

Project number: 2023-1-PL01-KA220-SCH-000154043



Co-funded by
the European Union

IoT4Schools

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

**Inteligentne kosze na śmieci: jak usprawnić gospodarkę
odpadami w inteligentnych miastach?**

Arkusze pracy dla uczniów

Autorzy: Angelika Tefelska, Dariusz Tefelski

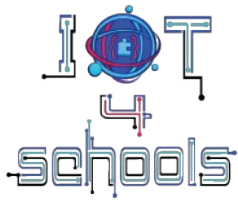
Instytucja: Warsaw University of Technology, Faculty of Physics

Licencja: CC BY-NC 4.0 LEGAL CODE, Attribution-NonCommercial 4.0 International



Co-funded by
the European Union

The European Commission's support to produce this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



Zespół:

1. Czas na burzę mózgów

Jak wygląda zarządzanie odpadami w mieście? (metody odbioru odpadów, przetwarzanie, statystyki recyklingu, statystyki zużycia paliwa przez śmieciarki) Wyszukaj informacje na ten temat w internecie.

.....

.....

.....

.....

.....

.....

.....

.....

Jak można ulepszyć obecny system zarządzania odpadami? Jak można ulepszyć odbiór odpadów z domów/mieszkań?

.....

.....

.....

.....

.....

.....

Jeśli chciałbyś/chciałabyś zbudować inteligentny kosz na śmieci, jak powinien on działać?

.....

.....

.....

.....

.....

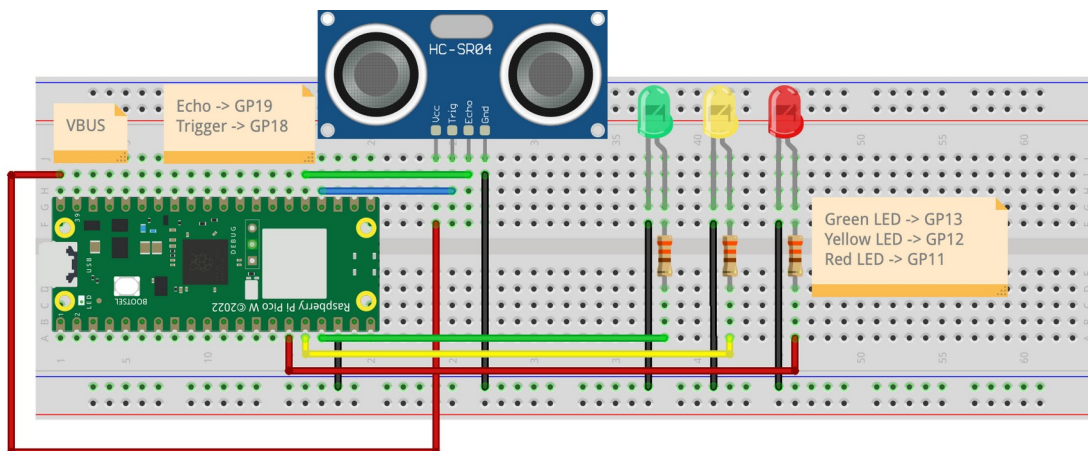
2. Czas zaprojektować własny inteligentny kosz na śmieci

Pomyśl, jak powinien wyglądać inteligentny kosz na śmieci, którego zadaniem będzie mierzenie poziomu zapełnienia kosza. W tym celu wykorzystamy ultradźwiękowy czujnik odległości, który mierzy odległość od przeszkód. Gdzie umieścić ten czujnik w koszu na śmieci, aby uzyskać informacje o zapełnieniu? Ponadto w projekcie zostaną wykorzystane trzy diody LED (czerwona, żółta i zielona). Gdzie umieścić te diody na koszu na śmieci, aby użytkownik wiedział, jak pełny jest kosz? Przy jakich wartościach zapełnienia kosza będą się świecić poszczególne diody? Czy przesyłanie danych o poziomie zapełnienia kosza na śmieci do chmury jest dobrym rozwiązaniem?

This image shows a full page of white paper with horizontal dotted lines, typical of notebook paper. The lines are evenly spaced and run across the width of the page. There is no handwriting or other markings on the paper.

3. Czas na wstępne ćwiczenia

Zanim przejdziemy do budowy inteligentnego kosza na śmieci, poznamy elementy, których będziemy używać. Aby to zrobić, wykonamy dwa krótkie zadania rozgrzewkowe. Pierwsze z nich dotyczy sygnalizacji świetlnej. Drugie dotyczy działania ultradźwiękowego czujnika odległości. Najpierw zbuduj obwód elektroniczny zgodnie z poniższym rysunkiem. Ten obwód zostanie użyty do dwóch wstępnych zadań.



3.1 Pierwsze zadania wstępne: sygnalizacja świetlna

Każdy program napisany w MicroPythonie do programowania Raspberry Pi Pico wygląda następująco:

```
1 import machine
2
3 #instructions that are to be executed only once here,
4 #e.g. configuration, creating variables
5
6 while True:
7     #instructions to be repeated over and over
```

Na początku dodajemy bibliotekę *machine*, która zawiera podstawowe funkcje do obsługi mikrokontrolera. Następnie musimy dodać instrukcje, które mają być wykonane tylko raz na samym początku, takie jak konfiguracja, tworzenie zmiennych lub tworzenie własnych funkcji. Następnie zawsze mamy nieskończoną pętlę, w której umieszczamy instrukcje, które będą powtarzane w kółko.

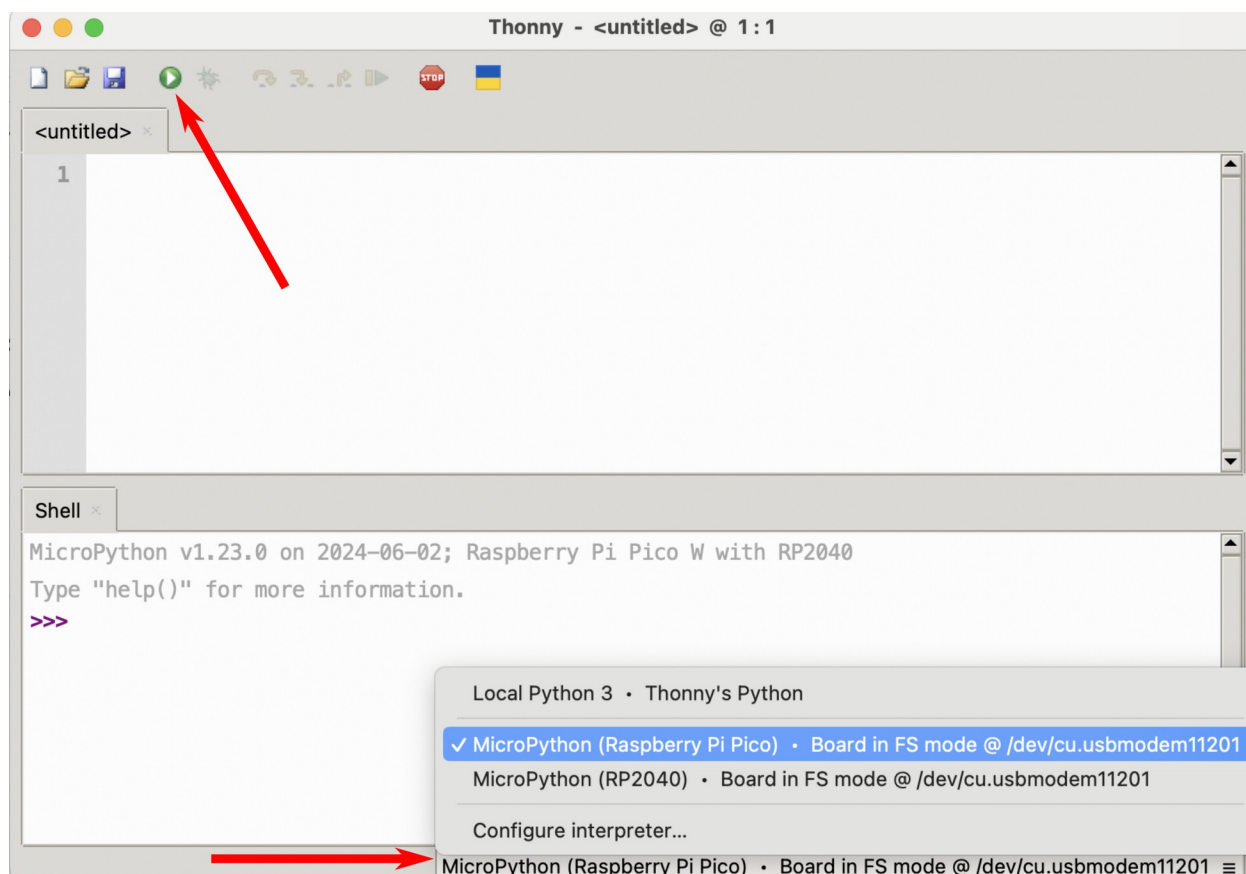
Podstawowe funkcje niezbędne do wykonania ćwiczenia to:

- **machine.Pin(numer pinu, machine.Pin.IN bądź machine.Pin.OUT)** - określa, czy dany pin powinien być wejściowy (*machine.Pin.IN*) czy wyjściowy (*machine.Pin.OUT*). Piny wejściowe są używane, gdy chcemy odczytać dane z elementu podłączonego do pinu, np. gdy mamy podłączony przycisk, odczytujemy, czy został naciśnięty (wartość 1) czy nie (wartość 0). Piny wyjściowe są używane, gdy chcemy kontrolować dany element podłączony do pinu, np. diodę LED, czy powinna być włączona (wartość 1) czy wyłączona (wartość 0).
- **value(0 bądź 1)** - funkcja ustawiająca wartość na danym pinie.
- **utime.sleep(czas w sekundach)** - funkcja, która zawiesza wykonywanie programu na dany czas. Aby jej użyć, musisz dołączyć bibliotekę *utime*: `import utime`.

Jeśli chcielibyśmy, aby dioda LED podłączona do pinu GP15 migiała co 1s, kod wyglądałby następująco:

```
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)
9     led.value(0)
10    utime.sleep(1)
```

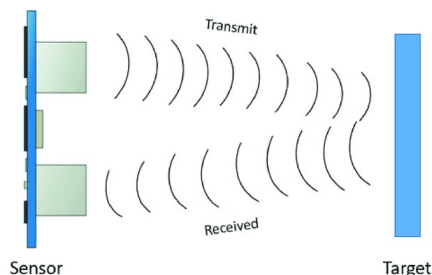
Otwórz edytor Thonny, a następnie wybierz „MicroPython (Raspberry Pi Pico)”, jak pokazano na rysunku. Po poprawnym połączeniu z płytą zielony przycisk Run powinien być aktywny (nie wyszarzony). Jeśli jest wyszarzony, wybierz ponownie „MicroPython (Raspberry Pi Pico)”.



Teraz spróbuj stworzyć program, który będzie symulował sygnalizację świetlną.

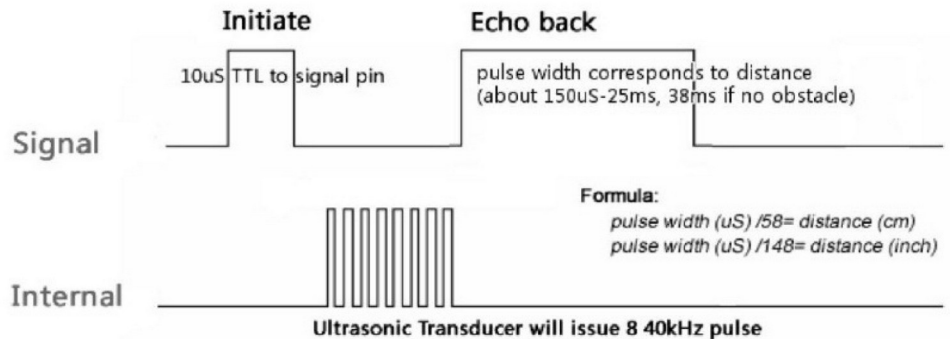
3.2 Wstępne zadanie nr 2: ultradźwiękowy czujnik odległości

Ultradźwiękowy czujnik odległości HC-SR04 pozwala mierzyć odległość od przeszkody. Ideę pomiaru pokazano na poniższym rysunku:



Źródło: https://www.researchgate.net/figure/A-block-diagram-of-Ultrasonic-sensor-working-principles_fig5_344385811

Zgodnie z dokumentacją do czujnika (patrz rysunek poniżej), aby dokonać pomiaru odległości należy ustawić niski sygnał na *trigger* przez krótki czas, np. 2µs. Następnie należy ustawić wysoki sygnał przez 10µs. Aby wygenerować opóźnienia w mikrosekundach, użyj funkcji: **utime.sleep_us()**. W następnym kroku należy ustawić niski sygnał na *trigger*.



Źródło: <https://www.electronicoscaldas.com/datasheet/HC-SR04.pdf>

Utwórz program, który odczytuje zmierzoną odległość przez ultradźwiękowy czujnik odległości i wyświetli ją w terminalu. Aby to zrobić:

1. Dodaj biblioteki: *machine* i *utime*.
2. Skonfiguruj pin, do którego podłączony jest trigger, jako wyjściowy.
3. Skonfiguruj pin, do którego podłączono echo, jako wejściowy.
4. W pętli while ustaw wartość 0 na pinie trigger-a.
5. Odczekaj 2 µs (funkcja **utime.sleep_us(czas)**).
6. Ustaw wartość 1 na pinie trigger-a.
7. Odczekaj 10 µs.
8. Ustaw wartość 0 na pinie trigger-a.
9. Teraz musimy zmierzyć, jak długo trwał wysoki sygnał na pinie echa, ponieważ czas trwania sygnału na pinie echa jest proporcjonalny do odległości. Aby to zrobić, użyjemy funkcji **utime.ticks_us()**, która mierzy, ile czasu upłynęło w µs od uruchomienia programu. Najpierw utworzymy pętlę while, która będzie wykonywała się tak długo, jak długo będzie niski sygnał na pinie echa. Wewnątrz umieścimy funkcję *tick_us()*. W ten sposób otrzymamy informacje o tym, kiedy sygnał był ostatnio niski. Dodaj ten fragment kodu do swojego programu:

```
while echo.value()==0:
    signal_off = utime.ticks_us()
```

10. Podobnie zmierz, kiedy sygnał był ostatnio wysoki i zapisz wartość w zmiennej o nazwie, np. **signal_on**.

11. Różnica między czasem ostatniego wystąpienia sygnału wysokiego i niskiego to czas trwania impulsu wysokiego na pinie echa. Jak zamienić czas trwania impulsu na odległość? Na początku emitowana jest fala dźwiękowa, która odbija się od obiektu i powraca do czujnika. Dlatego w zmierzonym czasie (t) fala pokonuje dwukrotność odległości między czujnikiem a obiektem i porusza się z prędkością około 340 m/s (prędkość dźwięku w powietrzu). Dlatego możemy zapisać następujące równanie:

$$v = \frac{2d}{t}$$

$$d = v \cdot \frac{t}{2} = 0.034 \frac{\text{cm}}{\mu\text{s}} \cdot \frac{t}{2} \approx \frac{t[\mu\text{s}]}{58}$$

Zatem zmierzony czas trwania impulsu należy podzielić przez 58, aby uzyskać odległość w centymetrach. Dodaj następujące wiersze do programu:

```
diff = signal_on-signal_off
distance = diff/58.0
print("Distance="+str(distance))
utime.sleep(0.1)
```

Teraz możesz przetestować działanie programu.

4. Inteligentny kosz na śmieci – poziom 1

Teraz stwórz program, w którym wykorzystasz umiejętności zdobyte podczas wstępnych zadań i zmierz poziom napełnienia kosza na śmieci. Wyświetl poziom napełnienia na diodach LED zgodnie z zakresami procentowymi zajętości kosza, które założyłeś na etapie projektowania kosza na śmieci.

Stwórz kosz na śmieci z nieużywanego pudełka po przesyłce/po butach i przetestuj, jak działa Twój kosz.

5. Inteligentny kosz na śmieci – poziom 2

Teraz zmodyfikujmy kosz na śmieci, aby działał jeszcze efektywniej. W tym celu wyślemy do chmury dane o zapełnieniu kosza na śmieci i jego lokalizacji. Dane te mogą być później wykorzystane do stworzenia algorytmu, który opracowuje trasę dla śmieciarki, aby zbierała śmieci tylko z tych miejsc, w których kosze są pełne, minimalizując w ten sposób ślad węglowy. W tym celu wykorzystamy chmurę Adafruit IO.


Najpierw należy założyć bezpłatne konto na <https://io.adafruit.com>. Następnie będziesz potrzebował/potrzebowała wysłać dwie wartości do chmury: zapełnienie kosza na śmieci i lokalizację do chmury. Aby to zrobić, należy utworzyć dwa *feeds*. *Feeds* to obiekty, które przechowują dane. Aby utworzyć *feed*, przejdź do zakładki „Feeds” i wybierz przycisk „New Feed”.

angtef / Feeds





Następnie pojawi się okno, w którym należy wpisać nazwę *feed* np. Filling czy Location.

Create a new Feed 


Name


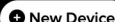
Maximum length: 128 characters. Used: 0


Description



 


Utwórz dwa *feeds*. Gdy to zrobisz, powinieneś/powinnaś zobaczyć *feeds*, które utworzyłeś na swojej stronie, podobnie jak na zrzucie ekranu poniżej:



Shop Learn Blog Forums IO LIVE! AdaBox Hi, Angelika Tefelska | Account  0



adafruit Devices Feeds Dashboards Actions Power-Ups  

angtef / Feeds 



Default  

Feed Name	Key	Last value	Recorded
<input type="checkbox"/> Filling	filling-the-waste-bin	71.62069	about 5 hours ago 
<input type="checkbox"/> Location	location	52.2297,21.0122	about 6 hours ago 

Następnym krokiem jest utworzenie pulpitu do wyświetlania danych. Aby to zrobić, wybierz zakładkę „Dashboards”, a następnie „New Dashboard”:

Shop Learn Blog Forums IO LIVE! AdaBox Hi, Angelika Tefelska | Account  0

adafruit Devices Feeds **Dashboards** Actions Power-Ups  


angtef / Dashboards 





Pojawi się okno, w którym należy wpisać nazwę wybranego pulpitu. Następnie po prawej stronie wybierz symbol ustawień i wybierz "Create New Block".

Shop Learn Blog Forums IO LIVE! AdaBox Account  0

adafruit Devices Feeds Dashboards Actions Power-Ups  

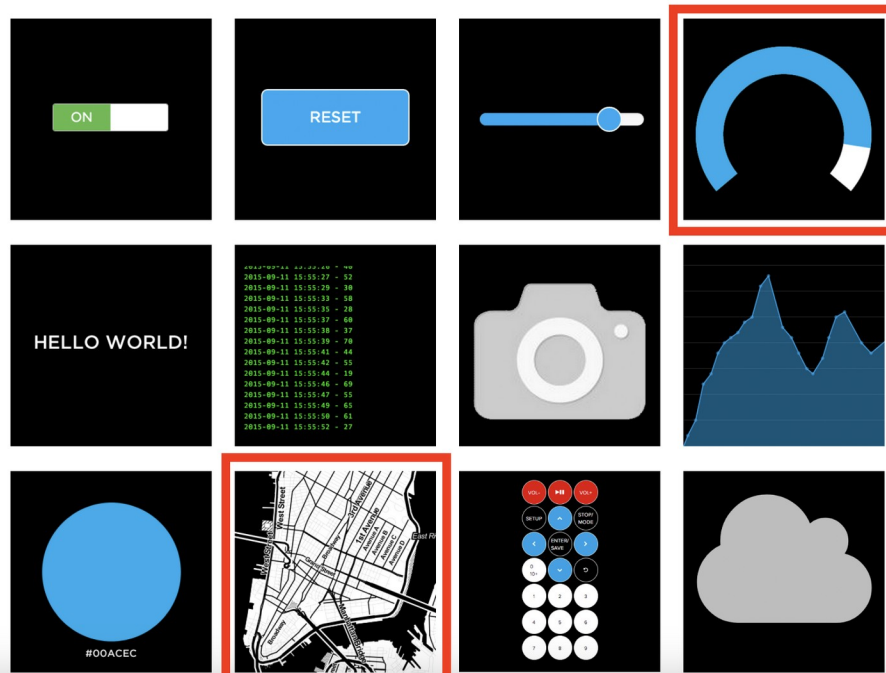
angtef / Dashboards / Test  

Adafruit IO ma różne bloki dostępne do wyświetlania danych (pola tekstowe, wskaźniki, wykresy, mapy itp.). W naszym przypadku wybierzmy wskaźnik i mapę:

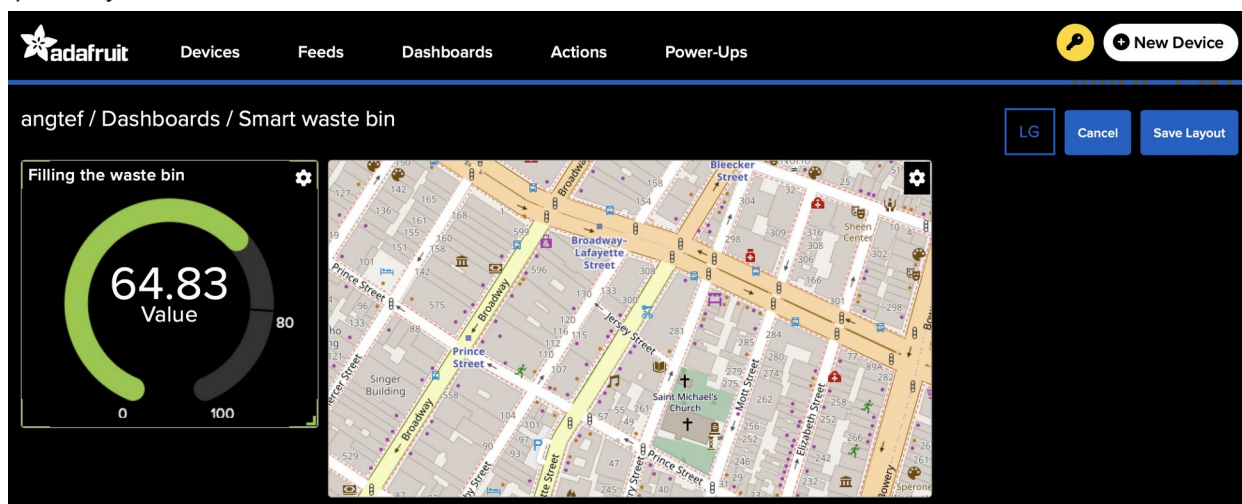
Create a new block



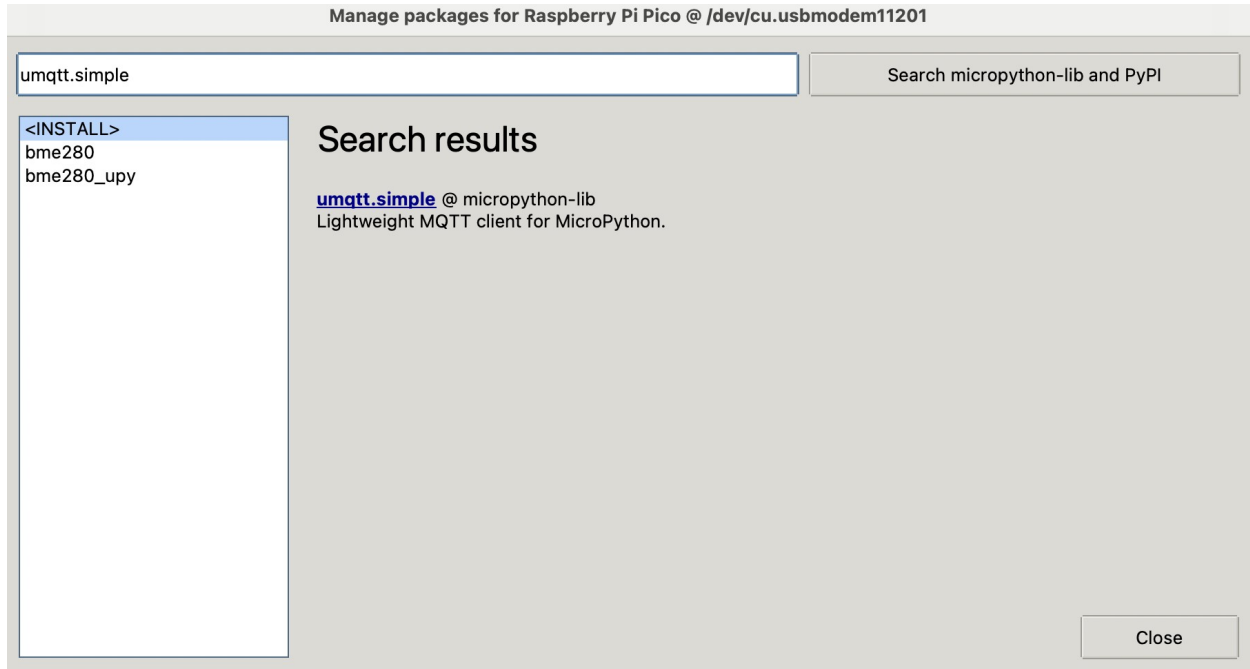
Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.



Następnie pojawi się okno z pytaniem, do którego *feed* chcemy podłączyć blok. Wybierz *feed* definiującą wypełnienie kosza na śmieci w przypadku wskaźnika i *feed* definiującą lokalizację w przypadku mapy. Następnie ponownie wybierz ikonę ustawień i wybierz „Edit Layout”. Umieść bloki na ekranie tak, jak uważasz za stosowne. Możesz również powiększać i zmniejszać bloki. Przykład wyglądu pokazano na poniższym rzucie ekranu:



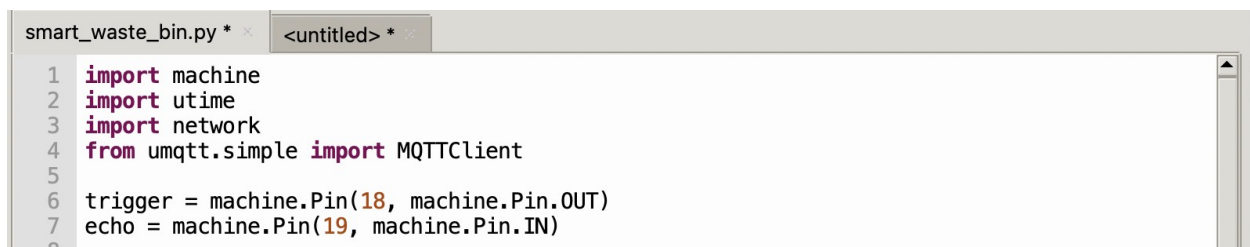
Przejdźmy teraz do edytora Thonny i zainstalujmy bibliotekę *umqtt.simple*. Aby to zrobić, wybierz „Narzędzia”, a następnie „Zarządzaj pakietami”. Pojawi się okno, w którym należy wpisać nazwę biblioteki, którą chcesz zainstalować, w tym przypadku *umqtt.simple*:



Po kliknięciu nazwy znalezionej biblioteki *umqtt.simple* otworzy się okno z danymi biblioteki. Będziesz mógł/a zainstalować bibliotekę, klikając przycisk Instaluj.

Teraz możemy wrócić do naszego kodu i dodać części niezbędne do wysłania danych do chmury. Aby to zrobić, wykonaj następujące kroki:

1. Dodaj niezbędne biblioteki (*network* i *MQTTClient* z biblioteki *umqtt.simple*):



2. Dodaj fragment kodu, który pozwoli Ci połączyć się z Twoją siecią WIFI. Tutaj musisz wypełnić wiersze 15-18. Najpierw wpisz nazwę swojej sieci WIFI, a następnie hasło.

```
smart_waste_bin.py × backup.py
1 import machine
2 import utime
3 import network
4 from umqtt.simple import MQTTClient
5
6 trigger = machine.Pin(18, machine.Pin.OUT)
7 echo = machine.Pin(19, machine.Pin.IN)
8
9 led_green = machine.Pin(13, machine.Pin.OUT)
10 led_yellow = machine.Pin(12, machine.Pin.OUT)
11 led_red = machine.Pin(11, machine.Pin.OUT)
12
13 depth = 100
14
15 WIFI_SSID = "your_wifi_name"
16 WIFI_PASSWORD = "your_wifi_password"
17 ADAFRUIT_IO_USERNAME = "username"
18 ADAFRUIT_IO_KEY = "key"
19
20 def connect_wifi():
21     wlan = network.WLAN(network.STA_IF)
22     wlan.active(True)
23     wlan.connect(WIFI_SSID, WIFI_PASSWORD)
24     while not wlan.isconnected():
25         print("Connecting to Wi-Fi...")
26         utime.sleep(1)
27     print("Connected to Wi-Fi:", wlan.ifconfig())
28
29 connect_wifi()
30
31 while True:
```

Ostatnie dwa parametry to dane z Adafruit IO. Wróć do strony Adafruit i kliknij symbol klucza. Kiedy to zrobisz, pojawi się Twój login i klucz. Skopiuj te dane do programu do wierszy 17-18:

YOUR ADAFRUIT IO KEY


Your Adafruit IO Key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your Adafruit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.


If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username

Active Key

[REGENERATE KEY](#)





+ New Device

LG

Cancel

Save Layout

3. Teraz użyjemy MQTT (*Message Queuing Telemetry Transport*), lekkiego protokołu komunikacyjnego opartego na modelu *publish/subscribe*. Został on zaprojektowany specjalnie do przesyłania danych w środowiskach o ograniczonych zasobach, takich jak urządzenia IoT (*Internet of Things*). Aby to zrobić, użyj poniższego kodu, zmieniając tylko nazwy *feeds* w wierszach 33-34. W poniższym przykładzie *feeds* nazwano: „Filling” i „Location”. Zastąp je własnymi. Pamiętaj tylko, aby zostawić */csv* w przypadku *Location*, ponieważ podczas korzystania z map musisz podać dane w formacie csv.

```
smart_waste_bin.py * x backup.py
13 depth = 100
14
15 WIFI_SSID = "your_wifi_name"
16 WIFI_PASSWORD = "your_wifi_password"
17 ADAFRUIT_IO_USERNAME = "username"
18 ADAFRUIT_IO_KEY = "key"
19
20 def connect_wifi():
21     wlan = network.WLAN(network.STA_IF)
22     wlan.active(True)
23     wlan.connect(WIFI_SSID, WIFI_PASSWORD)
24     while not wlan.isconnected():
25         print("Connecting to Wi-Fi...")
26         utime.sleep(1)
27     print("Connected to Wi-Fi:", wlan.ifconfig())
28
29 ADAFRUIT_IO_SERVER = "io.adafruit.com"
30 ADAFRUIT_IO_PORT = 1883 # Port MQTT
31 CLIENT_ID = "raspberrypi_pico"
32
33 FEED_FILL_LEVEL = "angtef/feeds/Filling"
34 FEED_LOCATION = "angtef/feeds/Location/csv"
35
36 client = MQTTClient(CLIENT_ID, ADAFRUIT_IO_SERVER, ADAFRUIT_IO_PORT, ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)
37
38 def connect_mqtt():
39     client.connect()
40     print("Connected to Adafruit IO!")
41
42 connect_wifi()
43 connect_mqtt()
```

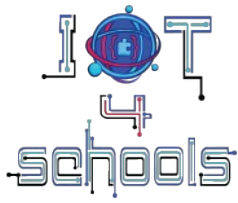
4. Teraz dodajmy funkcję wysyłającą dane do chmury:

```
38 def connect_mqtt():
39     client.connect()
40     print("Connected to Adafruit IO!")
41
42 connect_wifi()
43 connect_mqtt()
44
45 def send_data(Filling, Location):
46     client.publish(FEED_FILL_LEVEL, str(Filling))
47     print("Fill level sent:"+str(Filling))
48     client.publish(FEED_LOCATION, Location)
49     print("Location sent:"+str(Location))
50
```

Ostatnim krokiem jest przekazanie zmierzonego zapełnienia i lokalizacji do funkcji w pętli while. Możesz odczytać swoją lokalizację, na przykład z map Google i zastąpić ją w zmiennej „location”.

```
56 while True:
57     #your code
58
59     #value, latitude, longitude, altitude
60     location = "0, 52.2297,20.0122,0"
61     #fill_level - this is the measured fullness of the bin,
62     #correct the variable name to the one you used in your code
63     send_data(fill_level, location)
64
```

Przetestuj inteligentny kosz na śmieci. Pamiętaj, aby wrócić na stronę Adafruit IO do pulpitu, który utworzyłeś/aś, aby sprawdzić, czy poprawnie wysyłasz swoje dane do chmury.



6. Podsumowanie: refleksja nad funkcjonalnością i możliwymi usprawnieniami

Czy uważasz, że korzystanie z inteligentnych pojemników może usprawnić zbiórkę odpadów i zmniejszyć ślad węglowy? W projekcie nie optymalizowaliśmy trasy śmieciarek, ale możesz sobie wyobrazić, że istnieje system, który odczytuje dane z chmury z każdego pojemnika i przygotowuje optymalną trasę. . Jeśli jesteś zainteresowany tym, jak stworzyć taki algorytm, możesz zapoznać się z programami ze strony internetowej: <https://colab.research.google.com/drive/1aOq9jRh6c6fhaw1ahe0a1-yKVdMnO613?usp=sharing%2F> .

.....

.....

.....

.....

.....

Jakie inne zmiany można wprowadzić, aby inteligentne pojemniki działały jeszcze wydajniej?

.....

.....

.....

.....

.....