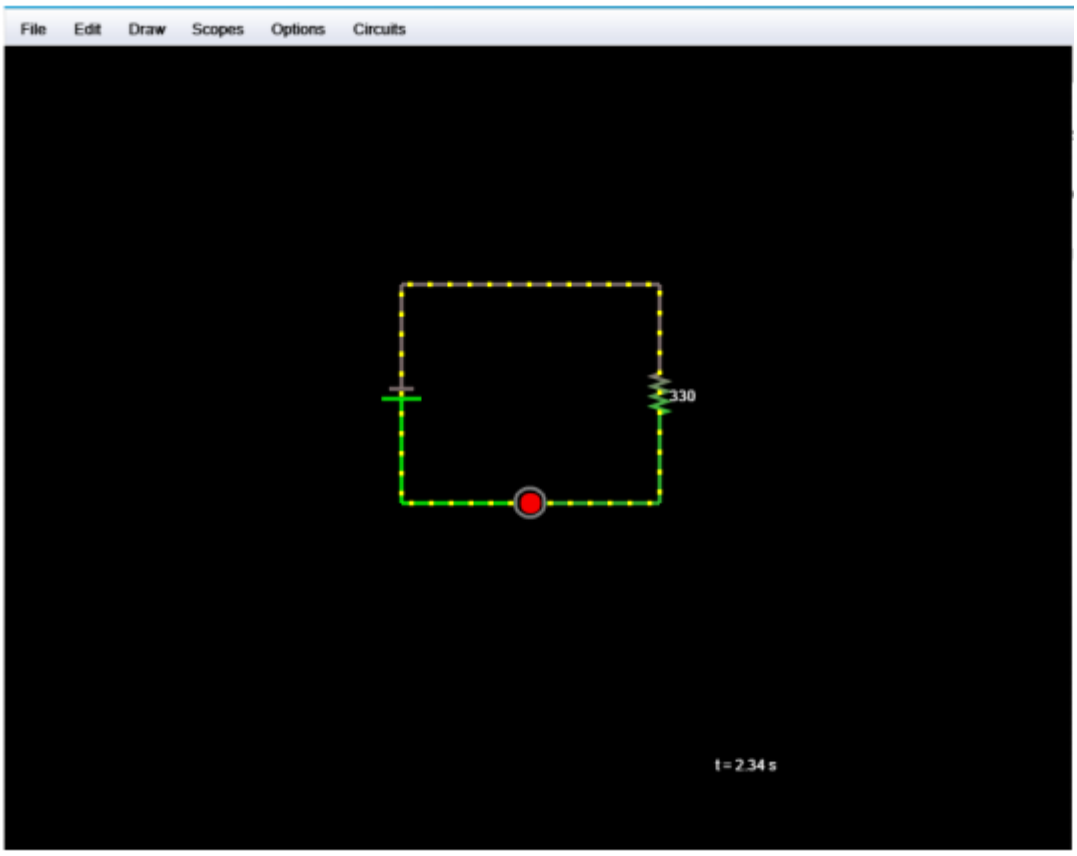
Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki	
Laboratorium 2 **The Digital Oscilloscope** **Packet Tracer - Sensors and the PT Microcontroller**	
Technologie loT rozproszone sieci sensoryczne	Autorzy: **Adrian Śmiglarski** **Patryk Tracz** Grupa: **3ID15A**

1. Lab - The Digital Oscilloscope

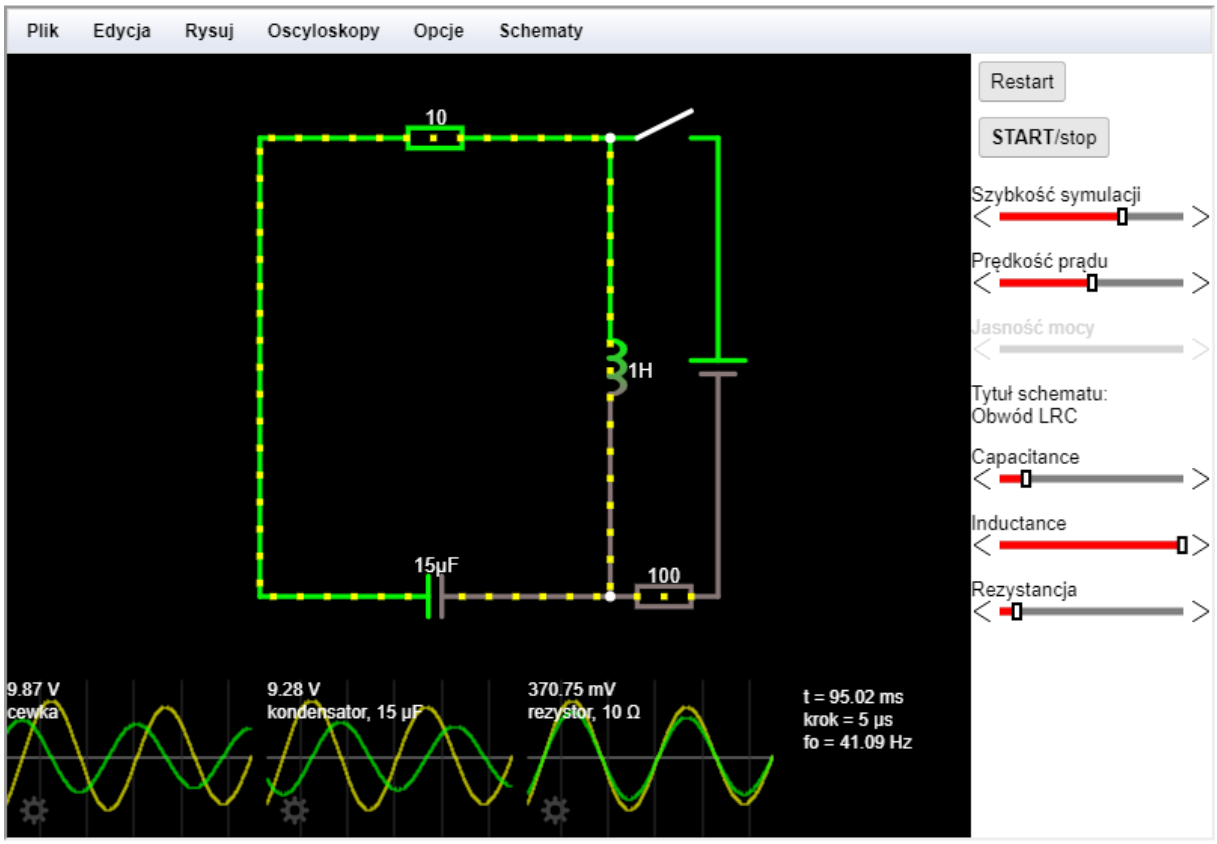
a) The Circuit Simulator and Basic Circuits

Topologia



Wartości napięcia na poszczególnych elementach obwodu:
LED- 1.78V
rezystor- 3.22V
bateria- 5V

b) Visualizing Signals



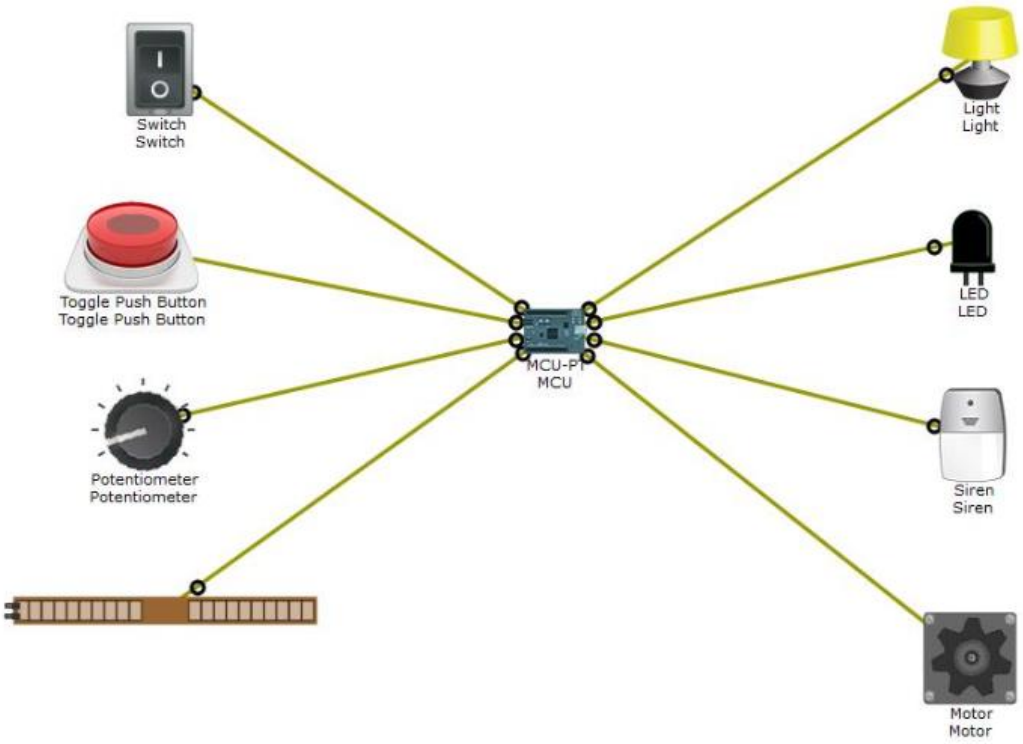
Challenge question: Alternating Current (AC) creates square waves or sine waves? Explain.

AC – prąd przemienny tworzy fale sinusoidalne ponieważ w tym przypadku wartości chwilowe podlegają zmianom w powtarzalny, okresowy sposób, z określoną częstotliwością.

Największe znaczenie praktyczne mają prąd i napięcie o **przebiegu sinusoidalnym**. W żargonie technicznym nazwa prąd przemienny często oznacza po prostu prąd sinusoidalny. Jeśli zakłócenia lub nieliniowość powodują zdeformowanie sinusoidalnego kształtu, wówczas taki niesinusoidalny przebieg nosi nazwę przebiegu odkształconego.

2. Packet Tracer - Sensors and the PT Microcontroller

Topologia:



W celu zmiany działania układu tak aby to przełącznik kontrolował diode LED a przycisk światło, musieliśmy zmodyfikować domyślny kod. Wystarczyło zmienić dwie linijki kodu:

Przed zmianami:

Reload | Copy | Paste | Undo | Redo | Find | Replace | Zoom: +

```
    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, otherwise false
        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 value is digital HIGH, otherw
        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 least half way
        customWrite(4, HIGH) # turn on the Siren
    else:
        customWrite(4, LOW) # turn off the Siren

    if (flexSensorValue > 0): # evaluates to True if the Flex Sensor is bent, otherwise false
        analogWrite(5, flexSensorValue) # turn on the motor with speed equal to the Flex Sensor value
    else:
        analogWrite(5, 0) # turn off the motor

def main(): # defines the main function
```

Po zmianach:

Reload Copy Paste Undo Redo Find Replace Zoom: +

```
    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, otherwise false
        digitalWrite(3, HIGH) # turn on the LED
    else:
        digitalWrite(3, LOW) # turn off the LED

    if (togglePushButtonValue == HIGH): # evaluates to True if the Toggle Push Button sensor value is digital HIGH, otherwi
        customWrite(2, "2") # turn on the Light
    else:
        customWrite(2, "0") # turn off the Light

    if (potentiometerValue > 512): # evaluates to True if the Potentiometer is turned at least half way
        customWrite(4, HIGH) # turn on the Siren
    else:
        customWrite(4, LOW) # turn off the Siren

    if (flexSensorValue > 0): # evaluates to True if the Flex Sensor is bent, otherwise false
        analogWrite(5, flexSensorValue) # turn on the motor with speed equal to the Flex Sensor value
    else:
        analogWrite(5, 0) # turn off the motor

def main(): # defines the main function
```

3. Wnioski

To zadanie zapoznało nas z podstawowymi elementami i pojęciami dotyczącymi obwodów elektrycznych. Dzięku stosowaniu narzędzi symulacyjnch takich jak Falstad’s Circuit Simulator Tool możemy badać przepływ prądu w obwodzie, oraz np. jak zmiany napięcia wpływają na poszczególne element obwodu.

Zadanie w Packet Tracer pokazało nam jak MCU może kontrolować działanie elementów układu IoT i jak ważne jest sprawne orientowanie się w kodzie w celu wprowadzenia pożądanych zmian.