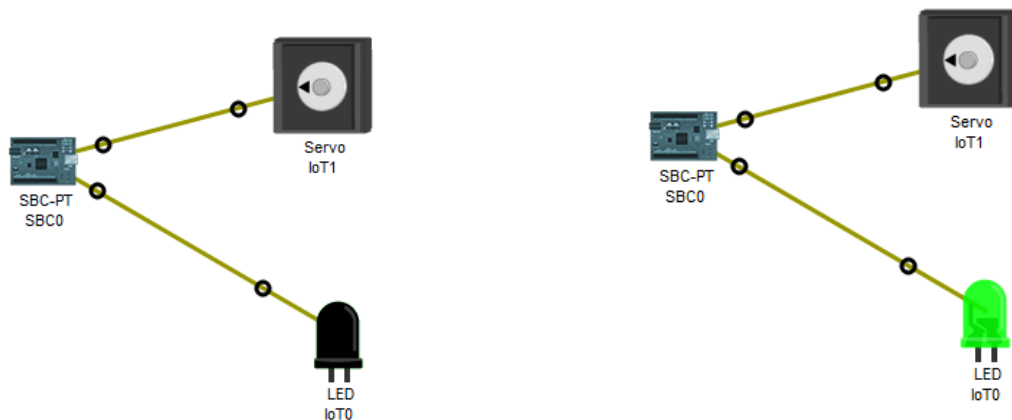


Politechnika Świętokrzyska w Kielcach	
Wydział Elektrotechniki, Automatyki i Informatyki	
Laboratorium IoT Rozproszone sieci sensoryczne	
Grupa: 3ID14B	Laboratorium 3
Data: 15.11.2018	Lesiak Karol

## Packet Tracer – Simulating IoT Devices

Topologia:



Modyfikacja kodu:

```

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

```

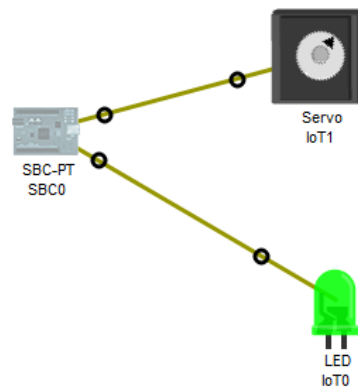
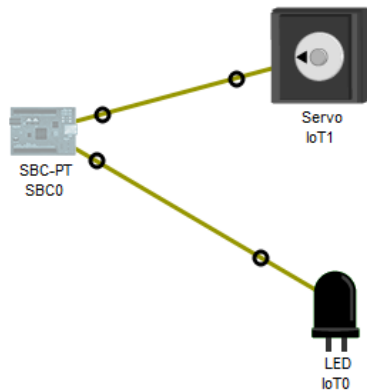
Starting Blink (Python)...

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.

The screenshot shows the SBC0 Programming window with the 'Blink (Python) - main.py' file open. 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, 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

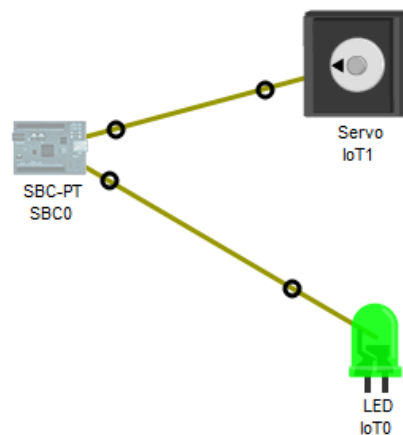
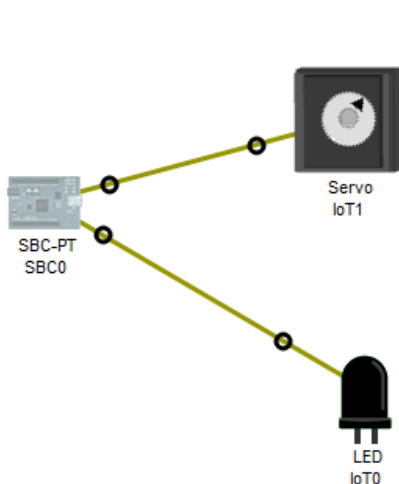
```

The output window at the bottom shows the following text:

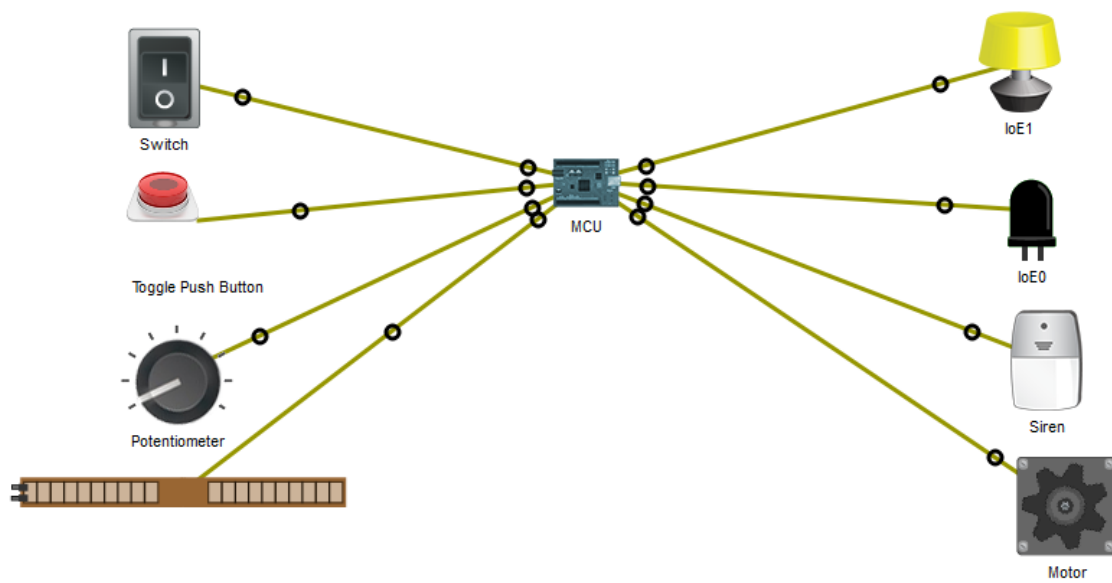
```

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

```

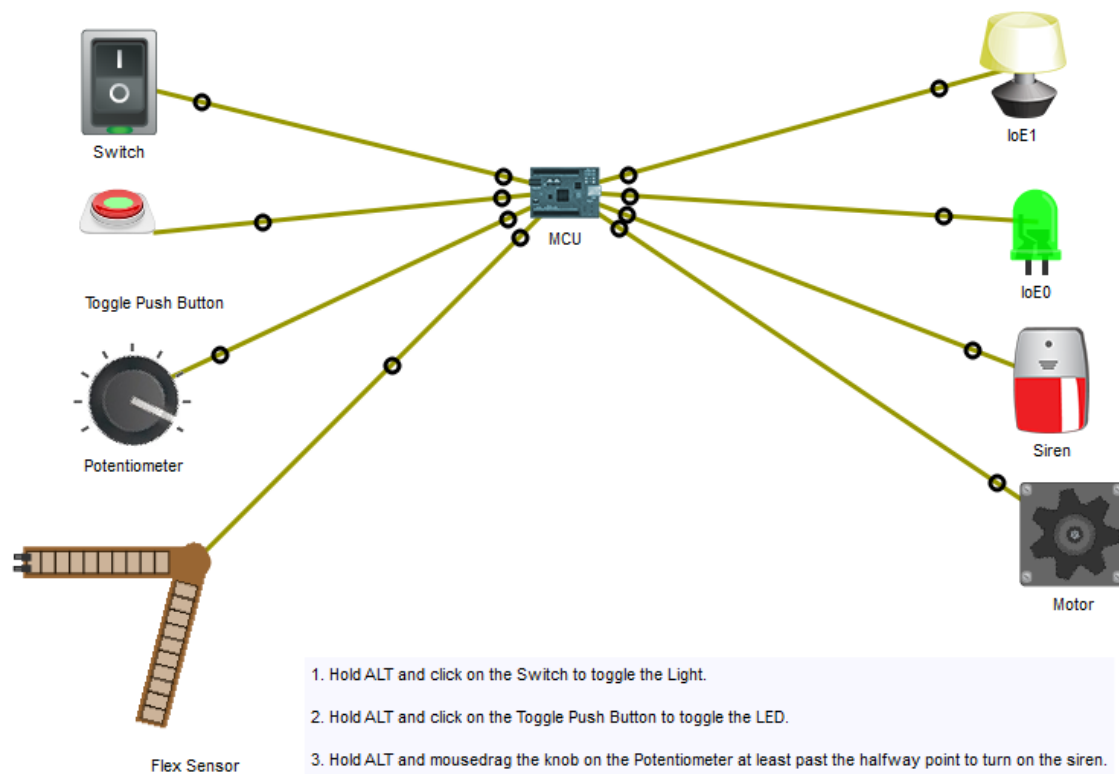


## Packet Tracer – Sensors and the PT Microcontroller



Flex Sensor

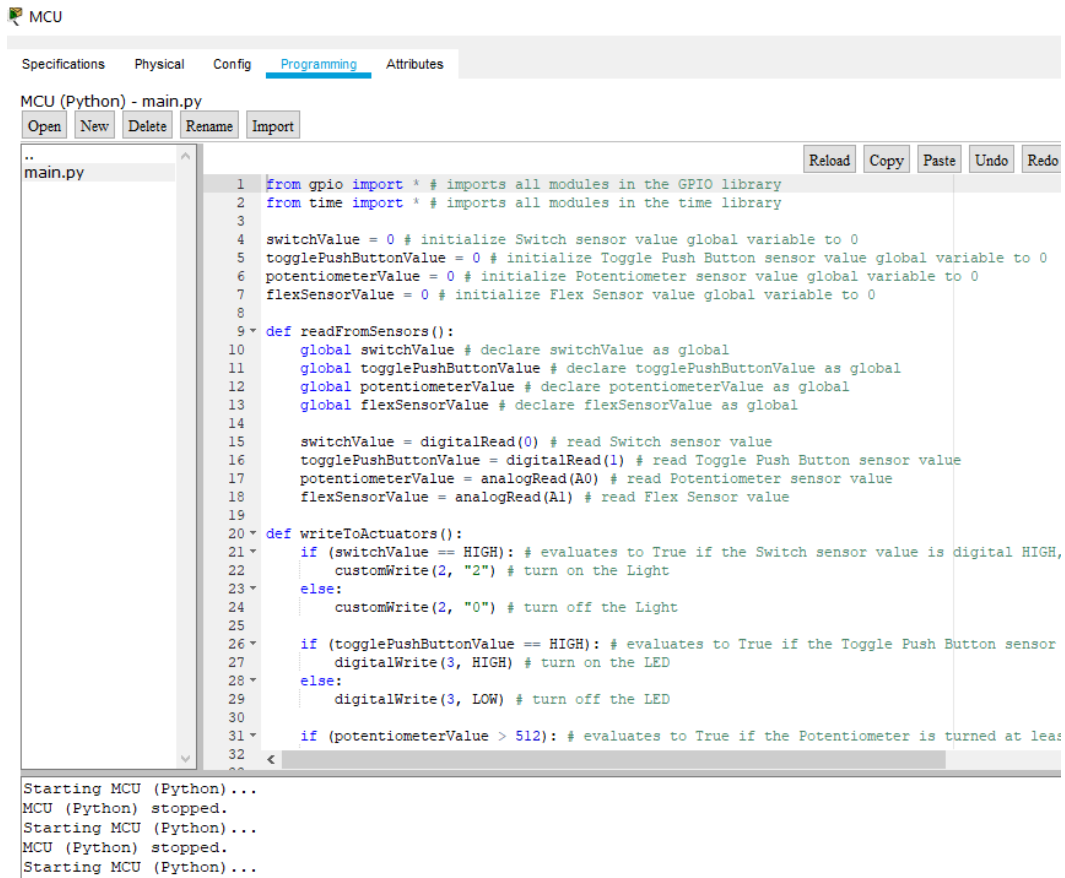
1. Hold ALT and click on the Switch to toggle the Light.
2. Hold ALT and click on the Toggle Push Button to toggle the LED.
3. Hold ALT and mousedrag the knob on the Potentiometer at least past the halfway point to turn on the siren.
4. Hold ALT and mousedrag the Flex Sensor to bend it and to move the motor.



Flex Sensor

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:



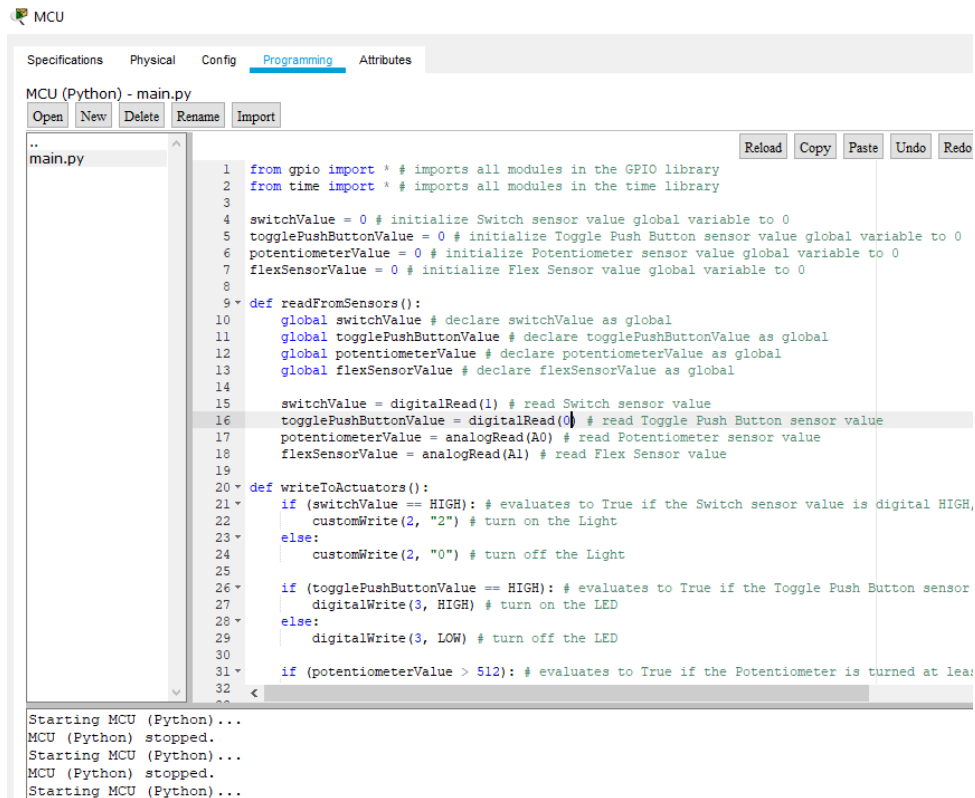
The screenshot shows the MCU Python IDE interface. The top bar includes tabs for Specifications, Physical, Config, Programming (selected), and Attributes. Below the tabs is the file name "MCU (Python) - main.py" and buttons for Open, New, Delete, Rename, and Import. The main editor area displays the code for main.py, with line numbers 1 through 32. The code includes imports for gpio and time, initialization of global variables for switchValue, togglePushButtonValue, potentiometerValue, and flexSensorValue, and two functions: readFromSensors() and writeToActuators(). The readFromSensors() function reads values from digital and analog sensors. The writeToActuators() function controls a light and an LED based on sensor inputs. The code is as follows:

```
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 digital 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
27         digitalWrite(3, HIGH) # turn on the LED
28     else:
29         digitalWrite(3, LOW) # turn off the LED
30
31     if (potentiometerValue > 512): # evaluates to True if the Potentiometer is turned at leas
32
```

Below the code editor, there is a terminal window showing the following output:

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

## Modyfikacja kodu:



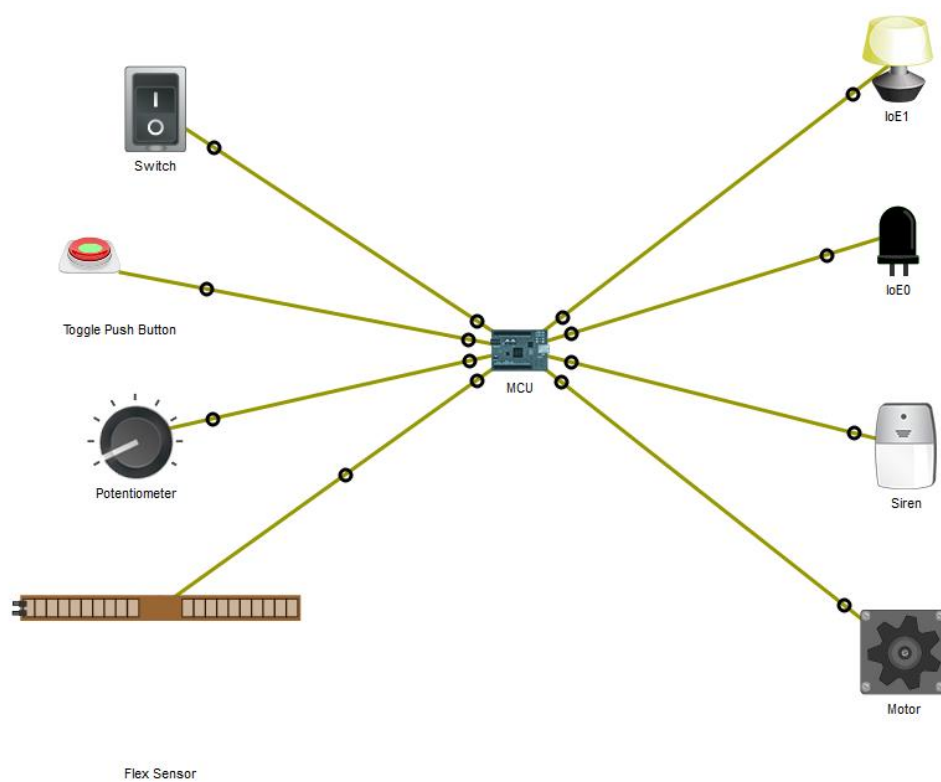
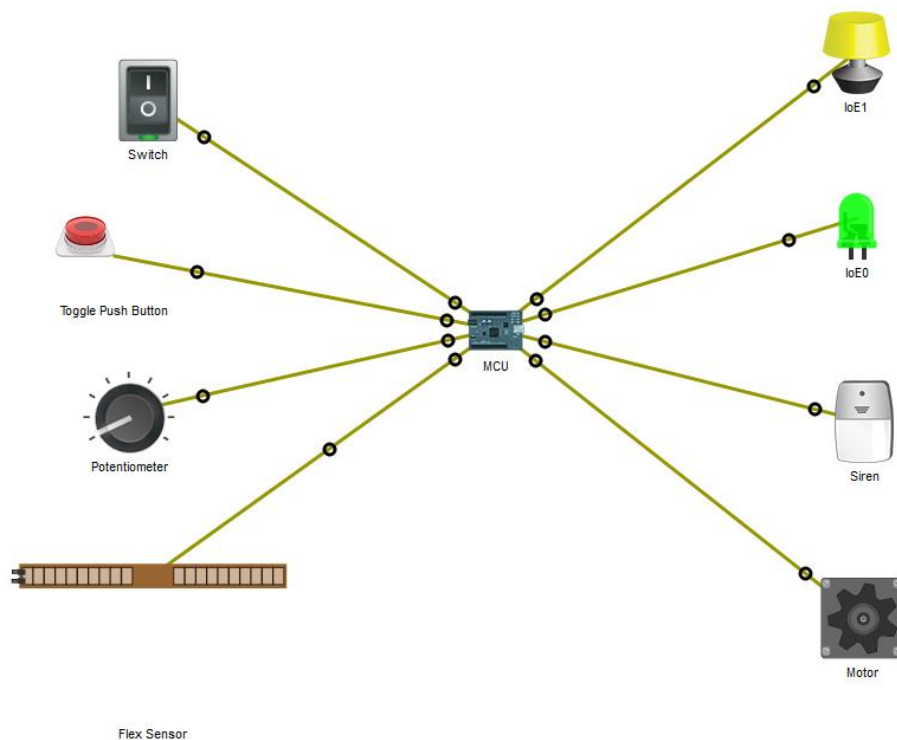
The screenshot shows the same MCU Python IDE interface as the previous one, but with modifications to the code in main.py. The changes are highlighted in the code editor. The modifications are as follows:

- Line 16: `togglePushButtonValue = digitalRead(0)` (previously 1)
- Line 18: `flexSensorValue = analogRead(A1)` (previously A0)

The rest of the code remains the same. The terminal window shows the same output as before:

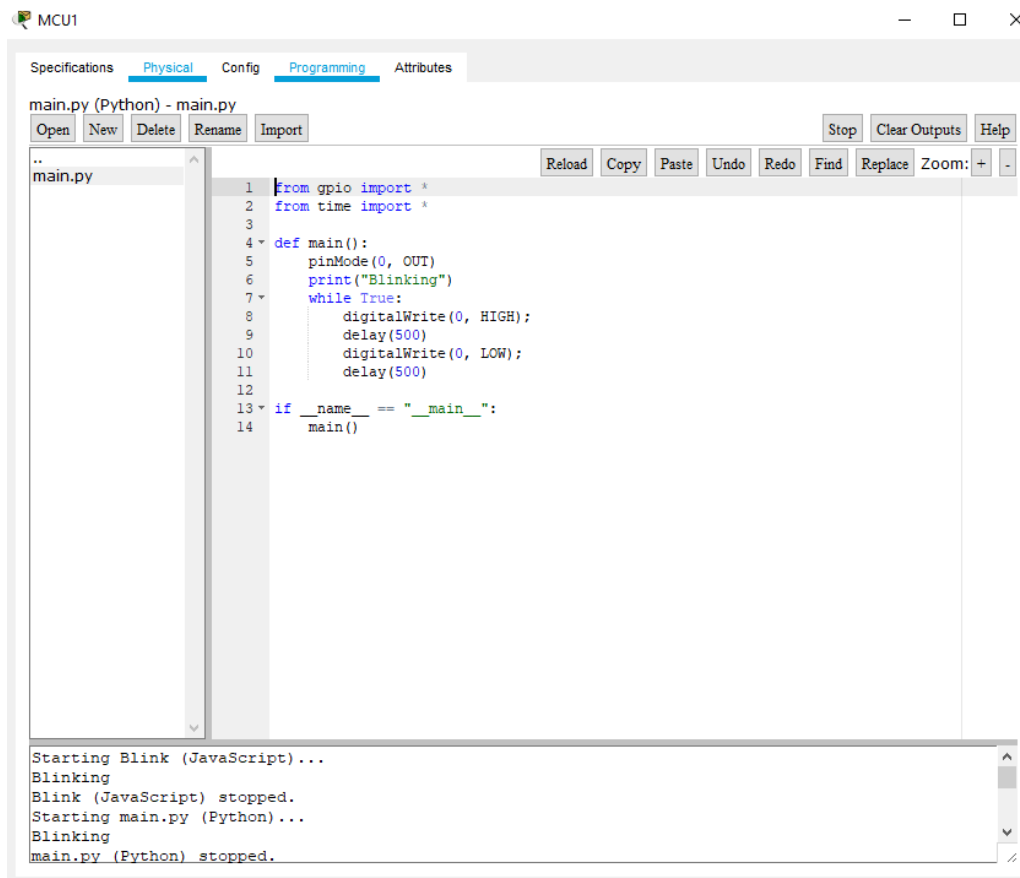
```
Starting MCU (Python)...
MCU (Python) stopped.
Starting MCU (Python)...
MCU (Python) stopped.
Starting MCU (Python)...
```

Aby przycisk kontrolował Lampę 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

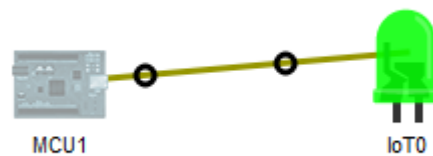


The screenshot shows the MCU1 IDE interface. The top bar includes tabs for Specifications, Physical, Config, Programming, and Attributes. The main window displays a Python file named 'main.py' with the following code:

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

The console at the bottom shows the execution log:

```
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

MCU1

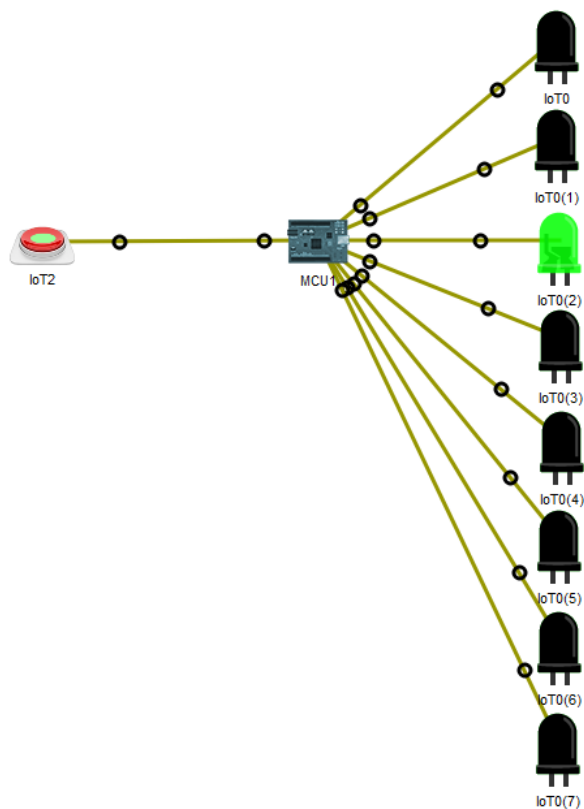
Specifications Physical Config **Programming** Attributes

main.py (Python) - main.py

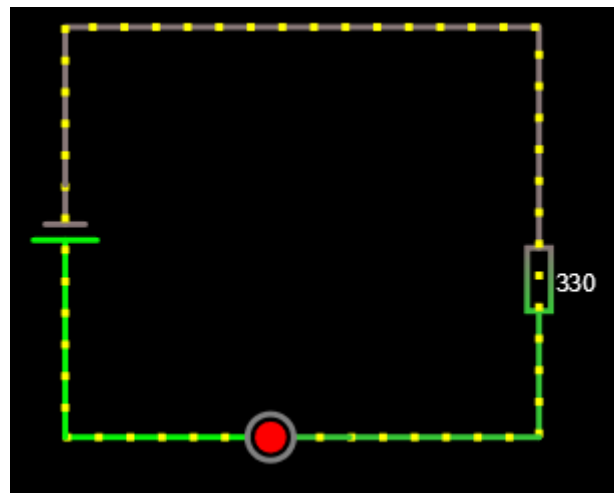
Open New Delete Rename Import Stop Clear Outputs Help

Reload Copy Paste Undo Redo Find Replace Zoom: + -

```
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, IN)
10    pinMode(0, OUT)
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(1)
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()
```



## Lab - The Digital Oscilloscope



źródło napięcia:

$$I = 9.75 \text{ mA}$$

$$V_d = 5 \text{ V}$$

$$(R = 512.73 \, \Omega)$$

$$P = -48.76 \text{ mW}$$

rezystor

$$I = 9.75 \text{ mA}$$

$$V_d = 3.22 \text{ V}$$

$$R = 330 \, \Omega$$

$$P = 31.38 \text{ mW}$$

LED

$$I = 9.75 \text{ mA}$$

$$V_d = 1.78 \text{ V}$$

$$P = 17.38 \text{ mW}$$

Napięcie na LED: 1.78 V

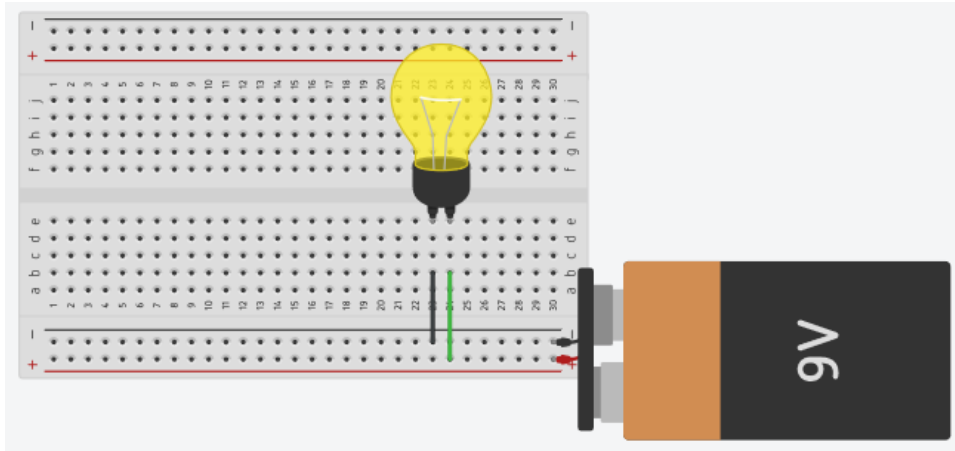
Napięcie na rezystorze: 3.22 V

Napięcie na baterii: 5 V

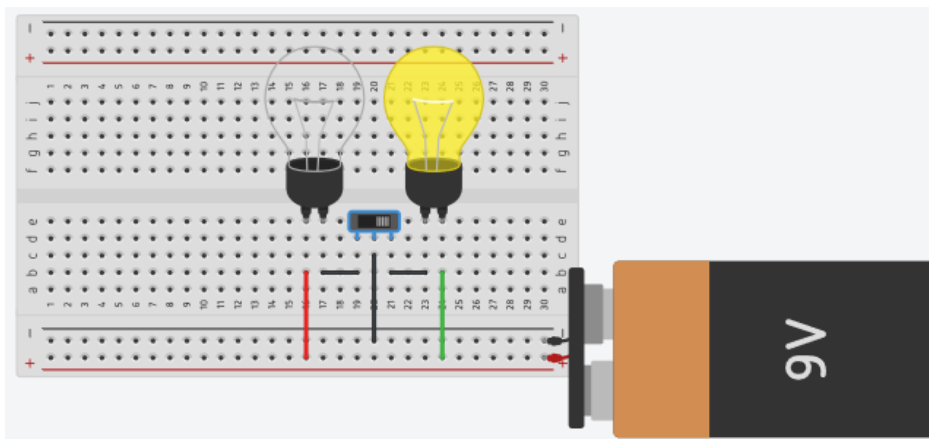
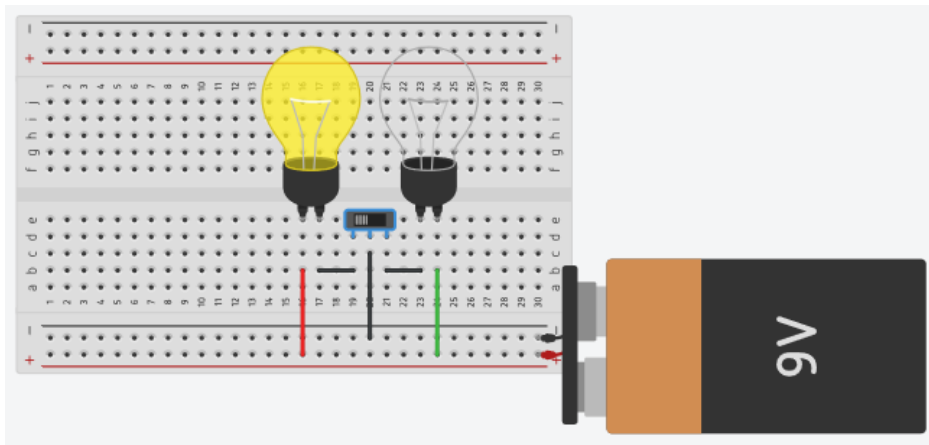


## Lab - Designing a Circuit from Start to Finish

Obwód 1:



Obwód 2:



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

