
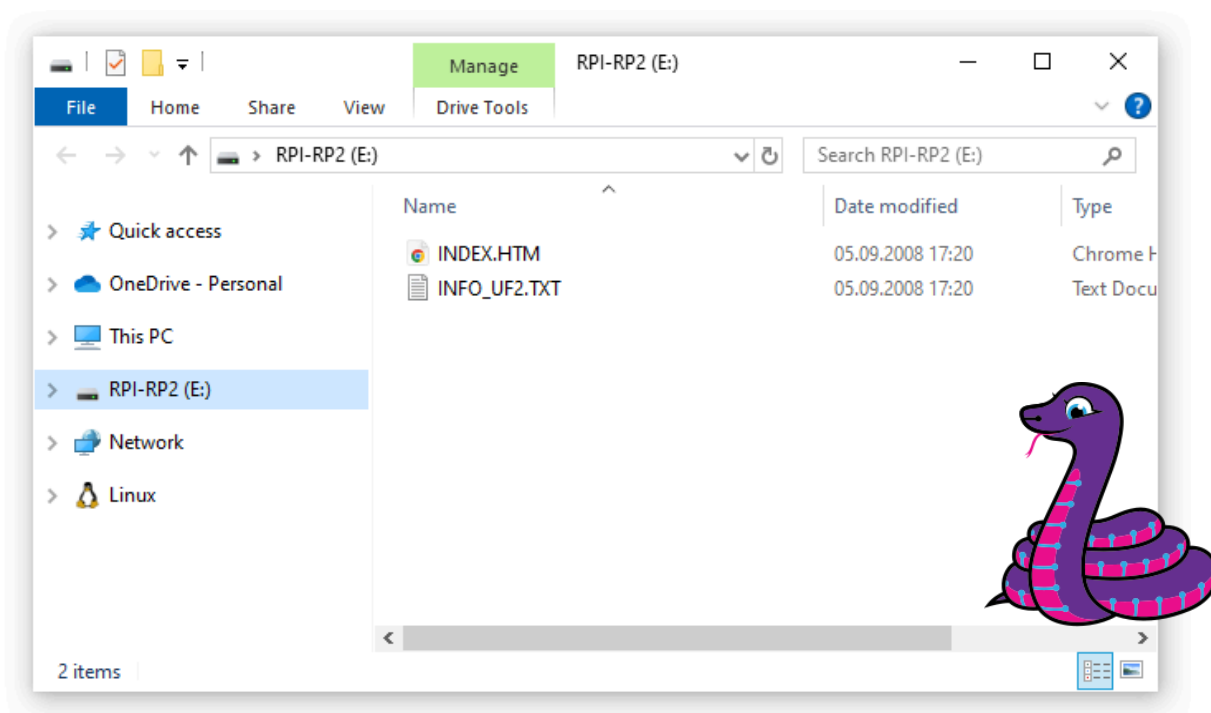


Tworzenie kontrolera do gier

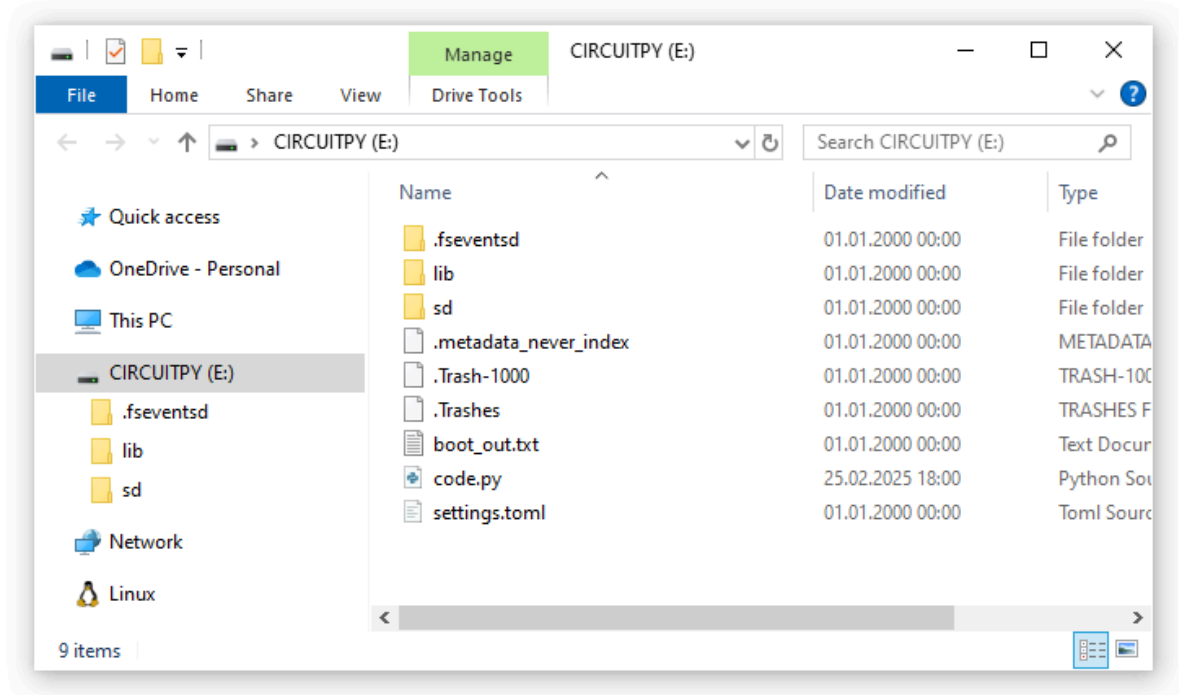
Instalowanie CircuitPython na RaspberryPi Pico

Pobierz [plik z rozszerzeniem .uf2](#) z oficjalnej strony. W przypadku różnych płytek może być potrzebna inna wersja systemu. Tutaj [bezpośredni link do pobrania](#) wersji na płytkę *pico*.

DOWNLOAD .UF2 NOW 



Przenieś (przeciągnij) pobrany plik do folderu **RPI-RP2**. CircuitPython powinien zainstalować się automatycznie.



Tak powinna wyglądać zawartość folderu po zakończeniu instalacji.

Środowisko programistyczne

Na warsztatach będziemy korzystać z edytora dostępnego w internecie przez przeglądarkę.

Przejdź na stronę <https://code.circuitpython.org/>

Aby połączyć się z płytką wybierz opcje:

USB ⇒ Connect to Device ⇒ CircuitPython CDC control ⇒ Connect ⇒ Use \

Podstawy programowania w Pythonie

Blink

Zacznijmy od czegoś prostego. Spróbujemy napisać program który zapala i gasi wbudowaną diodę co 1 sekundę. Najpierw trzeba zaimportować potrzebne biblioteki. Biblioteki zawierają skomplikowany kod który ktoś za nas napisał - nie musimy wiedzieć jak dokładnie działają, tylko jak ich używać. Importujemy je na samej górze pliku za pomocą słowa `import` i nazwy biblioteki.

```
import time
import board
import digitalio
```

Następnie tworzymy obiekt `led` i przypisujemy mu pin odpowiadający wbudowanej diodzie (`board.LED`). `LED` zastępuje nam numer pina (nóżki płytki) do której podpięta jest wbudowana dioda.

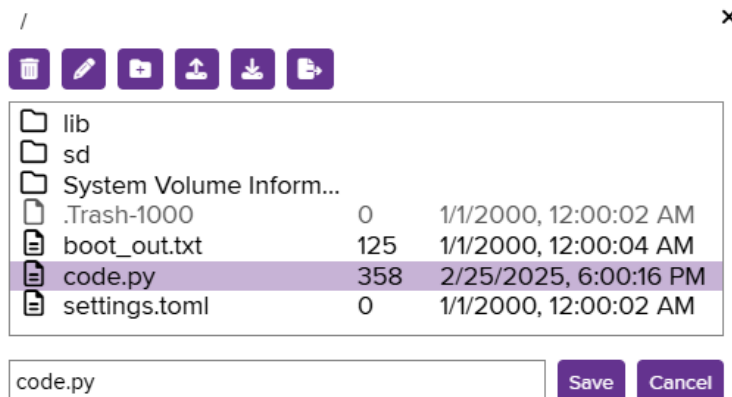
Ustawiamy ten pin jako wyjście (`OUTPUT`), dzięki czemu możemy nim sterować.

```
led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT
```

Teraz stworzymy nieskończoną pętlę za pomocą `while True`. Po wykonaniu instrukcji do końca, program będzie wracać do tej linijki i ponownie wykonywać instrukcje poniżej (i tak w nieskończoność). W tej pętli będziemy pisać całą logikę kontrolera. Kod który znajduje się w pętli (poniżej `while`) musi być wcięty (`tab` na klawiaturze).

```
while True:
    print("led on") # Wypisujemy informację w terminalu
    led.value = True # Włączamy led
    time.sleep(1)    # i czekamy 1 sekundę.
    led.value = False # Po sekundzie wyłączamy
    time.sleep(1)    # i znowu czekamy
```

Spróbujmy wgrać powyższy kod na płytkę przyciskiem **Save + Run** i zapisując go jako **code.py** w głównym katalogu. (Można podmienić plik który już tam jest.)



Powinno otworzyć się okienko terminala a w nim informacja o udanym wgraniu programu.

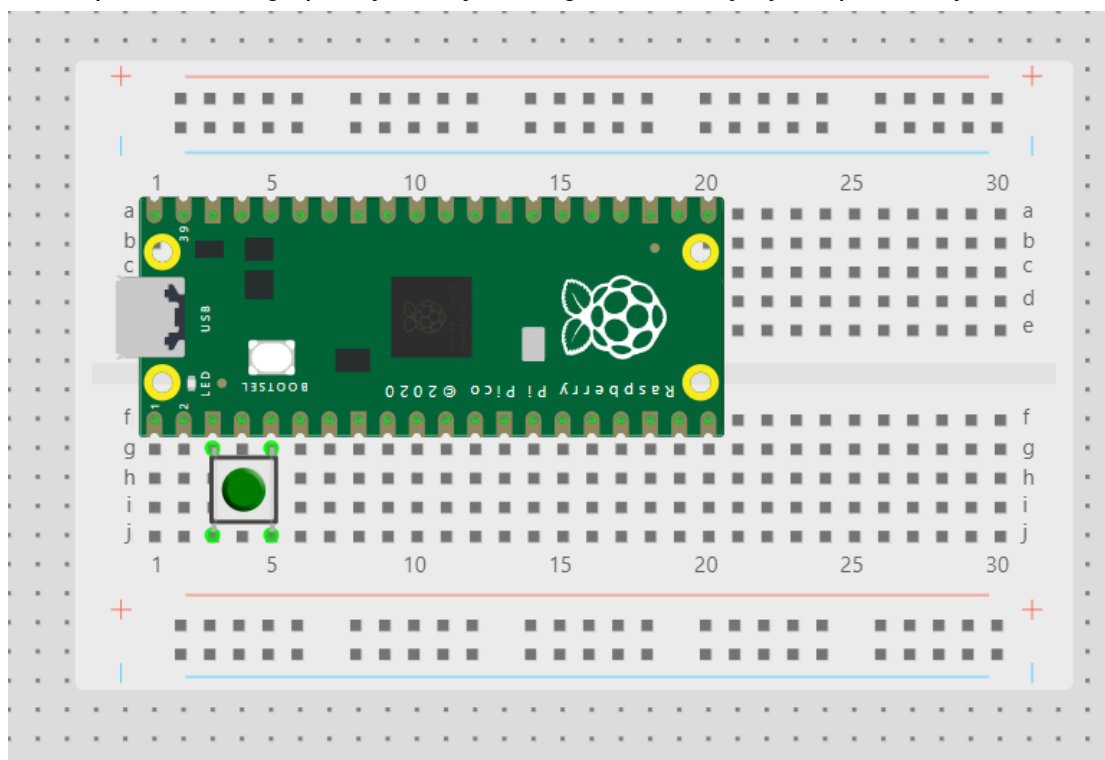
Co 2 sekundy powinna się również pojawiać informacja **"led on"**.

Jeśli w terminalu pojawia się błąd, cofnij się do kodu i sprawdź czy nie ma literówek lub błędów. Sprawdź też czy na początku linii nie ma żadnych niepotrzebnych spacji oraz czy zgadza się głębokość wcięć (1 tab == 4 spacje).

Kod do wszystkich etapów można znaleźć na [githubie projektu](#).

Button

Jeśli działa nam kod z poprzedniego zadania to spróbujemy teraz zapalać diodę na naciśnięcie przycisku. Zaczniemy od podłączenia przycisku do płytki. Ważne żeby jedna z nóżek przycisku była podłączona do dowolnego z pinów oznaczonych **GND**. Zapamiętaj numer pinu do którego podłączona jest druga - za chwilę będzie potrzebny.



W kodzie przycisk nie wymaga żadnych dodatkowych bibliotek, ale też musimy go zainicjalizować. Najpierw tworzymy obiekt `button` (najlepiej pod inicjalizacją diody) i podajemy pin do którego podpięliśmy przycisk. Na schemacie wyżej jest to pin 3 (czyli **GP3**). Później ustawiamy ten pin jako wejście (`INPUT`), co pozwoli nam odczytywać jego stan. Na koniec aktywujemy wewnętrzny rezystor podciągający (pull-up) w mikrokontrolerze (bez niego mogłyby występować losowe odczyty). Ważne jest to że odwróci nam on intuicyjne wartości przycisku - `True` kiedy przycisk nie jest wciśnięty i `False` kiedy jest.

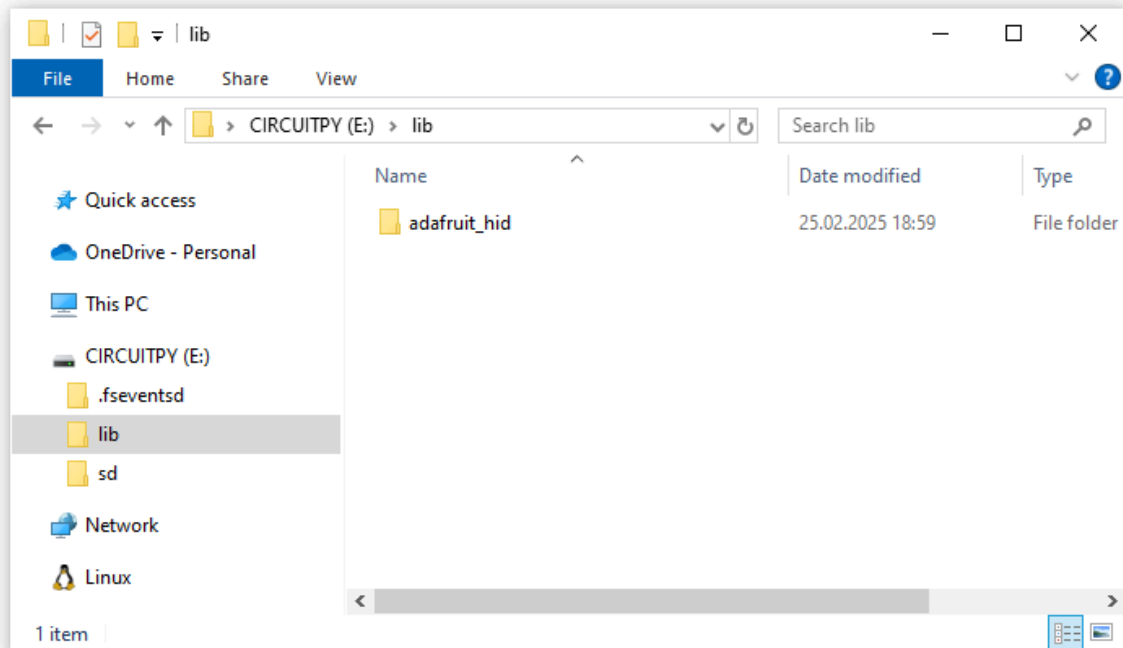
```
button = digitalio.DigitalInOut(board.GP3)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP
```

Możemy teraz zmodyfikować pętlę. Do sprawdzenia czy przycisk jest wciśnięty użyjemy instrukcji warunkowej `if-else`. `If` zadaje pytanie *jeżeli*. W naszym przypadku: *jeżeli* przycisk jest wciśnięty (ma wartość `False`): zapal LED i wypisz *click* w terminalu. To co napiszemy po `else` wykona się w każdym innym przypadku (czyli jeśli przycisk nie jest aktualnie wciśnięty). My chcemy wtedy wyłączyć diodę. Pętla może się wykonywać wiele razy na sekundę, ale żeby uniknąć błędów troszkę ją spowolnimy dodając na końcu krótkie zatrzymanie.

```
while True:
    if button.value == False: # Sprawdzamy czy przycisk jest wciśnięty
        led.value = True # Jeśli jest to włączamy led
        print("click") # i wypisujemy informacje w terminalu
    else:
        led.value = False # Jeśli nie to wyłączamy
        time.sleep(0.1) # zatrzymujemy się na sekundę żeby przycisk lepiej działał
```

Keyboard

Jesteśmy już blisko bardzo pierwszego najprostszego kontrolera. Musimy jeszcze tylko zasymulować kliknięcie przycisku na klawiaturze. Będziemy do tego potrzebować zewnętrznej biblioteki [adafruit_hid](#). Można ją pobrać w paczce [tutaj](#). Są w niej również inne biblioteki które będą potrzebne w dalszej części warsztatów. Aby dołączyć bibliotekę przenosimy odpowiedni plik lub folder (teraz jest to folder **adafruit_hid**) do folderu **lib** na płycie (**CIRCUITPY**).



Aby zaimportować te biblioteki dopiszemy na górze pliku (tam gdzie pozostałe **importy**):

```
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
```

i stworzymy obiekt klawiatury **keyboard**

```
keyboard = Keyboard(usb_hid.devices)
```

Pętla wymaga już tylko niewielkiej modyfikacji. Aby zasymulować kliknięcie przycisku użyjemy funkcji:

```
keyboard.press(Keycode.SPACE)
```

Trzeba też pamiętać o puszczeniu przycisku, inaczej on będzie wciśnięty cały czas, co bardzo utrudni nam dokończenie pisania programu 😊. Najlepiej użyć:

```
keyboard.release_all() # Puszcza wszystkie wciśnięte klawisze
```

Gotowy program

```
import time
import board
import digitalio
import usb_hid

from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode

# Dioda LED
led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

# Przycisk
button = digitalio.DigitalInOut(board.GP1) # Tu trzeba podać numer pinu do którego
wpinamy przycisk
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP

# Klawiatura
keyboard = Keyboard(usb_hid.devices)

while True:
    if not button.value: # Sprawdzamy czy przycisk jest wciśnięty
        keyboard.press(Keycode.SPACE) # Jeśli jest to wciskamy spację na klawiaturze
        led.value = True # włączamy led
        print("click") # i wypisujemy informacje w terminalu
    else:
        keyboard.release_all() # Jeśli nie to puszczamy spację (wszystkie wciśnięte
        klawisze)
        led.value = False # i wyłączamy led
        time.sleep(0.1) # zatrzymujemy się na sekundę żeby przycisk lepiej działał
```

Kod do wszystkich etapów zadania można znaleźć na [githubie projektu](#).

Testowanie kontrolera - easy

Udało się nam stworzyć bardzo prosty kontroler, teraz trzeba go przetestować.

Chrome Dino



Press space to play

chrome://dino/
(lub <https://chromedino.com/>)

Flappy Bird



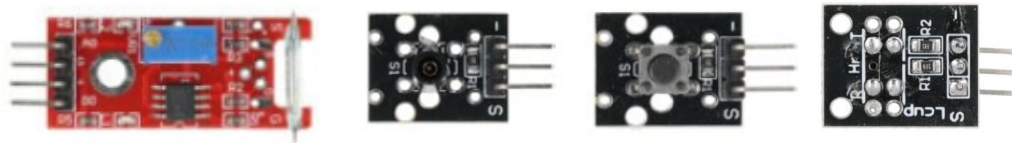
<https://flappybird.io/>

Sensory

To co wyróżni nasze kontrolery to zastosowanie nietypowych sensorów. Jest wiele różnych rodzajów sensorów i większość z nich wymaga zainstalowania konkretnych bibliotek.

Proste cyfrowe

Niektóre podstawowe czujniki możemy obsłużyć biblioteką **digitalio**, po prostu podmieniając przycisk na inny sensor, np. optyczny przełącznik krańcowy, czujnik wstrząsów/pochylenia, czujnik dźwięku czy magnetyczny kontaktron.



Analogowe

Kolejną grupę sensorów możemy obsłużyć biblioteką **analogio**. Użyjemy jej do sensorów które nie zwracają tylko wartości `True/False` (tak jak przycisk) ale jakąś wartość liczbową. Jako przykład użyjemy fotorezystora, czyli sensora natężenia światła, ale można go łatwo zamienić na potencjometr, magnetyczny sensor halla, analogowy czujnik dźwięku, czy czujniki temperatury/ciśnienia/wilgotności.

Będziemy kontynuować kod z poprzednich zadań. Zaczniemy od doimportowania biblioteki `analogio`:

```
import analogio
```

Później zainicjalizujemy sensor:

```
sensor = analogio.AnalogIn(board.A0)
```

Nie wszystkie piny mikrokontrolera potrafią odczytywać takie wartości liczbowe. Sensory analogowe trzeba podłączyć do pinów zaczynających się literą **A** (A0, A1, A2, A3).

Cofniemy się na chwilę i jedyne co zrobimy w pętli to wypisanie wartości jaką zwraca nasz sensor. Dla każdego sensora będzie to zupełnie inny przedział.

```
while True:
    sensor_value = sensor.value
    print("Wartość: ")
    print(sensor_value)
```

Teraz uruchomimy program i sprawdzimy jaki zakres wartości zwraca użyty sensor. Dla fotorezystora odczytują wartości od ~300 kiedy na sensor pada dużo światła, a 900 kiedy

jest zakryty. To znaczy że jeśli chcemy uzależnić od niego wciśnięcie klawisza musimy ustalić jakąś (jedną bądź kilka) wartość graniczną, powyżej której będzie on emulowany.

Do tego celu stworzymy zmienną `THRESHOLD`:

```
THRESHOLD = 600 # Dopasuj wartość do odczytów sensora
```

I do niej będziemy porównywać aktualny odczyt sensora. Jeśli jest powyżej - wciskamy spację, jeśli poniżej - nic się nie dzieje. Zaktualizujemy pętlę `while`:

```
while True:
    light_level = sensor.value # Read magnetic field level
    print("Light level:", light_level)

    if light_level > THRESHOLD: # If field strength is below threshold
        led.value = True # Turn LED on
        keyboard.press(Keycode.SPACE) # Emulate space key press
        print("Pressed SPACE")
    else:
        led.value = False # Turn LED off
        keyboard.release_all()

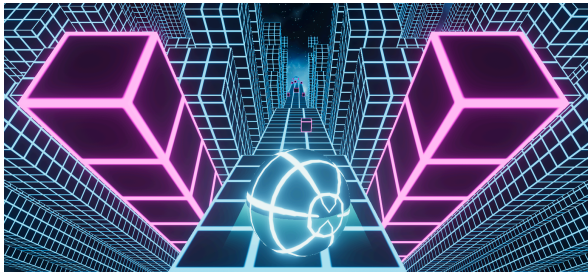
    time.sleep(0.1) # Small delay to debounce
```

Trochę bardziej inteligentne

Pozostają sensory które wymagają specjalnych własnych bibliotek i wiedzy, jak np. ultradźwiękowy sensor odległości, sensor kolorów, czujnik pulsu, czy enkodery. Ich działanie wytłumaczymy indywidualnie. D

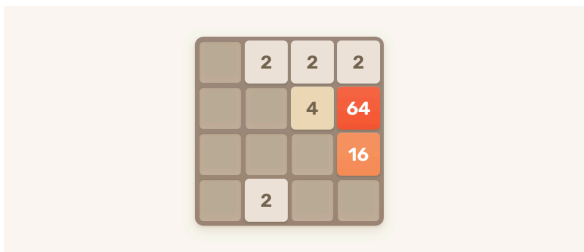
Testowanie kontrolera - normal

Slope 3



<https://slope3.com> (2x input)

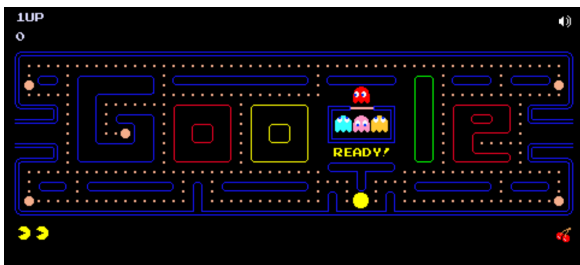
2048



<https://play2048.co> (4x input)

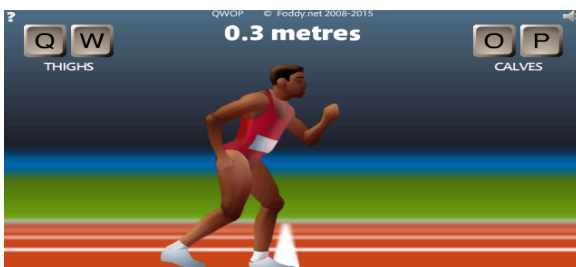
Testowanie kontrolera - **hardcore**

Pacman



<https://www.google.com/logos/2010/pacman10-i.html> (4 input)

QWOP



<https://www.foddy.net/Athletics.html> (4 input)