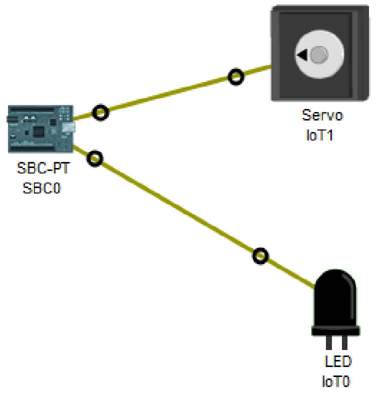
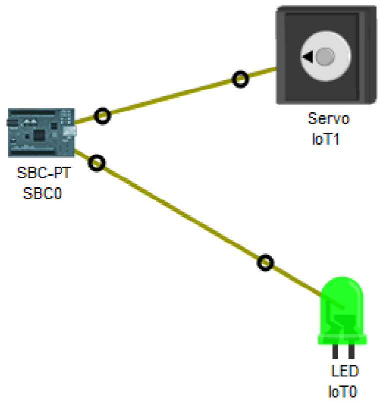
Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki	
Laboratorium IoT Rozproszone sieci sensoryczne	
Grupa: 3ID14B	Laboratorium 3
Data: 15.11.2018	Lesiak Karol

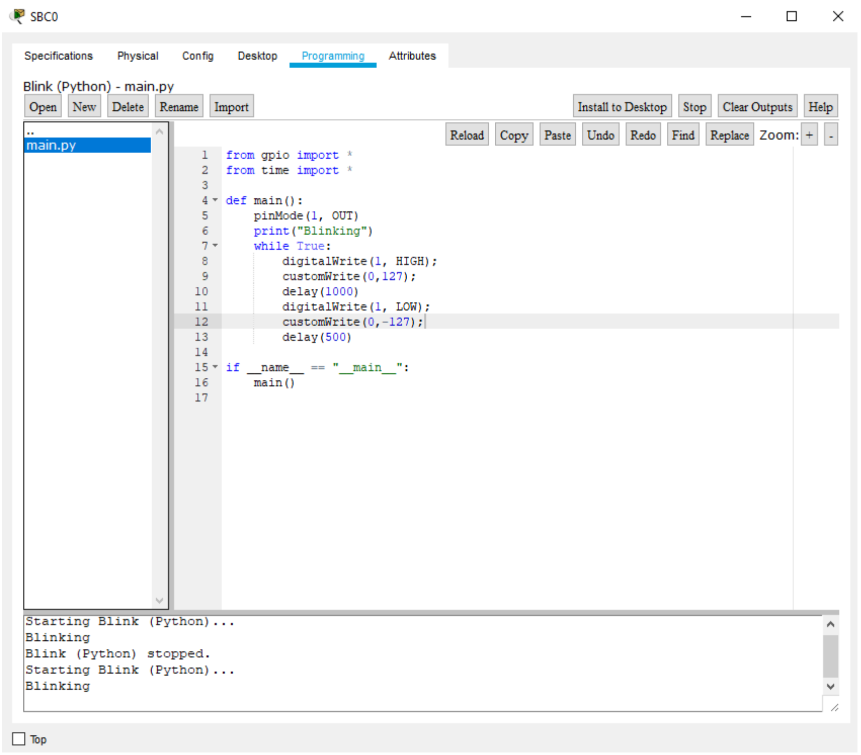
Packet Tracer – Simulating IoT Devices

Topologia:



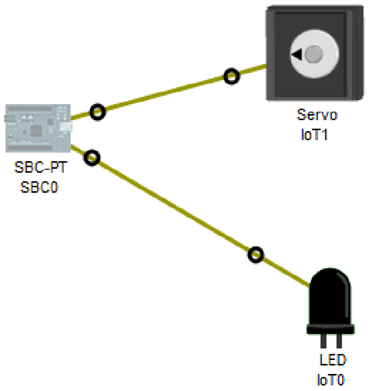


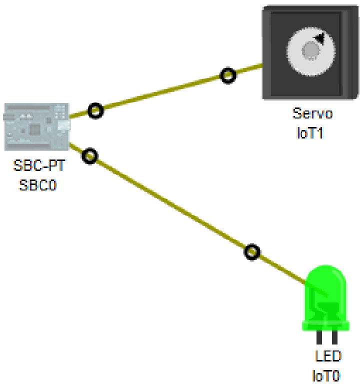
Modyfikacja kodu:



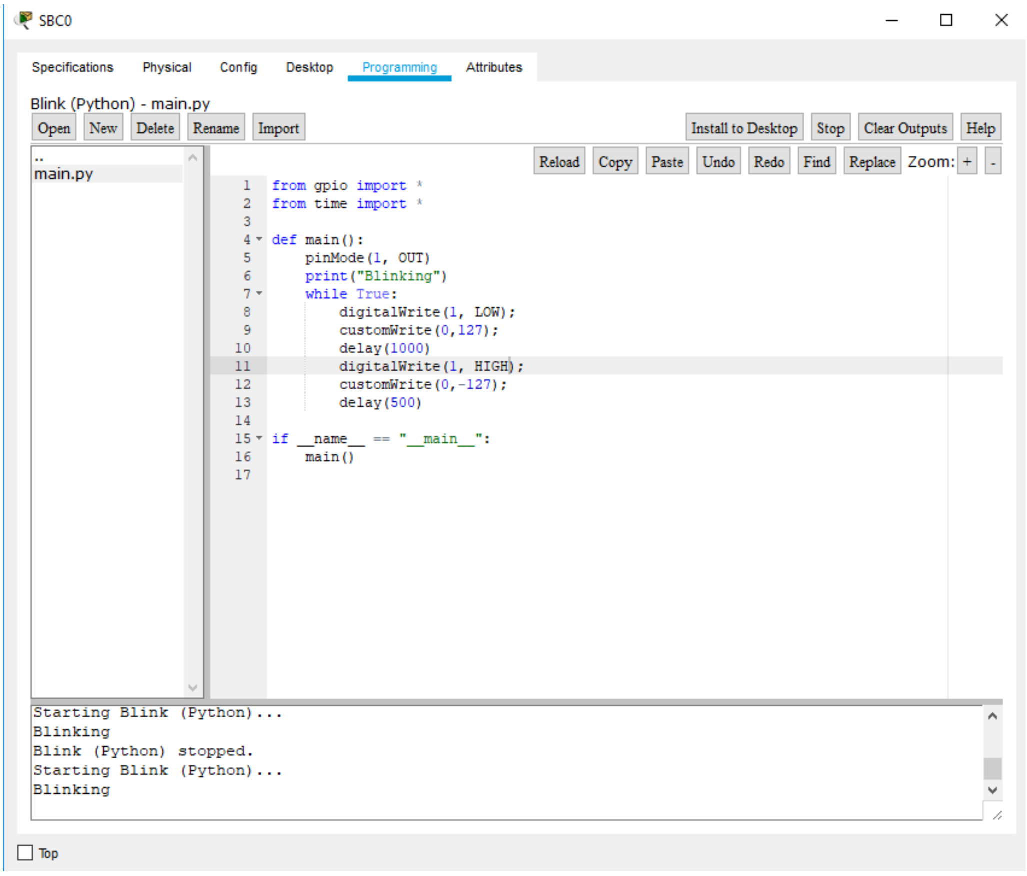
```
from gpio import *
from time import *

def main():
    pinMode(1, OUT)
    print("Blinking")
    while True:
        digitalWrite(1, HIGH);
        customWrite(0,127);
        delay(1000)
        digitalWrite(1, LOW);
        customWrite(0,-127);
        delay(500)

if __name__ == "__main__":
    main()

```

```
Starting Blink (Python)...
Blinking
Blink (Python) stopped.
Starting Blink (Python)...
Blinking
```





Aby servo obracało się w przeciwnym kierunku należy zmodyfikować kod programu, dokładnie w 8 i 11 linii kodu zamieniamy ze sobą wartości HIGH oraz LOW.

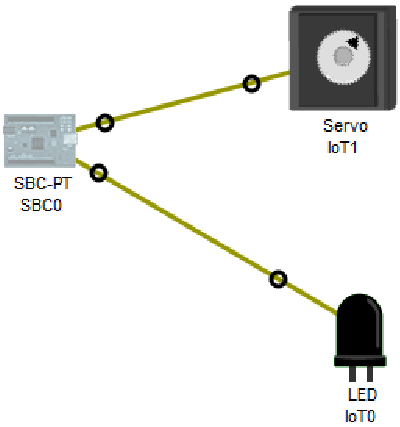


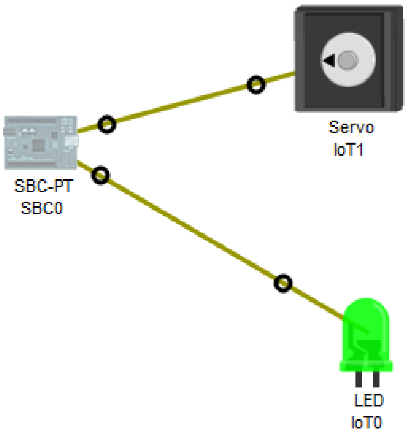
Blink (Python) - main.py

..
main.py

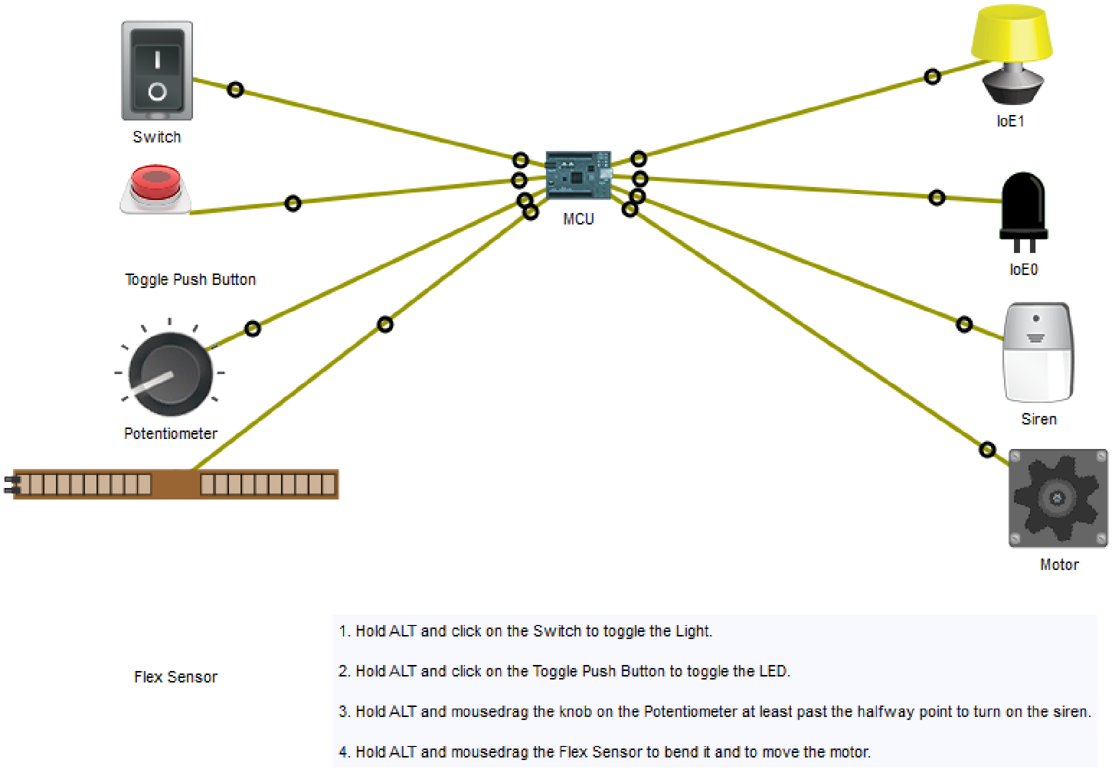
```
 1  from gpio import *
 2  from time import *
 3
 4  def main():
 5      pinMode(1, OUT)
 6      print("Blinking")
 7      while True:
 8          digitalWrite(1, LOW);
 9          customWrite(0,127);
10          delay(1000)
11          digitalWrite(1, HIGH);
12          customWrite(0,-127);
13          delay(500)
14
15  if __name__ == "__main__":
16      main()
17
```

```
Starting Blink (Python)...
Blinking
Blink (Python) stopped.
Starting Blink (Python)...
Blinking
```





Packet Tracer – Sensors and the PT Microcontroler

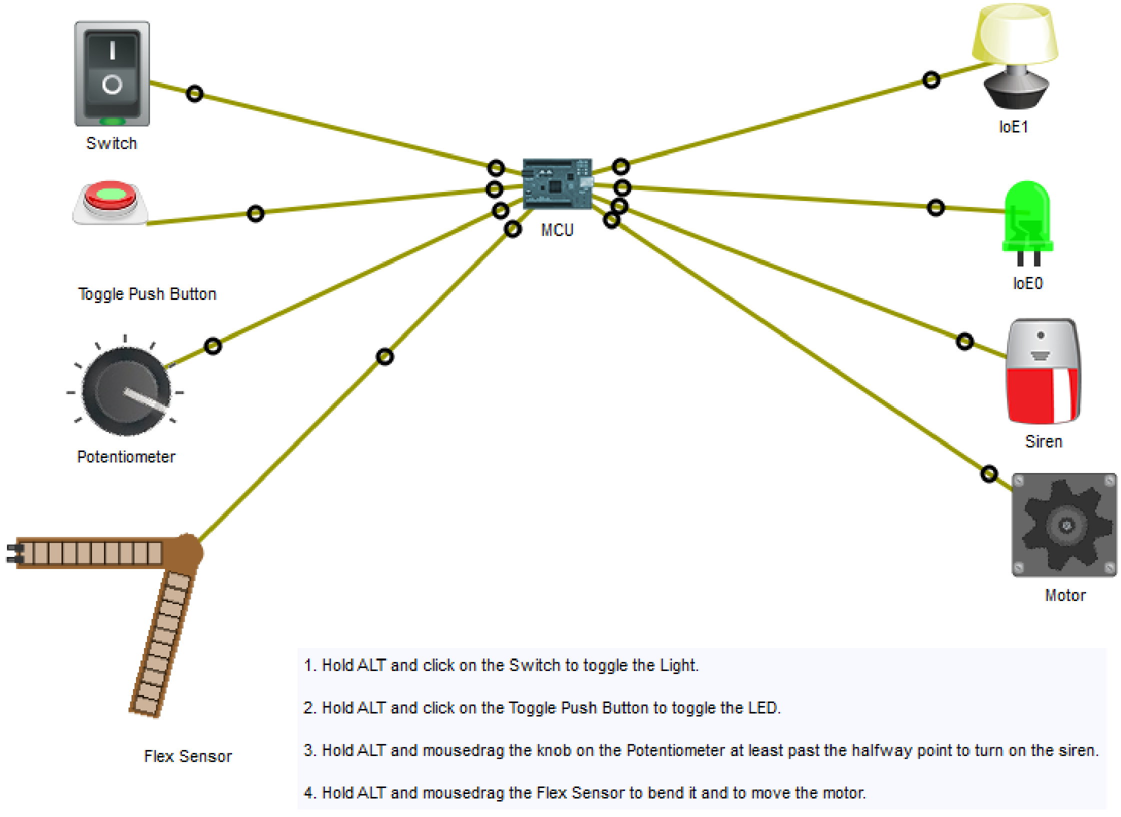


1. Hold ALT and click on the Switch to toggle the Light.

2. Hold ALT and click on the Toggle Push Button to toggle the LED.

3. Hold ALT and mousedrag the knob on the Potentiometer at least past the halfway point to turn on the siren.

4. Hold ALT and mousedrag the Flex Sensor to bend it and to move the motor.



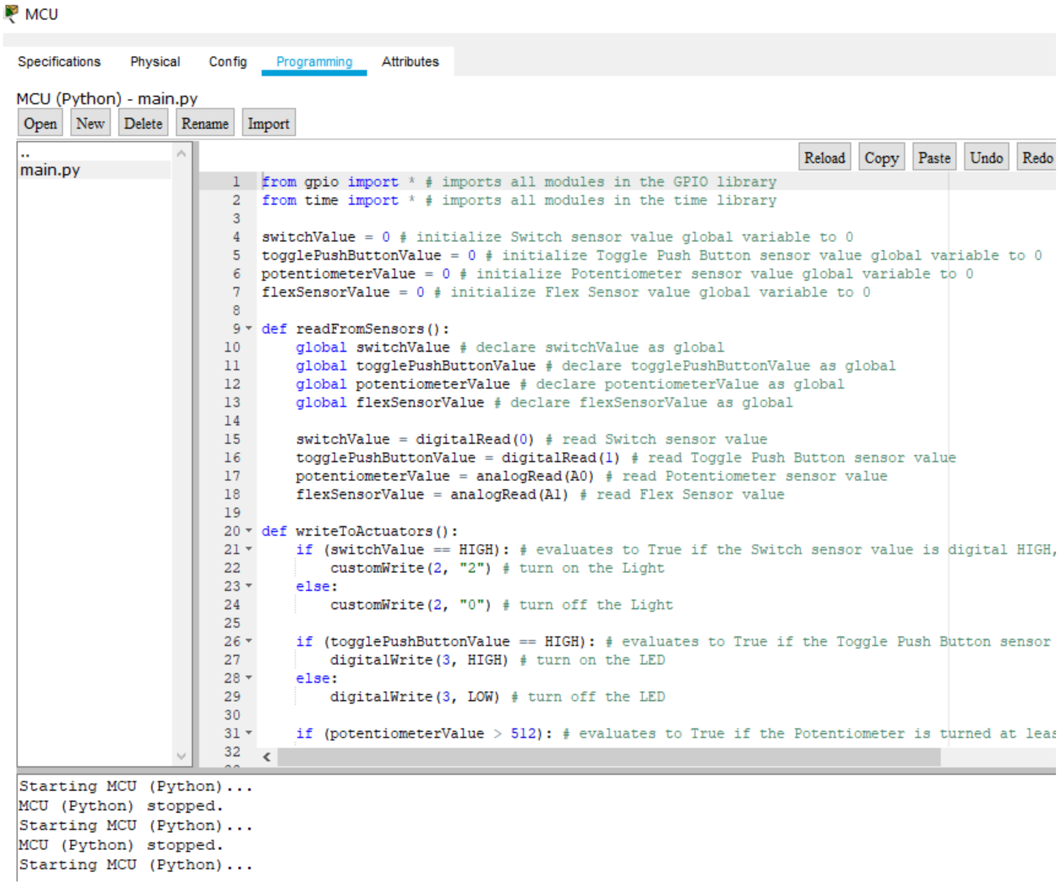
1. Hold ALT and click on the Switch to toggle the Light.

2. Hold ALT and click on the Toggle Push Button to toggle the LED.

3. Hold ALT and mousedrag the knob on the Potentiometer at least past the halfway point to turn on the siren.

4. Hold ALT and mousedrag the Flex Sensor to bend it and to move the motor.

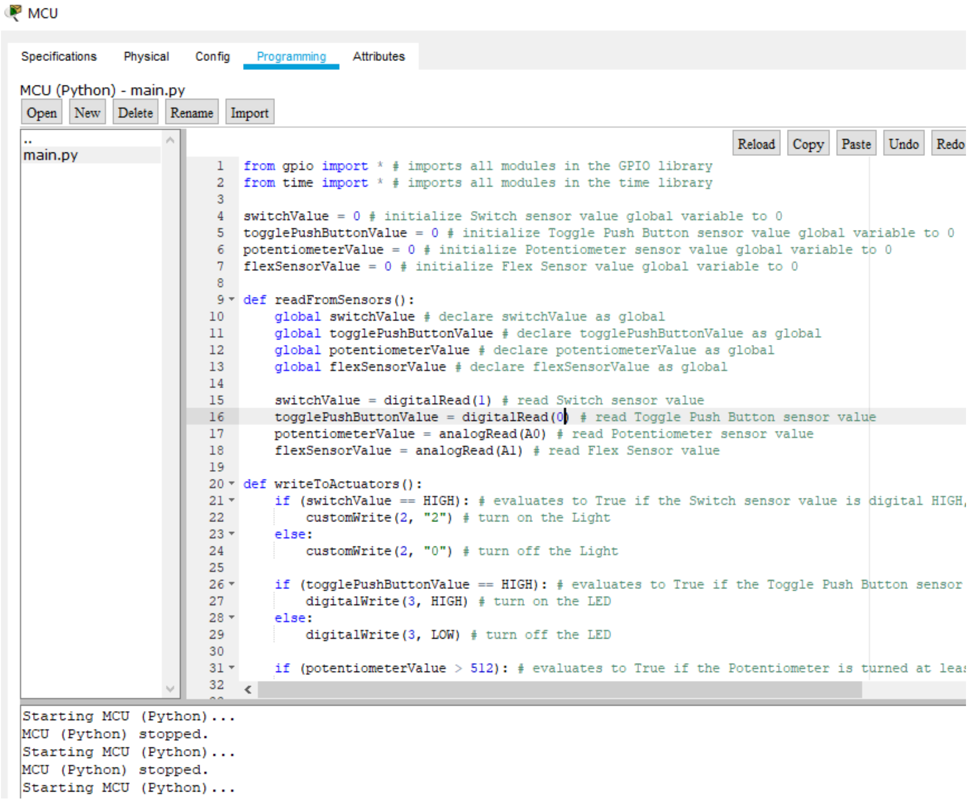
Kod MCU:



```
from gpio import * # imports all modules in the GPIO library
from time import * # imports all modules in the time library

switchValue = 0 # initialize Switch sensor value global variable to 0
togglePushButtonValue = 0 # initialize Toggle Push Button sensor value global variable to 0
potentiometerValue = 0 # initialize Potentiometer sensor value global variable to 0
flexSensorValue = 0 # initialize Flex Sensor value global variable to 0

def readFromSensors():
    global switchValue # declare switchValue as global
    global togglePushButtonValue # declare togglePushButtonValue as global
    global potentiometerValue # declare potentiometerValue as global
    global flexSensorValue # declare flexSensorValue as global

    switchValue = digitalRead(0) # read Switch sensor value
    togglePushButtonValue = digitalRead(1) # read Toggle Push Button sensor value
    potentiometerValue = analogRead(A0) # read Potentiometer sensor value
    flexSensorValue = analogRead(A1) # read Flex Sensor value

def writeToActuators():
    if (switchValue == HIGH): # evaluates to True if the Switch sensor value is digital HIGH,
        customWrite(2, "2") # turn on the Light
    else:
        customWrite(2, "0") # turn off the Light

    if (togglePushButtonValue == HIGH): # evaluates to True if the Toggle Push Button sensor
        digitalWrite(3, HIGH) # turn on the LED
    else:
        digitalWrite(3, LOW) # turn off the LED

    if (potentiometerValue > 512): # evaluates to True if the Potentiometer is turned at leas
```

```
Starting MCU (Python)...
MCU (Python) stopped.
Starting MCU (Python)...
MCU (Python) stopped.
Starting MCU (Python)...
```

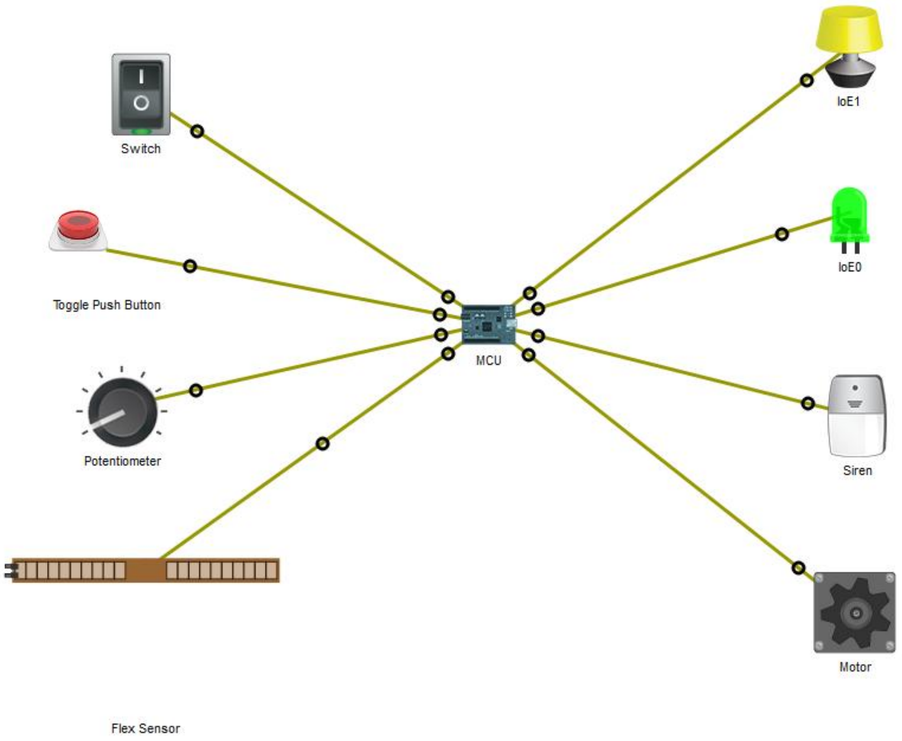
Modyfikacja kodu:

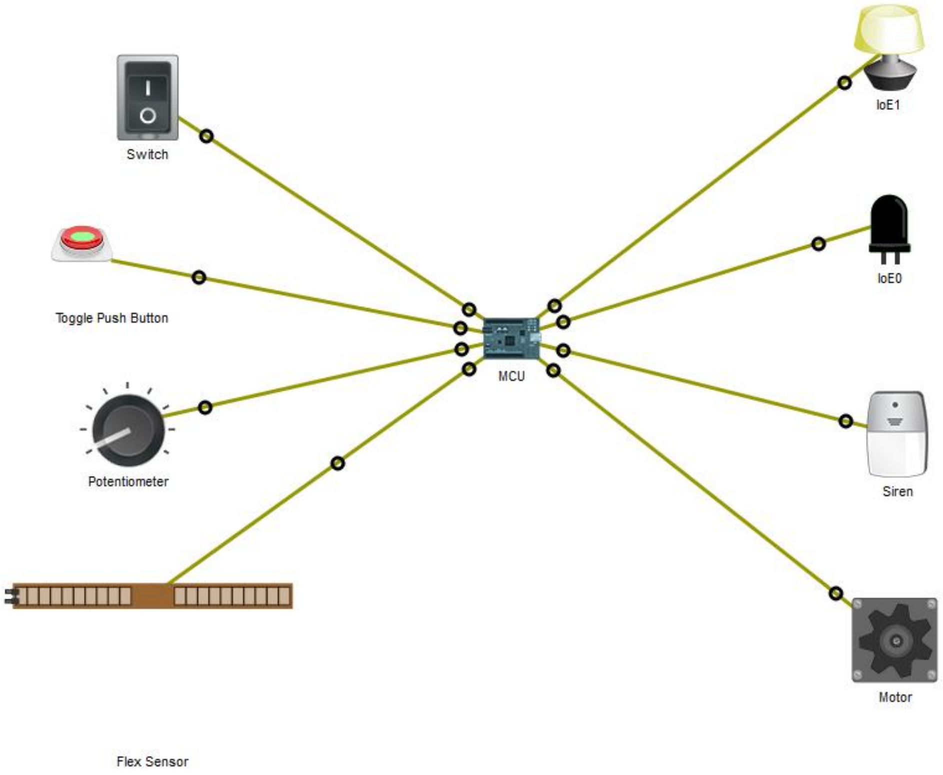


```
from gpio import * # imports all modules in the GPIO library
from time import * # imports all modules in the time library

switchValue = 0 # initialize Switch sensor value global variable to 0
togglePushButtonValue = 0 # initialize Toggle Push Button sensor value global variable to 0
potentiometerValue = 0 # initialize Potentiometer sensor value global variable to 0
flexSensorValue = 0 # initialize Flex Sensor value global variable to 0

def readFromSensors():
    global switchValue # declare switchValue as global
    global togglePushButtonValue # declare togglePushButtonValue as global
    global potentiometerValue # declare potentiometerValue as global
    global flexSensorValue # declare flexSensorValue as global

    switchValue = digitalRead(1) # read Switch sensor value
    togglePushButtonValue = digitalRead(0) # read Toggle Push Button sensor value
    potentiometerValue = analogRead(A0) # read Potentiometer sensor value
    flexSensorValue = analogRead(A1) # read Flex Sensor value

def writeToActuators():
    if (switchValue == HIGH): # evaluates to True if the Switch sensor value is digital HIGH,
        customWrite(2, "2") # turn on the Light
    else:
        customWrite(2, "0") # turn off the Light

    if (togglePushButtonValue == HIGH): # evaluates to True if the Toggle Push Button sensor
        digitalWrite(3, HIGH) # turn on the LED
    else:
        digitalWrite(3, LOW) # turn off the LED

    if (potentiometerValue > 512): # evaluates to True if the Potentiometer is turned at lea
```

```
Starting MCU (Python)...
MCU (Python) stopped.
Starting MCU (Python)...
MCU (Python) stopped.
Starting MCU (Python)...
```

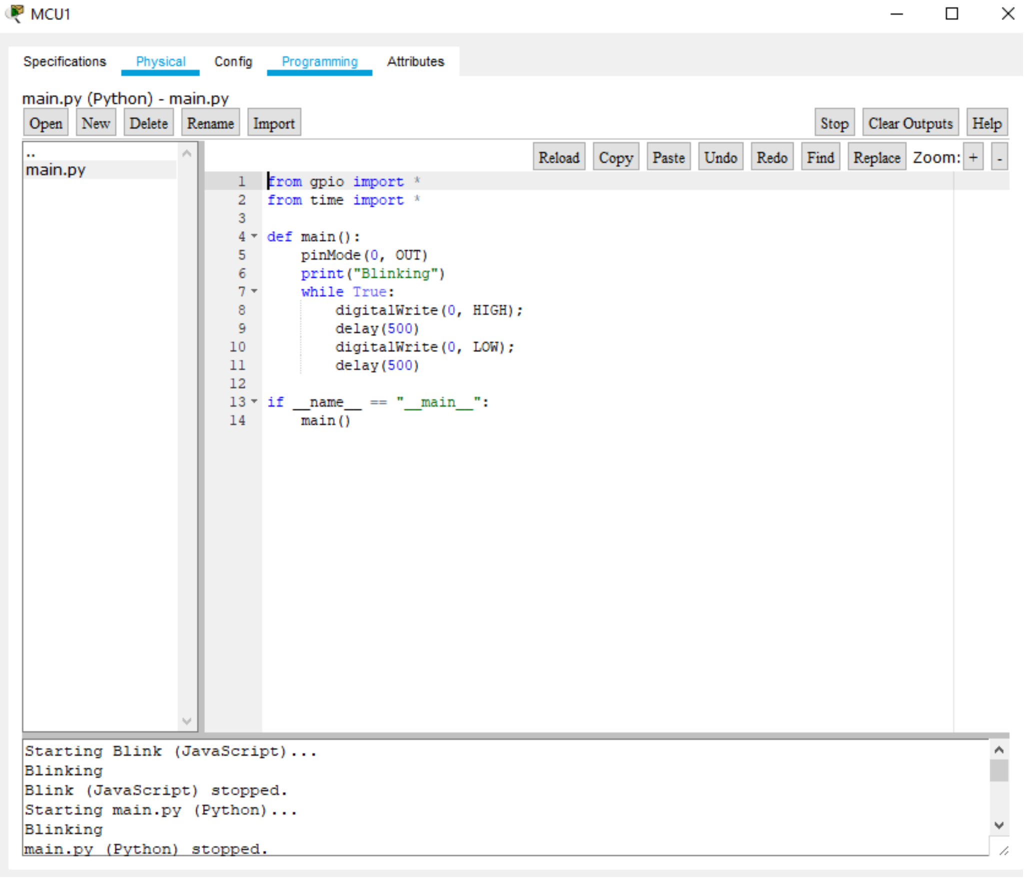
Aby przycisk kontrolował Lampe a przełącznik kontrolował LED należy w kodzie programu w linii 15 zmienić wartość switchValue = digitalRead(0) na 1 i w linii 16 togglePushButtonValue = digitalRead(1) na 0.





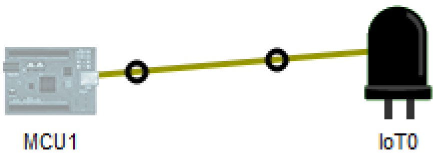
Wyzwanie 1: Migająca dioda

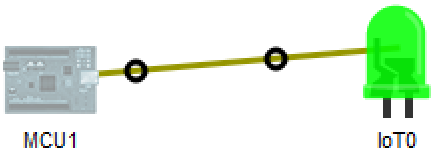
modyfikowanie kodu – MCU



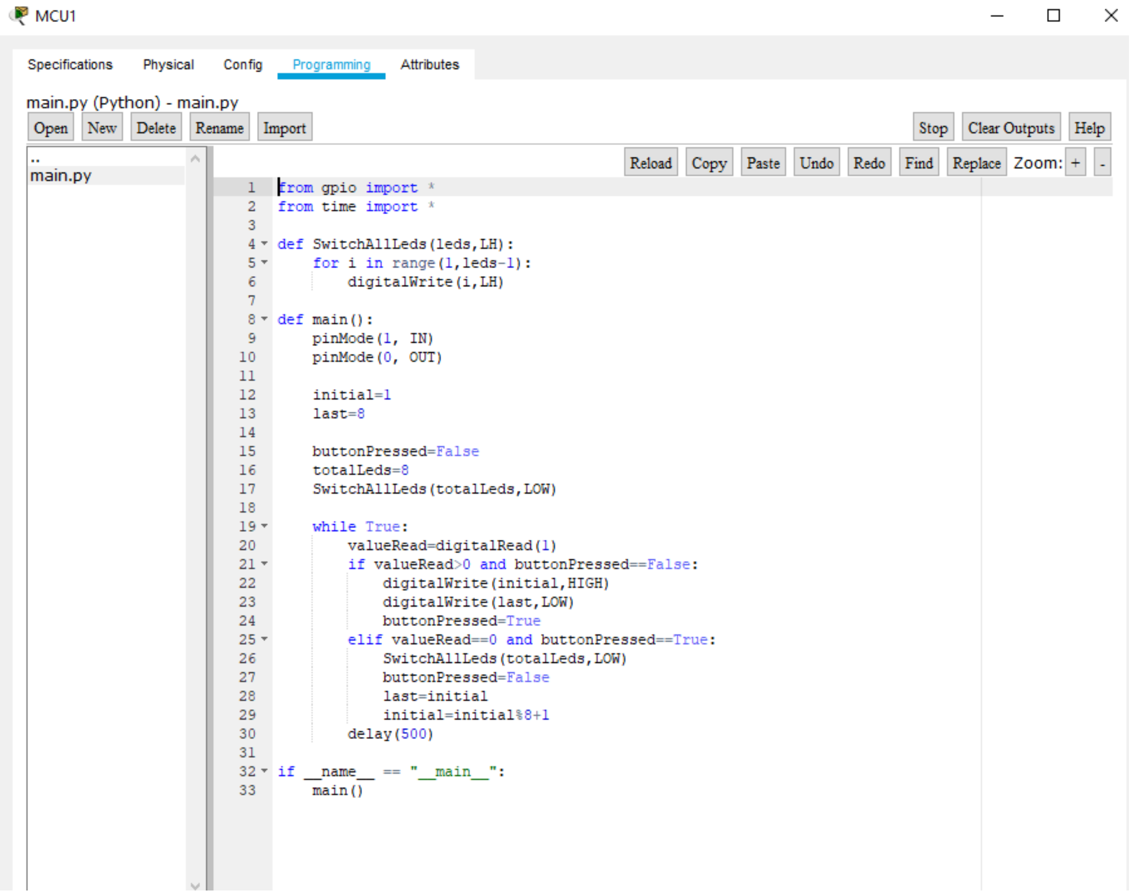
```
from gpio import *
from time import *

def main():
    pinMode(0, OUT)
    print("Blinking")
    while True:
        digitalWrite(0, HIGH);
        delay(500)
        digitalWrite(0, LOW);
        delay(500)

if __name__ == "__main__":
    main()
```

```
Starting Blink (JavaScript)...
Blinking
Blink (JavaScript) stopped.
Starting main.py (Python)...
Blinking
main.py (Python) stopped.
```

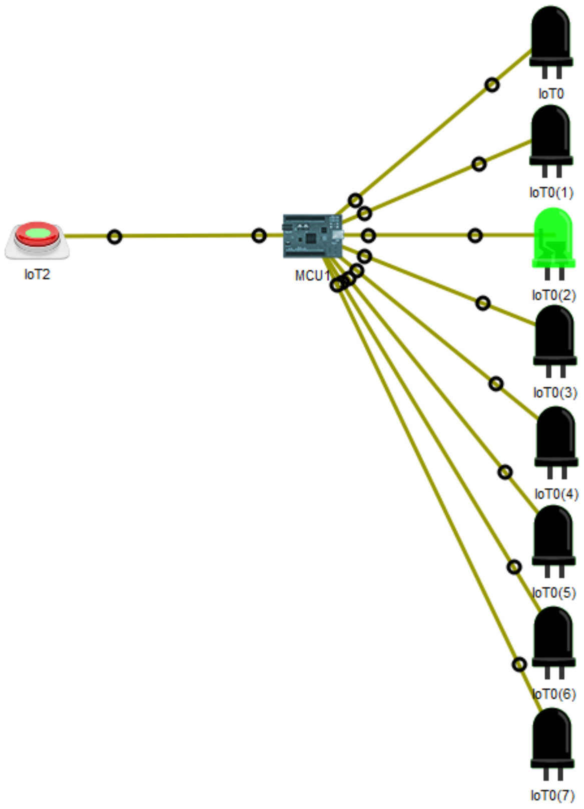




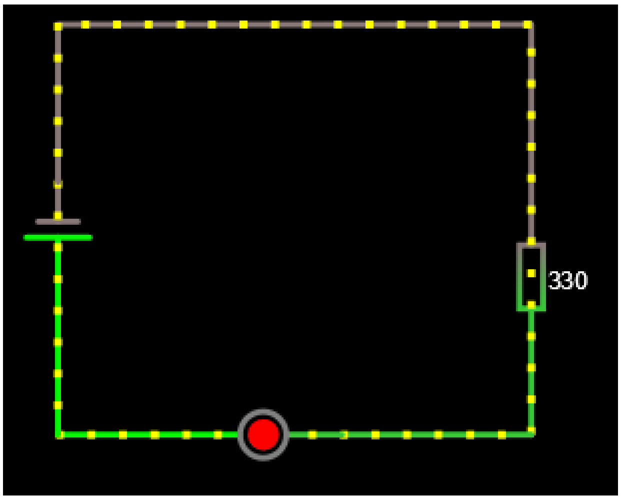
Wyzwanie 2 Podświetlenie po kolei 8 LED po każdym naciśnięciu przycisku



```python
from gpio import *
from time import *

def SwitchAllLeds(leds,LH):
    for i in range(1,leds-1):
        digitalWrite(i,LH)

def main():
    pinMode(1, IN)
    pinMode(0, OUT)

    initial=1
    last=8

    buttonPressed=False
    totalLeds=8
    SwitchAllLeds(totalLeds,LOW)

    while True:
        valueRead=digitalRead(1)
        if valueRead>0 and buttonPressed==False:
            digitalWrite(initial,HIGH)
            digitalWrite(last,LOW)
            buttonPressed=True
        elif valueRead==0 and buttonPressed==True:
            SwitchAllLeds(totalLeds,LOW)
            buttonPressed=False
            last=initial
            initial=initial%8+1
        delay(500)

if __name__ == "__main__":
    main()
```



Lab - The Digital Oscilloscope



źródło napięcia:
I = 9.75 mA
Vd = 5 V
(R = 512.73 Ω)
P = -48.76 mW

rezystor
I = 9.75 mA
Vd = 3.22 V
R = 330 Ω
P = 31.38 mW

LED
I = 9.75 mA
Vd = 1.78 V
P = 17.38 mW

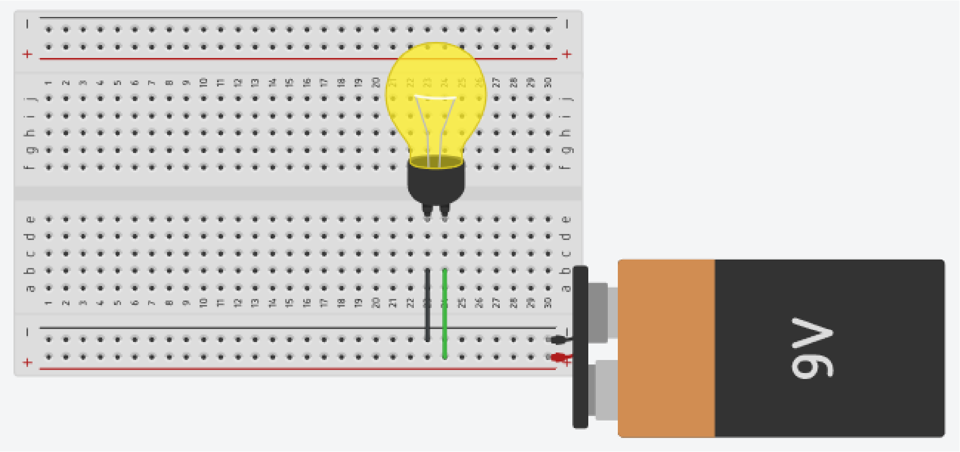
Napięcie na LED: 1.78 V

Napięcie na rezystorze: 3.22 V

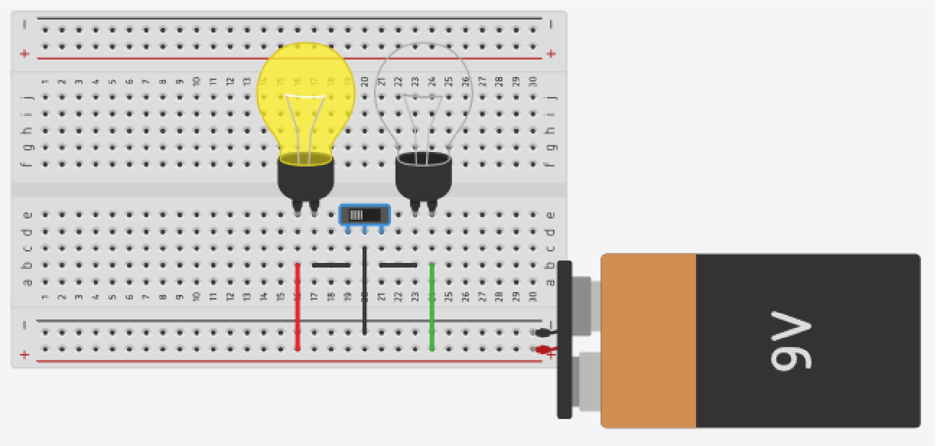
Napięcie na baterii: 5 V

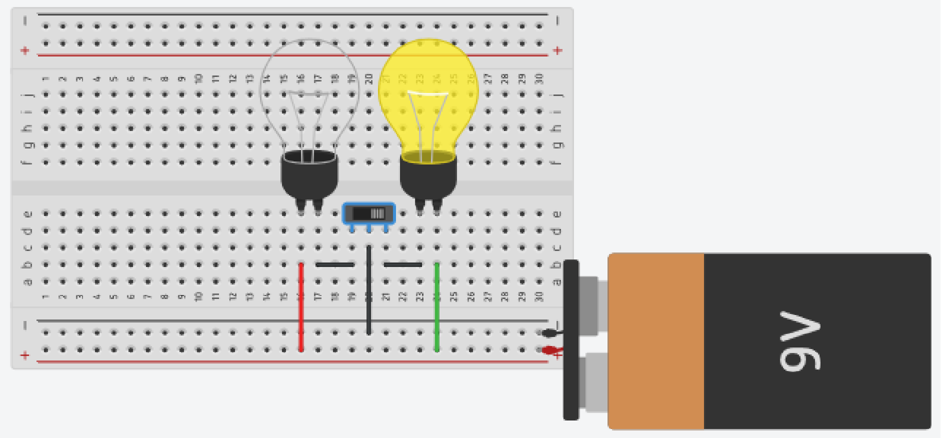
Lab - Designing a Circuit from Start to Finish

Obwód 1:



Obwód 2:





Zastępujemy przełącznik suwakowy potencjometrem co pozwala nam na regulacje napięcia dostarczanego do żarówek co w przypadku przełącznika było niemożliwe.