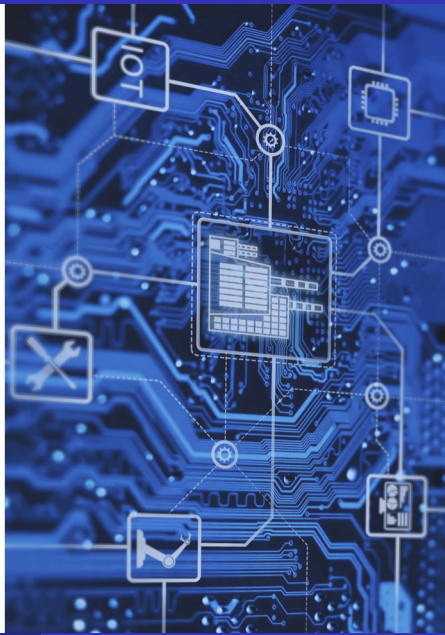
PROJECT NUMBER: 2023-1-PL01-KA220-SCH-000154043

Bringing the Internet of Things in school education as a tool to address 21st century challenges

Internet Rzeczy na lekcjach fizyki - od pomiarów do chmury danych. Warsztaty z Raspberry Pi Pico W



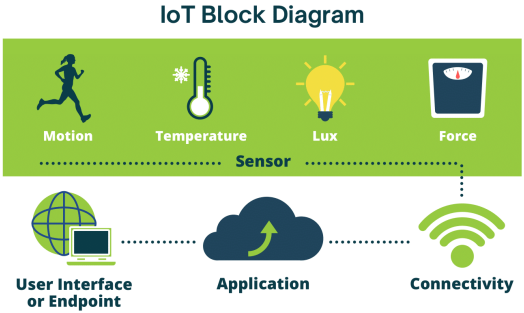
Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Czym IoT jest?

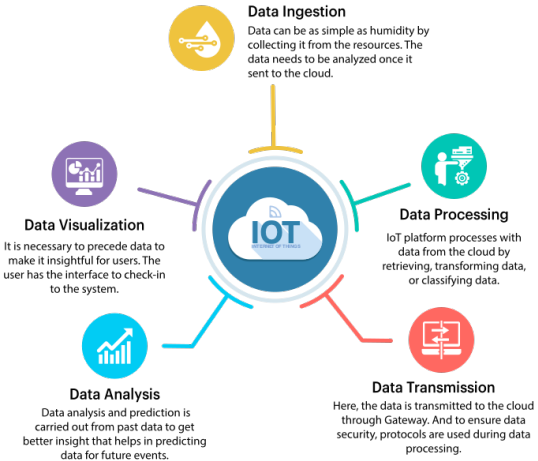
- **Internet of things (IoT)** jest koncepcją, w której urządzenia wyposażone w czujniki zbierają dane i przesyłają je za pośrednictwem sieci.
- Przesyłane dane mogą być wykorzystywane do wizualizacji i/lub podejmowania decyzji dotyczących zarządzania innymi komponentami.
- Jakie warunki musi spełniać system IoT?

1. posiadać czujniki zbierające dane,
2. posiadać urządzenie, które odczyta dane z czujników, wyśle je dalej i w razie potrzeby podejmie działania, np. Raspberry Pi Pico, Microbit,
3. posiadać sposób komunikacji urządzeń z Internetem, np. przez moduł WiFi.



Source: https://tacunasystems.com

Zagadnienia związane z IoT

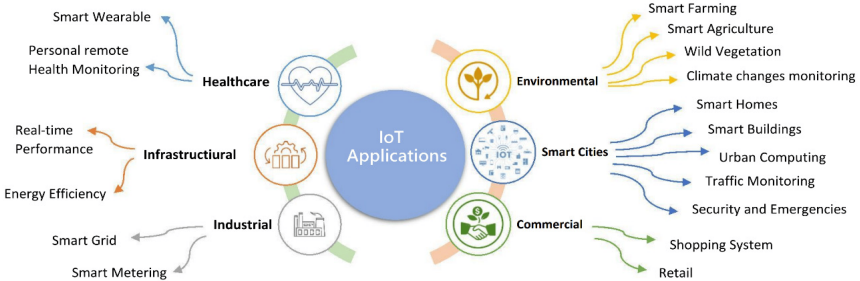


Source: https://www.sigmadatasys.com

Zastosowania technologi IoT

W projekcie IoT4schools skupiamy się na zastosowaniach technologi IoT w następujących obszarach:

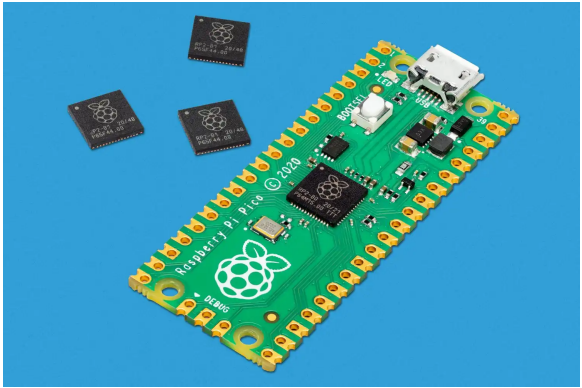
- monitorowanie zdrowia,
- ochrona środowiska,
- inteligentne miasta,
- automatyzacja domowa.



Source: https://www.mdpi.com

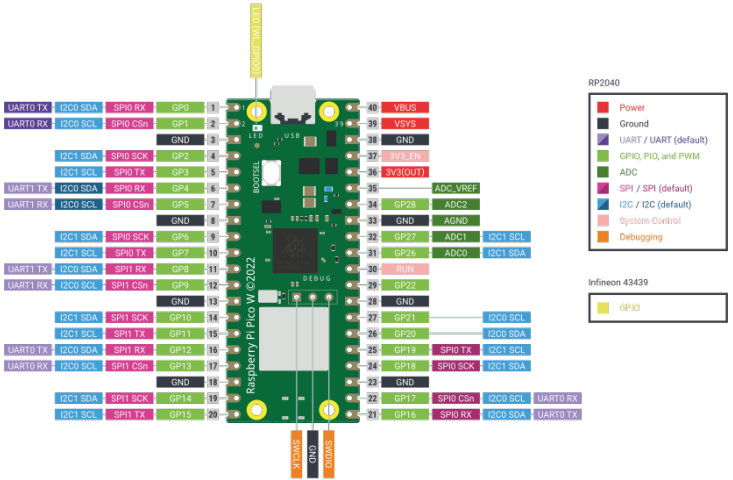
Raspberry Pi Pico

- W 2021 r. Raspberry Pi Foundation ogłosiło premierę płytki **Raspberry Pi Pico**.
- Płytka ta wyposażona jest w autorski układ **RP2040** - dwurdzeniowy mikrokontroler **ARM Cortex M0+** pracujący z częstotliwością 133 MHz.
- Płytkę można programować używając:
  - C/C++ SDK
  - MicroPython



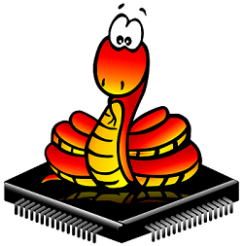
Źródło: https://www.raspberrypi.com/products/raspberry-pi-pico/

Układ płytki Raspberry Pi Pico W



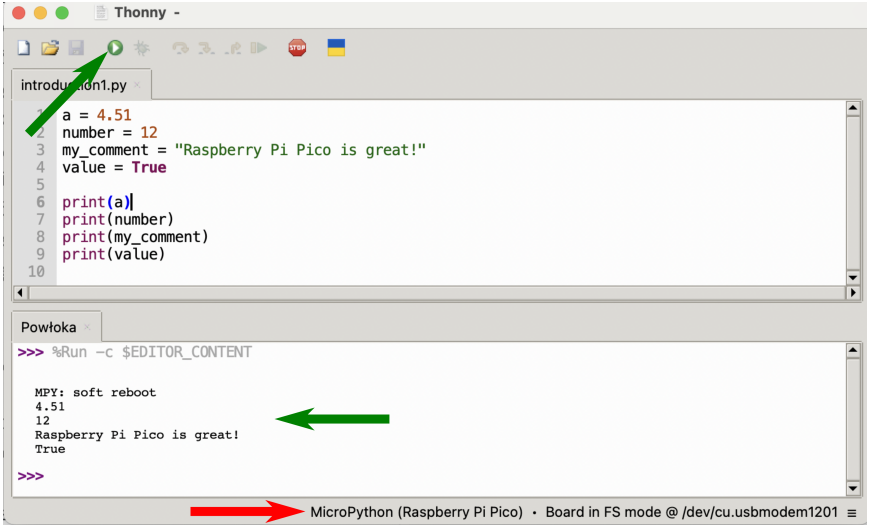
Źródło: https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html

- **MicroPython** to zmodyfikowana wersja języka Python, dedykowana do programowania mikrokontrolerów.
- Najważniejsze elementy języka Python:
  - **Zmienne** - umożliwiają przechowywanie wartości pod wybraną nazwą zmiennej np. zamiast w kodzie używać kilka razy wartości 3.14 można utworzyć zmienną *pi=3.14*.
  - Rodzaje zmiennych:



Źródło: https://randomnerdtutorials.com/getting-started-raspberry-pi-pico-w/

	Typ zmiennej	Przykład definicji
Liczby całkowite	int	LED=2
Liczby zmiennoprzecinkowe	float	pi=3.14
Wartości logiczne	boolean	status=True
Ciąg znaków	string	powitanie="Witaj"


Thonny -
introduction1.py
1 a = 4.51
2 number = 12
3 my_comment = "Raspberry Pi Pico is great!"
4 value = True
5
6 print(a)
7 print(number)
8 print(my_comment)
9 print(value)
10
Powłoka
>>> %Run -c $EDITOR_CONTENT
MPY: soft reboot
4.51
12
Raspberry Pi Pico is great!
True
>>>
MicroPython (Raspberry Pi Pico) • Board in FS mode @ /dev/cu.usbmodem1201

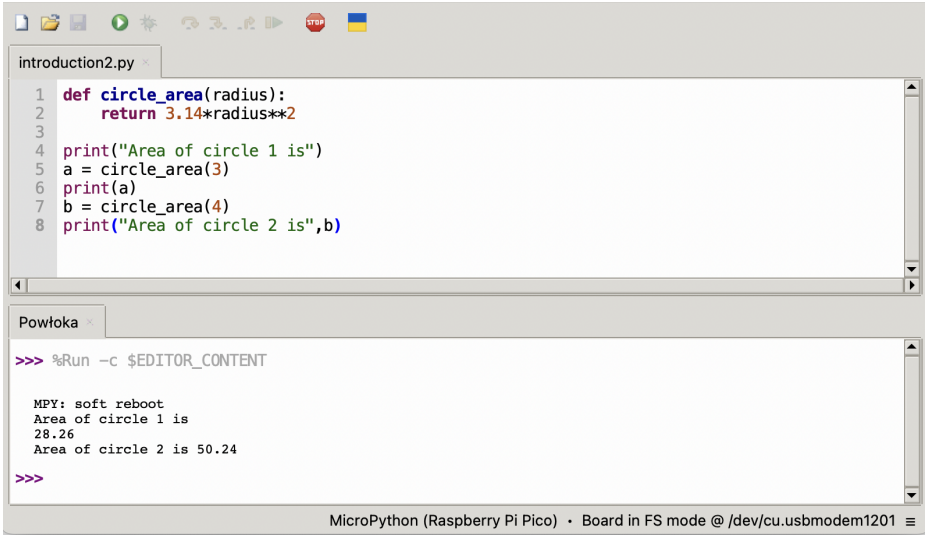
- **Funkcje** - są podprogramami, które wykonują określone operacje.
- Funkcja może przyjmować dane wejściowe zwane **argumentami** funkcji i zwracać wynik.
- Na przykład możesz napisać funkcję, która oblicza pole koła, wtedy argumentem funkcji będzie promień a rezultatem wartość pola.
- **Ważne:** Wszystko co ma być wewnątrz funkcji musi być przesunięte o tabulator (4 spacje) względem *def*.
- Potęgowanie: $x^y \rightarrow x**y$.

```
1 def nazwaFunkcji(argument1,
    ↪argument2):
2     polecenie1
3     polecenie2
4     ...
5     return wynik
```

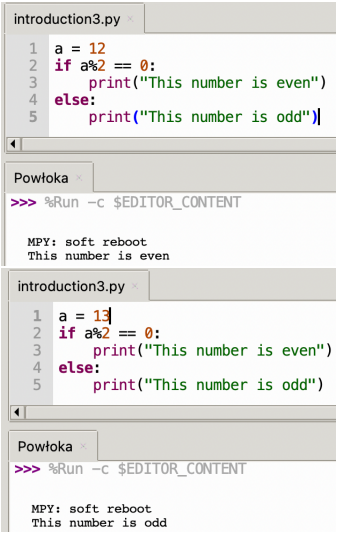
```
1 def circle_area(radius):
2     area =      3.14*radius**2
3     return area
```

```
1 def circle_area(radius):
2     return 3.14*radius**2
```

Wstęp do języka MicroPython



- **Struktura if...else** - umożliwia wykonywanie różnych fragmentów kodu w zależności od spełnionego warunku.
- Wszystkie polecenia, które umieścimy wewnątrz *if* (po wcięciu), zostaną wykonane tylko wtedy, gdy warunek zostanie spełniony.
- Następnie możemy, ale nie musimy, dodać blok *else*, który wykona się tylko wtedy, gdy warunek w *if* nie został spełniony.
- Podstawowe metody porównania:
  - $a == b$ - sprawdzenie czy a jest równe b,
  - $a > b$ - sprawdzenie czy a jest większe niż b,
  - $a >= b$ - sprawdzenie czy a jest większe bądź równe b,
  - $a < b$ - sprawdzenie czy a jest mniejsze niż b,
  - $a <= b$ - sprawdzenie czy a jest mniejsze bądź równe b.

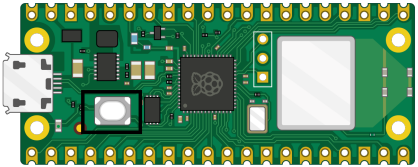


Ogólna struktura każdego programu

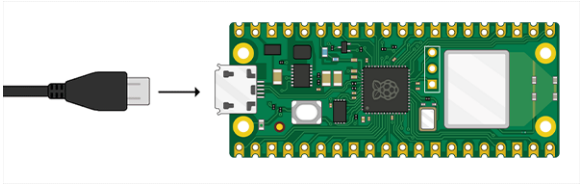
```
#Dodanie niezbędnych bibliotek, które zawierają przydatne funkcje
import nazwa_biblioteki

#Utworzenie zmiennych
#Wykonanie poleceń, które mają wykonać się tylko raz np. różnego
   ↪rodzaju konfiguracje

#Nieskończona pętla
while True:
        #Polecenia, które mają się ciągle wykonywać
```

Instalacja płytki Raspberry Pi Pico na komputerze

1. Przytrzymaj przycisk BOOSTEL na płytce Raspberry Pi Pico:

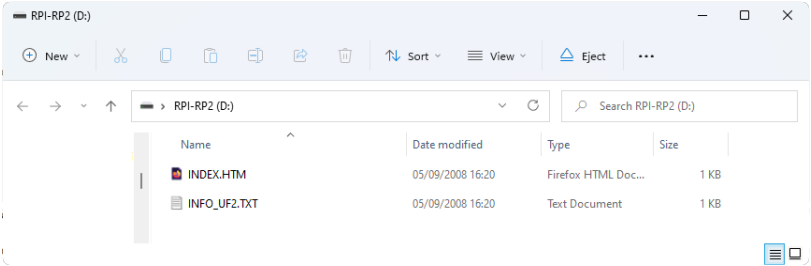


2. Podłącz płytkę Raspberry Pi Pico W do komputera trzymając przycisk BOOSTEL:



Instalacja płytki Raspberry Pi Pico na komputerze

3. Płytka Raspberry Pi Pico W będzie widoczna analogicznie jak pendrive. Po otwarciu katalogu *RPI* zobaczysz takie pliki:

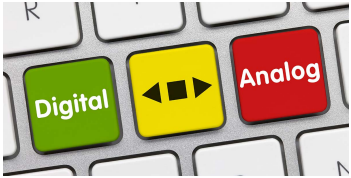


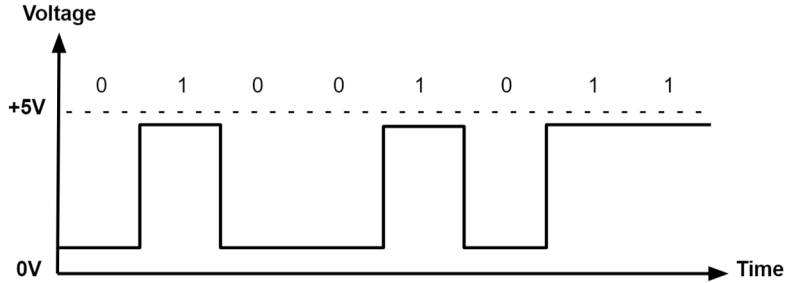
4. Pobierz plik **.UF2* ze strony `https://rpf.io/pico-w-firmware`.
5. Pobrany plik wklej do katalogu *RPI*.
6. Zainstaluj edytor *Thonny*. Instalator znajdziesz tutaj: `https://thonny.org`.

Rodzaje sygnałów

Rozpatrzmy 2 rodzaje sygnałów:

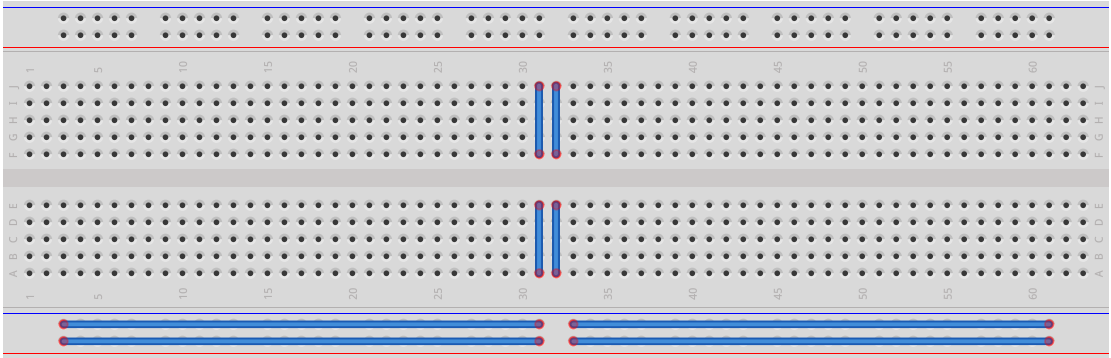
- cyfrowe,
- analogowe.





Źródła obrazków: https://www.nutsvolts.com, https://www.monolithicpower.com

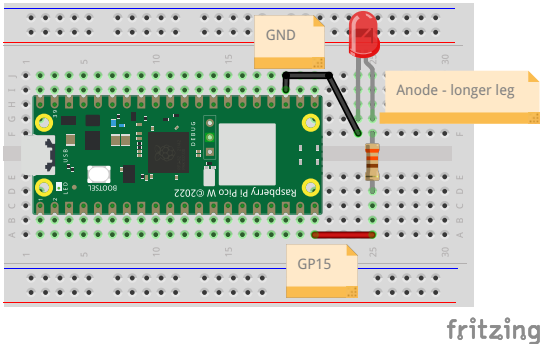
Płytka stykowa



Zadanie 1- migająca dioda LED

- Cel: napisz program, który będzie cyklicznie zapalał i gasił diodę LED co 1s.
- Typowa dioda LED do jasnego świecenia wymaga napięcia na poziomie ok 1,7 V (U_z) i natężenia prądu od 1 do 15 mA. Aby natężenie prądu nie przekraczało 15 mA, należy podłączyć rezystor o wartości np.:

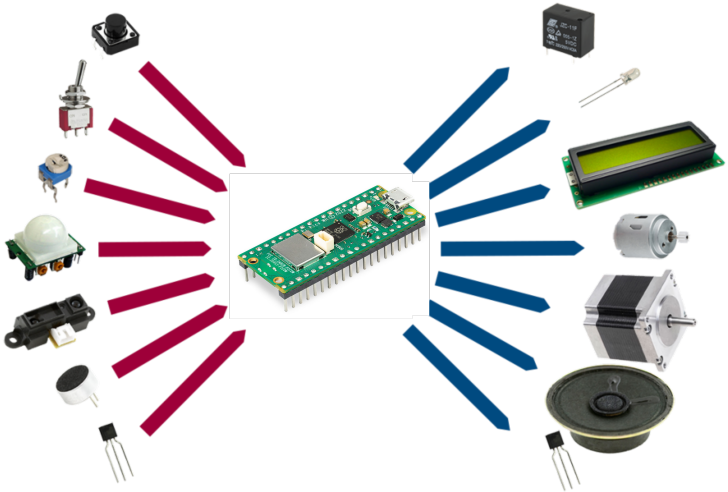
$$R = \frac{3,3V - U_z}{I} = \frac{3,3V - 1,7V}{4,9mA} \approx 330\Omega \quad (1)$$



Konfiguracja pinów

Piny, do których podłączane są elementy, mogą być:

- wejściowe (*input*),
- wyjściowe (*output*).



Źródło oryginalnego obrazka: https://blog.codebender.cc/2014/03/07/lesson-1-inputs-and-outputs/

Zadanie 1

1. Na początku należy dodać bibliotekę, która zawiera niezbędne funkcje do komunikacji z płytką Raspberry Pi Pico:

```
1 import machine
```

2. Dioda LED jest elementem **wyjściowym** gdyż to mikrokontroler wysyła do niej sygnał wysoki, by zapalić diodę, bądź niski by ją zgasić. Zatem należy skonfigurować pin GP15 jako wyjściowy. Do tego służy funkcja: *machine.Pin()*.

Konfiguracja pinów

Funkcja **machine.Pin(numer_pinu, rodzaj_pinu)** służy do konfiguracji pinu jako wejściowy (**machine.Pin.IN**) bądź wyjściowy (**machine.Pin.OUT**).

```
1 import machine
2
3 led = machine.Pin(15, machine.Pin.OUT)
```

Zadanie 1

3. Następnie należy umieścić nieskończoną pętlę, w której znajdą się instrukcje powtarzane w kółko przez Raspberry Pi Pico W:

```
1 import machine
2
3 led = machine.Pin(15, machine.Pin.OUT)
4
5 while True:
```

4. Następnie zapalamy diodę LED wysyłając sygnał wysoki (1) na pin GP15:

```
1 import machine
2
3 led = machine.Pin(15, machine.Pin.OUT)
4
5 while True:
6     led.value(1)
```

Zadanie 1

5. Teraz program powinien odczekać 1s zanim zgasimy diodę. W tym celu należy skorzystać z funkcji *sleep(opoźnienie)*, która jest dostępna w bibliotece *utime*. Stąd najpierw należy dodać bibliotekę *utime* a potem opóźnienie 1s:

```
1 import machine
2 import utime
3
4 led = machine.Pin(15, machine.Pin.OUT)
5
6 while True:
7         led.value(1)
8         utime.sleep(1)
```

Zadanie 1

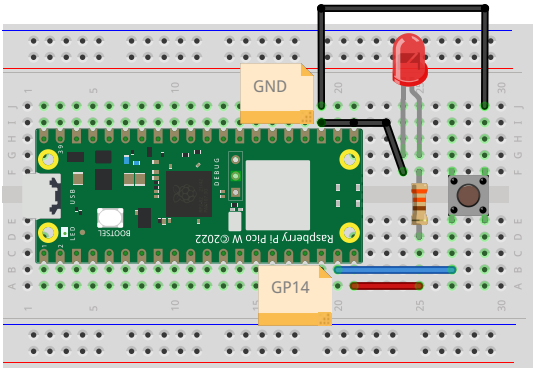
6. Ostatni krok to zgaszenie diody LED wysyłając sygnał niski (0) na pin GP15 oraz odczekanie 1s:

```
import machine
import utime

led = machine.Pin(15, machine.Pin.OUT)

while True:
        led.value(1)
        utime.sleep(1)
        led.value(0)
        utime.sleep(1)
```

7. Program jest już gotowy. Uruchom go i przetestuj działanie!

Zadanie 2

- Cel: Zapalenie/zgaszenie diody LED po naciśnięciu przycisku.
- Podłącz układ zgodnie z poniższym rysunkiem.



Zadanie 2

1. Na początku należy dołączyć niezbędne biblioteki:

```
1 import machine
2 import utime
```

2. Następnie należy skonfigurować pin GP15, do którego podłączona jest dioda LED, jako wyjściowy oraz zgasić diodę LED:

```
1 import machine
2 import utime
3
4 LED = machine.Pin(15, machine.Pin.OUT)
5 LED.value(0)
```

Zadanie 2

3. Następnie należy skonfigurować pin GP14, do którego podłączony jest przycisk, jako element wejściowy (opcja **machine.Pin.IN**), ponieważ będziemy odczytywać wartość z przycisku.
4. Przycisk należy również podłączyć do rezystorów podciągających (*pull-up*), które są już zamontowane wewnątrz płytki Raspberry Pi Pico i podłączone do wszystkich pinów GPIO. Rezystory pull-up łączą przycisk z 3.3V, co oznacza, że jeśli przycisk nie jest naciśnięty, zostanie odczytana wartość 1. Gdy jest naciśnięty to odczytamy wartość 0. Aby przycisk podłączyć do rezystorów *pull-up* wystarczy podać jako trzeci argument funkcji opcje: **machine.Pin.PULL_UP**.

```
1 import machine
2 import utime
3
4 LED = machine.Pin(15, machine.Pin.OUT)
5 LED.value(0)
6 but=machine.Pin(14,machine.Pin.IN,machine.Pin.PULL_UP)
```

Zadanie 2

4. Teraz utwórzmy główną pętlę programu i sprawdźmy w niej stan przycisku. Jeśli wartość zwrócona przez przycisk jest równa 0, oznacza to, że przycisk został naciśnięty. Następnie zmieńmy stan diody LED na przeciwny po naciśnięciu przycisku:

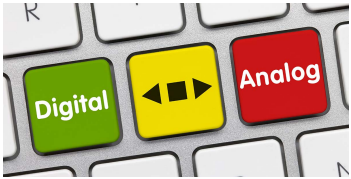
```
1  import machine
2
3  LED = machine.Pin(15, machine.Pin.OUT)
4  LED.value(0)
5  but=machine.Pin(14,machine.Pin.IN,machine.Pin.PULL_UP)
6
7  while True:
8      if but.value()==0:
9          LED.toggle()
10         utime.sleep(1)
```

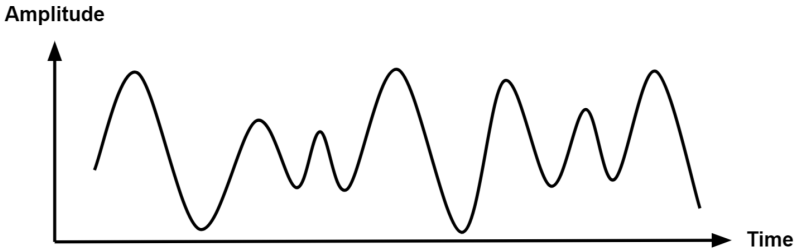
Program jest gotowy! Przetestuj jego działanie.

Rodzaje sygnałów

Rozpatrzmy 2 rodzaje sygnałów:

- cyfrowe,
- analogowe.

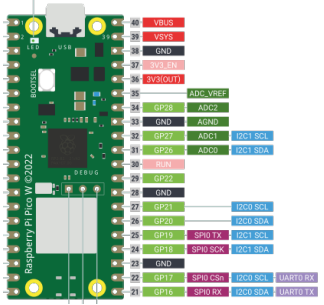




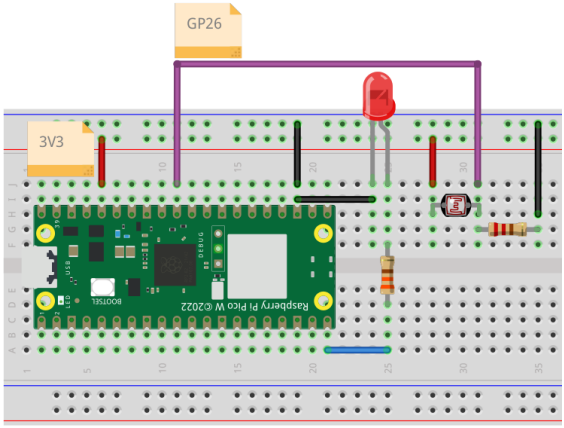
Źródła obrazków: https://www.nutsvolts.com, https://www.monolithicpower.com

Zadanie 3

- Cel: Zapalenie diody LED tylko gdy jest ciemno.
- Fotorezystor jest elementem, którego rezystancja maleje wraz ze wzrostem natężenia oświetlenia.
- Napięcie sygnału analogowego można zmierzyć za pomocą przetworników ADC podłączonych do pinów GP26, GP27 i GP28.



Źródło: https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html



Zadanie 3

1. Na początku dodajemy niezbędne biblioteki oraz konfigurujemy pin GP26 aby pracował jako pin analogowy. Do tego służy funkcja *ADC(numer pinu)*:

```
import machine
import utime
LDR = machine.ADC(26)
```

2. Następnie odczytujemy napięcie na pinie GP26 korzystając z funkcji *read_u16()*. Funkcja ta zwraca wartość 16bitową czyli maksymalna wartość to 65535, która odpowiada napięciu 3.3V.

```
import machine
import utime
LDR = machine.ADC(26)
while True:
        print(LDR.read_u16())
        utime.sleep(1)
```

Uruchom program i przetestuj jakie wartości odczytasz gdy zasłonisz fotorezystor ręką a gdy poświecisz latarką.

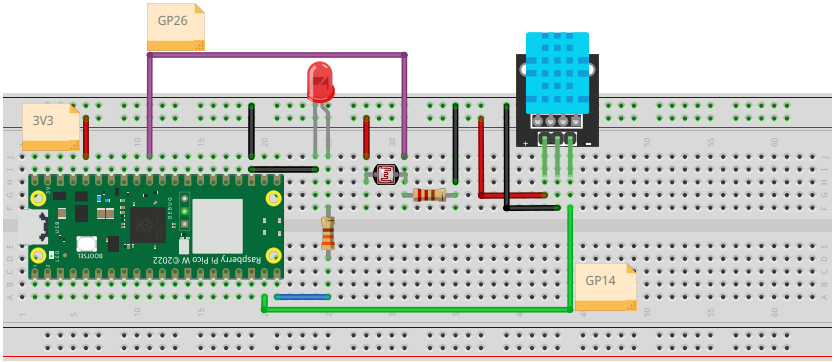
Zadanie 3

3. Teraz dodajmy konfiguracje diody LED oraz zapalmy ją gdzie jest ciemno. Wartość wpisaną do zmiennej ***ciemno*** zastąp na własną jaką uznasz za najlepszą:

```
1  import machine
2  import utime
3
4  LDR = machine.ADC(26)
5  LED = machine.Pin(15, machine.Pin.OUT)
6  LED.value(0)
7  ciemno = 5000
8
9  while True:
10     if LDR.read_u16() < ciemno:
11         LED.value(1)
12     else:
13         LED.value(0)
14     utime.sleep(1)
```

Zadanie 4

- Oprócz prostych czujników cyfrowych zwracających wartość 0 bądź 1, istnieją czujniki, które potrzebują przesłać konkretną wartość. W takich wypadkach wykorzystują one magistrale komunikacyjne.
- Umożliwiają one przesłanie wartości zgodnie z ustalonym protokołem. Jedną z takich magistrali jest 1-wire.
- Czujnik DHT11 umożliwia pomiar temperatury i wilgotności. Przesyła on dane korzystając z magistrali 1-wire. Stwórzmy program, który odczyta wartość temperatury i wilgotności.



Zadanie 4

1. Zainstaluj bibliotekę **dht** z menedżera pakietów w Thonny. W tym celu wybierz: *Tools* → *Manage Packages*. A następnie wyszukaj bibliotekę **dht** i ją zainstaluj.
2. Przejdźmy teraz do pisania kodu. Najpierw dodajmy niezbędne biblioteki oraz utwórzmy obiekt klasy DHT11:

```
import machine
import utime
from dht import DHT11
dht = DHT11(14)
```

3. Następnie wymuśmy by czujnik DHT11 dokonał pomiaru:

```
import machine
import utime
from dht import DHT11
dht = DHT11(14)

while True:
    dht.measure()
```

Zadanie 4

4. Teraz odczytajmy z czujnika jaką wartość temperatury i wilgotności zmierzył:

```
import machine
import utime
from dht import DHT11

dht = DHT11(14)

while True:
    dht.measure()
    temp = dht.temperature()
    hum = dht.humidity()
    print("Temperatura:", temp, "°C")
    print("Wilgotnosc:", hum, "%")
    utime.sleep(2)
```

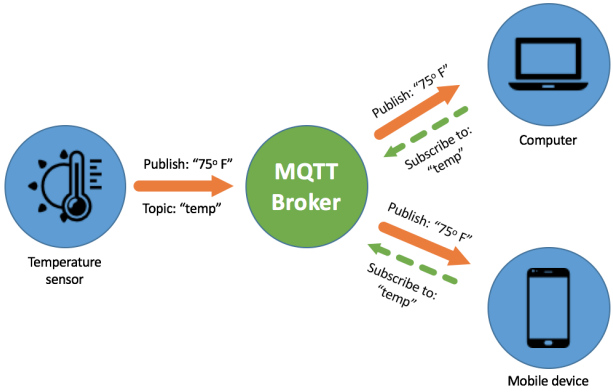
5. Program jest gotowy! Przetestuj jego działanie.

Adafruit IO

- Układy IoT wysyłają zebrane dane do chmury.
- Obecnie do dyspozycji jest wiele różnych chmur, które różnią się możliwościami, ceną oraz trudnością w używaniu.
- Na tych warsztatach skupimy się na chmurze **Adafruit IO**, która w wersji darmowej zapewnia możliwość zbierania danych z 10 różnych czujników i utworzenie 5 różnych pulpitów do wyświetlania danych.
- Chmura Adafruit IO oparta jest o protokół MQTT (*Message Queuing Telemetry Transport*).

MQTT

- Protokół MQTT opiera się na modelu publikuj/subskrybuj.
- Klient "A" publikuje informacje w danym temacie, a każdy inny klient, który zasubskrybował ten temat, odczyta dane opublikowane przez klienta "A".



Źródło: https://www.akcp.com/blog/scaling-mqtt-network-for-better-operational-output/

Schemat postępowania

Zadanie 5

Cel

Przesłanie zmierzonej temperatury i wilgotności przez czujnik DHT11 do chmury Adafruit IO.

Kroki:

1. Na początku utwórz konto na platformie Adafruit IO: `https://io.adafruit.com`
2. Aby przesłać dane do chmury, należy utworzyć *feed*.
3. *Feed* to obiekt, który przechowuje dane. Aby utworzyć *feed*, należy przejść do zakładki *Feed* na platformie Adafruit IO i kliknąć na *New Feed* a następnie pojawi się okienko, w którym należy nadać nazwę *feed* np. *Temp*:

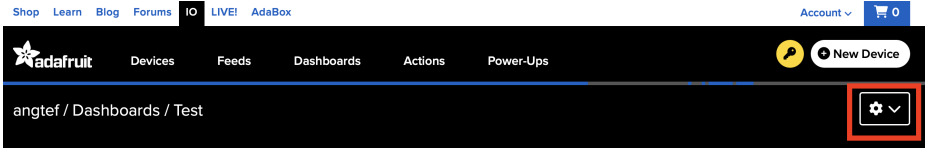
4. Stwórz dwa *feeds*, jeden do przechowywania temperatury (*Temp*) a drugi do wilgotności (*humidity*).

Zadanie 5

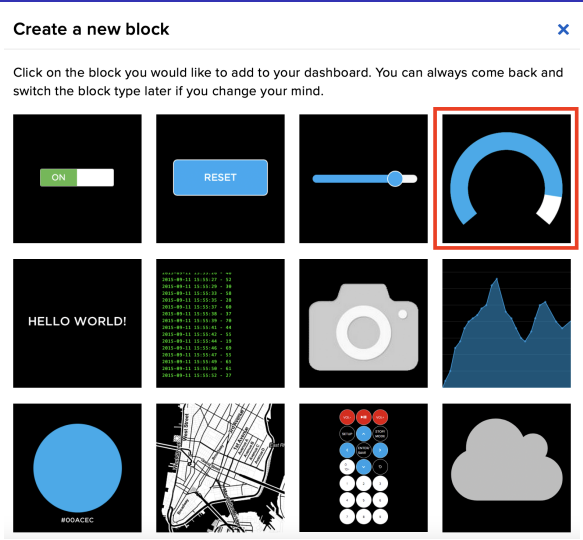
4. Następnie należy utworzyć pulpit. W tym celu należy przejść do zakładki *Dashboards* i wybrać przycisk *New Dashboard*:

5. Następnie wyskoczy okno z wyborem nazwy pulpitu.

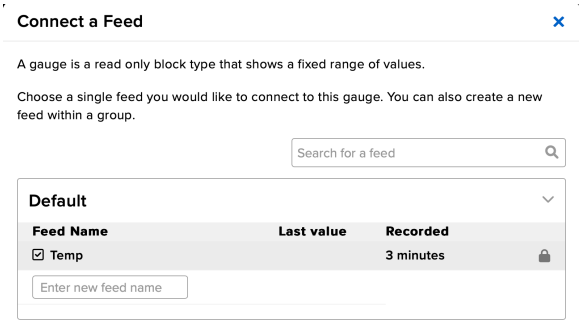
6. Następnie po prawej stronie kliknij na ikonę ustawień i wybierz: *Create New Block*:



Zadanie 5


Create a new block
Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.
ON
RESET
HELLO WORLD!
#0DACEC

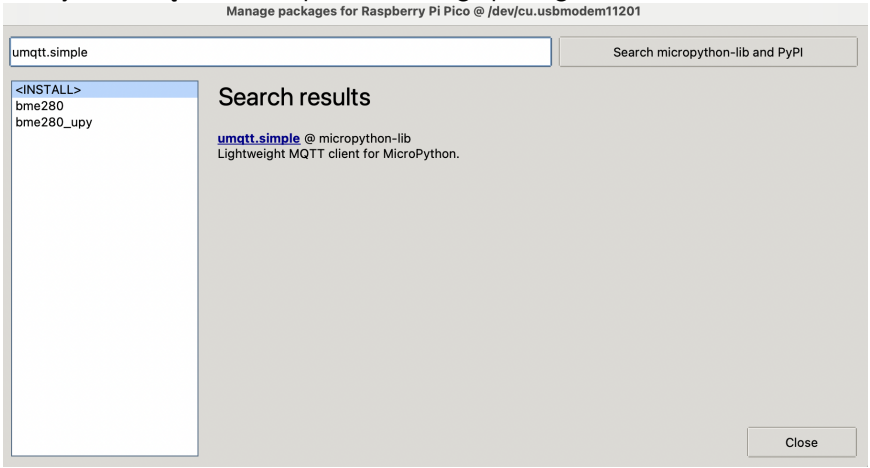
Zadanie 5



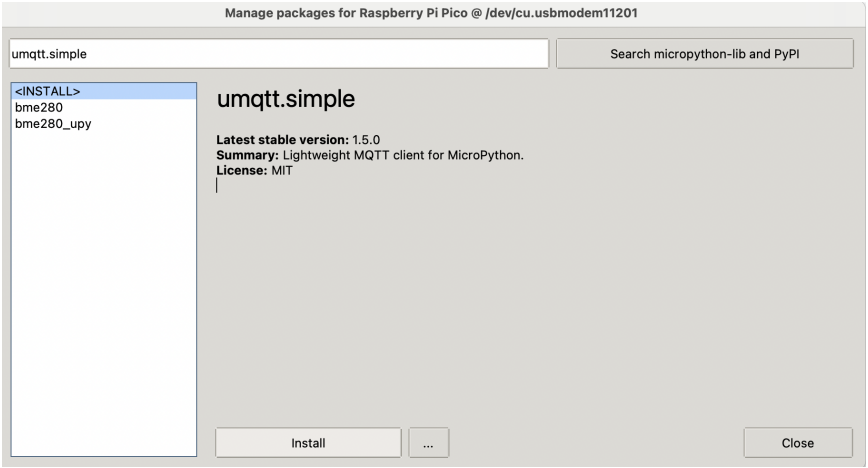
7. Analogicznie należy dodać blok dla pomiaru wilgotności.

Zadanie 5

8. Teraz powróćmy do edytora Thonny i zainstalujmy bibliotekę *umqtt.simple*. W tym celu wybierz w edytorze Thonny zakładkę "Tools" a potem "Manage packages":



Zadanie 5


Manage packages for Raspberry Pi Pico @ /dev/cu.usbmodem11201
umqtt.simple
Search micropython-lib and PyPI
<INSTALL>
bme280
bme280_upy
umqtt.simple
Latest stable version: 1.5.0
Summary: Lightweight MQTT client for MicroPython.
License: MIT
Install
...
Close

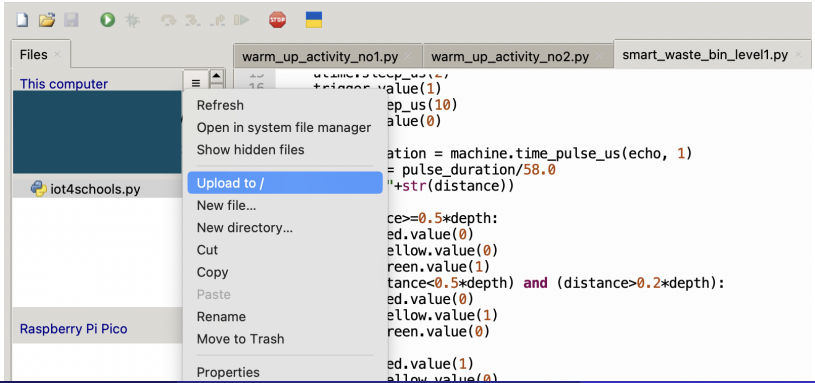
Zadanie 5

9. Powróćmy do kodu z zadania 4. Dodajmy niezbędne biblioteki by połączyć się przez WiFi i do protokołu MQTT:

```
import machine
import utime
from dht import DHT11
import network
from umqtt.simple import MQTTClient

dht = DHT11(14)

while True:
    dht.measure()
    temp = dht.temperature()
    hum = dht.humidity()
    print("Temperatura:", temp, "°C")
    print("Wilgotnosc:", hum, "%")
    utime.sleep(2)
```

Zadanie 5

0. Pobierz bibliotekę *iot4schools.py* z repozytorium github: https://github.com/angsam/iot4schools
1. Wybierz w edytorze Thonny: *View* → *Files*.
2. Wgraj zewnętrzną bibliotekę *iot4schools.py* na płytkę Raspberry Pi Pico. Aby wgrać bibliotekę należy:



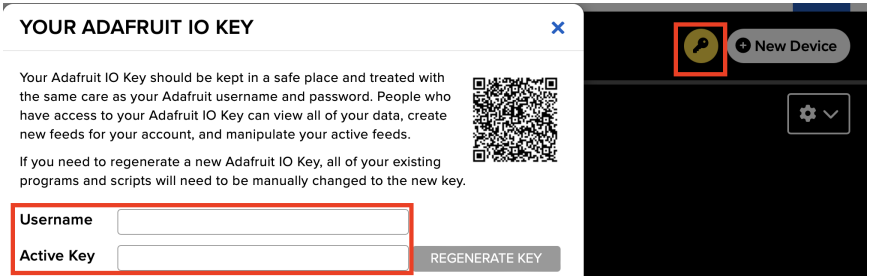
Zadanie 5

2. Zaimportuj bibliotekę *iot4schools* oraz utwórz zmienne, które będą przechowywały dane sieci WiFi oraz klucz do konta Adafruit IO:

```
1  import machine
2  import utime
3  from dht import DHT11
4  import network
5  from umqtt.simple import MQTTClient
6  import iot4schools
7
8  dht = DHT11(14)
9  WIFI_SSID = "name of your wifi"
10 WIFI_PASSWORD = "password to your wifi"
11 ADAFRUIT_IO_USERNAME = "your username"
12 ADAFRUIT_IO_KEY = "key"
13
14 while True:
15     ... // Trzy kropki oznaczaja, ze dalszy kod nie umieszczono na slajdzie z
           ↪powodu braku miesjca ale go nie usuwaj
```

Zadanie 5

4. Uzupełnij powyższe zmienne z linijek 9-10 danymi o sieci WiFi przekazanymi przez prowadzącego bądź dane własnej sieci udostępnionej z telefonu.
5. Uzupełnij powyższe zmienne z linijek 11-12 danymi Twojego konta na platformie Adafruit IO. Klucz uzyskasz klikając na symbol klucza na platformie Adafruit IO.



Zadanie 5

15. Teraz należy podłączyć się z siecią WiFi:

```
1  import machine
2  import utime
3  from dht import DHT11
4  import network
5  from umqtt.simple import MQTTClient
6  import iot4schools
7
8  dht = DHT11(14)
9  WIFI_SSID = "name of your wifi"
10 WIFI_PASSWORD = "password to your wifi"
11 ADAFRUIT_IO_USERNAME = "your username"
12 ADAFRUIT_IO_KEY = "key"
13
14 iot4schools.connect_wifi(WIFI_SSID, WIFI_PASSWORD)
15 while True:
16     ...
```

Zadanie 5

15. Teraz przejdźmy do protokołu MQTT. Na początku należy stworzyć klienta MQTT a potem połączyć się z serwerem:

```
1  import machine
2  import utime
3  from dht import DHT11
4  import network
5  from umqtt.simple import MQTTClient
6  import iot4schools
7
8  ...
9
10 iot4schools.connect_wifi(WIFI_SSID, WIFI_PASSWORD)
11 client = iot4schools.create_mqtt_client(ADAFRUIT_IO_USERNAME,
       ↪ADAFRUIT_IO_KEY)
12 iot4schools.connect_mqtt(client)
13
14 while True:
15     ...
```

Zadanie 5

- Teraz stwórzmy zmienne przechowujące informacje o stworzonych *feed*, które będą przechowywać wysłane dane do chmury:

```
...
import iot4schools

dht = DHT11(14)
WIFI_SSID = "your_wifi_name"
WIFI_PASSWORD = "your_wifi_password"
ADAFRUIT_IO_USERNAME = "username"
ADAFRUIT_IO_KEY = "key"
FEED1 = "your_username/feeds/Temp"
FEED2 = "your_username/feeds/humidity"

iot4schools.connect_wifi(WIFI_SSID, WIFI_PASSWORD)
client = iot4schools.create_mqtt_client(ADAFRUIT_IO_USERNAME,
    ↪ADAFRUIT_IO_KEY)
iot4schools.connect_mqtt(client)
```

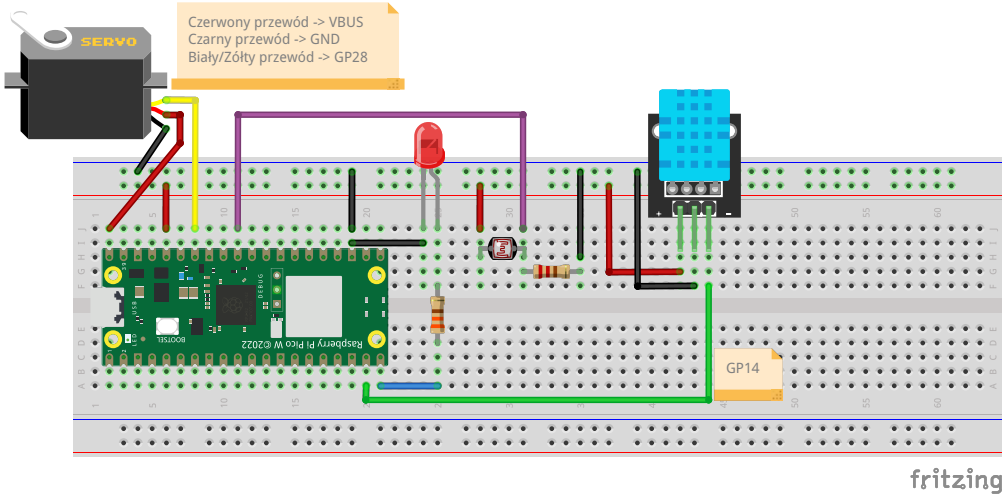
- Teraz stwórzmy funkcję wysyłającą wartość temperatury i wilgotności do chmury oraz w pętli while wyślijmy odczytaną temperaturę do chmury:

```
...
iot4schools.connect_mqtt(client)

def send_data(temp, hum):
    client.publish(FEED1, str(temp))
    print("Temp. wyslana:"+str(temp))
    client.publish(FEED2, str(hum))
    print("Wilg. wyslana:"+str(hum))

while True:
        ...
        temp = dht.temperature()
        hum = dht.humidity()
        print("Temperatura:", temp, "°C")
        print("Wilgotnosc:", hum, "%")
        send_data(temp, hum)
        utime.sleep(300)
```

Zadanie 6

- W ostatnim zadaniu stworzyliśmy program, który jest pierwszym krokiem do automatyki domowej.
- Mamy już pomiar wilgotności i ciśnienia. Zatem na ich podstawie możemy sterować oknami np. gdy temperatura jest odpowiednio wysoka to okna otwieramy a gdy niska to zamykamy.
- Do tego zadania wykorzystamy serwomechanizm.

Zadanie 6 - podłączenie



Zadanie 6

1. Najpierw należy zainstalować bibliotekę **micropython-servo**. Wybierz w menu edytora Thonny: Narzędzia → Zarządzaj pakietami, a następnie wyszukaj bibliotekę i ją zainstaluj.
2. Teraz powróćmy do kodu z zadania 5 i dodajmy konfiguracje serwomechanizmu:

```
1  import machine
2  import utime
3  from dht import DHT11
4  import network
5  from umqtt.simple import MQTTClient
6  import iot4schools
7  from servo import Servo
8
9  servo = Pin(28)
10 angle = Servo(pin_id=servo)
11 angle.write(0)
12 sleep(0.5)
13 dht = DHT11(14)
14 ...
```

Zadanie 6

3. Dodajmy teraz zamykanie i otwieranie okien w zależności do temperatury:

```
...
while True:
        dht.measure()
        temp = dht.temperature()
        hum = dht.humidity()
        print("Temperatura:", temp, "°C")
        print("Wilgotnosc:", hum, "%")
        send_data(temp, hum)
        if temp > 22:
                angle.write(90)
        else:
                angle.write(0)
        utime.sleep(300)
```

4. Program jest gotowy!

Zadanie 7

1. Dodajmy teraz możliwość sterowania otwieraniem i zamykaniem okien z chmury Adafruit IO. W tym celu przejdź do chmury i utwórz kolejny *feed* nazwany **servo**:

2. Następnie przejdź do pulpitu chmury Adafruit IO i stwórz nowy blok typu suwak. Powiąż go z feed: *servo*.

Zadanie 7

3. Wróćmy do kodu z zadania 6 i dodajmy zmienną przechowującą nazwę *feed*:

```
...
import iot4schools

dht = DHT11(14)
WIFI_SSID = "your_wifi_name"
WIFI_PASSWORD = "your_wifi_password"
ADAFRUIT_IO_USERNAME = "username"
ADAFRUIT_IO_KEY = "key"
FEED1 = "your_username/feeds/Temp"
FEED2 = "your_username/feeds/humidity"
FEED_SERVO = "your_username/feeds/servo"

iot4schools.connect_wifi(WIFI_SSID, WIFI_PASSWORD)
...
```

Zadanie 7

4. Dodajmy funkcje do odbierania danych z chmury Adafruit IO:

```
...
iot4schools.connect_mqtt(client)
def send_data(temp, hum):
    client.publish(FEED1, str(temp))
    print("Temp. wyslana:"+str(temp))
    client.publish(FEED2, str(hum))
    print("Wilg. wyslana:"+str(hum))

def Receive_Data(feed, msg):
    if FEED_SERVO in feed:
        action = str(msg, "utf-8")
        print("action = ".format(action))
        angle.write(float(action))

client.set_callback(Receive_Data)
client.subscribe(FEED_SERVO)
```

Zadanie 7

5. Ostatni krok to sprawdzenie czy dane z chmury są dostępne do odczytu:

```
...
while True:
        ...
    print("Temperatura:", temp, "°C")
    print("Wilgotnosc:", hum, "%")
    send_data(temp, hum)
    if temp > 22:
        angle.write(90)
    else:
        angle.write(0)
    try:
        client.check_msg()
    except KeyboardInterrupt:
        print("Ctrl-C pressed...exiting")
        client.disconnect()
        sys.exit()
    utime.sleep(300)
```

Zadanie 8

1. W kolejnym zadaniu dodamy do naszej automatyki możliwość włączenia/wyłączenia oświetlenia z poziomu chmury. W tym celu przejdź do chmury i utwórz kolejny *feed* nazwany **leds**.

2. Następnie przejdź do pulpitu chmury Adafruit IO i stwórz nowy blok typu przycisk. Powiąż go z feed: *leds*.

Zadanie 8

3. Wróćmy do kodu z zadania 7 i dodajmy zmienną przechowującą nazwę *feed* i konfiguracje diody LED:

```
1  ...
2  import iot4schools
3
4  LED = machine.Pin(15, machine.Pin.OUT)
5  dht = DHT11(14)
6  WIFI_SSID = "your_wifi_name"
7  WIFI_PASSWORD = "your_wifi_password"
8  ADAFRUIT_IO_USERNAME = "username"
9  ADAFRUIT_IO_KEY = "key"
10 FEED1 = "your_username/feeds/Temp"
11 FEED2 = "your_username/feeds/humidity"
12 FEED_SERVO = "your\_username/feeds/servo"
13 FEED_LEDS = "your_username/feeds/leds"
14
15 iot4schools.connect_wifi(WIFI_SSID, WIFI_PASSWORD)
```

Zadanie 8

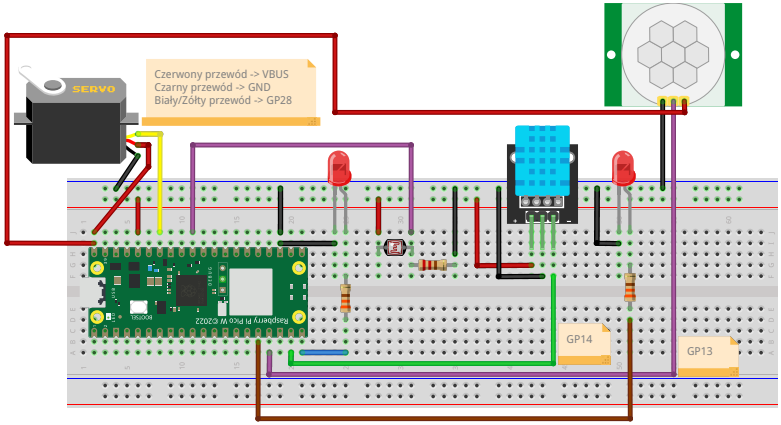
4. Zmodyfikujmy funkcje do odbierania danych z chmury Adafruit IO tak by sprawdzała stan przycisku z pulpitu chmury:

```
1  ...
2  def Receive_Data(feed, msg):
3      if FEED_SERVO in feed:
4          action = str(msg, "utf-8")
5          print("action = {} ".format(action))
6          angle.write(float(action))
7      if FEED_LEDS in feed:
8          action = str(msg, "utf-8")
9          print("action = {} ".format(action))
10         if action == "ON":
11             LED.value(1)
12         else:
13             LED.value(0)
14
15 client.set_callback(Receive_Data)
16 client.subscribe(FEED_SERVO)
17 client.subscribe(FEED_LEDS)
```

5. Przetestuj program!

Zadanie 9

- Ostatni element automatyki domowej polega na dodaniu alarmu. W tym celu wykorzystamy czujnik ruchu i ostrzegawczą diodę LED.
  - Czujnik ruchu zwraca 1 gdy wykryje ruch. W przeciwnym wypadku zwraca wartość 0.



Zadanie 9

1. Zacznijmy do konfiguracji czujnika ruchu PIR:

```
1  import machine
2  import utime
3  from dht import DHT11
4  import network
5  from umqtt.simple import MQTTClient
6  import iot4schools
7  from servo import Servo
8
9  PIR = machine.Pin(13, machine.Pin.IN)
10
11 servo = Pin(28)
12 angle = Servo(pin_id=servo)
13 angle.write(0)
14 sleep(0.5)
15 dht = DHT11(14)
16 ...
```

Zadanie 9

2. Dodajmy teraz konfiguracje kolejnej diody LED:

```
import machine
import utime
from dht import DHT11
import network
from umqtt.simple import MQTTClient
import iot4schools
from servo import Servo

PIR = machine.Pin(13, machine.Pin.IN)
LED2 = machine.Pin(12, machine.Pin.OUT)

servo = Pin(28)
angle = Servo(pin_id=servo)
angle.write(0)
sleep(0.5)
dht = DHT11(14)
...
```

Zadanie 9

3. Ostatni krok to sprawdzenie czy wykryto ruch, jeśli tak to zostanie zapalona dioda LED:

```
...
while True:
    ...
    print("Temperatura:", temp, "°C")
    print("Wilgotnosc:", hum, "%")
    send_data(temp, hum)
    if temp > 22:
        angle.write(90)
    else:
        angle.write(0)
        if PIR.value() == 1:
            print("Motion Detected")
            ledred.value(1)
        else:
            ledred.value(0)
        ...
```

Cały układ automatyki jest gotowy! Miłego testowania!

Jak skończysz to proszę zostaw swoją opinię o warsztatach: https://forms.gle/eZnnZd7E2rmdQ5aq9

Więcej materiałów na temat IoT w języku PL i ENG jest dostępnych tutaj: https://www.iot.fizyka.pw.edu.pl