

Projectnummer: 2023-1-PL01-KA220-SCH-000154043



Co-funded by
the European Union

IoT4Schools

**"Het internet der dingen in het onderwijs brengen middel om
de uitdagingen van de 21^{ste} eeuw aan te gaan"**

Slimme afvalbakken: hoe afvalbeheer verbeteren in slimme steden?

Richtlijnen voor docenten

Auteurs: Angelika Tefelska, Dariusz Tefelski, Adam Kowalczyk (fotografie)

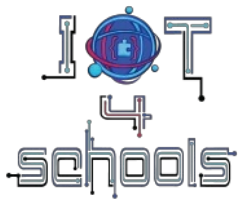
Organisatie: Technische Universiteit Warschau, Faculteit Natuurkunde

Licentie: CC BY-NC 4.0 LEGAL CODE, Naamsvermelding-NietCommercieel 4.0 Internationaal



Co-funded by
the European Union

De steun van de Europese Commissie voor de productie van deze publicatie houdt geen goedkeuring in van de inhoud, die uitsluitend de standpunten van de auteurs weergeeft, en de Commissie kan niet verantwoordelijk worden gehouden voor het gebruik van de informatie die erin is vervat.



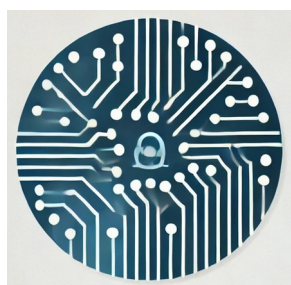
Inhoudsopgave

1 Inleiding tot het project.....	3
1.1 Scenario en reikwijdte van het project	3
1.2 Leerdoelen	4
1.3 Leertraject - Implementatiefasen	4
1.4 Voorwaarden voor leren	5
1.5 Hardware en software.....	5
1.6 Tijdsplanning	6
2 Uitvoering van het project.....	7
2.1 Nivea1	7
2.1.1 Circuit maken	7
2.1.2 Programmeren	7
2.1.3 Crafting	16
2.2 Nivea2.....	16
2.2.1 Programmeren.....	16
3 Tips en aanbevelingen.....	23

1 Inleiding tot het project

1.1 Scenario en reikwijdte van het project

Het doel van dit project is om studenten kennis te laten maken met de Internet of Things (IoT)-technologie in de context van slimme steden. Sinds enkele jaren is het concept van slimme steden ontwikkeld, waarbij IoT-oplossingen gebruikt worden en zullen worden om de leefomstandigheden te verbeteren en de negatieve impact van steden op het milieu te minimaliseren. Een van de belangrijke problemen van moderne steden is vuilnisophaling. Momenteel halen de meeste steden afval op volgens een vast schema, waarbij ze langs alle eigendommen rijden, ongeacht de mate waarin de bakken gevuld zijn. Hetzelfde gebeurt met bakken die langs de stoepen staan en door voorbijgangers worden gebruikt om afval in te gooien. Een dergelijk systeem is inefficiënt en veroorzaakt een aanzienlijk brandstofverbruik. Dit project wil laten zien hoe je een intelligente afvalbak kunt maken met behulp van een Raspberry Pi Pico-bord en een ultrasone afstandssensor om de mate van vulling van de afvalbak te meten. De mate van vulling van de vuilnisbak wordt naar de cloud gestuurd om de route van de vuilniswagen te optimaliseren en zo de ecologische voetafdruk te minimaliseren. Dergelijke oplossingen worden al langzaam ingevoerd in sommige steden, bijvoorbeeld in Antwerpen (zie foto's hieronder).



Het project heeft als doel de digitale competenties van studenten te ontwikkelen. IoT-technologie combineert veel gebieden: elektronica (sensoren meten), programmeren (in dit geval de programmeertaal MicroPython - een versie van Python speciaal voor het programmeren van microcontrollers), netwerken (clouds gebruiken) en gegevensanalyse. Tijdens het project ontwikkelen de studenten de bovengenoemde digitale en technologische competenties.



Als onderdeel van het project leren de leerlingen over het probleem van afvalbeheer in de stad, een uiterst belangrijk onderwerp om de negatieve invloed van steden op het milieu te minimaliseren door de koolstofvoetafdruk te verkleinen (minder brandstofverbruik door vuilniswagens). Daarnaast zal het onderwerp recycling worden besproken.



Het project wordt in teams uitgevoerd door studenten met verschillende achtergronden en landen. Door gezamenlijk samen te werken hopen we dat studenten integreren ondanks culturele verschillen of taalbarrières.

1.2 Leerdoelen

Door dit project zullen de leerlingen in staat zijn om:

- Maak een eenvoudig programma in de programmeertaal MicroPython.
- Bouw een elektronisch circuit met het Raspberry Pi Pico-bord.
- Bouw en programmeer een apparaat dat het vullen van afvalbakken kan meten.
- Leer sensoren zoals de ultrasone sensor te gebruiken.
- Begrijpen hoe IoT-apparaten gegevens kunnen verzamelen en doorsturen naar de cloud.
- Begrijpen en uitleggen hoe gegevens in realtime kunnen worden gecontroleerd.
- Oplossingen aangeven voor een efficiënter en ecologischer afvalbeheer in steden.
- De voordelen van recycling en de huidige problemen bij afvalverwerking aangeven.

1.3 Leertraject - Implementatiefasen

Als onderdeel van het project zullen studenten oplossingen bedenken om het afvalbeheer in steden te verbeteren en zo hun ecologische voetafdruk te minimaliseren. Hiervoor gebruiken ze een Raspberry Pi Pico-bord en een ultrasone afstandssensor.

Hier volgen enkele suggesties om het project van de slimme prullenbak soepel en effectief te implementeren met je leerlingen:

1. **Groepsvorming:** Verdeel je leerlingen in teams van twee of drie.
2. **Brainstormen:** Stimuleer elk team om informatie te zoeken over mogelijke manieren om afval in de stad slimmer te beheren (methode van afvalinzameling, routeoptimalisatie, voor- en nadelen van verschillende oplossingen, recyclagestatistieken, brandstofverbruik van vuilniswagens, enz.)



3. **Discussie en toewijzing van de activiteit:** Moedig elk team aan om hun bevindingen en ideeën te delen over hoe het afvalbeheersysteem kan worden verbeterd om het slimmer te maken. Introduceer na de discussie het specifieke doel van het project. (Opmerking: Het is aan te raden om het specifieke doel van het project na het brainstormen te introduceren om uw leerlingen aan te moedigen het project van de slimme afvalbak in een bredere context te bekijken).
4. **Planning:** Stimuleer elk team om na te denken over hoe ze het apparaat zullen bouwen (hoe de bak zal worden gemaakt; waar de afstandssensor zal worden geplaatst; welke gegevens naar de cloud moeten worden verzonden, wanneer de gegevens moeten worden verzonden en hoe vaak).
5. **Creatie:** Stimuleer elk team om hun eigen slimme afvalbak te maken met behulp van de werkbladen voor de leerlingen. Afhankelijk van de vaardigheden van de leerlingen kun je overwegen om rollen toe te wijzen.
6. **Testen - optimalisatie:** Moedig je leerlingen na afloop van het project aan om hun slimme prullenbak te testen. Op basis van de testresultaten kun je elk team aanmoedigen om hun project te optimaliseren. Als het project niet te moeilijk was voor de leerlingen, kun je overwegen om uitbreidingen te doen, waarbij je een algoritme ontwikkelt om de route te optimaliseren (het reizende koopmanprobleem).
7. **Presentatie - delen:** Moedig je leerlingen aan om hun projecten plenair te presenteren en vraag hen om te reflecteren op de hele ervaring. Moedig alle teams aan om na te denken over de impact van zulke slimme afvalbakken op het functioneren van steden en het milieu.

1.4 Voorwaarden voor leren

Leerlingen moeten bekend zijn met de basisprincipes van programmeren in een willekeurige programmeertaal. Als ze die ervaring niet hebben, is het nog steeds mogelijk om het project te voltooien omdat de materialen zo zijn voorbereid dat iedereen het project kan voltooien.

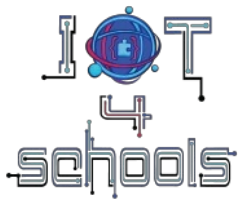
1.5 Hardware en software

Hardware:

- Raspberry Pi Pico W-bord
- Broodplank (contactplaat)
- Aansluitdraden vrouwelijk - mannelijk en mannelijk - mannelijk
- Rode, gele en groene LED-diodes
- 330 Ohm weerstanden
- Externe voedingsbron: 2AAA batterijhouder of een powerbank met laag voltage (tot 5V) - optioneel
- Ultrasonische afstandssensor HC-SR04

Software:

- Thonny redacteur
- Adafruit IO wolk



1.6 Tijdsplanning

Naar schatting heb je 4 tot 6 uur nodig om het project af te ronden:

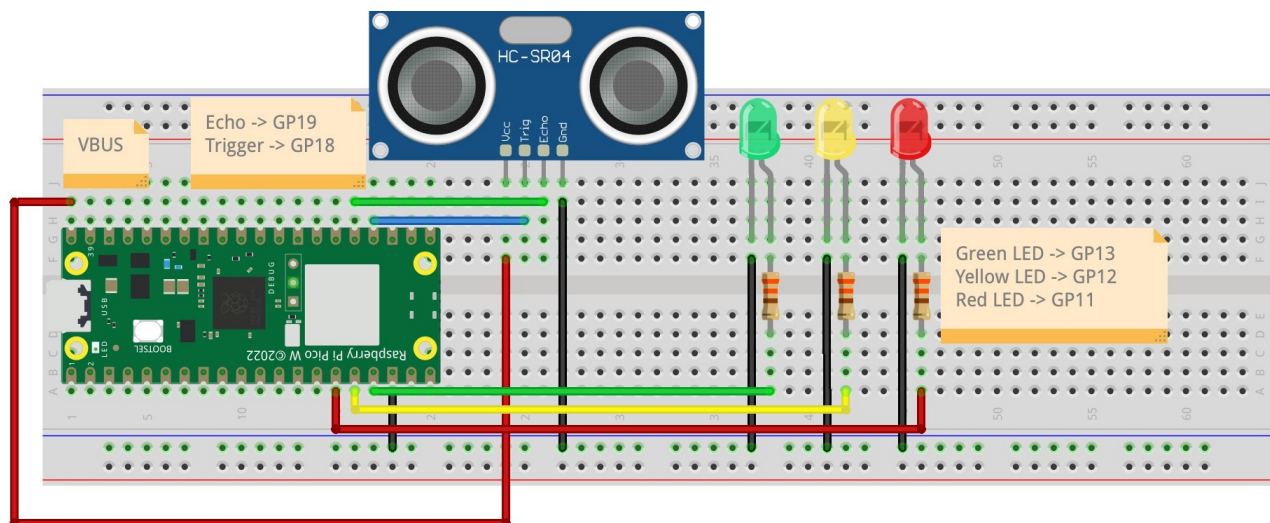
- 45-90 minuten om het project te introduceren (inclusief brainstormen en discussie), planning en opwarmactiviteit
- 45 minuten om Niveau 1 te voltooien
- 45 minuten voor niveau 2
- 45-90 minuten voor verlengingen
- 30 minuten voor afronding en discussie

2 Uitvoering van het project

2.1 Niveau 1

2.1.1 Circuit maken

Raspberry Pi Pico W moet worden verbonden met een ultrasone afstandssensor en drie LED's via weerstanden van 330. Een voorbeeldaansluiting is te zien in de figuur hieronder. In het begin kun je het systeem testen aangesloten op de computer via USB. Uiteindelijk heb je echter een powerbank of batterijen nodig om het systeem van stroom te voorzien. Om ecologische en economische redenen raden we een powerbank aan, die je na de lessen kunt opladen en opnieuw kunt gebruiken.

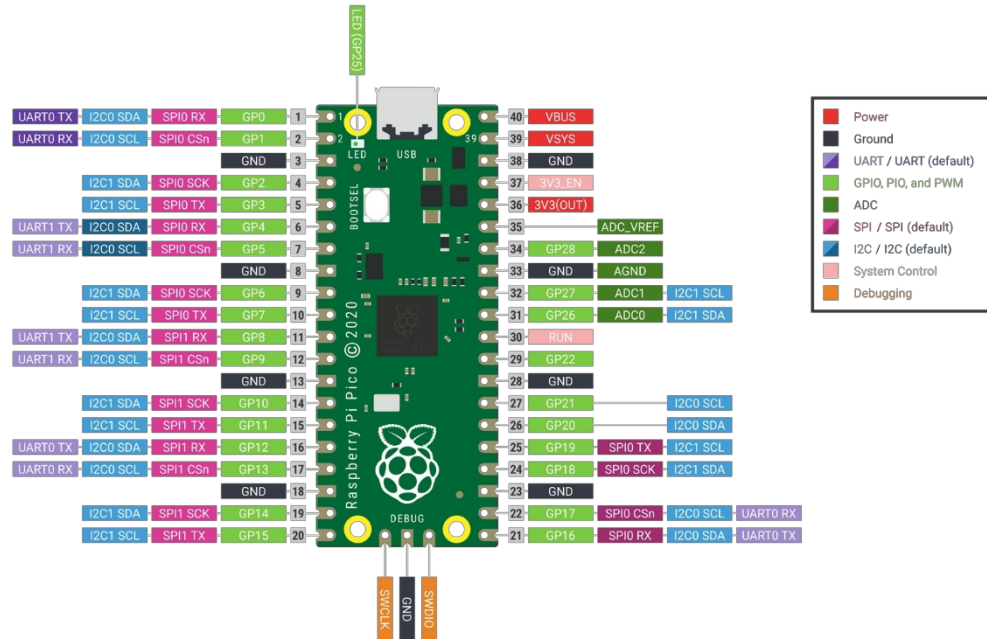


fritzing

Merk op dat op sommige breadboards de pinnen onder de blauwe streep die we gebruikten voor GND niet over de hele lengte van de printplaat zijn aangesloten. Als er een gat zit in de blauwe streep, betekent dit dat de pinnen maar op een deel van het bord zijn aangesloten. Let hierop bij het aansluiten van de printplaat.

2.1.2 Programmeren

In dit project gebruiken we GPIO-pinnen (General Purpose Input/Output), die worden gebruikt om digitale signalen te genereren of te lezen, d.w.z. signalen die slechts twee mogelijke waarden hebben: 0 of 1. GPIO zijn pinnen die zowel als ingangspinnen kunnen worden gebruikt, waarvan we informatie kunnen lezen, bijvoorbeeld over het aanzetten van een knop, als uitgangspinnen, waarop we digitale signalen kunnen genereren, bijvoorbeeld om een LED te laten knipperen. De figuur hieronder toont de 40 pinnen van het Raspberry Pi Pico-bord, die genummerd zijn vanaf linksboven. Pinnen die dienen als GPIO worden afgekort tot GPx (groene rechthoeken), waarbij x het rangtelwoord van de GPIO-pin is, beginnend bij 0 tot 28. Meer informatie in de gids: "Technische gids over Raspberry Pi Pico en MicroPython" beschikbaar op: <https://www.iot.fizyka.pw.edu.pl/results/>.

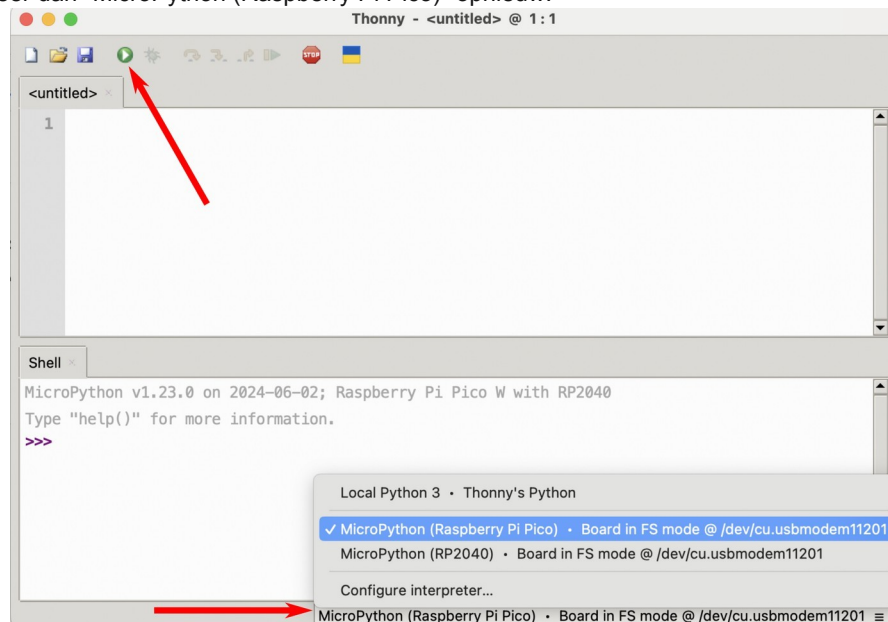


Bron: <https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html>.

Om leerlingen zoveel mogelijk zelf te laten doen, raden we aan om eerst twee opwarmoefeningen te doen.

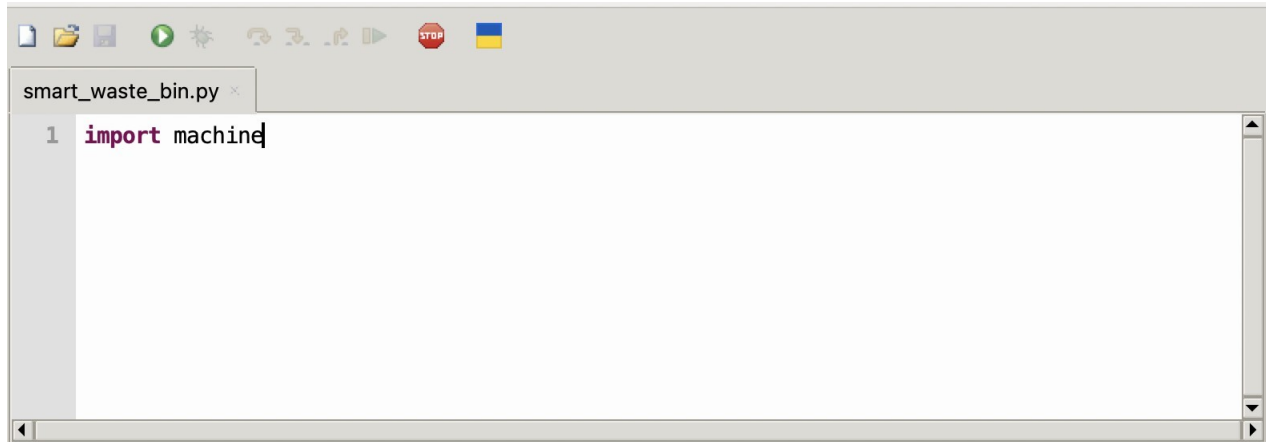
Opwarmactiviteit 1:

De eerste opwarmoefening is het maken van een programma dat verkeerslichten simuleert. Om dit te doen, open je de Thonny editor en selecteer je "MicroPython (Raspberry Pi Pico)" zoals in de afbeelding. Nadat je op de juiste manier verbinding hebt gemaakt met het bord, moet de groene Run-knop actief zijn (niet grijs weergegeven). Als het grijs is, selecteer dan "MicroPython (Raspberry Pi Pico)" opnieuw.



Laten we nu een programma voor verkeerslichten schrijven. Volg deze stappen om dit te doen:

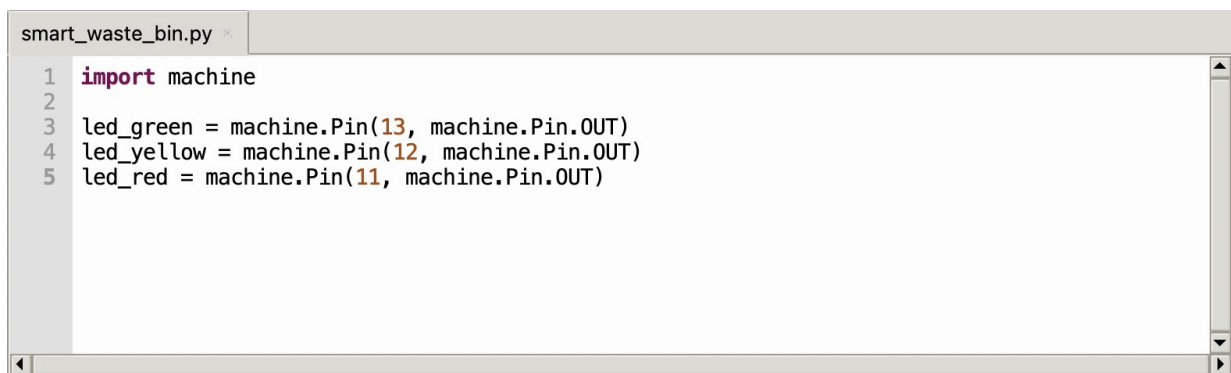
1. Laten we beginnen met het opnemen van de bibliotheek in MicroPython:



```
smart_waste_bin.py x
1 import machine
```

De machinebibliotheek bevat alle nodige instructies om te communiceren met de Raspberry Pi Pico. Het commando import voegt de gespecificeerde bibliotheek toe aan het project.

2. Vervolgens moet je de GP13, GP12 en GP11 configureren zodat ze als uitgangspennen werken, omdat we de LED's besturen door een waarde van 1 of 0 naar de GPIO-pinnen te sturen. Hiervoor wordt de Pin-functie uit de machinebibliotheek gebruikt. Om in Micropython functies uit een bibliotheek aan te roepen, geven we eerst de naam van de bibliotheek op en na de punt de naam van de functie uit deze bibliotheek, dus *machine.Pin()*. De Pin-functie neemt twee argumenten. Het eerste is het GPIO-pinnummer. Het tweede is de specificatie of de GPIO-pin werkt als ingang (*machine.Pin.IN*) of als uitgang (*machine.Pin.OUT*). In dit geval zou de functieaanroep er dus uitzien als *machine.Pin(13, machine.Pin.OUT)*. Het object dat wordt teruggegeven door de Pin-functie wordt toegewezen aan de aangemaakte variabele *led_green*.



```
smart_waste_bin.py x
1 import machine
2
3 led_green = machine.Pin(13, machine.Pin.OUT)
4 led_yellow = machine.Pin(12, machine.Pin.OUT)
5 led_red = machine.Pin(11, machine.Pin.OUT)
```

3. In de volgende stap plaatsen we een oneindige lus, die het hoofdgedeelte van het programma zal bevatten:

```
smart_waste_bin.py x
1 import machine
2
3 led_green = machine.Pin(13, machine.Pin.OUT)
4 led_yellow = machine.Pin(12, machine.Pin.OUT)
5 led_red = machine.Pin(11, machine.Pin.OUT)
6
7 while True:
```

4. In de volgende stap laten we de rode LED oplichten door de waarde 1 naar de GP11 pin te sturen. De functie value() wordt gebruikt om de waarde op de pin in te stellen. Deze neemt de waarde 0 of 1 als argument.

```
smart_waste_bin.py * x
1 import machine
2
3 led_green = machine.Pin(13, machine.Pin.OUT)
4 led_yellow = machine.Pin(12, machine.Pin.OUT)
5 led_red = machine.Pin(11, machine.Pin.OUT)
6
7 while True:
8     led_red.value(1)
9     |
```

5. Nu moet het programma 1s wachten zodat we het effect van het oplichten van de diode kunnen zien. Hiervoor gebruiken we de utime-bibliotheek, die functies voor vertragingen bevat. Eerst moeten we de bibliotheek toevoegen:

```
smart_waste_bin.py x
1 import machine
2 import utime
3
4 led_green = machine.Pin(13, machine.Pin.OUT)
5 led_yellow = machine.Pin(12, machine.Pin.OUT)
6 led_red = machine.Pin(11, machine.Pin.OUT)
7
8 while True:
9     led_red.value(1)
10    |
```

6. Met de slaapfunctie van de utime-bibliotheek kun je het programma een seconden vertragen. Laten we een vertraging van 1s instellen nadat de diode oplicht:

```
smart_waste_bin.py *
1 import machine
2 import utime
3
4 led_green = machine.Pin(13, machine.Pin.OUT)
5 led_yellow = machine.Pin(12, machine.Pin.OUT)
6 led_red = machine.Pin(11, machine.Pin.OUT)
7
8 while True:
9     led_red.value(1)
10    utime.sleep(1)
11    |
```

7. Laten we nu de rest van de code toevoegen die de resterende LED's in de juiste volgorde laat oplichten.

```
smart_waste_bin.py
1 import machine
2 import utime
3
4 led_green = machine.Pin(13, machine.Pin.OUT)
5 led_yellow = machine.Pin(12, machine.Pin.OUT)
6 led_red = machine.Pin(11, machine.Pin.OUT)
7
8 while True:
9     led_red.value(1)
10    utime.sleep(1)
11
12    led_red.value(0)
13    led_yellow.value(1)
14    utime.sleep(1)
15
16    led_yellow.value(0)
17    led_green.value(1)
18    utime.sleep(1)
19
20    led_green.value(0)
21
```

Dankzij deze activiteit leren leerlingen hoe ze LED's kunnen gebruiken, die we zullen gebruiken om een slimme afvalbak te maken.

Opwarmactiviteit 2:

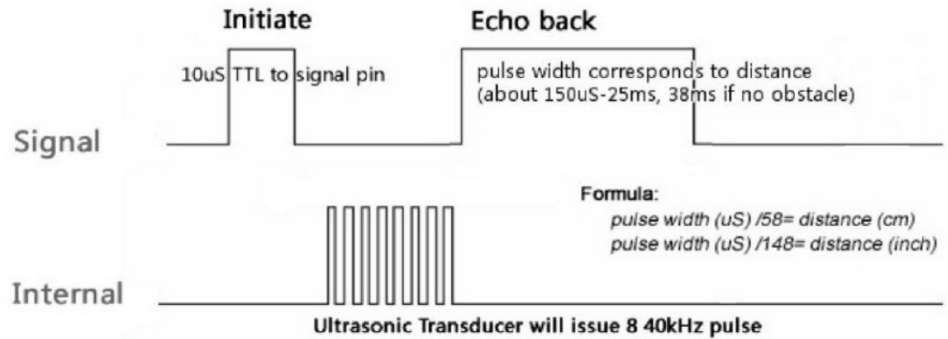
Laten we nu een programma maken dat de afstand afleest met behulp van een ultrasone afstandssensor. Volg hiervoor deze stappen:

1. Laten we eerst de benodigde bibliotheken toevoegen, dat wil zeggen machine en utime, en de pinnen configureren: GP18 (trigger) als uitgang en GP19 (echo) als ingang.

```
smart_waste_bin.py
1 import machine
2 import utime
3
4 trigger = machine.Pin(18, machine.Pin.OUT)
5 echo = machine.Pin(19, machine.Pin.IN)
6
7 |
```

2. Vervolgens voegen we in de hoofdloop een codefragment toe om de afstand tot de ultrasone afstandssensor te meten. Volgens de documentatie voor de HC-SR04 sensor (zie de afbeelding hieronder), stel je eerst de lage

signaal op de trigger voor een korte tijd, bijvoorbeeld 2µs. Zet dan het hoge signaal voor 10 µs. Om vertragingen in microseconden te genereren, gebruik je de functie: `utime.sleep_us()`. Stel in de volgende stap het lage signaal in op de trigger.



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

```
smart_waste_bin.py
1 import machine
2 import utime
3
4 trigger = machine.Pin(18, machine.Pin.OUT)
5 echo = machine.Pin(19, machine.Pin.IN)
6
7 while True:
8     trigger.value(0)
9     utime.sleep_us(2)
10    trigger.value(1)
11    utime.sleep_us(10)
12    trigger.value(0)
```

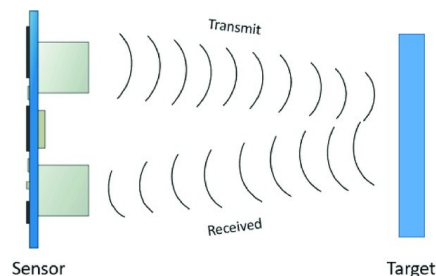
3. Nu moeten we meten hoe lang het hoge signaal op de echo pin duurde, omdat de duur van het signaal op de echo pin gerelateerd is aan de afstand. Om dit te doen zullen we de functie `utime.ticks_us()` gebruiken, die meet hoeveel tijd er verstreken is in µs sinds het programma werd gestart. Eerst maken we een while-lus die wordt uitgevoerd zolang het signaal laag is. Daarbinnen plaatsen we de functie `tick_us()`. Op deze manier krijgen we informatie over wanneer het signaal voor het laatst laag was.

```
smart_waste_bin.py
1 import machine
2 import utime
3
4 trigger = machine.Pin(18, machine.Pin.OUT)
5 echo = machine.Pin(19, machine.Pin.IN)
6
7 while True:
8     trigger.value(0)
9     utime.sleep_us(2)
10    trigger.value(1)
11    utime.sleep_us(10)
12    trigger.value(0)
13
14    while echo.value() == 0:
15        signal_off = utime.ticks_us()
```

4. Op dezelfde manier zullen we meten wanneer het signaal voor het laatst hoog was op de echo-pen:

```
smart_waste_bin.py
1 import machine
2 import utime
3
4 trigger = machine.Pin(18, machine.Pin.OUT)
5 echo = machine.Pin(19, machine.Pin.IN)
6
7 while True:
8     trigger.value(0)
9     utime.sleep_us(2)
10    trigger.value(1)
11    utime.sleep_us(10)
12    trigger.value(0)
13
14    while echo.value()==0:
15        signal_off = utime.ticks_us()
16
17    while echo.value()==1:
18        signal_on = utime.ticks_us()
```

Het verschil tussen de tijd van het laatste hoog en laag signaal is de duur van de hoge puls op de echo pin. Hoe vertalen we de pulsduur in afstand? Kijk naar de onderstaande figuur, die het idee weergeeft van het meten van afstand met een ultrasone afstandssensor.



Bron: www.researchgate.net/figure/werking-principes_fig5_344385811

<https://A-block-diagram-of-Ultrasonic-sensor->

Aan het begin wordt een geluidsgolf uitgezonden die door het object weerkaatst en terugkeert naar de sensor. Daarom legt de golf in de gemeten tijd t tweemaal de afstand tussen de sensor en het object af en beweegt met een snelheid van ongeveer 340 m/s (de geluidssnelheid in lucht). Daarom kunnen we de vergelijking voor de snelheid schrijven:

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

$$d = v \cdot \frac{t}{2} = 0,034 \frac{\text{cm}}{\mu\text{s}} \cdot \frac{t}{2} = 58$$

Daarom moet de verkregen pulsduur worden gedeeld door 58 om de afstand in centimeters te krijgen. Om een waarde weer te geven in de Thonny editor terminal, moet je de printfunctie gebruiken. De str-functie converteert een variabele met drijvende komma naar een tekenreeks, die nodig is om de waarde in de terminal weer te geven. Met het plusteken kun je je eigen tekst combineren met variabelen:

```
smart_waste_bin.py
1 import machine
2 import utime
3
4 trigger = machine.Pin(18, machine.Pin.OUT)
5 echo = machine.Pin(19, machine.Pin.IN)
6
7 while True:
8     trigger.value(0)
9     utime.sleep_us(2)
10    trigger.value(1)
11    utime.sleep_us(10)
12    trigger.value(0)
13
14    while echo.value()==0:
15        signal_off = utime.ticks_us()
16
17    while echo.value()==1:
18        signal_on = utime.ticks_us()
19
20    diff = signal_on - signal_off
21    distance = diff/58.0
22    print("d="+str(distance))
```

```
Shell
d=10.86207
d=11.10345
d=11.34483
d=11.31034
d=11.31034
```

Nu kunnen de leerlingen alle elementen gebruiken die nodig zijn om een slimme afvalbak te bouwen. Laten we verder gaan met het project.

Smart waste bin project:

Bij het maken van een slim prullenbakproject beginnen we met het aanpassen van het programma van opwarmactiviteit 2. Om dit te doen voegen we drie LED-diodes toe. Om dit te doen voegen we drie LED diodes toe:

```
smart_waste_bin.py *
1 import machine
2 import utime
3
4 trigger = machine.Pin(18, machine.Pin.OUT)
5 echo = machine.Pin(19, machine.Pin.IN)
6
7 led_green = machine.Pin(13, machine.Pin.OUT)
8 led_yellow = machine.Pin(12, machine.Pin.OUT)
9 led_red = machine.Pin(11, machine.Pin.OUT)
10
11 while True:
12     trigger.value(0)
13     utime.sleep_us(2)
14     trigger.value(1)
15     utime.sleep_us(10)
16     trigger.value(0)
```

Laten we vervolgens een variabele diepte maken die de diepte van de prullenbak weergeeft. Laten we de waarde tijdelijk instellen op 100 cm en deze in een latere stap aanpassen aan de werkelijke diepte van de prullenbak die we gaan maken:

```
smart_waste_bin.py
1 import machine
2 import utime
3
4 trigger = machine.Pin(18, machine.Pin.OUT)
5 echo = machine.Pin(19, machine.Pin.IN)
6
7 led_green = machine.Pin(13, machine.Pin.OUT)
8 led_yellow = machine.Pin(12, machine.Pin.OUT)
9 led_red = machine.Pin(11, machine.Pin.OUT)
10
11 depth = 100
12
13 while True:
14     trigger.value(0)
15     utime.sleep_us(2)
16     trigger.value(1)
17     utime.sleep_us(10)
18     trigger.value(0)
```

Afhankelijk van hoe vol de bak is, laten we een andere LED oplichten. Als hij minder dan 50% vol is, gaat de groene diode branden, als hij meer dan 50% vol is maar minder dan 80%, gaat de gele diode branden en als hij meer dan 80% vol is, gaat de rode diode branden. Onthoud dat 0% vol staat voor de diepte van de bak, d.w.z. in dit voorbeeld 100 cm, en 50% vol zal zijn wanneer we de afstand van 50 cm aflezen door de sensor, omdat de sensor helemaal bovenaan in het deksel van de bak wordt geplaatst.

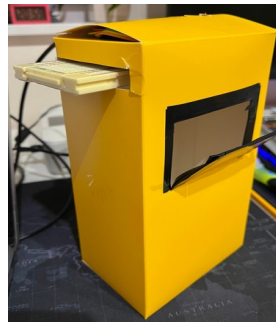


Bron: https://www.researchgate.net/figure/Ultrasonic-fill-level-sensor_fig3_304711436

```
13 while True:
14     trigger.value(0)
15     utime.sleep_us(2)
16     trigger.value(1)
17     utime.sleep_us(10)
18     trigger.value(0)
19
20     while echo.value()==0:
21         signal_off = utime.ticks_us()
22
23     while echo.value()==1:
24         signal_on = utime.ticks_us()
25
26     diff = signal_on - signal_off
27     distance = diff/58.0
28     print("d="+str(distance))
29
30     if distance>=0.5*depth:
31         led_red.value(0)
32         led_yellow.value(0)
33         led_green.value(1)
34     elif (distance<0.5*depth) and (distance>0.2*depth):
35         led_red.value(0)
36         led_yellow.value(1)
37         led_green.value(0)
38     else:
39         led_red.value(1)
40         led_yellow.value(0)
41         led_green.value(0)
42     utime.sleep(0.1)
```


2.1.3 Crafting

Nu is het tijd om je eigen slimme afvalcontainer te maken. Gebruik hiervoor een willekeurige verzend-/schoenendoos, enz. Plaats een breadboard met een ultrasone sensor in het bovenste deksel. Vergeet niet om de sensor naar beneden te richten. Knip aan de voorkant een gat voor het afval. Je kunt de diodes in de bovenkant van de bak plaatsen of bij het gat voor de prullenbak. Je kunt de powerbank aansluiten op de vuilnisbak of de voeding van de USB van je computer gebruiken. Op de foto's hieronder zie je een voorbeeldimplementatie. Onthoud dat je na het bouwen van de slimme afvalcontainer moet aflezen welke afstand (diepte) de sensor teruggeeft als de afvalcontainer leeg is, en de waarde in de dieptevariabele moet corrigeren. Anders de slimme afvalcontainer de verkeerde vulling weergeven.



2.2 Niveau 2

In niveau 2 voegen we het verzenden van gegevens over het vullen van de afvalcontainer en over de locatie van de afvalcontainer naar de cloud toe. We zullen de locatie handmatig invoeren in een variabele in het programma, ervan uitgaande dat de afvalcontainer altijd bij een bepaald huis/blok hoort. Hieronder volgt een stapsgewijze beschrijving van het programma.

2.2.1 Programmeren

Er zijn een paar verschillende clouds die je kunt gebruiken. Wij raden echter aan om Adafruit IO te gebruiken. Eerst moet je een gratis account aanmaken op <https://io.adafruit.com>. Vervolgens wil je twee gegevens naar de cloud sturen: het vullen van de vuilcontainer en de locatie. Om dit te doen, moet je twee feeds aanmaken. Feeds zijn objecten die gegevens opslaan. Om een feed te maken, ga je naar het tabblad "Feed" en selecteer je de knop "Nieuwe feed".



Vervolgens verschijnt er een venster waarin je de naam van de feed moet invoeren, bijvoorbeeld Vullen of Locatie.

Create a new Feed

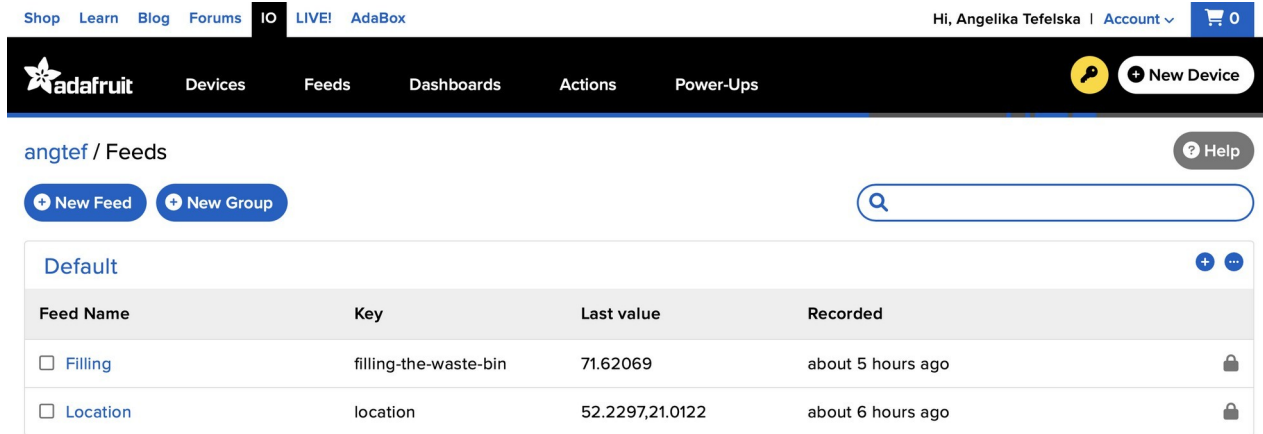
Name

Maximum length: 128 characters. Used: 0

Description

Cancel Create

Maak twee feeds aan. Zodra je dat hebt gedaan, zou je de feeds die je hebt gemaakt op je pagina moeten zien, zoals in de schermafbeelding hieronder:



The screenshot shows the Adafruit IO interface. The top navigation bar includes links for Shop, Learn, Blog, Forums, IO (selected), LIVE!, and AdaBox. The user is logged in as 'Hi, Angelika Tefelska' with an account menu and a shopping cart icon showing 0 items. The main navigation bar has links for Devices, Feeds, Dashboards, Actions, and Power-Ups. The 'Feeds' section is active, showing a 'Default' group with two feeds:

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

De volgende stap is het maken van een dashboard. Selecteer hiervoor het tabblad "Dashboards" en vervolgens "Nieuw dashboard":



The screenshot shows the Adafruit IO interface with the 'Dashboards' tab selected in the main navigation bar. The 'New Dashboard' button is highlighted with a red box. The breadcrumb trail shows 'angtef / Dashboards'.

Er verschijnt een venster waarin je de naam van het geselecteerde dashboard moet invoeren. Selecteer vervolgens aan de rechterkant het symbool Instellingen en kies "Nieuw blok maken".



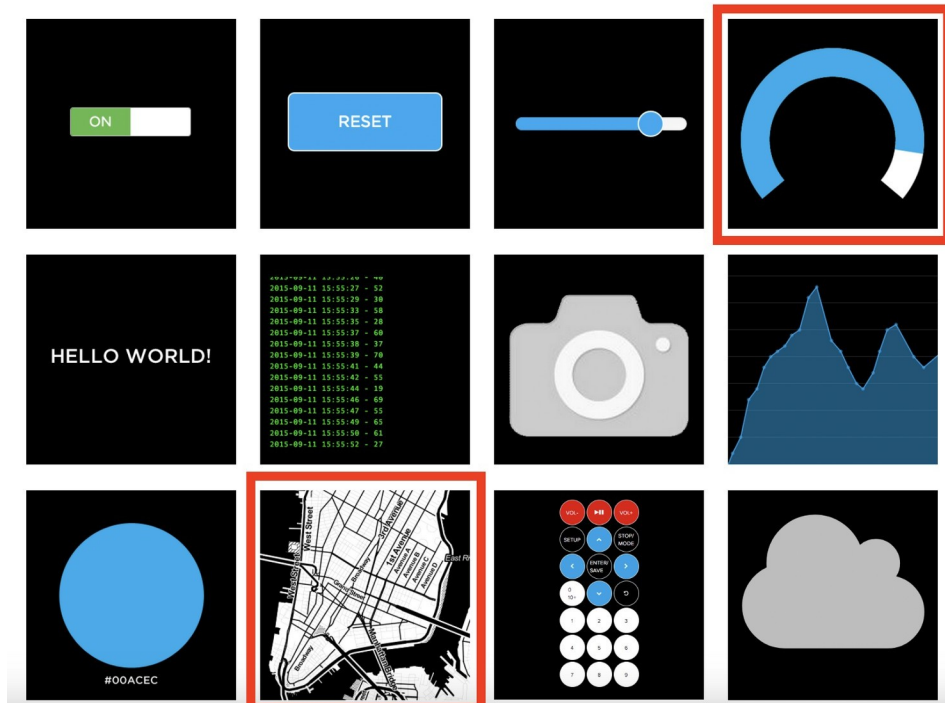
The screenshot shows the Adafruit IO interface with the 'Test' dashboard selected. The breadcrumb trail shows 'angtef / Dashboards / Test'. The settings icon (gear) is highlighted with a red box.

Adafruit IO heeft verschillende blokken beschikbaar om gegevens weer te geven (tekstvakken, meters, grafieken, kaarten, enz.). In ons geval kiezen we voor een meter en een kaart:

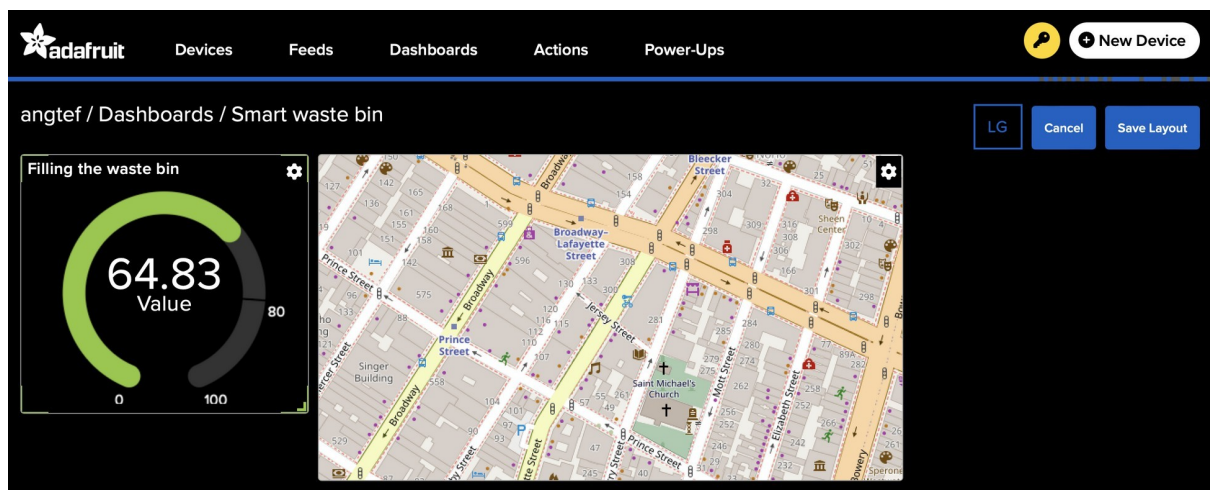
Create a new block



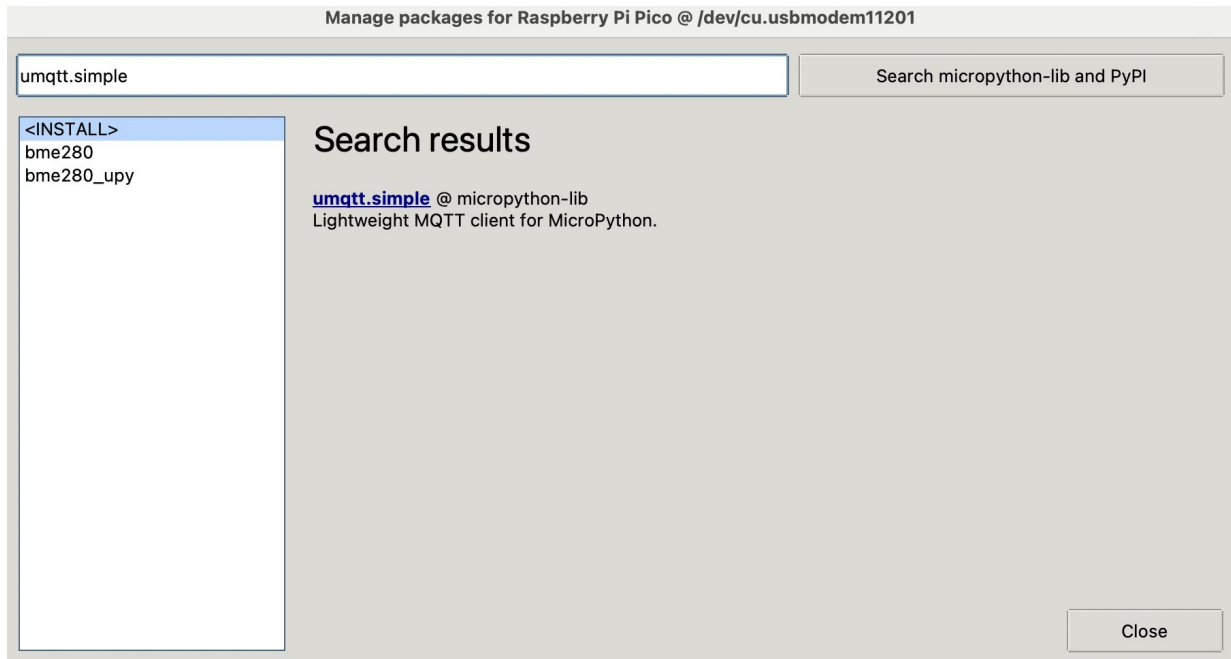
Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.



Vervolgens verschijnt er een venster waarin wordt gevraagd met welke feed we het blok willen verbinden. Selecteer de feed die de afvalbakvulling definieert naar de indicator en de feed die de locatie definieert naar de kaart. Selecteer vervolgens weer het instellingenpictogram en kies "Lay-out bewerken". Rangschik de blokken naar wens op het scherm. Je kunt de blokken ook vergroten en verkleinen. Een voorbeeld van het uiterlijk zie je in de schermafbeelding hieronder:



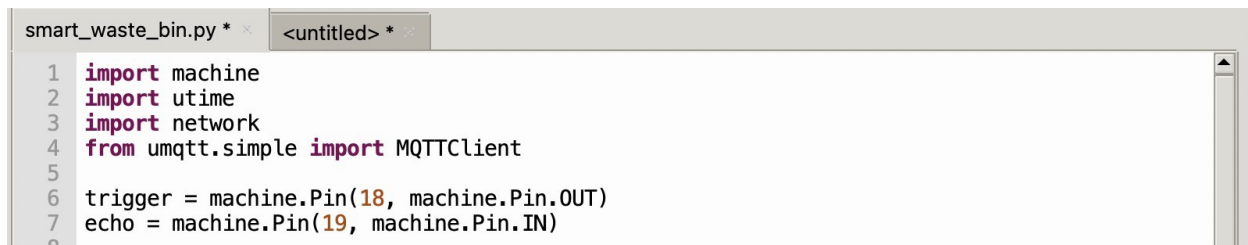
Laten we nu naar de Thonny editor gaan en de **umqtt.simple** bibliotheek installeren. Selecteer hiervoor "Tools" en vervolgens "Manage packages". Er verschijnt een venster waarin je de naam van de bibliotheek die je wilt installeren moet invoeren, in dit geval **umqtt.simple**:



Als je op de naam van de gevonden umqtt.simple bibliotheek klikt, wordt een venster met bibliotheekgegevens geopend en kun je de bibliotheek installeren door op de knop Installeren te klikken.

Nu kunnen we teruggaan naar onze code en de stukken toevoegen die nodig zijn om gegevens naar de cloud te sturen. Volg deze stappen om dit te doen:

1. Voeg de benodigde bibliotheken toe:



2. Voeg een stukje code toe waarmee je verbinding kunt maken met je WIFI. Hier moet je de regels 15-18 invullen. Voer eerst de naam van je WIFI in en daarna het wachtwoord.

```
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:
```

De laatste twee parameters zijn de gegevens van Adafruit IO. Ga terug naar de Adafruit website en klik op het sleutelsymbool. Wanneer je dit doet, verschijnen je login en sleutel. Kopieer deze gegevens naar het programma op regel 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.

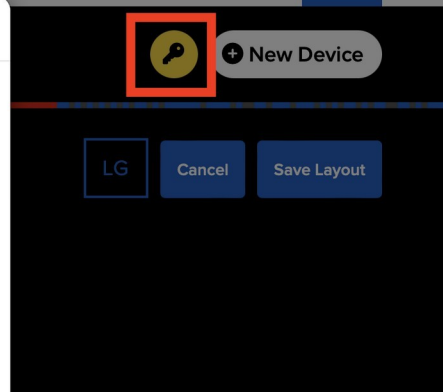
If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username

Active Key

REGENERATE KEY





3. Nu gebruiken we MQTT (Message Queuing Telemetry Transport), een lichtgewicht communicatieprotocol gebaseerd op het publish/subscribe model. Het is speciaal ontworpen voor het verzenden van gegevens in omgevingen met beperkte middelen, zoals IoT-apparaten (Internet of Things). Meer details in "Technische handleiding over Raspberry Pi Pico en microPython". Gebruik hiervoor de onderstaande code, waarbij je alleen de feednamen in regel 33-34 verandert. Het voorbeeld gebruikt feeds met de namen: "Vullen" en "Locatie". Vervang ze door je eigen feeds. Vergeet niet om /csv te laten staan in het geval van Locatie, want als je kaarten gebruikt, moet je gegevens in csv-formaat aanleveren.

```
smart_waste_bin.py * 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. Laten we nu functies toevoegen om gegevens naar de cloud te sturen:

```
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
```

De laatste stap is het doorgeven van de gemeten bezetting en locatie aan de functie:

```
68 if distance>=0.5*depth:
69     led_red.value(0)
70     led_yellow.value(0)
71     led_green.value(1)
72 elif (distance<0.5*depth) and (distance>0.2*depth):
73     led_red.value(0)
74     led_yellow.value(1)
75     led_green.value(0)
76 else:
77     led_red.value(1)
78     led_yellow.value(0)
79     led_green.value(0)
80
81 location = "0, 52.2297,21.0122,0" #value, latitude, longitude, altitude
82 fill_level = ((depth - distance)/depth)*100
83 send_data(fill_level, location)
84
85 utime.sleep(10)
```

Het programma is klaar. Start het en ga naar het aangemaakte dashboard op de Adafruit IO-pagina.

De hele code ziet er als volgt uit:

```
smart_waste_bin.py x 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 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"
```

```
smart_waste_bin.py * backup.py
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()
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
51 while True:
52     trigger.value(0)
53     utime.sleep_us(2)
54     trigger.value(1)
55     utime.sleep_us(10)
56     trigger.value(0)
57
58     while echo.value()==0:
59         signal_off = utime.ticks_us()
60
61     while echo.value()==1:
62         signal_on = utime.ticks_us()
63
64     diff = signal_on - signal_off
65     distance = diff/58.0
```



```

64     diff = signal_on - signal_off
65     distance = diff/58.0
66     print("d="+str(distance))
67
68     if distance>=0.5*depth:
69         led_red.value(0)
70         led_yellow.value(0)
71         led_green.value(1)
72     elif (distance<0.5*depth) and (distance>0.2*depth):
73         led_red.value(0)
74         led_yellow.value(1)
75         led_green.value(0)
76     else:
77         led_red.value(1)
78         led_yellow.value(0)
79         led_green.value(0)
80
81     location = "0, 52.2297,21.0122,0" #value, latitude, longitude, altitude
82     fill_level = ((depth - distance)/depth)*100
83     send_data(fill_level, location)
84
85     utime.sleep(10)
86

```

3 Tips en aanbevelingen

Het project kan in twee richtingen worden uitgebreid:

1. Door GPS toe te voegen om de werkelijke locatie van slimme vuilniscontainers te meten. Een voorbeeld van het lezen van gegevens van GPS is te vinden in de "Technische handleiding over Raspberry Pi Pico en microPython".
2. Door gegevens uit de cloud te lezen en code te schrijven, bijvoorbeeld in Python, om de optimale route voor de vuilniswagen te bepalen. Voorbeeldcodes die hiervoor kunnen worden gebruikt zijn hier:
<https://colab.research.google.com/drive/1aOq9jRh6c6fhaw1ahe0a1-yKVdMnO613?usp=sharing/>