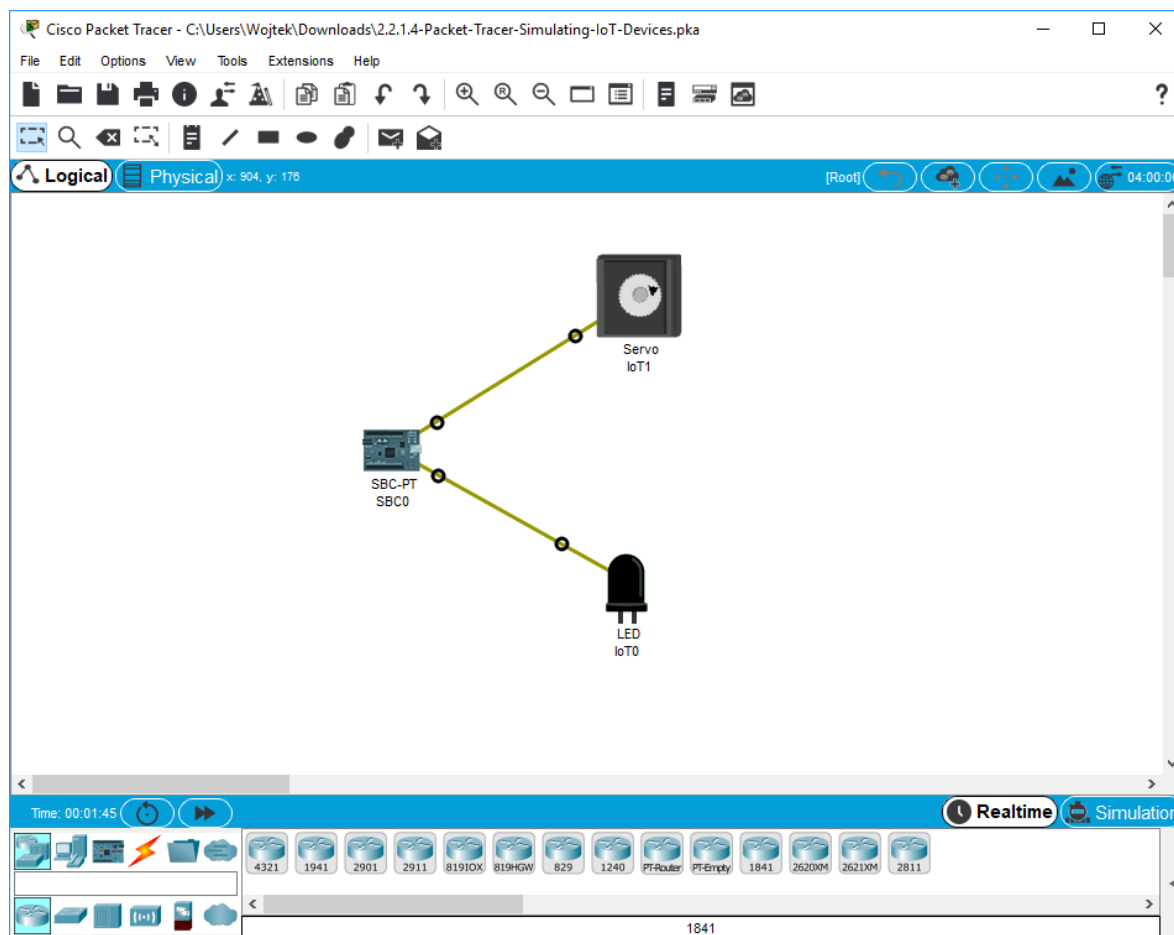


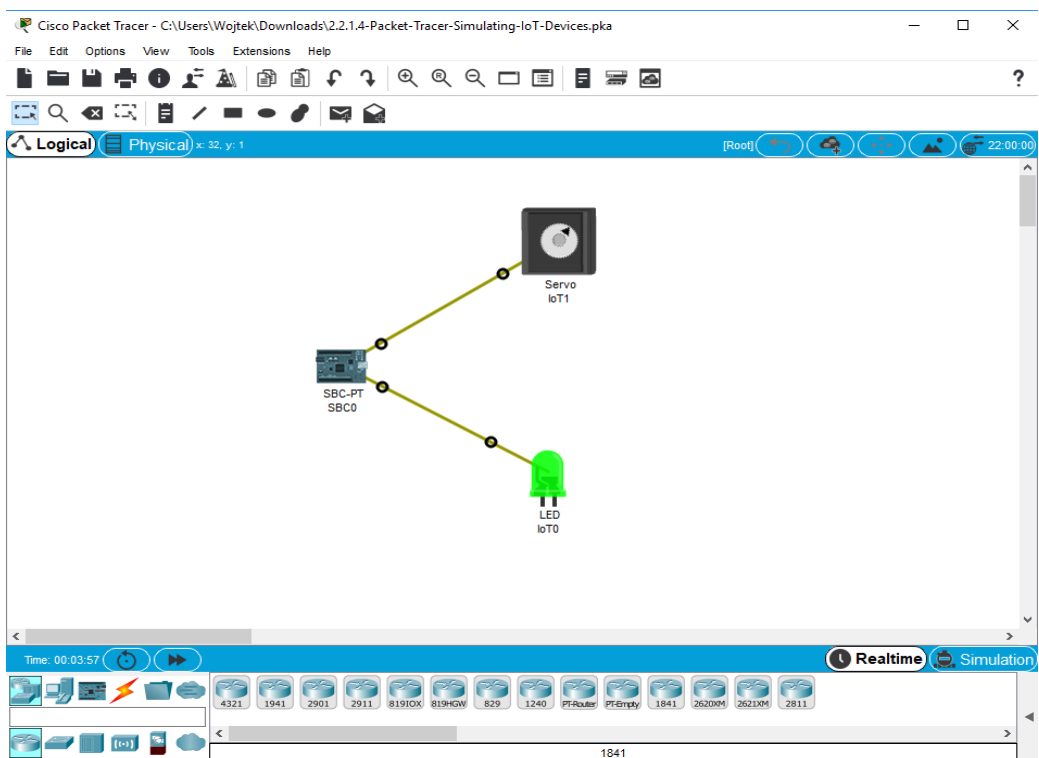
<p style="text-align: center;">Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki</p>	
<p style="text-align: center;">Laboratorium : Technologie IoT rozproszone sieci sensor</p>	
<p>Moduł 2</p>	<p>Autor: Wojciech Harabin Damian Domański Grupa: 3ID15B</p>
<p>Numer laboratorium: 2</p>	<p>Data wykonania: 15.11.2018</p>

Packet Tracer – Simulating IoT Devices

Topologia:



Uruchamianie obwodu:



Modyfikacja domyślnego programu:

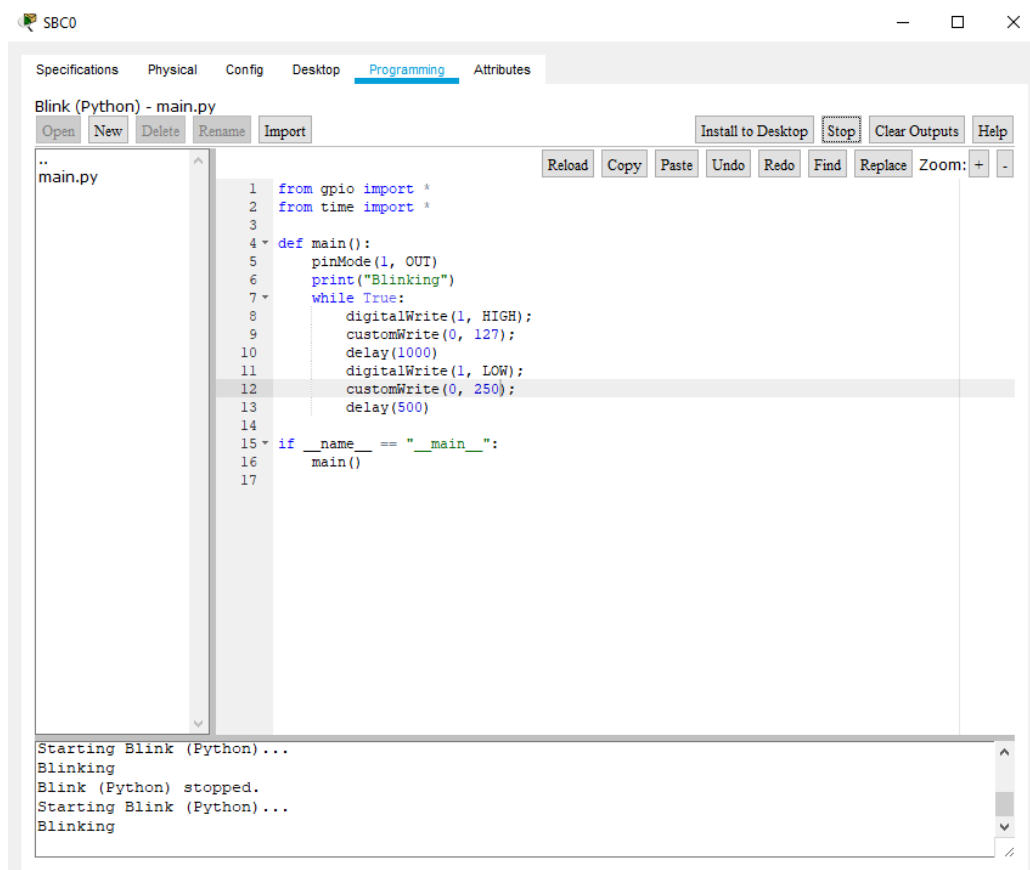
The screenshot shows the SBC0 Programming window. The main tab is 'Programming', and the file 'main.py' is open. The script is a Python program for a Blinking LED. The code is as follows:

```
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, HIGH);
9         customWrite(0, 127);
10        delay(1000)
11        digitalWrite(1, LOW);
12        customWrite(0, -127);
13        delay(500)
14
15 if __name__ == "__main__":
16     main()
17
```

The output window at the bottom shows the following text:

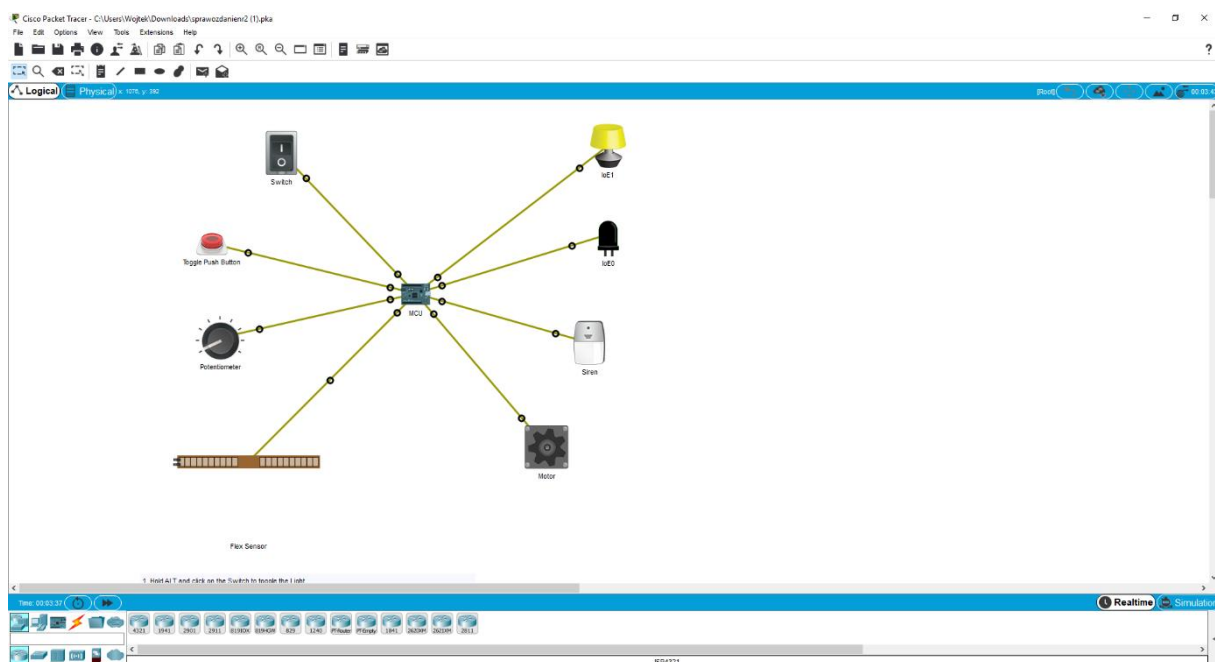
```
Starting Blink (Python)...
Blinking
Blink (Python) stopped.
Starting Blink (Python)...
Blinking
```

Aby serwo obracało się w przeciwnym kierunku należy zmodyfikować program tzn. w zakładce programing w 12 linie kodu programu zastępujemy wartość -127 dowolną wartością większą od 127 co sprawi że kierunek obracania serwo zmieni się.

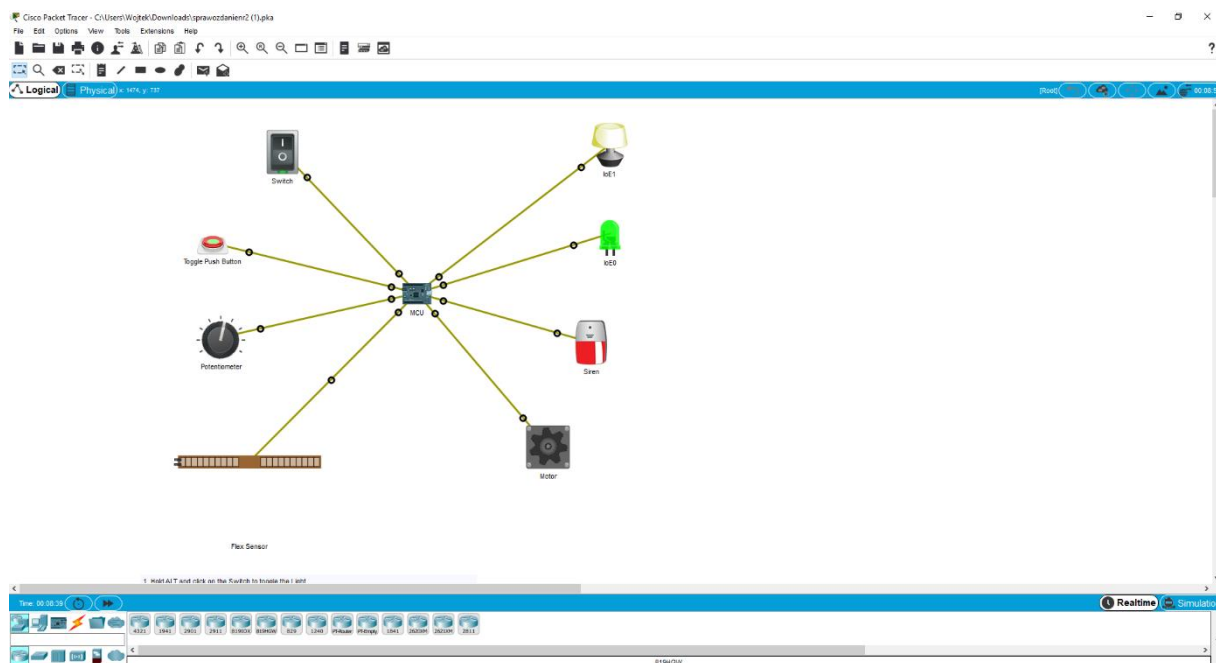


Packet Tracer - Sensors and the PT Microcontroller

Topologia:



Czujniki i MCU PT:



Trzymając przycisk Alt klikamy na poszczególny komponent znajdujący się po lewej stronie aby zasłała interakcja między nim a komponentem znajdującym się po prawej stronie.

Programowanie MCU:

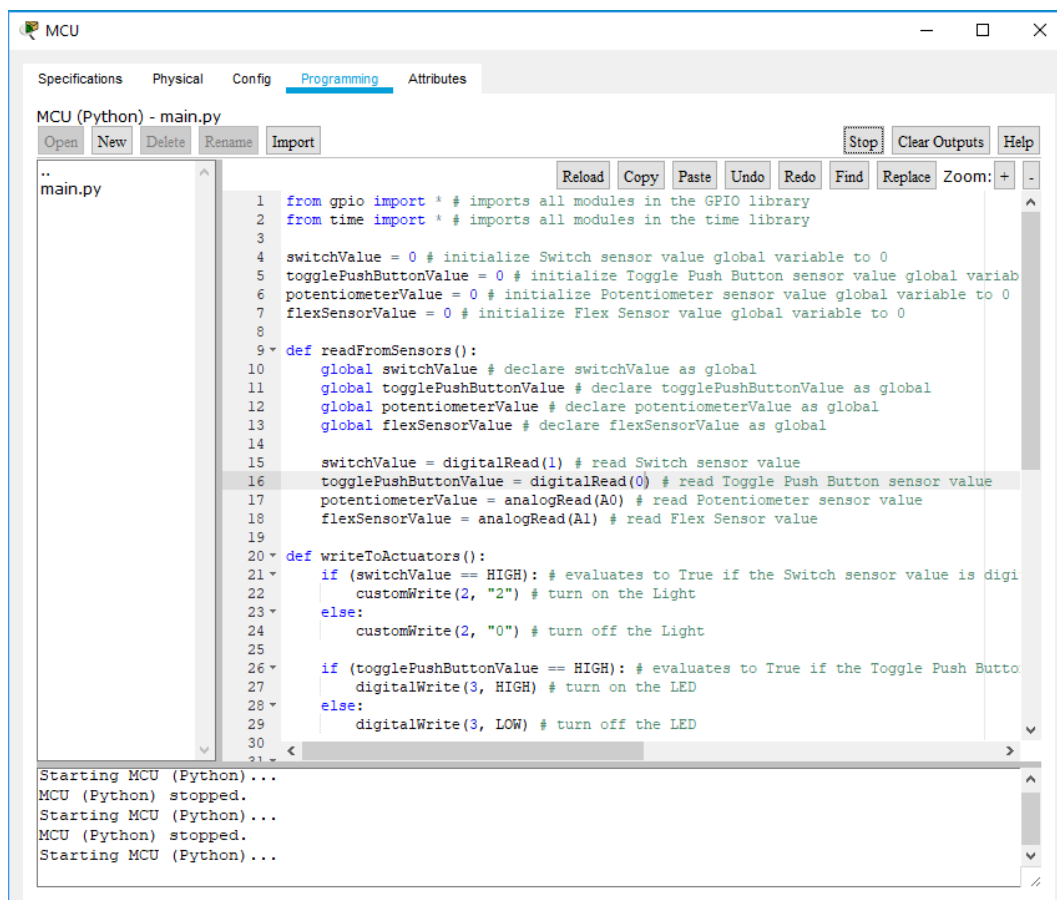
The screenshot shows the MCU Programming window in Cisco Packet Tracer. The window has tabs for Specifications, Physical, Config, Programming, and Attributes. The Programming tab is active, showing a Python script for the MCU. The script is titled "MCU (Python) - main.py" and contains the following code:

```
1 from gpio import * # imports all modules in the GPIO library
2 from time import * # imports all modules in the time library
3
4 switchValue = 0 # initialize Switch sensor value global variable to 0
5 togglePushButtonValue = 0 # initialize Toggle Push Button sensor value global variable to 0
6 potentiometerValue = 0 # initialize Potentiometer sensor value global variable to 0
7 flexSensorValue = 0 # initialize Flex Sensor value global variable to 0
8
9 def readFromSensors():
10     global switchValue # declare switchValue as global
11     global togglePushButtonValue # declare togglePushButtonValue as global
12     global potentiometerValue # declare potentiometerValue as global
13     global flexSensorValue # declare flexSensorValue as global
14
15     switchValue = digitalRead(0) # read Switch sensor value
16     togglePushButtonValue = digitalRead(1) # read Toggle Push Button sensor value
17     potentiometerValue = analogRead(A0) # read Potentiometer sensor value
18     flexSensorValue = analogRead(A1) # read Flex Sensor value
19
20 def writeToActuators():
21     if (switchValue == HIGH): # evaluates to True if the Switch sensor value is digitalRead(0) == HIGH
22         customWrite(2, "2") # turn on the Light
23     else:
24         customWrite(2, "0") # turn off the Light
25
26     if (togglePushButtonValue == HIGH): # evaluates to True if the Toggle Push Button sensor value is digitalRead(1) == HIGH
27         digitalWrite(3, HIGH) # turn on the LED
28     else:
29         digitalWrite(3, LOW) # turn off the LED
30
31 while True:
32     readFromSensors()
33     writeToActuators()
34     time.sleep(0.1)
```

At the bottom of the window, there is a console area showing the following output:

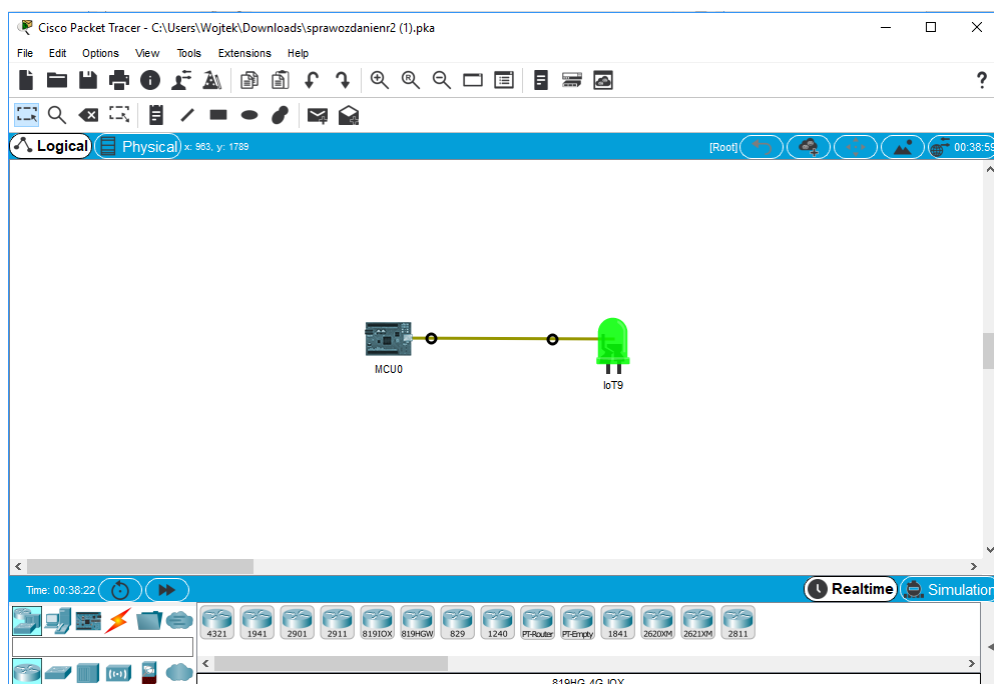
```
Starting MCU (Python)...
MCU (Python) stopped.
Starting MCU (Python)...
```

Modyfikacja kodu:



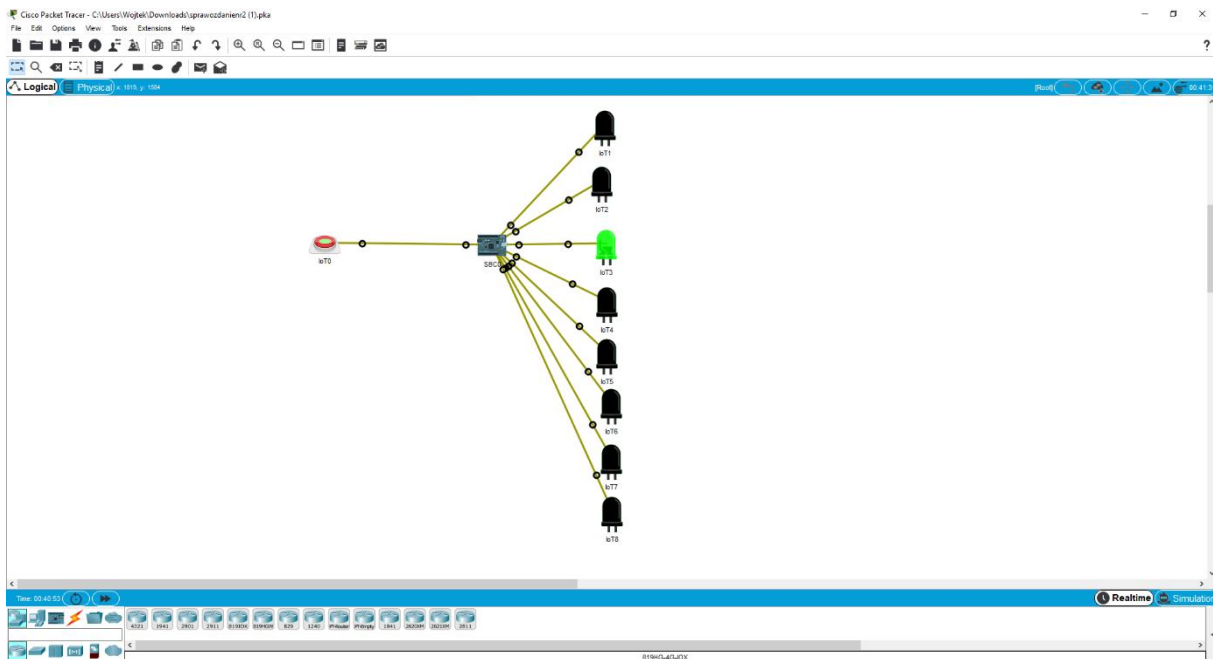
Aby przełącznik kontrolował lampę LED a przycisk do naciskania kontrolował lampę należy w zakładce programing w linie 15 zamienić wartość w nawiasach `switchValue = digitalRead(0)` na 1 i w linie 16 `togglePushbuttonValue = digitalRead(1)` na 0.

Wyzwanie 1: Migająca dioda



Wyzwanie 2: Podświetlanie kolejno jedną z ośmiu diod LED za każdym naciśnięciem przycisku

Topologia:



Programowanie MCU:

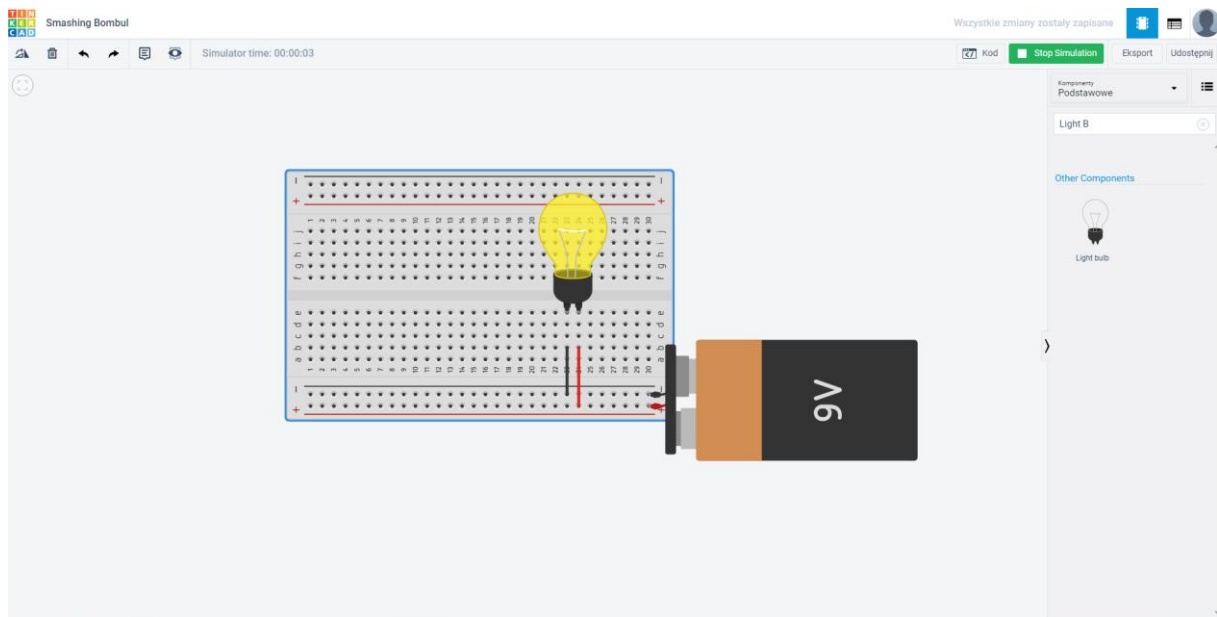
```
Specifications Physical Config Desktop Programming Attributes
Blink (Python) - main.py
Open New Delete Rename Import Install to Desktop Stop Clear Outputs Help
Reload Copy Paste Undo Redo Find Replace Zoom: + -
..
main.py
1 from gpio import *
2 from time import *
3
4 def SwitchAllLeds(leds,LH):
5     for i in range(1,leds-1):
6         digitalWrite(i,LH)
7
8 def main():
9     pinMode(1, OUT)
10    pinMode(0, IN)
11
12    initial=1
13    last=8
14
15    buttonPressed=False
16    totalLeds=8
17    SwitchAllLeds(totalLeds,LOW)
18
19    while True:
20        valueRead=digitalRead(0)
21        if valueRead>0 and buttonPressed==False:
22            digitalWrite(initial, HIGH)
23            digitalWrite(last, LOW)
24            buttonPressed=True
25        elif valueRead==0 and buttonPressed==True:
26            SwitchAllLeds(totalLeds,LOW)
27            buttonPressed=False
28            last=initial
29            initial=initial%8+1
30            delay(500)
31
32 if __name__ == "__main__":
33     main()
34
```

Starting Blink (Python)...

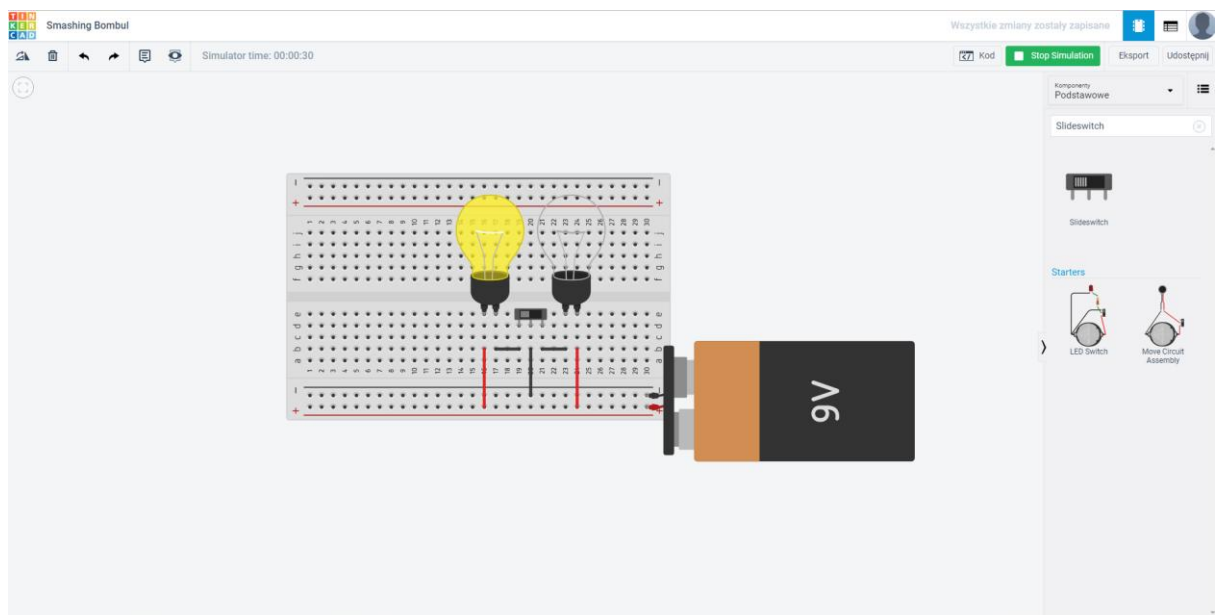
☐ Top

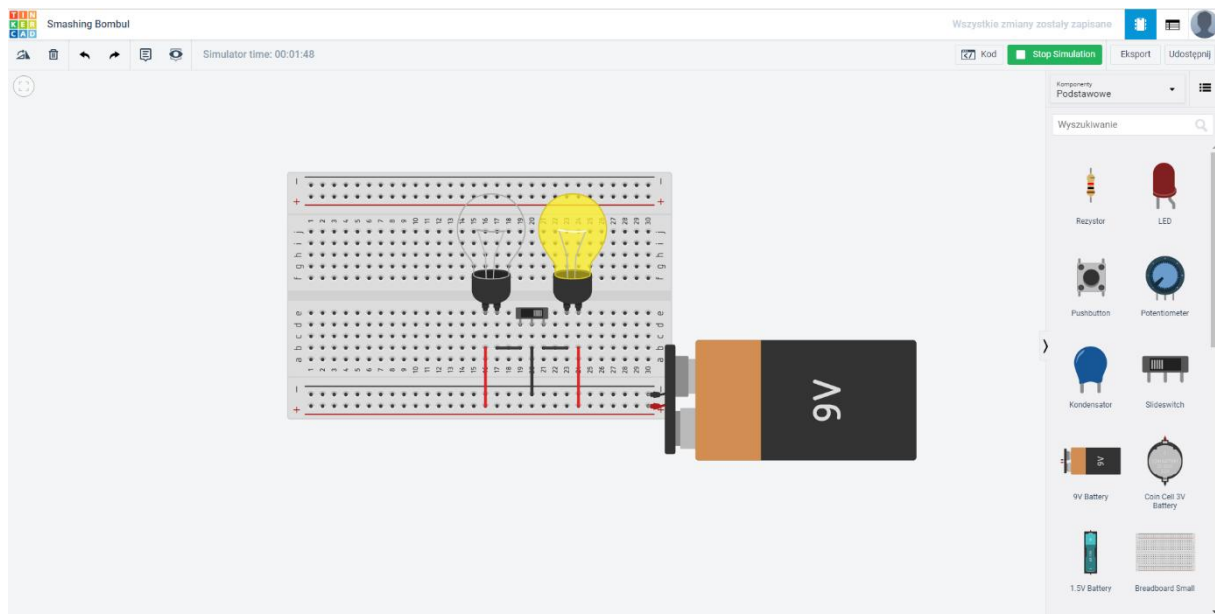
Lab – Designing a Circuit from Start to Finish

Obwód nr.1:

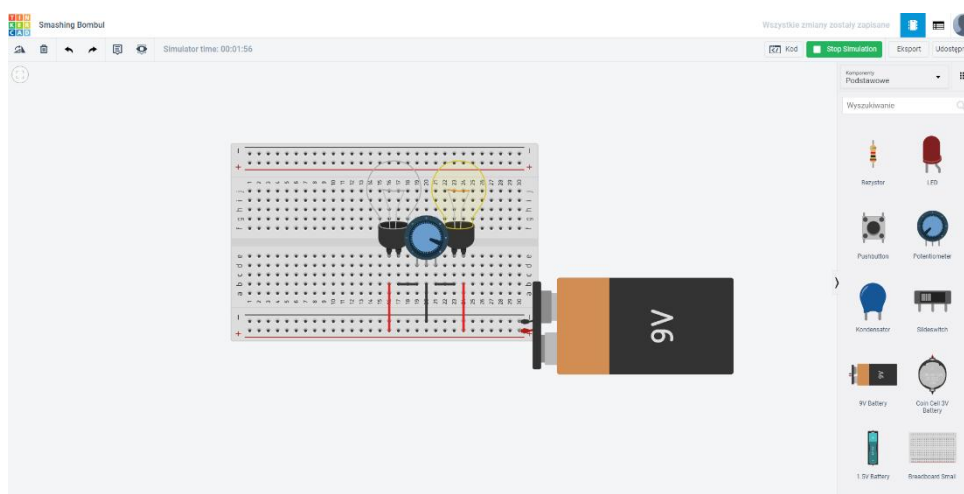
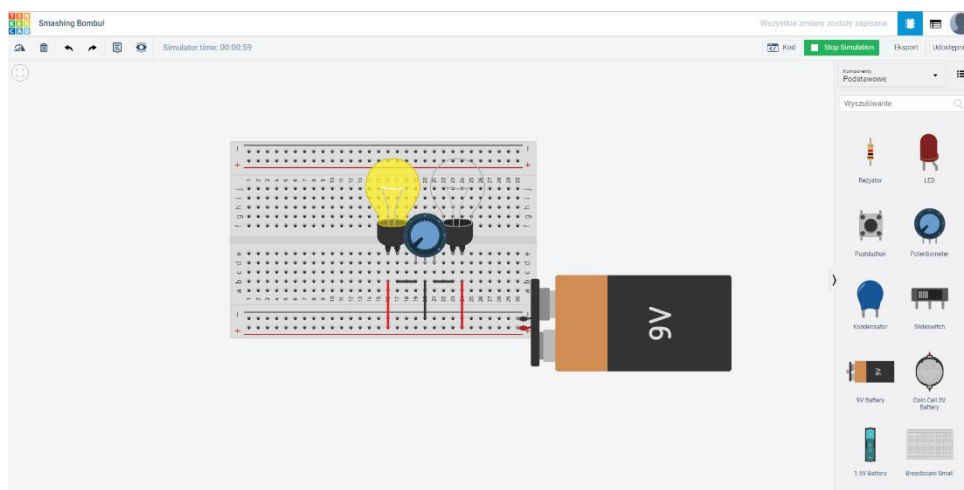


Obwód nr.2:





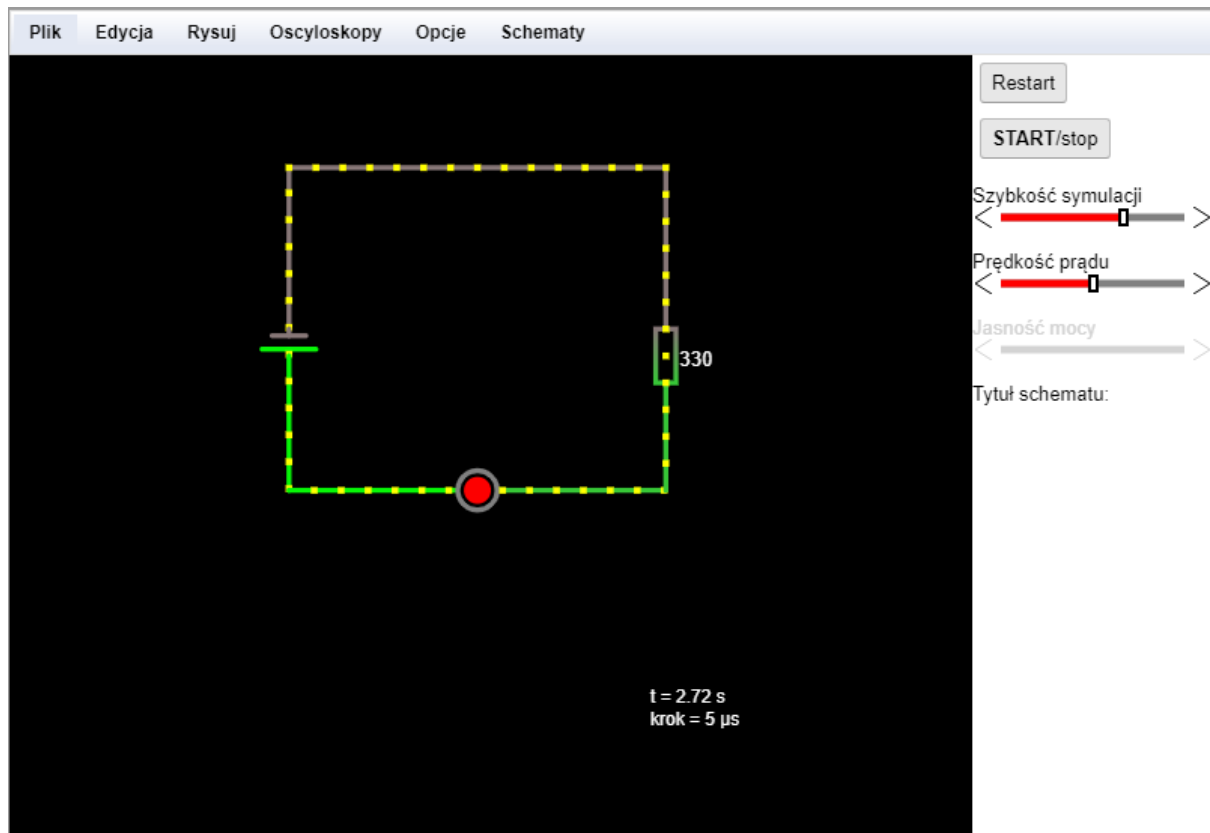
Zastąpienie przełącznika suwakowego potencjometrem:



Użycie potencjometru pozwala nam na regulację napięcia dostarczanego do żarówek w przeciwieństwie do przełącznika gdzie było ono stałe.

Lab - The Digital Oscilloscope

Topologia:



Symulator obwodu i obwody podstawowe:

- a) $I=9.75$
 $V_d=1.78$
 $P=17.38\text{mW}$

- b) $I=9.75\text{ mA}$
 $V_d=3.22\text{ V}$
 $R=330\ \Omega$
 $P=31.38\text{ mW}$

Napięcie na LED: 1.78V

Napięcie na rezystorze: 3.22V

Napięcie na baterii: 5V