5. Arduino

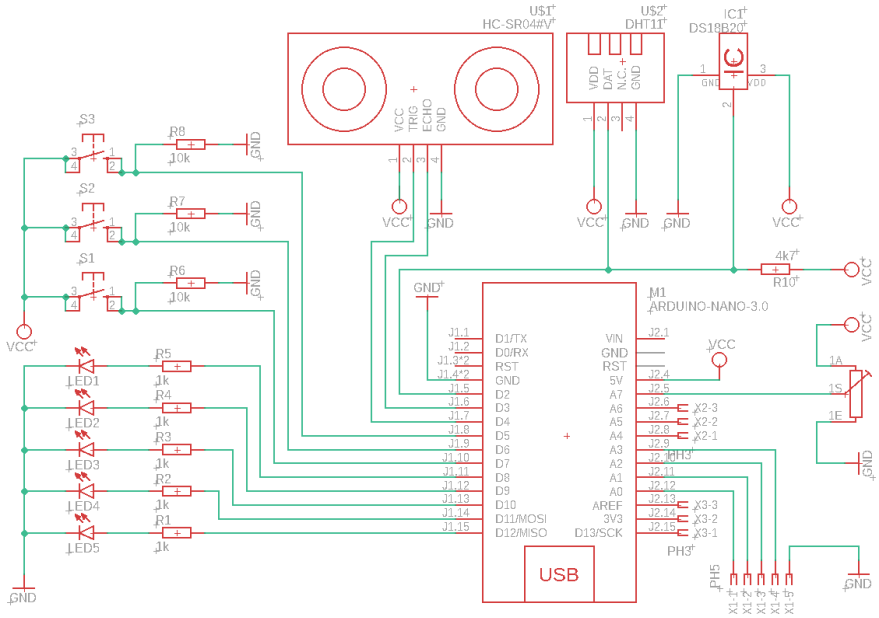
Do nauki programowania przy użyciu Arduino potrzebne są pewne podstawowe elementy:

- Komputer z oprogramowaniem Arduino IDE (https://www.arduino.cc/en/Main/Software),
- Arduino Nano lub UNO (lub klon) z kablem USB,
- Układ przycisków, diod i innych podzespołów.

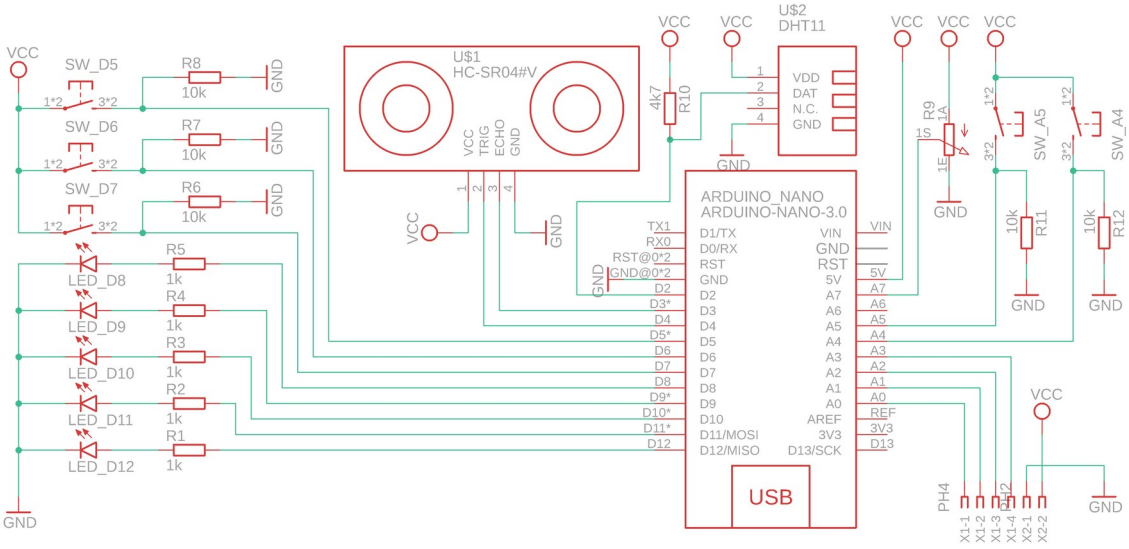
LUB symulator (tinkercad.com) gdy nie mamy dostępu do fizycznych części.

W ramach zajęć wykorzystywane będą gotowe układy, wg poniższych schematu:

Wersja 2:

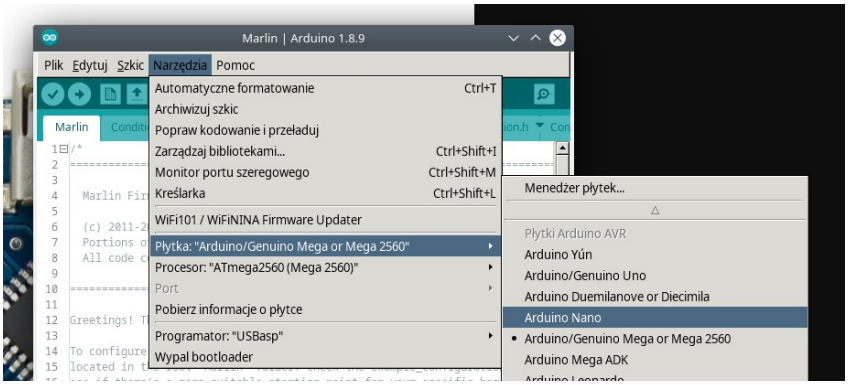


Wersja 3:



Arduino IDE

Aby sprawdzić czy *wszystko działa* należy nie podłączać Arduino i uruchomić Arduino IDE, następnie w menu *Narzędzia* wybrać odpowiednią płytkę (na zajęciach *Arduino Nano).*

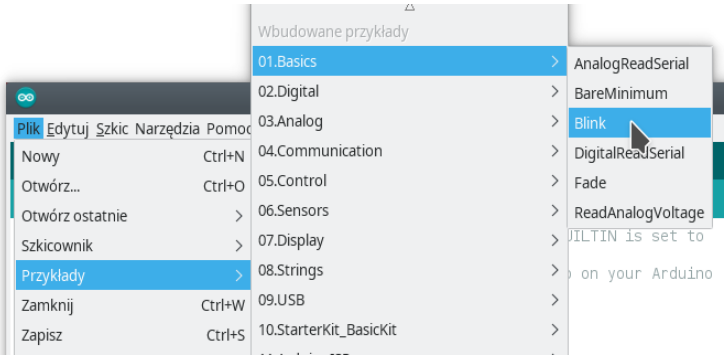


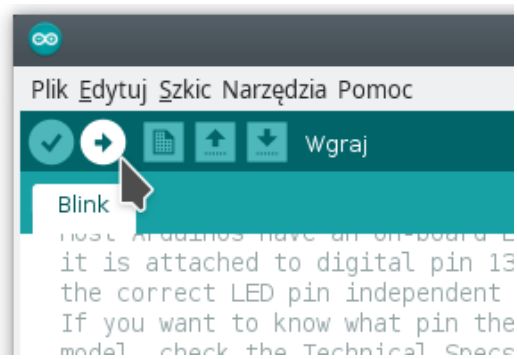
Jeśli menu *Port* jest aktywny, należy zapamiętać jego *zawartość*. Następnie podłączyć Arduino, poczekać aż sterowniki zostaną zainstalowane i ponownie wybrać menu *Narzędzia*, następnie z menu *Port* wybrać nowy port.

MacBook: jeśli nie pojawia się nowy port, należy zainstalować sterowniki ze strony wtd.zabałaganionemiejsce.pl/Automatyka/Arduino (plik pkg).

Windows: jeśli nie pojawia się nowy port, należy zainstalować sterowniki ze strony wtd.zabałaganionemiejsce.pl/Automatyka/Arduino (plik exe).

Z menu *Procesor* należy wybrać opcję *ATmega328P (Old Bootloader)*. Następnie wgrać program przykładowy *Blink*.

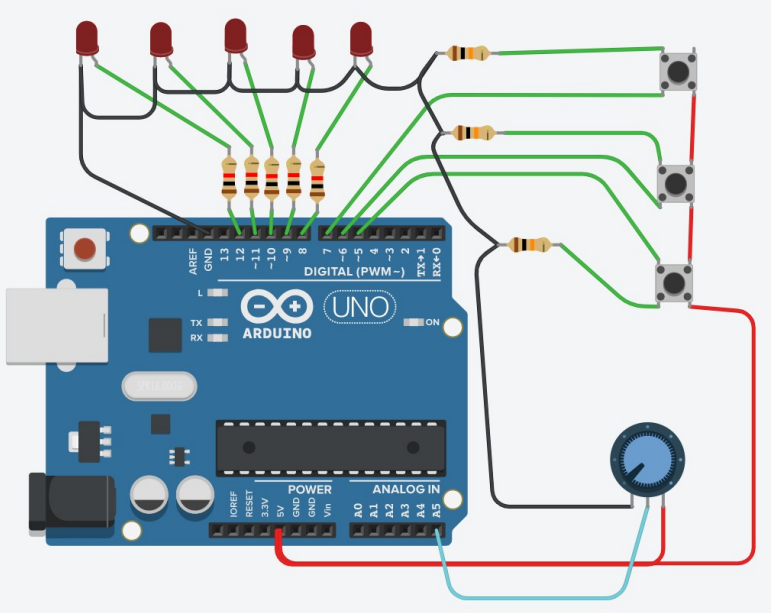




Linux: jeśli występują błędy przy wgrywaniu związane z brakiem dostępu (*Permission denied*) należy wykonać komendę: `sudo usermod -a -G dialout $USER`
Możliwe, że niezbędne będzie przelogowanie się by zmiany zadziałały!

Arduino on-line

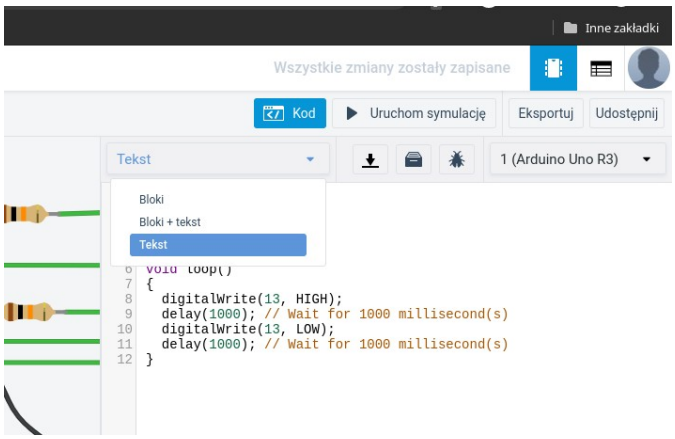
W przypadku braku dostępu do fizycznego sprzętu, można się posiłkować symulatorem online (www.tinkercad.com/circuits):



Należy mieć jednak na uwadze, że:

- obecnie dostępne jest jedynie Arduino UNO (które jest kompatybilne z NANO),
- Arduino UNO nie posiada wyjścia A7, potencjometr trzeba podłączyć do innego wyjścia oznaczonego literką A i pamiętać o tym podczas pisania programów.
- na początek wystarczy podłączyć diody, przyciski oraz potencjometr wraz z rezystorami.

W celu faktycznego programowania należy z prawej strony wybrać *Kod* oraz przełączyć widok blokowy na tekstowy.

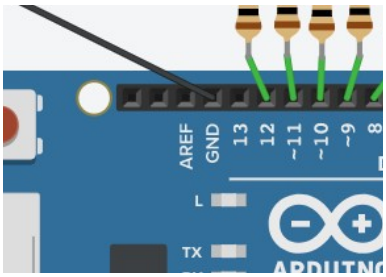


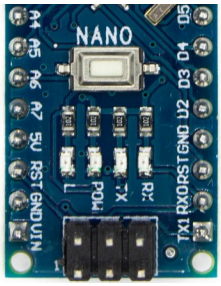
Warto porządnie rozmieścić sobie elementy w symulatorze, ponieważ po rejestracji układy są automatycznie zapisywane, więc zawsze można do nich wrócić bez ponownego rysowania.

Ustawianie/odczytywanie portów

```
pinMode(port, STAN);          - ustawia portu jako wejście (INPUT) lub
                                wyjście (OUTPUT)
digitalRead(port);            - odczytuje wartość (0 lub 1) z portu
digitalWrite(port, STAN);     - ustawia port, na logiczne 0 lub 1 (5V)
```

port to numer pinu z pominięciem literki D.
W przypadku portów Ax, literkę A należy podać w programie.





`LED_BUILTIN`

w przypadku Arduino NANO oraz UNO, jest to zmienna przyjmująca wartość 13, do którego dołączona jest dioda L zamontowana na obu Arduino

Zadanie 5.1

IDE: Po uruchomieniu Arduino IDE oraz podłączeniu Arduino, z menu Plik, wybrać Przykłady → 01.Basics → Blink. Przeanalizować program, a następnie wgrać program na Arduino.

Symulator: Program blink dostępny jest również na stronie Arduino, aby użyć go w symulatorze, należy go skopiować z www.arduino.cc/en/tutorial/blink. Przeanalizować program, a następnie uruchomić symulację.

Czy program zachowuje się zgodnie z przewidywaniami?

Zadanie 5.2

a) Co należy zmienić w programie, aby zamiast diody L mrugała dioda podłączona do D12?

b) Zmodyfikować program w taki sposób, by diody mrugały:

- wszystkie naraz,
- mrugały jedna po drugiej,
- mrugały tam i z powrotem.

c) Narysować wykres stanów logicznych na wyjściach dwóch diod, z których jedna na przemian zapala się na 1s i gaśnie na 1s; druga – zapala na 0,5s i gaśnie na 0,5s.

d) Napisać program realizujący *miganie* z podpunktu c).

Łączenie warunków

Jeśli wewnątrz funkcji if (lub innej warunkowej) chcemy umieścić więcej niż jeden warunek:

`(digitalRead(3) || digitalRead(4))` – oznacza prawdę, jeśli na wejściu 3 LUB 4 jest logiczne 1 (PRAWDA)

`(digitalRead(3) && digitalRead(4))` – oznacza prawdę, jeśli na wejściu 3 ORAZ 4 jest logiczne 1 (PRAWDA)

Instrukcja warunkowa `if/else/else if`

Instrukcja warunkowa **if** może być rozbudowana o **else**, którego ciało wykonywane jest gdy poprzednie warunki są niespełnione lub **else if**(warunek), które wykonywane jest gdy wcześniejsze warunki są niespełnione ORAZ spełniony jest *nowy* warunek.

```
if(warunek1){
        //wykonywane, gdy spełniony warunek1
}else if(warunek2){
        //wykonywane, gdy niespełniony warunek 1 i spełniony warunek2
}else{
        //wykonywane, gdy niespełnione wcześniejsze warunki
}
```

Po *warunku* nie należy wpisywać średnika!

Zadanie 5.3

Napisać program, który zapala tylko jedną diodę:

a) po wciśnięciu jednego z przycisków od 1 do 3 zapala odpowiadającą diodę,

b) jeśli zostaną wciśnięte dwa przyciski – zapala się czwarta dioda, jeśli 1, 2 oraz 3 – piąta dioda.

c) Jeśli wcześniej nie zostało to zrobione: zadeklarować na początku programu intuicyjne zmienne przechowujące numery portów do których podłączone są diody (np. `int dioda5 = 12;`), a następnie podmienić w dalszej części programu *surowe* (np. 12) wartości portów na zmienne (np. `dioda5`).

Zadanie 5.4 (tylko dla stanowisk w wersji 3 oraz on-line)

Sprawdzić czy program napisany na poprzednich zajęciach (na tablicy) działa jak powinien.

Sprawozdania (w PDF) należy zaopatrzyć w napisane programy z kometarzem.

Forma zdjęć jest niedopuszczalna.

Programy należy umieścić w formie zrzutów ekranu lub w formie tekstowej po wcześniejszym pokolorowaniu. Do automatycznego kolorowania tekstu może posłużyć strona tohtml.com/cpp.

Zmienne

Zmienne służą do wygodnego zapamiętywania danych programu pod jakąś nazwą. Ponieważ są różne rodzaje danych, są też różne rodzaje zmiennych. Podstawowym typem (dla nas) będzie:

- `int` – ang. „integer” – liczba całkowita, przechowuje liczby od -32768 do 32767

Wszystkie zmienne będziemy traktować jako zmienne globalne, czyli dostępne w każdym miejscu w programie. Zmienne globalne deklaruje się na początku programu (przed `void setup()`):

```
typ_zmiennej nazwa_zmiennej;
```

lub, gdy chcemy od razu zadeklarować jej wartość:

```
typ_zmiennej nazwa_zmiennej = wartość;
```

Przykładowo deklaracja: `int dioda5 = 12;` sprawi, że wszędzie gdzie w programie użyjemy nazwy *dioda5*, komputer będzie wiedział, że mieliśmy na myśli liczbę 12 (póki jej nie zmienimy).

Obliczenia na zmiennych

Obliczenia na zmiennych zachowują się tak jak uniwersalne znaki obliczeń: `+` (dodawanie), `-` (odejmowanie), `*` (mnożenie), `/` (dzielenie), `%` (reszta z dzielenia). Znak równości (`=`) oznacza przypisanie „tego co z prawej” do tego co „z lewej” – w przeciwieństwie do porównania (`==`). Np.:

```
zmienna8 = zmienna5 + 3;
```

Jeśli zmienna `zmienna5` miała wartość 5, to po wykonaniu tej komendy, `zmienna8` będzie przechowywała wartość 8. W ów czas komenda `digitalWrite(zmienna8, HIGH);` ustawi port D8 w stan wysoki (i np. zapali diodę do niego podpiętą).

Zadanie dodatkowe

Napisać program, który po włączeniu arduino zapali diodę trzecią, następnie, gdy zostanie wciśnięty przycisk 1, gaśnie *obecna* dioda i zapala *poprzednia*. Po wciśnięciu przycisku 2, gaśnie *obecna* i zapala *następna*. Czyli:

Zdefiniować nową *zmienną* typu `int` o dowolnej nazwie. Zwiększyć wartość *zmiennej* o 1, gdy zostanie wciśnięty przycisk 1 oraz zmniejszyć wartość *zmiennej*, jeśli zostanie wciśnięty przycisk 2. W zależności od tego, jaką wartość przyjmuje *zmienna* zapalić odpowiednią diodę (i zgasić inne).

Zadanie dodatkowe 2

Funkcja `millis()` zwraca zmienną `unsigned long` zawierającą czas od uruchomienia arduino w milisekundach. Wykonać zadanie 5.2 podpunkt c) beż użycia funkcji `delay()`.

Źródła:
Autodesk Eagle, Version 9.3.2, © 2019 Autodesk, Inc. All rights reserved
http://akademia.nettigo.pl/zmienne_podstawy_jezyka_arduino/
https://www.tinkercad.com/

Patryk Król

v3.2