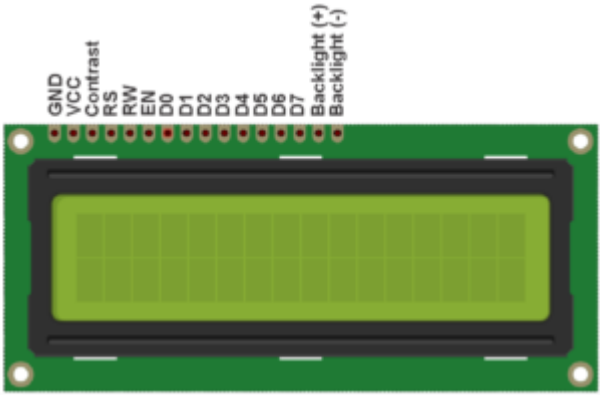
Jest ona zbudowana z wielu przycisków połączonych w następujący sposób:



naciśnięcie przycisku 1 powoduje zwarcie pomiędzy X1 i Y1, naciśnięcie 2 powoduje zwarcie X1 i Y2, naciśnięcie 4 powoduje zwarcie między X2 a Y1 i tak dalej. Jedyną wadą tego typu klawiatur jest zajmowanie wielu portów w arduino.

1.14 Wyświetlacz LCD

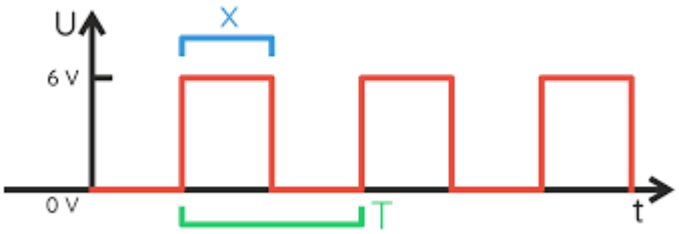
Wyświetlacz lcd pozwala na wyświetlanie dowolnych znaków poprzez podświetlenie pikseli z których zbudowane są segmenty, najczęściej są ułożone w dwa rzędy po 16 segmentów.



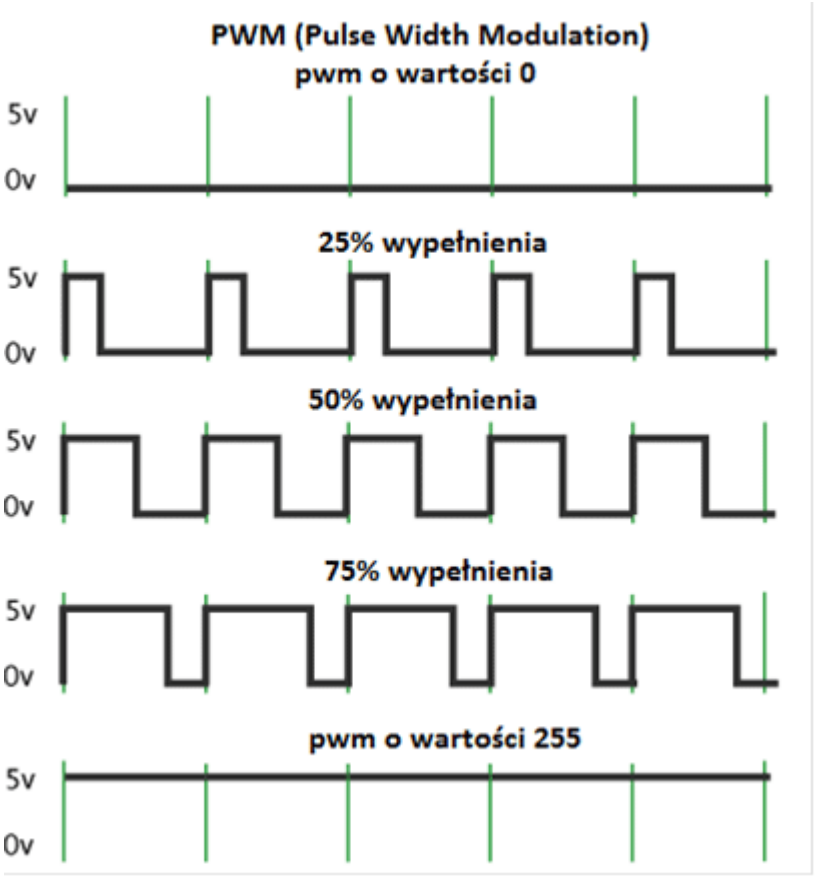
GND - masa
Vcc - zasilanie, 5V
Contrast - regulacja kontrastu
RS - wybór rejestrów (komenda, dane)
RW - wybór opcji odczyt/zapis
E - zezwolenie na zapis do rejestrów
D0 - dane
D1 - dane
D2 - dane
D3 - dane
D4 - dane
D5 - dane
D6 - dane
D7 - dane
Vpod - zasilanie podświetlenia
GNDpod - masa podświetlenia

1.15 Serwo

serwo pozwala na precyzyjne poruszanie elementami. zbudowane jest z silnika, przekładni,potencjometru oraz układu elektronicznego który steruje pracą silnika. Serwerm sterujemy poprzez zadawanie mu kąta na który ma się ustawić, robimy to poprzez sygnał pwm.



serwo oblicza zadany kąt poprzez obliczenie stosunku długości okresu X do okresu T, stosunek ten nazywamy wypełnieniem sygnały PWM.



1.16 Czytnik RFID

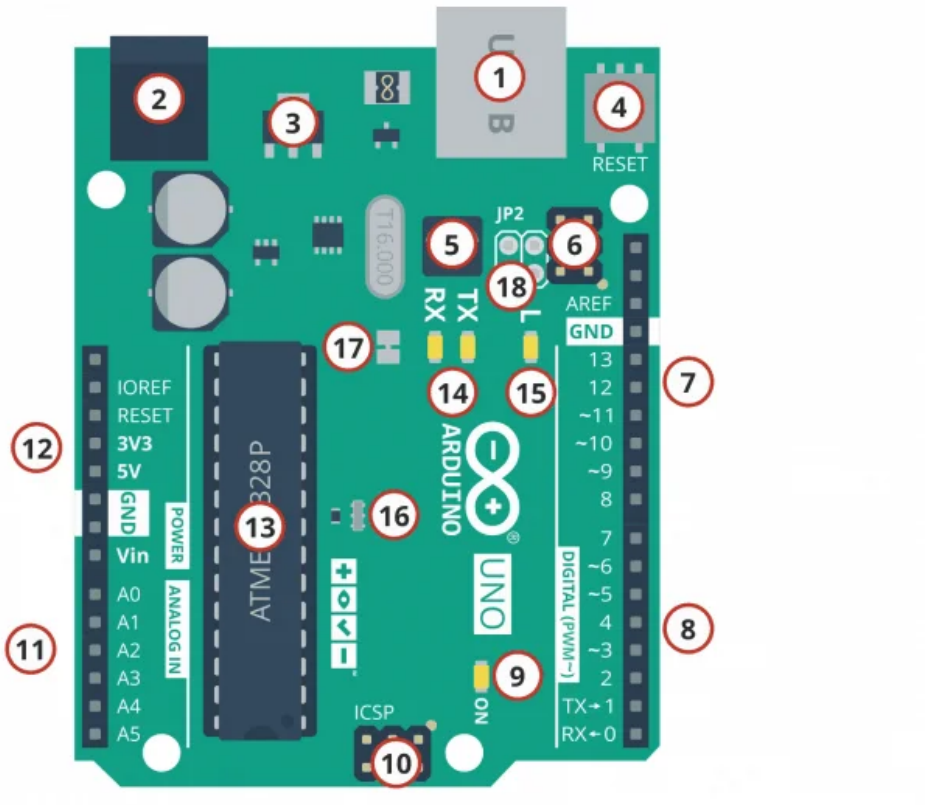
Technologia RFID pozwala na przesyłanie danych na małe dystanse. Korzystamy z niej na przykład przy otwieraniu drzwi kartą. W odbiornik wbudowana jest cewka która rozsiewa z daną częstotliwością fale elektromagnetyczne. W karcie umieszczony jest nadajnik który zostaje zasilony tymi falami i odsyła swój własny sygnał tym razem zawierający dane zapisane na karcie.

Pin 1 : VCC
Pin 2 : RST
Pin 3 : GND
Pin 4 : IRQ
Pin 5 : MISO/SCL/TX
Pin 6 : MOSI
Pin 7 : SCK
Pin 8 : SS/SDA/RX

VCC-zasilanie, 5V
RST- Reset
GND-masa
IRQ-powiadamia kontroler o tym, że tag znajduje się w pobliżu czytnika
MISO-przekazywanie danych
MOSI-przekazywanie danych
SCK-zegar
SS-przekazywanie danych

2.Wprowadzenie do arduino

Arduino zostało wyposażone w mikroprocesor ATmega 328p który w przeciwieństwie do wersji 328 jest mniejszy i pobiera mniej mocy zachowując te same parametry oraz częstotliwość na poziomie 16MHz co oznacza że może wykonać 16 milionów operacji w każdej sekundzie. Po bokach płytki znajdują się wyprowadzenia najważniejszych sygnałów dla płytki. Znajduje się tam 14 wyjść/wejść cyfrowych z czego 6 może zostać wykorzystane jako PWM a kolejne 6 jako wyjścia analogowe.



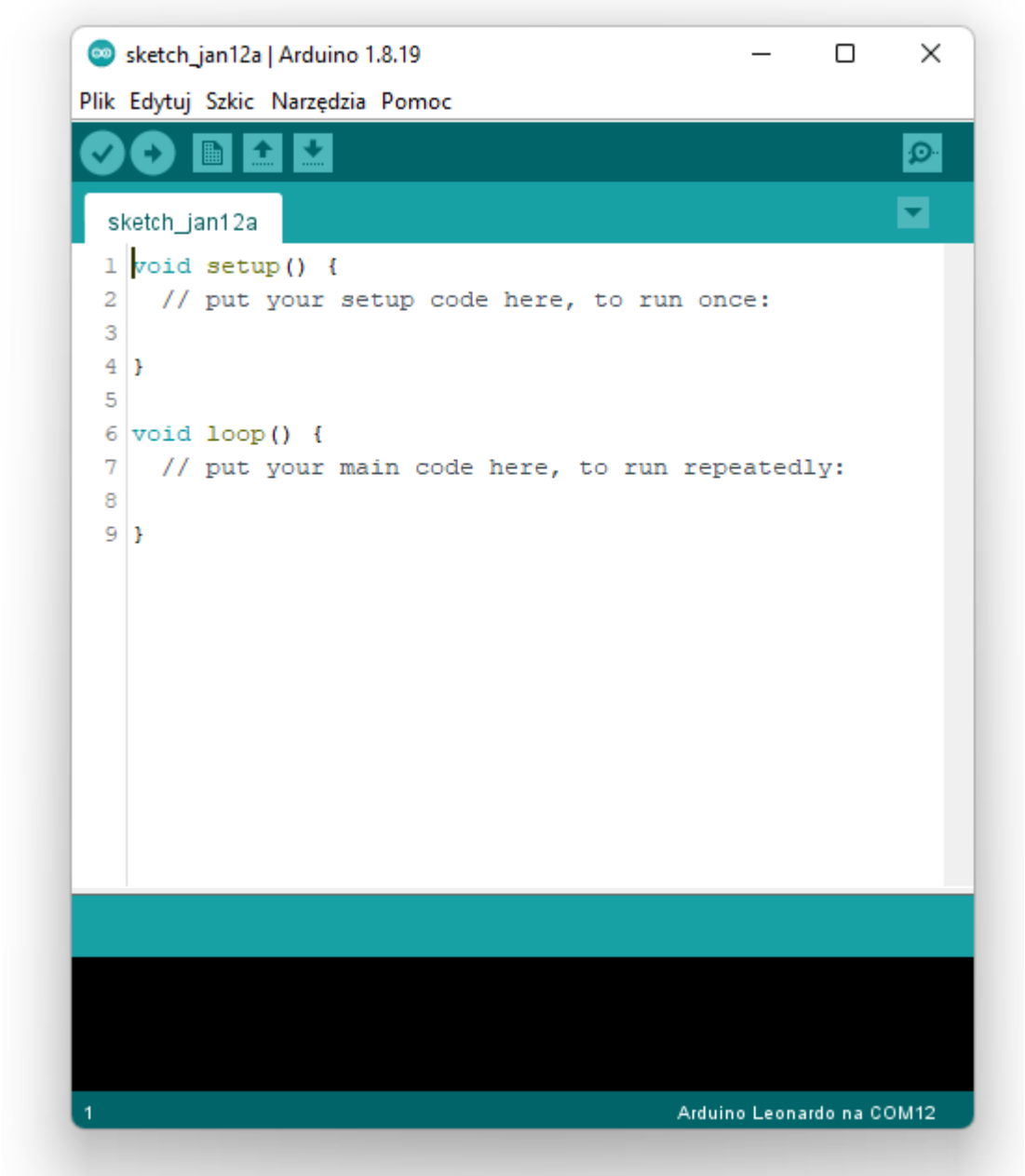
1.Złącze USB - wykorzystywane zapisywania programów na płytce oraz zasilania jej
2.Złącze zasilania 7V-12V
3.Stabilizator napięcia- zmniejsza napięcie do 5V
4.reset - odłącza zasilanie od płytki
5.Mikrokontroler komunikujący się z komputerem
6.Złącze programowania
7.Złącza cyfrowe
8.Złącza cyfrowe PWM
9.Sygnalizacja pracy arduino
10.Wyjście programatora
11.Złącza analogowe
12.Złącza zasilania
13.Mikrokontroler ATmega328p
14.Dioda sygnalizująca komunikację z komputerem
15.Dioda do dyspozycji użytkownika
16.Rezonator ceramiczny
17.Zworka której przerwanie wyłącza automatyczne resetowanie arduino
18.Pola lutownicze

3.Arduino IDE

3.1 Instalacja środowiska

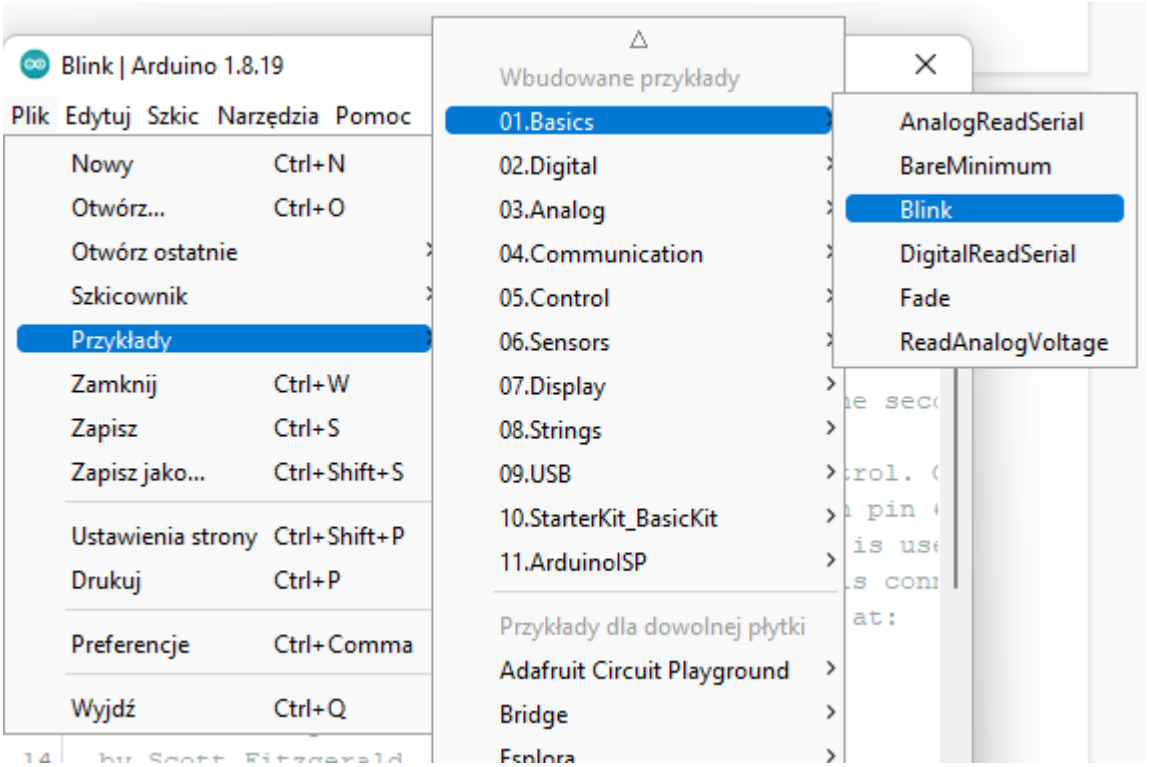
1.Pobieramy najnowszą wersję darmowego środowiska z oficjalnej strony Arduino IDE.
2. Uruchamiamy instalator i akceptujemy Umowę licencyjną

3.następnie klikamy Dalej, wybieramy gdzie chcemy zainstalować środowisko i naciskamy “instaluj”.
4. po kilku minutach Arduino IDE powinno być zainstalowane i gotowe do pracy.

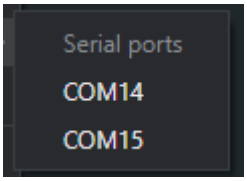


3.2 Konfiguracja Arduino IDE

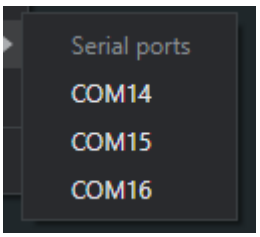
Uruchamiamy edytor, automatycznie powinien uruchomić się nowy plik, wchodzimy w zakładkę Plik->przykłady->basic-Blink



Powinno otworzyć się nowe okno z programem Blink.ino.
W zakładce Tools znajdują się dwie opcje które musimy ustawić:
Board oraz Port, Board to płytka na której pracujemy, w naszym przypadku arduino uno, ustawienie portu może sprawiać trochę więcej problemów, aby sprawdzić na jakim porcie podpięliśmy nasze arduino wystarczy sprawdzić dostępne porty przed wpięcie płytki:



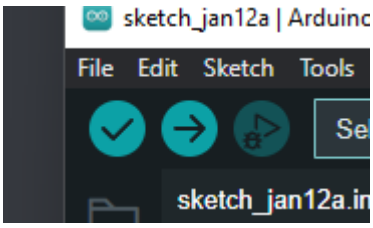
a następnie wpiąć płytkę:



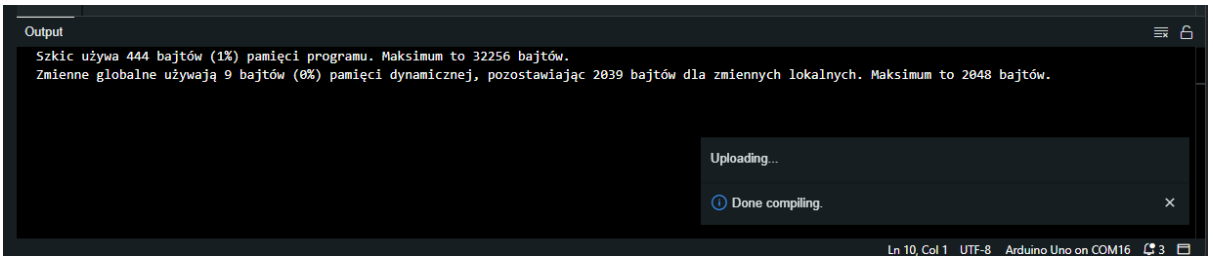
pojawił się dodatkowy port COM 16 a więc to najprawdopodobniej nasza płytka.

3.3 Pierwsze uruchomienie arduino

Teraz kiedy mamy już wybrany program, płytkę możemy przejść do wgrywania programy do mikrokontrolera w tym celu używamy przycisków w lewym górnym narożniku



pierwszy z nich służy do kompilowania programu czyli zamianie programu na język binarny oraz sprawdzeniu ewentualnych błędów
drugi z nich służy do skompilowania programu oraz wgrania go na płytkę. Wybieramy ten drugi i czekamy na zakończenie procesu kompilacji oraz wgrywania, w konsoli na dole pojawiają się informacje odnośnie całego procesu



po zakończeniu wgrywania dioda oznaczona “L” powinna zacząć migać z częstotliwością 0.5HZ

4. Podstawy programowania Arduino

4.1 Zmienne

Wyróżniamy 11 typów danych, zostały one przedstawione w poniższej tabeli:

void	Określa pusty zestaw wartości i jest używany tylko do deklarowania funkcji	void setup( )
		{
		. //
		}

int	LIczby całkowite do 2^{15}	2147483647, -2147483647, 8, 5
char	Pojedyncze znaki, możliwy jest zapis poprzez liczbę(numer znaku)	‘B’, ‘C’, 56, ‘5’, ‘8’, 4
float	liczby zmiennoprzecinkowe do 3.4028235E+38	3.123, 5.2, 4.0, 0.0, -8.4298
double	bardziej precyzyjne liczby zmiennoprzecinkowe, jest dwa razy większy niż float	3.123, 572627.2, 4.0, 0.0, -88263836.4298
unsigned int	przechowuje liczby całkowite tylko dodatnie do 65535	3.12314, 2387, 0
short	przechowuje liczby całkowite do 32767, jest dwa razy mniejszy niż unsigned int	12, -12, 32768, -32768
long	większe liczby całkowite do 2147483648	2147483648, -2147483648, 8, 0 ,-4 ,1
unsigned long	nie przechowuje liczb ujemnych	2147483648, 8, 0, 1
byte	przechowuje wartości od 0 do 255	0, 16, 56, 255
word	liczby od 0 do 65535	0, 17, 74, 65535

4.2 Operatory

Arytmetyczne:

+	2+2	4	dodawanie
-	2-1	1	odejmowanie
*	2*2	4	mnożenie
/	4/2	2	dzielenie
%	6%4	2	reszta z dzielenia

relacyjne:

==	X==Y	porównanie, sprawdź czy X jest równy Y
!=	X!=Y	Zaprzeczenie, sprawdź czy X NIE jest równy Y
<	X<Y	mniejszość, sprawdź czy X jest mniejszy od Y
>	X>Y	większość, sprawdź czy X jest większy od Y
<=	X<=Y	mniejsze lub równe, sprawdź czy X jest mniejszy lub równy Y
>=	X>=Y	większe lub równe, sprawdź czy X jest większy lub równy Y
++	X++	inkrementacja, zwiększ X o 1
--	X–	dekrementacja, zmniejsz X o 1
+=	X+=5	zwiększenie, zwiększ X o 5
-=	X-=5	zmniejszenie, zmniejsz X o 5
*=	X*=5	mnożnik, pomnóż X przez 5
/=	X/=5	dzielnik, podziel X przez 5

logiczne

&&	X<5 && X>9	logiczne “i”
||	X<5 || X>9	logiczne “lub”

4.3 instrukcje warunkowe

służą do wykonywania poleceń jeżeli spełnione są dane warunki, dla przykładu:

```
if(x<5){
X=3;
}
```

Jeżeli X będzie mniejszy od 5, zostanie ustawiony na wartość 3.

```
if(x<5){
X=3;
}else{
X=2;
}
```

Jeżeli X będzie mniejszy od 5, zostanie ustawiony na wartość 3. W przeciwnym wypadku zostanie ustawiony na 2.

```
if(x<5){
X=3;
}elseif(X == 2){
X=-4;
}else{
X=2;
}
```

Jeżeli X będzie mniejszy od 5, zostanie ustawiony na wartość 3. Następnie nastąpi sprawdzenie czy jest równy 2 jeżeli tak zostanie ustawiony na -4, w przeciwnym wypadku zostanie ustawiony na 2.

4.4 pętle

Pętle służą do powtarzania danych komend, wyróżniamy dwa rodzaje pętli, while oraz for.

Pętla while będzie wykonywała się dopuki w jej klamrach znajduje się wartość True, na przykład:

```
While(x<10){
X++;
}
```

Pętla ta będzie wykonywała się dopóki osiągnięty jest warunek x<10.

Pętle for są trochę bardziej skomplikowane, składnia pętli for wygląda w następujący sposób:

```
for(int i=0;i<6;i++){
X++;
}
```

Pierwsze co ustawiamy to parametr który będziemy używali w pętli, może to być już wykorzystywany parametr, jednak najczęściej tworzy się nowy.

Drugi parametr to warunek, który musi być spełniony aby pętla wykonywała się w tym przypadku i<6.

Trzeci parametr to komenda która zostanie wykonana po każdym wykonaniu pętli, w tym przypadku „i” zostaje zwiększone o 1.

Powyższa pętla wykona się 6 razy.

4.5Tablice

Tablica to zbiór zmiennych tego samego typu, dla przykładu:

Int liczby [] = {10, 30, 60, 5, 78}

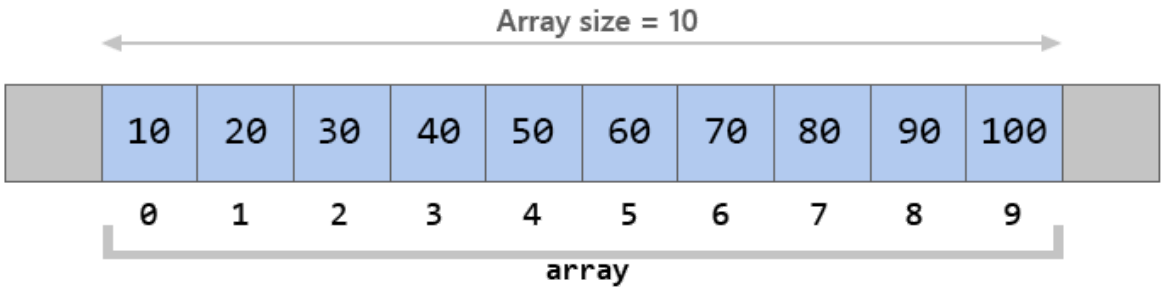
Aby odwołać się do wybranej komórki tablicy wystarczy użyć:

liczby[2]

liczby[0]

liczby[7] = 3

Tablice numeruje się od 0!



4.6 Funkcje

Czasami zdarza się tak że tego samego zestawu komend używamy wiele razy, w takim przypadku możemy użyć funkcji:

```
int ObwodProstokata(int a, int b){
return (a*2)+(b*2);
}
```

Aby wykonać funkcje:

```
ObwodProstokata(5,8);
```

Używanie funkcji ma dwie główne zalety, po pierwsze skraca długość kodu, a po drugie umożliwia szybsze wprowadzanie zmian.

5.Projekty

przystkie projekty dostępne są również na GitHub'ie:

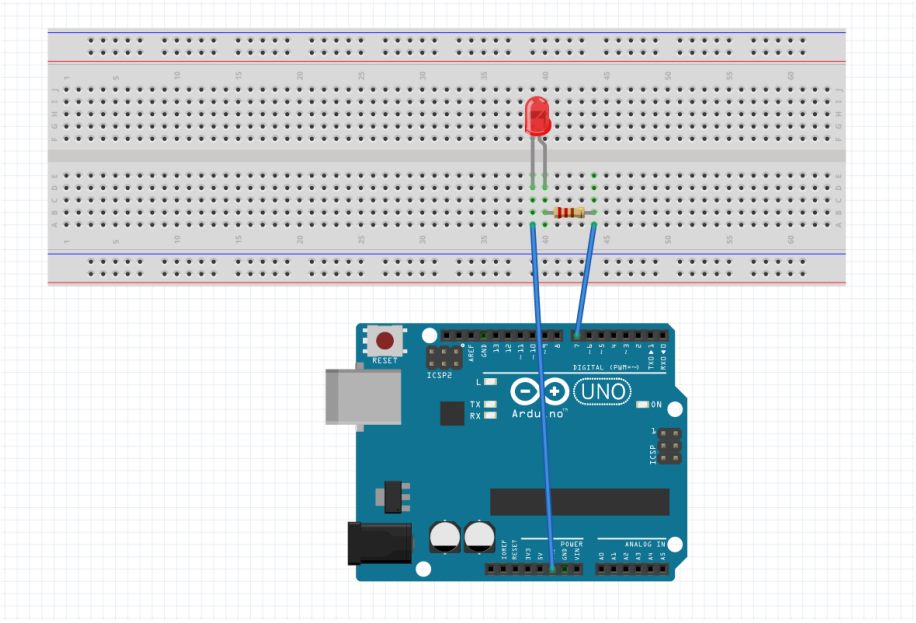
5.1 Migająca dioda LED

opis:
Poniższy program miga diodą z zadaną częstotliwością zapisaną w zmiennej "czestotliwosc"

wykaz elementów:

1. Dioda LED czerwona
2. Rezystor 220R
3. płytka stykowa
4. Arduino UNO R3

schemat połączeń:



program:

```
double czestotliwosc = 10.0; //częstotliwośc w HZ. 1HZ = 1 cykl w ciągu
sekundy

void setup() {
 pinMode(7, OUTPUT); //ustawienie pinu 7 jako wyjście dla diody
}

void loop() {
  digitalWrite(7, HIGH); //zapalenie diody
  delay(((1/czestotliwosc)/2*1000)); // Zamiana HZ na milisekundy i
odczekanie połowy cyklu
  digitalWrite(7, LOW); //zapalenie diody
  delay(((1/czestotliwosc)/2*1000)); // Zamiana HZ na milisekundy i
odczekanie połowy cyklu
}
```

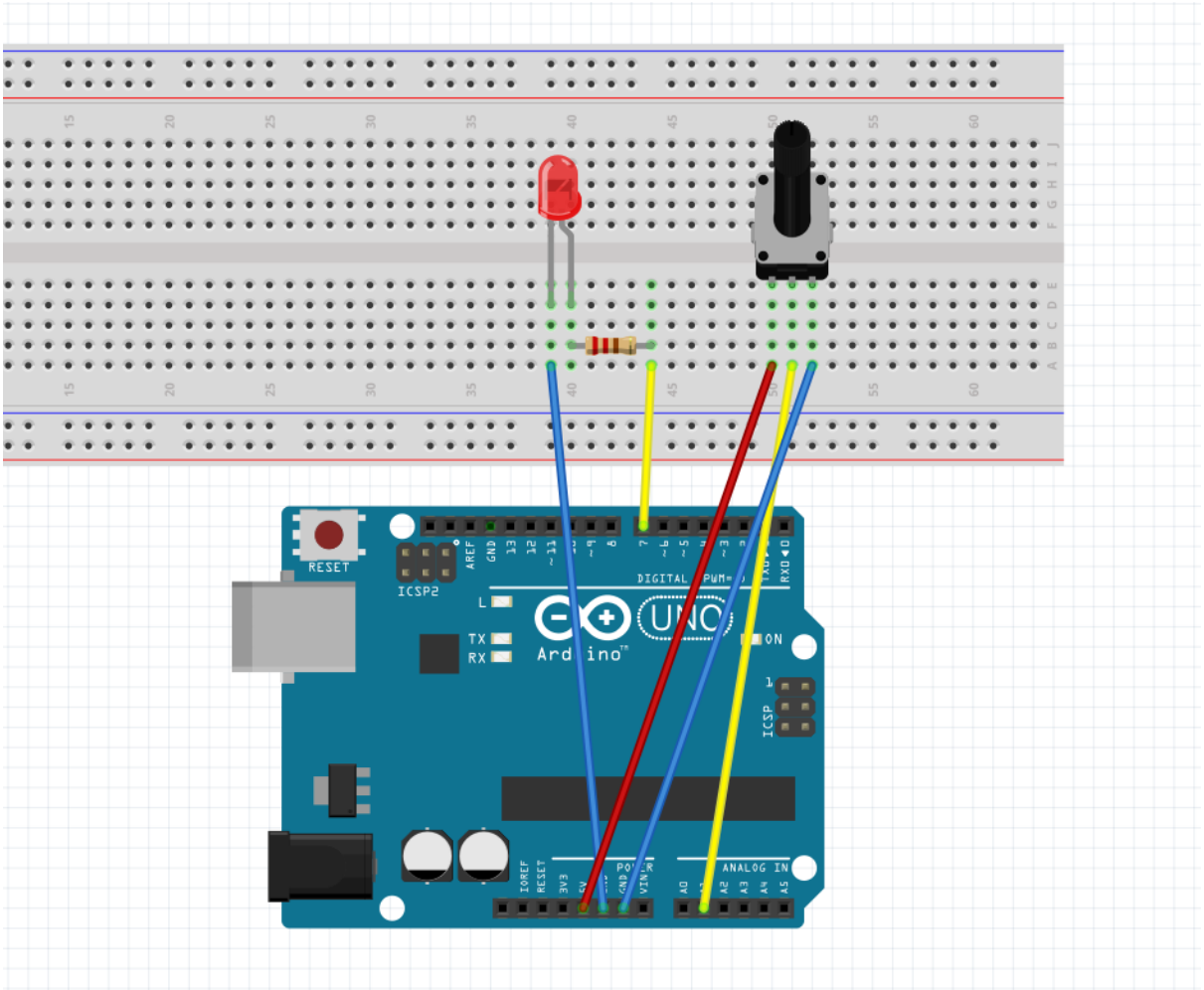
5.2 Regulacja częstotliwości potencjometrem

opis:
Poniższy program miga diodą z częstotliwością zależną od rezystancji potencjometru.

wykaz elementów:

1. Dioda Led
2. potencjometr
3. rezystor 220R
4. płytka stykowa
5. arduino Uno r3

schemat połączeń:



program:

```
double czestotliwosc = 1; //częstotliwośc w HZ. 1HZ = 1 cykl w ciągu
sekundy
 //wartość odczytana z potencjometru
void setup() {
 pinMode(7, OUTPUT); //ustawienie pinu 7 jako wyjście dla diody
 Serial.begin(9600);//Uruchomienie komunikacji przez USART
}

void loop() {
  czestotliwosc = analogRead(A1)* 0.034+1;// odczytanie wartości
potencjometru, zmniejszenie jej 34 krotnie oraz zmiększenie o 1(w
przeciwnym wypadku w linijce niżej nastąpiło by dzieleie przez 0)
  digitalWrite(7, HIGH); //zapalenie diody
  delay(((1/czestotliwosc)/2*1000)); // Zamiana HZ na milisekundy i
odczekanie połowy cyklu
  digitalWrite(7, LOW); //zapalenie diody
  czestotliwosc = analogRead(A1)* 0.034+1;
  delay(((1/czestotliwosc)/2*1000)); // Zamiana HZ na milisekundy i
odczekanie połowy cyklu
}
```

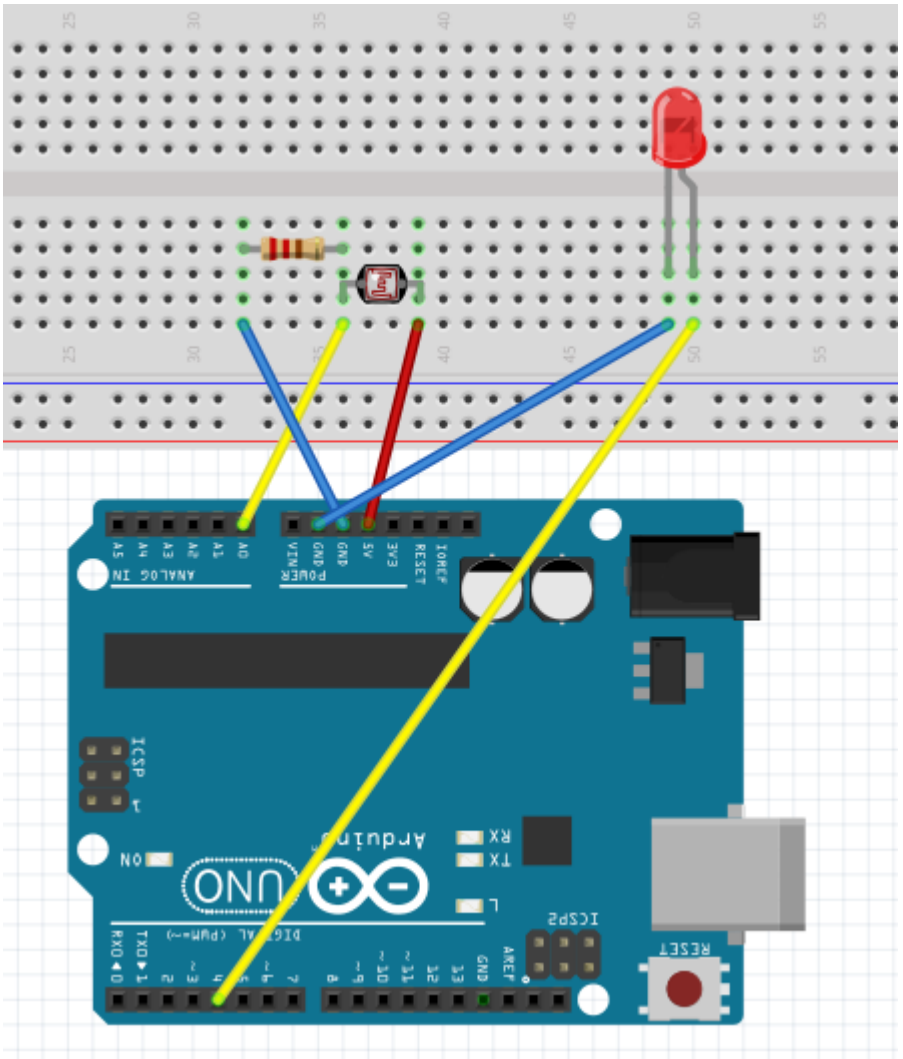
5.3 Fotorezystor

opis:
Poniższy program włącza lub wyłącza diodę led w zależności od ilości światła odbieranego przez fotorezystor.

wykaz elementów:

1. Fotorezystor
2. Dioda LED
3. płytka stykowa
4. Arduino UNO R3
5. 2x rezystor 220R

schemat połączeń:



program:

```
int wartoscGraniczna = 200; //Ustawienie wartości granicznej
uruchomienia diody LED z zakresu 0-1024

void setup() {
}

void loop() {
 int a = analogRead(A1);
 if(a<600){
   analogWrite(4, 1024);
}else{
   analogWrite(4, 0);
 }
}
```

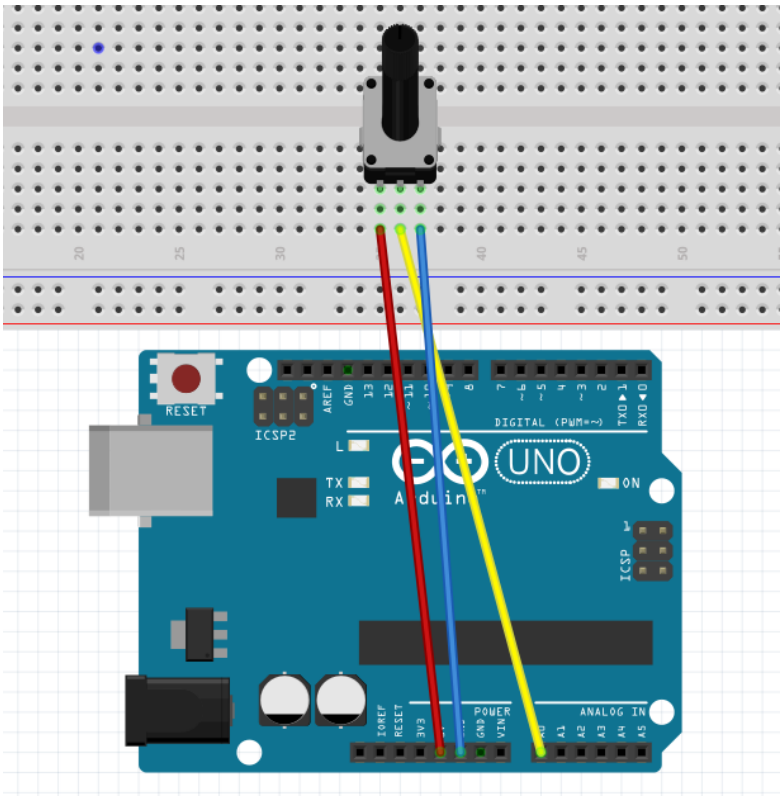
5.4 Serial

Opis:
program wysyła wartość rezystancji potencjometru przez port szeregowy.

wykaz elementów:

1. potencjometr
2. płytka prototypowa
3. arduino uno R3

schemat połączeń:

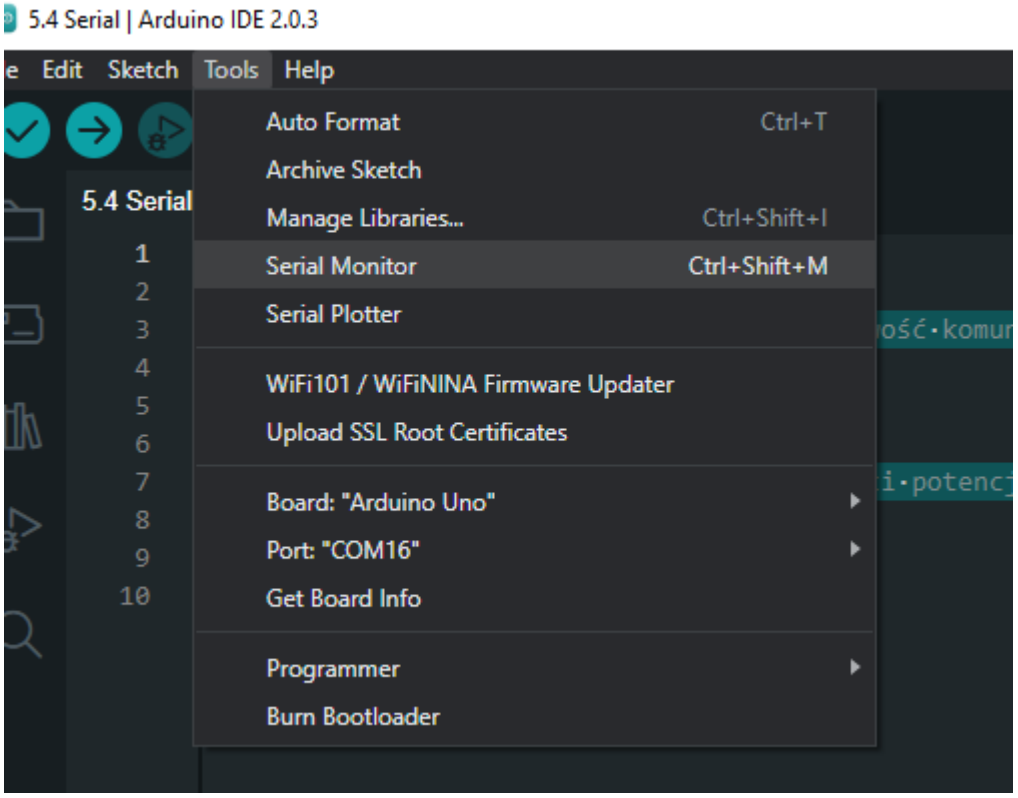


program:

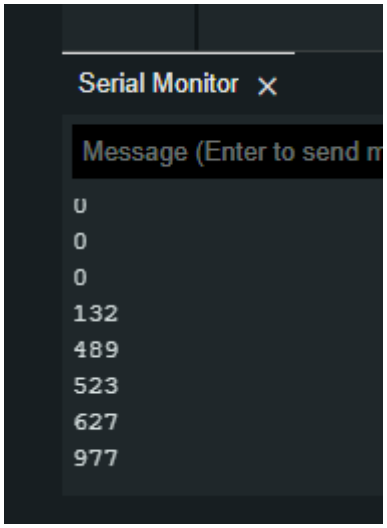
```
void setup() {
  Serial.begin(9600); //Ustawiamy częstotliwość komunikacjii
}

void loop() {
  int a = analogRead(A0); //Pobranie wartości potencjometru
  Serial.println(a); //wysłanie wartośći
}
```

Jak uruchomić monitor portu szeregowego:



lub skrót Ctrl+Shift+M



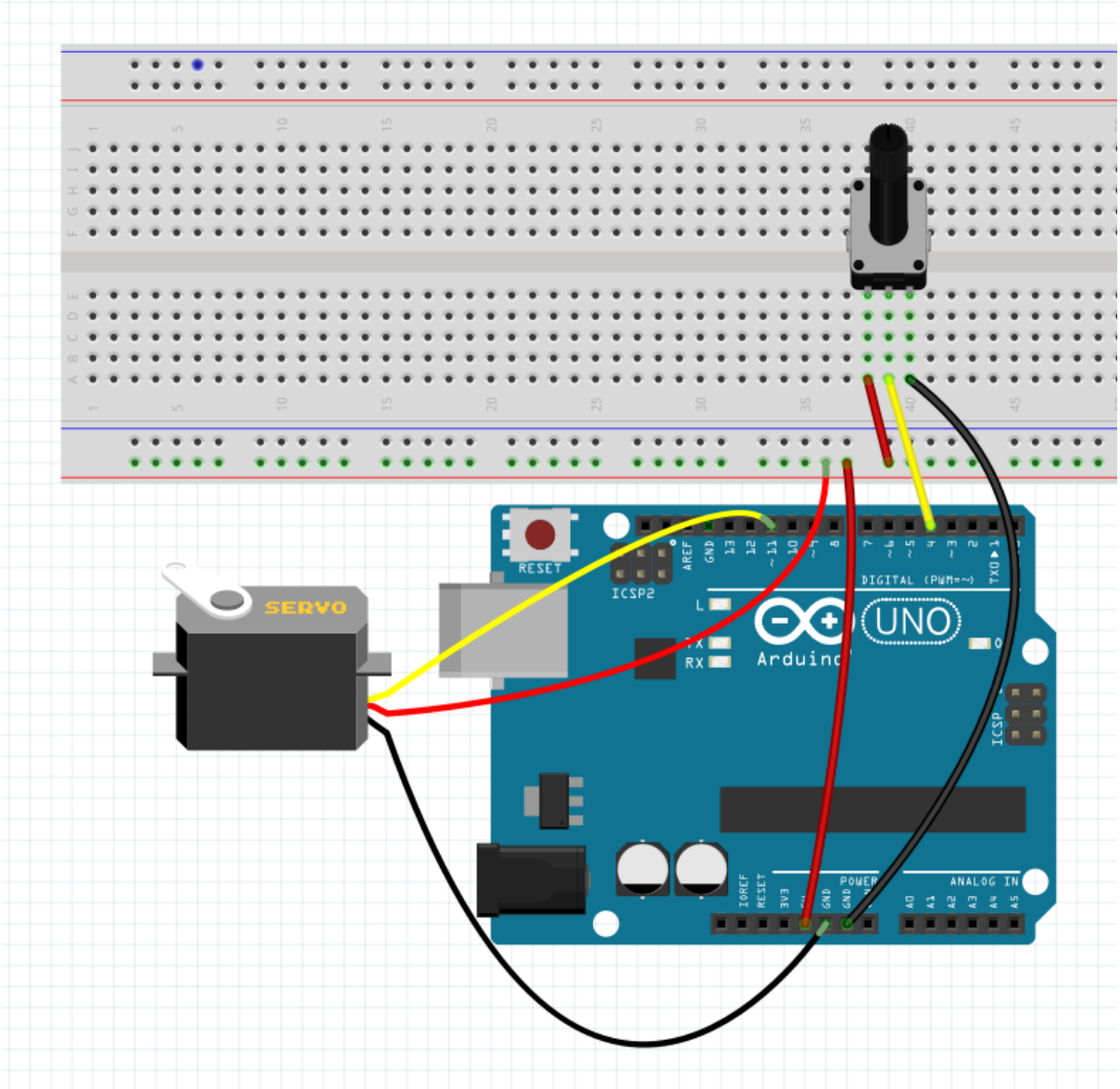
5.5 Serwo

Opis:
Program ustawia serwo w zależności od pozycji potencjometru.

wykaz elementów:

1. Serwo
2. potencjometr
3. płytka prototypowa
4. arduino uno r3

schemat połączeń:



Program:

```
#include <Servo.h>

Servo myservo;  // utworzenie obiektu serwa
int potpin = 4;  // pin używany do odczytywania wartości
int val;    // wartość przesyłana do serwa

void setup() {
  myservo.attach(11);  // podłączenie serwa do portu 11
}

void loop() {
  val = analogRead(potpin);            //odczytanie wartości z
potencjometru 0-1023
  val = map(val, 0, 1023, 0, 180);     // skalowanie wartości z zakresu
0 - 1023 do wartości 0 - 180
  myservo.write(val);                  // ustawienie serwa na zadaną
wartość
  delay(15);                           // odczekanie aż serwo usatwi
się na zadaną wartość
}
```

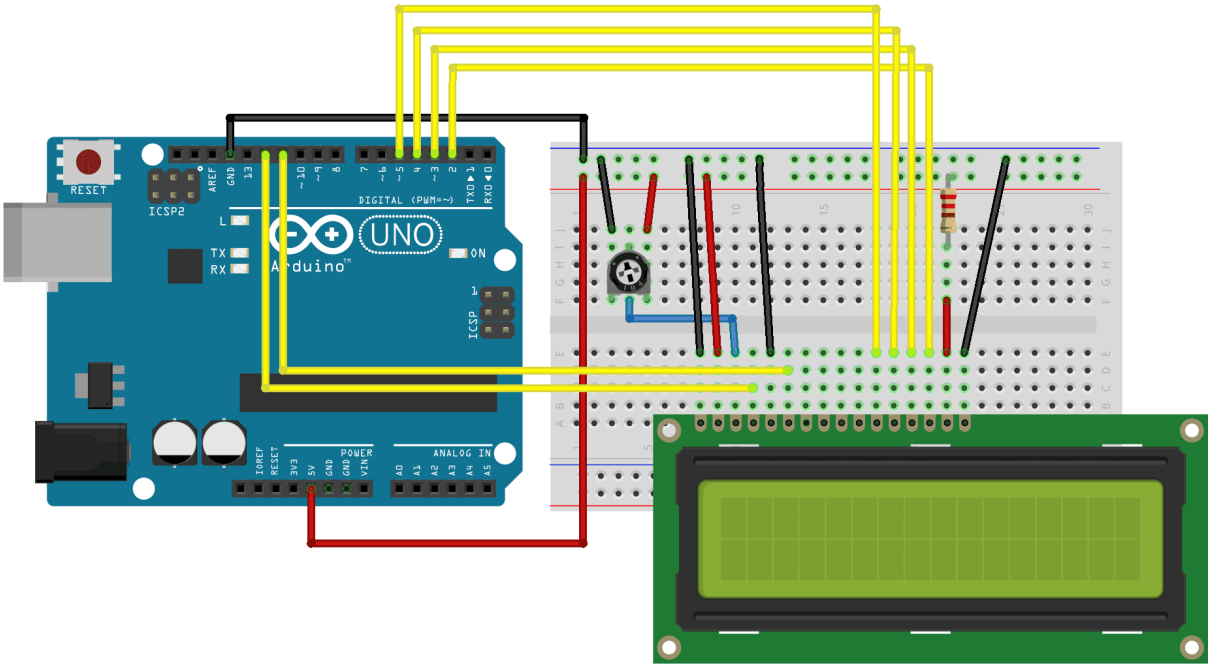
5.6 Ekran LCD

opis: Sterowanie ekranem lcd

wykaz elemetów:

1. ekran lcd
2. płytka stykowa
3. arduino

schemat połączeń:



program:
Najpierw należy pobrać(link) i dodać bibliotekę do środowiska arduino(szkic->dołącz bibliotekę->dołącz biblioteke .ZIP).

```
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("essaTech"); // tekst do wyświetlenia
}
```

```
void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

5.7 Czujnik DHT11

Opis:
Program odczytuje wartość temperatury i wilgotności powietrza.

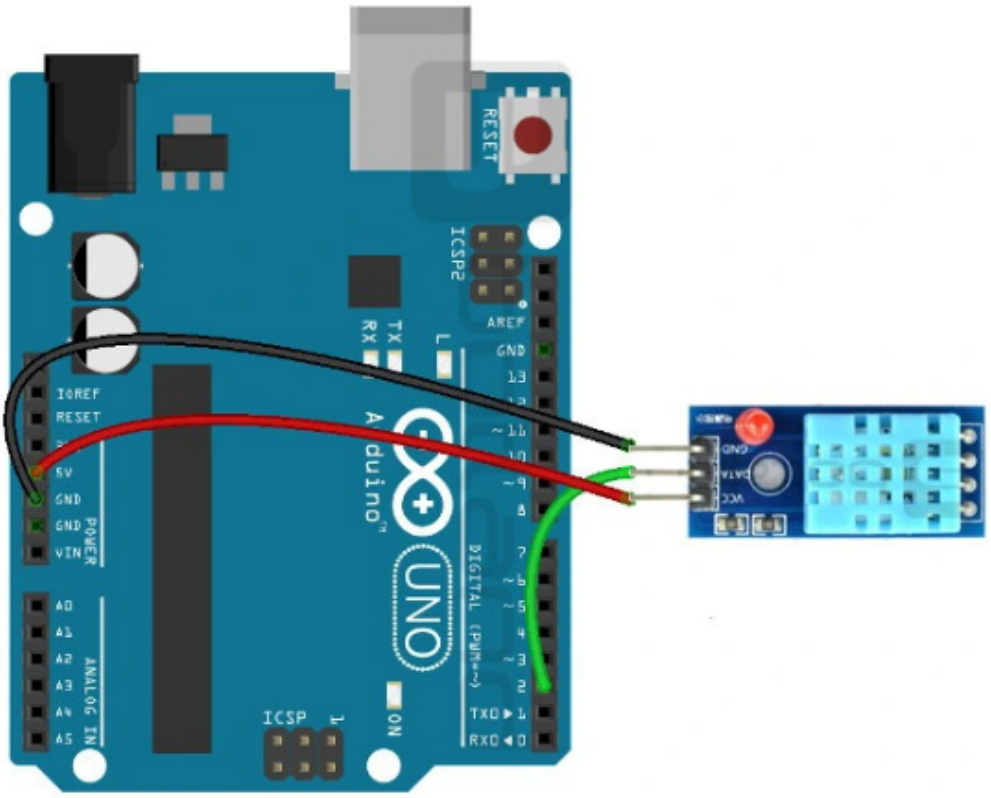
Wykaz elementów:

- Arduino Uno
- Czujnik DHT1
- Płytka stykowa
- Przewody połączeniowe męsko-męskie
- Biblioteka DHT11

Połączenie DHT11 z Arduino:

Pin DHT11	**Pin Arduino**
VCC	5 V
DATA	2
GND	GND

Schemat połączeń:



Program:
Na początku należy pobrać (Link CODE->Download ZIP) i dodać bibliotekę do środowiska Arduino (Szkic -> Include Library -> Add .ZIP Library...).

```
#include "DHT.h"
#define DHT11_PIN 2
DHT dht;

void setup()
{
  Serial.begin(9600);
  dht.setup(DHT11_PIN);
}

void loop()
{
  //Pobranie informacji o wilgotnosci
  int wilgotnosc = dht.getHumidity();
  //Pobranie informacji o temperaturze
  int temperatura = dht.getTemperature();
```

```
  if (dht.getStatusString() == "OK") {
    Serial.print(wilgotnosc);
    Serial.print("%RH | ");
    Serial.print(temperatura);
    Serial.println("*C");
  }

  //Odczekanie wymaganego czasu
  delay(dht.getMinimumSamplingPeriod());
}
```

5.8 RFID

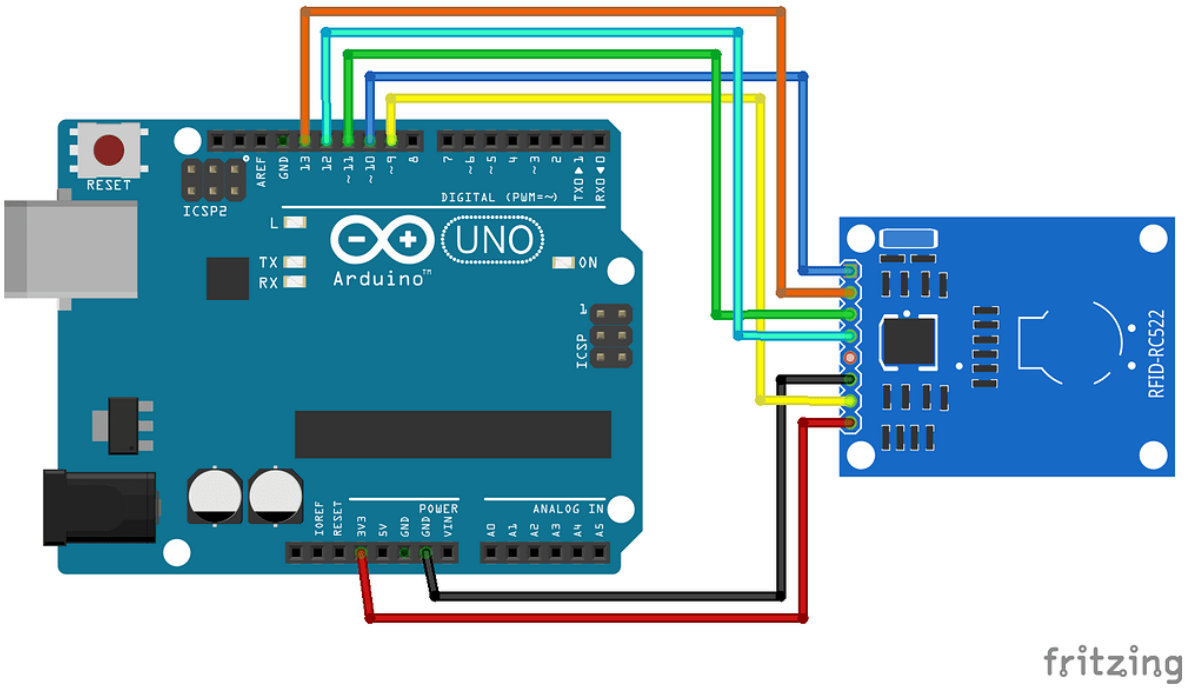
opis:
Program odczytuje zawartość karty RFID i wyświetla ją w Serial Monitor.

wykaz elementów:

- czytnik RFID
- karta RFID
- brelok RFID
- płytka stykowa
- Arduino uno r3

schemat połączeń:

UWAGA! Czytnik jest zasilany z linii 3.3V.



program:
Najpierw należy pobrać(link) i dodać bibliotekę do środowiska arduino(szkic->dołącz bibliotekę->dołącz biblioteke .ZIP).

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9

MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class

MFRC522::MIFARE_Key key;

// Init array that will store new NUID
byte nuidPICC[4];

void setup() {
  Serial.begin(9600);
  SPI.begin(); // Init SPI bus
  rfid.PCD_Init(); // Init MFRC522

  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
```

```
  Serial.println(F("This code scan the MIFARE Classsic NUID."));

}

void loop() {

  // Reset the loop if no new card present on the sensor/reader. This
saves the entire process when idle.
  if ( ! rfid.PICC_IsNewCardPresent())
    return;

  // Verify if the NUID has been readed
  if ( ! rfid.PICC_ReadCardSerial())
    return;

  Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
  Serial.println(rfid.PICC_GetTypeName(piccType));

  // Check is the PICC of Classic MIFARE type
  if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
    piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
    piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("Your tag is not of type MIFARE Classic."));
    return;
  }


    Serial.println(F("A new card has been detected."));

    // Store NUID into nuidPICC array
    for (byte i = 0; i < 4; i++) {
      nuidPICC[i] = rfid.uid.uidByte[i];
    }

    Serial.println(F("The NUID tag is:"));
    Serial.print(F("In hex: "));
    printHex(rfid.uid.uidByte, rfid.uid.size);
    Serial.println();
    Serial.print(F("In dec: "));
    printDec(rfid.uid.uidByte, rfid.uid.size);
    Serial.println();
```

```
  // Halt PICC
  rfid.PICC_HaltA();

  // Stop encryption on PCD
  rfid.PCD_StopCrypto1();
}

/**
 * Helper routine to dump a byte array as hex values to Serial.
 */
void printHex(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
  }
}

/**
 * Helper routine to dump a byte array as dec values to Serial.
 */
void printDec(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(' ');
    Serial.print(buffer[i], DEC);
  }
}
```