

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



Co-funded by
the European Union

IoT4Schools

**“Φέρνοντας το Διαδίκτυο των Πραγμάτων στη σχολική
εκπαίδευση ως εργαλείο για την αντιμετώπιση των
προκλήσεων του 21^{ου} αιώνα”**

**Έξυπνοι κάδοι απορριμμάτων: πώς να βελτιώσουμε τη
διαχείριση απορριμμάτων στις έξυπνες πόλεις;**

Οδηγός για τον Εκπαιδευτικό

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

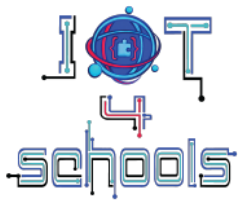
Organization: Warsaw University of Technology, Faculty of Physics

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



Co-funded by
the European Union

Η υποστήριξη της Ευρωπαϊκής Επιτροπής για την παραγωγή αυτής της δημοσίευσης δεν συνιστά έγκριση του περιεχομένου, το οποίο αντικατοπτρίζει μόνο τις απόψεις των συγγραφέων και η Επιτροπή δεν μπορεί να θεωρηθεί υπεύθυνη για οποιαδήποτε χρήση των πληροφοριών που περιέχονται σε αυτήν.



Πίνακας Περιεχομένων

1 Εισαγωγή στο έργο	3
1.1 Σενάριο και Σκοπός του έργου	3
1.2 Μαθησιακοί στόχοι	4
1.3 Διαδρομή μάθησης – Στάδια υλοποίησης	4
1.4 Προαπαιτούμενες γνώσεις	5
1.5 Υλικό και λογισμικό	5
1.6 Χρονοπρογραμματισμός	6
2 Υλοποίηση του έργου	7
2.1 Επίπεδο 1	7
2.1.1 Διαδικασία κατασκευής κυκλώματος	7
2.1.2 Προγραμματισμός	7
2.1.3 Κατασκευή (χειροτεχνία)	16
2.2 Επίπεδο 2	16
2.2.1 Προγραμματισμός	16
3 Συμβουλές και συστάσεις	23

1 Εισαγωγή στο έργο

1.1 Σενάριο και σκοπός του έργου

Στόχος αυτού του έργου είναι να εισαγάγει τους μαθητές στην τεχνολογία Διαδικτύου των Πραγμάτων (IoT) στο πλαίσιο των έξυπνων πόλεων. Εδώ και αρκετά χρόνια, έχει αναπτυχθεί η έννοια των έξυπνων πόλεων, στην οποία οι λύσεις IoT χρησιμοποιούνται και θα χρησιμοποιηθούν για τη βελτίωση των συνθηκών διαβίωσης και την ελαχιστοποίηση των αρνητικών επιπτώσεων των πόλεων στο περιβάλλον. Ένα από τα σημαντικά προβλήματα των σύγχρονων πόλεων είναι η αποκομιδή των απορριμμάτων. Επί του παρόντος, οι περισσότερες πόλεις συλλέγουν τα απορρίμματα με καθορισμένο χρονοδιάγραμμα, περνώντας από όλα τα κτίρια ανεξάρτητα από το επίπεδο πλήρωσης των κάδων. Το ίδιο συμβαίνει και με τους κάδους που βρίσκονται κατά μήκος των πεζοδρομίων και χρησιμοποιούνται από τους περαστικούς για να πετάξουν τα σκουπίδια. Ένα τέτοιο σύστημα είναι αναποτελεσματικό και προκαλεί σημαντική κατανάλωση καυσίμου. Το έργο στοχεύει να δείξει πώς να δημιουργήσετε έναν έξυπνο κάδο απορριμμάτων χρησιμοποιώντας μια πλακέτα Raspberry Pi Pico και έναν αισθητήρα απόστασης υπερήχων για τη μέτρηση του επιπέδου πλήρωσης του κάδου. Το επίπεδο πλήρωσης του κάδου θα αποσταλεί στο cloud για να βοηθήσει στη βελτιστοποίηση της διαδρομής του απορριμματοφόρου και έτσι να ελαχιστοποιήσει το αποτύπωμα άνθρακα. Τέτοιες λύσεις ήδη εισάγονται σιγά σιγά σε ορισμένες πόλεις, π.χ. στην Αμβέρσα (δείτε φωτογραφίες παρακάτω).



Το έργο στοχεύει στην ανάπτυξη των ψηφιακών ικανοτήτων των μαθητών. Η τεχνολογία IoT συνδυάζει πολλούς τομείς: ηλεκτρονικά (αισθητήρες μέτρησης), προγραμματισμό (στη συγκεκριμένη περίπτωση, γλώσσα προγραμματισμού MicroPython - μια έκδοση της Python για τον προγραμματισμό μικροελεγκτών), δικτύωση (με χρήση σύννεφων/clouds) και ανάλυση δεδομένων. Κατά τη διάρκεια του έργου, οι μαθητές θα αναπτύξουν τις ψηφιακές και τεχνολογικές ικανότητες που αναφέρονται παραπάνω.



Ως μέρος του έργου, οι μαθητές θα μάθουν για το πρόβλημα της διαχείρισης των απορριμμάτων στην πόλη, ένα εξαιρετικά σημαντικό ζήτημα για την ελαχιστοποίηση των αρνητικών επιπτώσεων των πόλεων στο περιβάλλον με τη

μείωση του αποτυπώματος άνθρακα (λιγότερη κατανάλωση καυσίμου από τα απορριμματοφόρα). Επιπλέον, θα συζητηθεί το θέμα της ανακύκλωσης.



Το έργο θα πραγματοποιηθεί σε ομάδες από μαθητές από διαφορετικά υπόβαθρα και χώρες. Μέσω κοινής συνεργασίας, ελπίζουμε ότι οι μαθητές θα ενσωματωθούν παρά τις πολιτισμικές διαφορές ή τα γλωσσικά εμπόδια.

1.2 Μαθησιακοί στόχοι

Μέσα από αυτό το έργο οι μαθητές θα είναι σε θέση να:

- Δημιουργήσουν ένα απλό πρόγραμμα στη γλώσσα προγραμματισμού MicroPython.
- Κατασκευάσουν ένα ηλεκτρονικό κύκλωμα χρησιμοποιώντας την πλακέτα Raspberry Pi Pico.
- Κατασκευάσουν και προγραμματίσουν μια συσκευή που μπορεί να μετρήσει το επίπεδο πλήρωσης των κάδων απορριμμάτων.
- Μάθουν πώς να χρησιμοποιούν αισθητήρες όπως τον αισθητήρα υπερήχων.
- Κατανοήσουν πώς οι συσκευές IoT μπορούν να συλλέγουν και να μεταδίδουν δεδομένα στο cloud.
- Κατανοήσουν και εξηγήσουν πώς τα δεδομένα μπορούν να παρακολουθούνται σε πραγματικό χρόνο.
- Υποδεικνύουν λύσεις που στοχεύουν σε πιο αποτελεσματική και οικολογική διαχείριση των απορριμμάτων στις πόλεις.
- Αναφέρουν τα πλεονεκτήματα της ανακύκλωσης και τις τρέχουσες δυσκολίες στην επεξεργασία των απορριμμάτων.

1.3 Διαδρομή μάθησης – Στάδια υλοποίησης

Ως μέρος του έργου, οι μαθητές θα δημιουργήσουν λύσεις για τη βελτίωση της διαχείρισης των απορριμμάτων στις πόλεις προκειμένου να ελαχιστοποιήσουν το αποτύπωμα άνθρακα. Για να γίνει αυτό, θα χρησιμοποιήσουν μια πλακέτα Raspberry Pi Pico και έναν αισθητήρα απόστασης υπερήχων.

Ακολουθούν μερικά προτεινόμενα στάδια για την ομαλή και αποτελεσματική εφαρμογή του έργου έξυπνου κάδου απορριμμάτων με τους μαθητές σας:

1. **Σχηματισμός ομάδας:** Χωρίστε τους μαθητές σας σε ομάδες των δύο ή τριών.

2. **Καταιγισμός ιδεών:** Ενθαρρύνετε κάθε ομάδα να αναζητήσει πληροφορίες σχετικά με πιθανούς τρόπους διαχείρισης των απορριμμάτων στην πόλη για να γίνει πιο έξυπνη (μέθοδος συλλογής απορριμμάτων, βελτιστοποίηση διαδρομής, πλεονεκτήματα και μειονεκτήματα διαφορετικών λύσεων, στατιστικά στοιχεία ανακύκλωσης, κατανάλωση καυσίμου απορριμματοφόρων κ.λπ.).
3. **Συζήτηση και ανάθεση της δραστηριότητας:** Ενθαρρύνετε κάθε ομάδα να μοιραστεί τα ευρήματά της και τις ιδέες της σχετικά με τον τρόπο βελτίωσης του συστήματος διαχείρισης απορριμμάτων για να γίνει πιο έξυπνο. Μετά τη συζήτηση, εισαγάγετε τον συγκεκριμένο στόχο του έργου. (Σημείωση: Συνιστάται ο συγκεκριμένος στόχος του έργου να εισαχθεί μετά τον καταιγισμό ιδεών, προκειμένου να ενθαρρύνετε τους μαθητές σας να εξετάσουν το έργο του έξυπνου κάδο απορριμμάτων σε ένα ευρύτερο πλαίσιο).
4. **Σχεδιασμός:** Ενθαρρύνετε κάθε ομάδα να σκεφτεί πώς θα κατασκευάσει τη συσκευή (πώς θα γίνει ο κάδος, πού θα τοποθετηθεί ο αισθητήρας απόστασης, ποια δεδομένα πρέπει να μεταδίδονται στο cloud, πότε πρέπει να αποστέλλονται δεδομένα και πόσο συχνά).
5. **Δημιουργία:** Χρησιμοποιώντας τα φύλλα εργασίας των μαθητών, ενθαρρύνετε κάθε ομάδα να δημιουργήσει τον δικό της έξυπνο κάδο απορριμμάτων. Ανάλογα με τις δεξιότητες των μαθητών, μπορεί να θέλετε να εξετάσετε την κατανομή ρόλων.
6. **Δοκιμή - βελτιστοποίηση:** Μετά την ολοκλήρωση του έργου, ενθαρρύνετε τους μαθητές σας να δοκιμάσουν τον έξυπνο κάδο απορριμμάτων τους. Με βάση τα αποτελέσματα των δοκιμών, μπορείτε να ενθαρρύνετε κάθε ομάδα να βελτιστοποιήσει το έργο της. Εάν το έργο δεν ήταν πολύ δύσκολο για τους μαθητές, σκεφτείτε να κάνετε επεκτάσεις, όπου αναπτύσσετε έναν αλγόριθμο για τη βελτιστοποίηση της διαδρομής (το πρόβλημα του ταξιδιώτη πωλητή).
7. **Παρουσίαση – κοινή χρήση:** Ενθαρρύνετε τους μαθητές σας να παρουσιάσουν τα έργα τους στην ολομέλεια και ζητήστε τους να αναστοχαστούν την όλη εμπειρία. Ενθαρρύνετε όλες τις ομάδες να εξετάσουν τον αντίκτυπο τέτοιων έξυπνων κάδων απορριμμάτων στη λειτουργία των πόλεων και στο περιβάλλον.

1.4 Προαπαιτούμενα μάθησης

Οι μαθητές θα πρέπει να είναι εξοικειωμένοι με τις βασικές έννοιες του προγραμματισμού σε οποιαδήποτε γλώσσα προγραμματισμού. Ακόμη και αν δεν έχουν ανάλογη προηγούμενη εμπειρία, είναι δυνατό να ολοκληρώσουν το έργο καθώς τα υλικά είναι προετοιμασμένα με τέτοιο τρόπο ώστε όλοι να μπορούν να ολοκληρώσουν το έργο.

1.5 Υλικό και λογισμικό

Υλικό:

- Πλακέτα Raspberry Pi Pico W
- Breadboard (contact plate)
- Καλώδια σύνδεσης Θηλυκό-Αρσενικό και Αρσενικό - Αρσενικό
- Κόκκινοι, Κίτρινοι και Πράσινοι δίοδοι LED
- Αντιστάσεις 330 Ohm

- Εξωτερική πηγή τροφοδοσίας: 2 θήκες μπαταρίας AAA ή ένα power bank εξόδου χαμηλής τάσης (μέχρι 5V) - προαιρετικά
- Αισθητήρας απόστασης υπερήχων HC-SR04

Λογισμικό:

- Thonny editor
- Adafruit IO cloud

1.6 Χρονοπρογραμματισμός

Υπολογίζεται ότι θα χρειαστείτε 4 με 6 ώρες για να ολοκληρώσετε το έργο.

Συγκεκριμένα, εκτιμάται ότι θα χρειαστείτε:

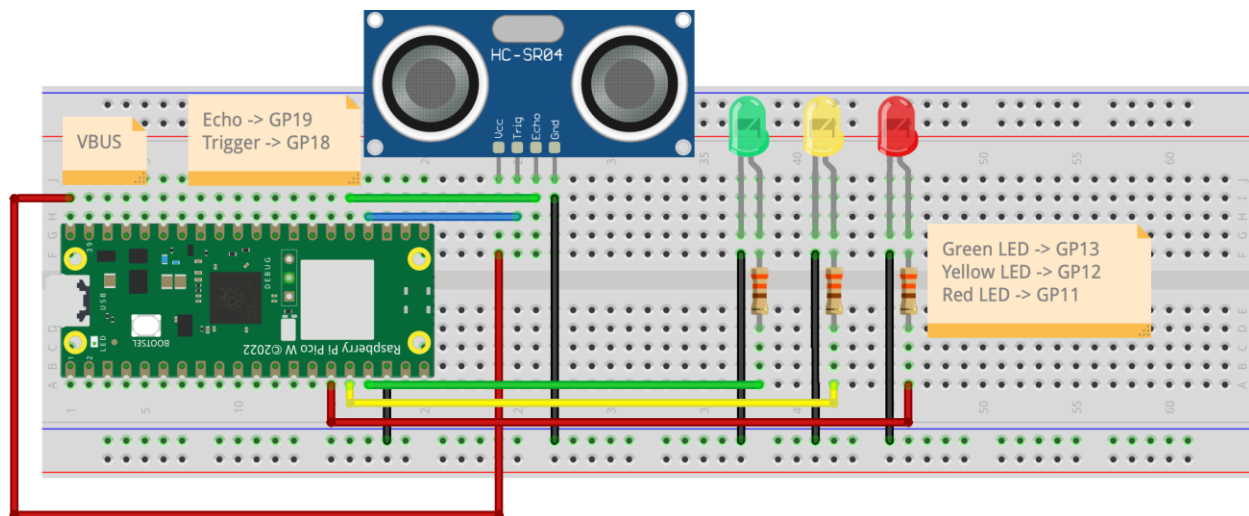
- 45-90 λεπτά για την εισαγωγή του έργου (συμπεριλαμβανομένου του καταιγισμού ιδεών και της συζήτησης), του σχεδιασμού και της δραστηριότητας προθέρμανσης
- 45 λεπτά για να ολοκληρώσετε το Επίπεδο 1
- 45 λεπτά για το Επίπεδο 2
- 45-90 λεπτά για παρατάσεις
- 30 λεπτά για σύνοψη και συζήτηση

2 Υλοποίηση του έργου

2.1 Επίπεδο 1

2.1.1 Διαδικασία δημιουργία κυκλώματος

Το Raspberry Pi Pico W θα πρέπει να συνδεθεί με έναν αισθητήρα απόστασης υπερήχων και τρεις λυχνίες LED μέσω αντιστάσεων 330Ω. Ένα παράδειγμα σύνδεσης φαίνεται στο παρακάτω σχήμα. Αρχικά, μπορείτε να δοκιμάσετε το σύστημα που είναι συνδεδεμένο στον υπολογιστή μέσω USB. Τελικά, ωστόσο, θα χρειαστείτε ένα power bank ή μπαταρίες για να τροφοδοτήσετε το σύστημα. Για οικολογικούς και οικονομικούς λόγους, προτείνουμε ένα power bank, το οποίο μπορεί να φορτιστεί μετά τα μαθήματα και να χρησιμοποιηθεί ξανά.

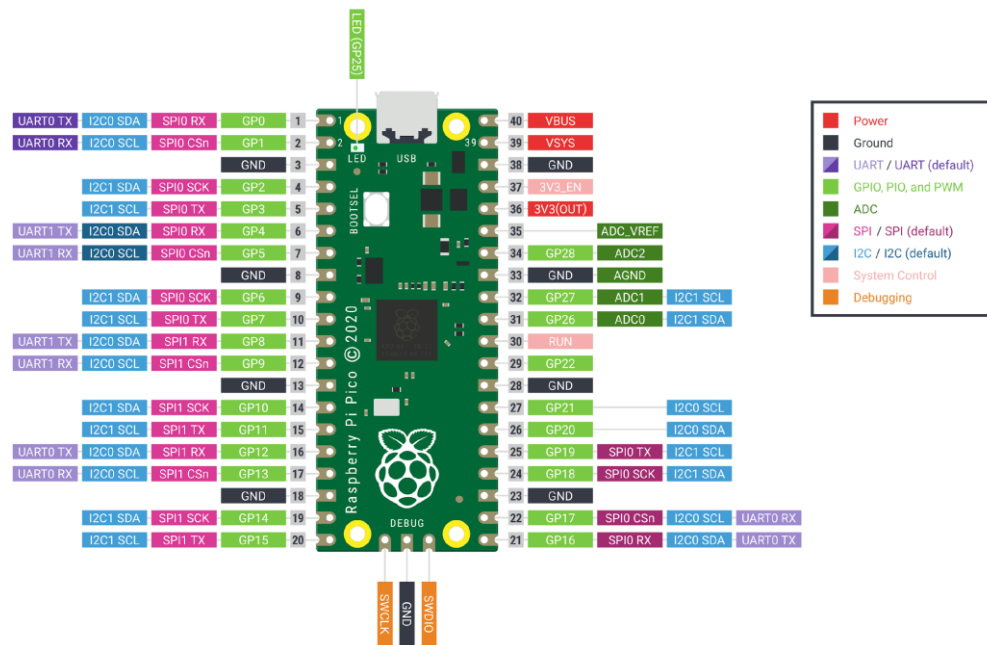


fritzing

Σημειώστε ότι σε ορισμένα breadboard οι ακίδες κάτω από τη μπλε γραμμή που χρησιμοποιήσαμε για το GND δεν συνδέονται σε όλο το μήκος της πλακέτας. Εάν υπάρχει κενό στη μπλε λωρίδα, σημαίνει ότι μόνο ένα μέρος της πλακέτας έχει τις ακίδες συνδεδεμένες. Δώστε προσοχή σε αυτό όταν συνδέετε την πλακέτα.

2.1.2 Προγραμματισμός

Σε αυτό το έργο, θα χρησιμοποιήσουμε ακροδέκτες εισόδου/εξόδου (GPIO) γενικής χρήσης, οι οποίοι χρησιμοποιούνται για να δημιουργήσουν ή να διαβάσουν ψηφιακά, δηλαδή σήματα που έχουν μόνο δύο πιθανές τιμές: 0 ή 1. Το GPIO είναι ακροδέκτες που μπορούν να χρησιμοποιηθούν και οι δύο ως ακίδες εισόδου, από τις οποίες μπορούμε να διαβάσουμε πληροφορίες, π.χ. σχετικά με την ενεργοποίηση ενός κουμπιού, ή ως ακίδες εξόδου, όπου μπορούμε να δημιουργήσουμε ψηφιακά σήματα, π.χ. να αναβοσβήνει ένα LED. Το παρακάτω σχήμα δείχνει τις 40 ακίδες από τον πίνακα Raspberry Pi Pico, οι οποίες είναι αριθμημένες από πάνω αριστερά. Οι ακίδες που χρησιμεύουν ως GPIO έχουν συντομογραφία GPx (πράσινα ορθογώνια), όπου x είναι ο τακτικός αριθμός της ακίδας GPIO, ξεκινώντας από το 0 έως το 28. Περισσότερες πληροφορίες στον οδηγό: "Τεχνικός οδηγός για το Raspberry Pico και MicroPython" διαθέσιμος στη διεύθυνση: <https://www.iot.fizyka.pw.edu.pl/results/>.

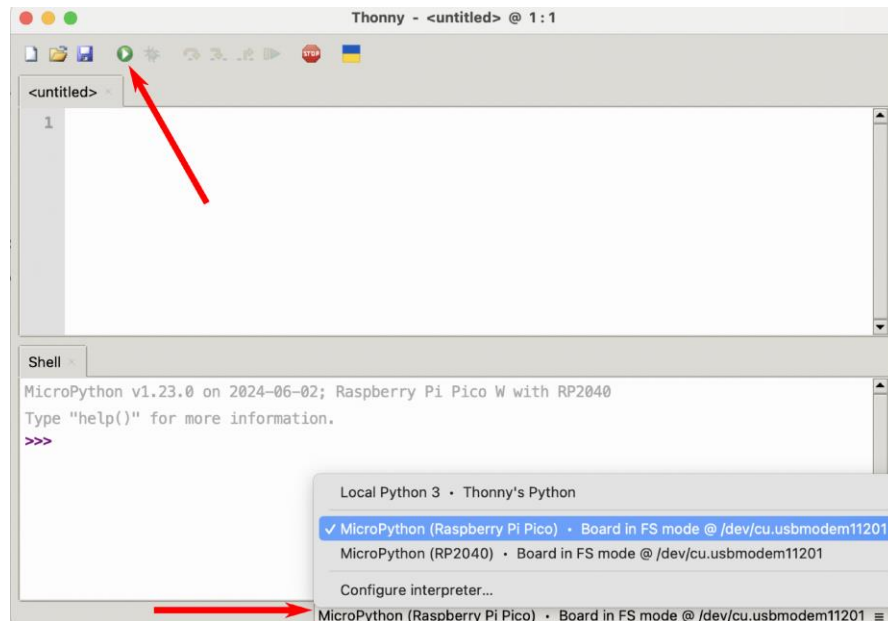


Πηγή: <https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html>.

Προκειμένου οι μαθητές να κάνουν όσο το δυνατόν περισσότερα μόνοι τους, προτείνουμε να κάνουν πρώτα δύο ασκήσεις προθέρμανσης.

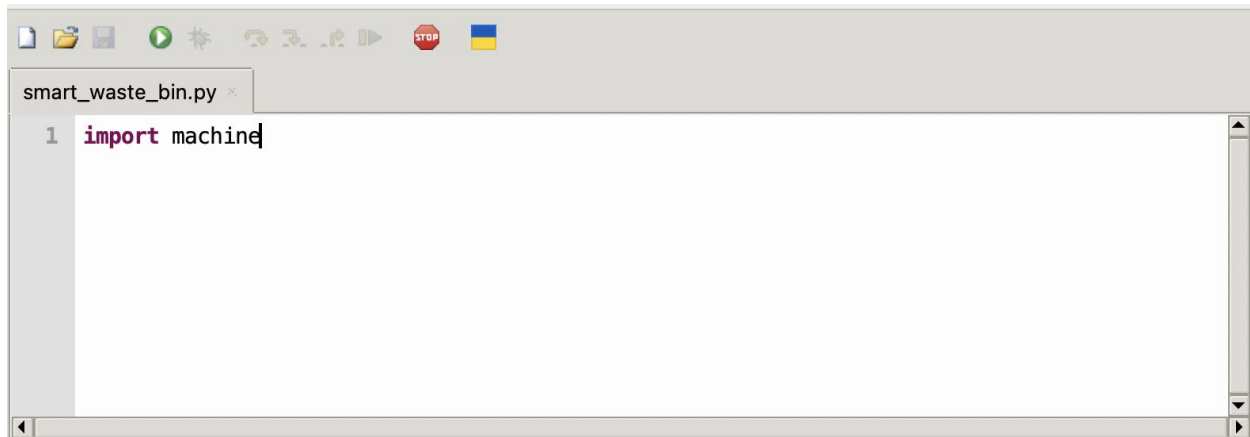
Δραστηριότητα προθέρμανσης 1:

Η πρώτη άσκηση προθέρμανσης είναι η δημιουργία ενός προγράμματος που θα προσομοιώνει τα φανάρια ελέγχου κυκλοφορίας. Για να το κάνετε αυτό, ανοίξτε το πρόγραμμα επεξεργασίας Thonny και στη συνέχεια επιλέξτε "MicroPython (Raspberry Pi Pico)" όπως φαίνεται στην εικόνα. Μετά τη σωστή σύνδεση στην πλακέτα, το πράσινο κουμπί Run θα πρέπει να είναι ενεργό (όχι γκριζαρισμένο). Εάν είναι γκριζαρισμένο, επιλέξτε ξανά "MicroPython (Raspberry Pi Pico)".



Τώρα ας γράψουμε ένα πρόγραμμα για φανάρια ελέγχου κυκλοφορίας. Για να το κάνουμε αυτό, ακολουθούμε τα εξής βήματα:

1. Ας ξεκινήσουμε συμπεριλαμβάνοντας τη βιβλιοθήκη στο MicroPython:



Η βιβλιοθήκη *machine* περιέχει όλες τις απαραίτητες οδηγίες για την επικοινωνία με το Raspberry Pi Pico. Η εντολή `import` καθιστά την καθορισμένη βιβλιοθήκη διαθέσιμη στο έργο.

2. Στη συνέχεια, πρέπει να διαμορφώσουμε τις ακίδες GP13, GP12 και GP11 ώστε να λειτουργούν ως ακίδες εξόδου, επειδή με αυτές ελέγχουμε τις λυχνίες LED στέλνοντας μια τιμή 1 ή 0 στις ακίδες GPIO. Για το σκοπό αυτό χρησιμοποιείται η συνάρτηση `Pin` από τη βιβλιοθήκη *machine*. Για να καλέσουμε συναρτήσεις από τη βιβλιοθήκη στη γλώσσα MicroPython, πρώτα δίνουμε το όνομα της βιβλιοθήκης, ακολουθεί μια τελεία και μετά το όνομα της συνάρτησης από αυτήν τη βιβλιοθήκη, π.χ `machine.Pin()`. Η συνάρτηση `Pin` παίρνει δύο ορίσματα. Το πρώτο είναι ο αριθμός `pin` GPIO. Το δεύτερο είναι η δήλωση για το εάν η ακίδα GPIO θα λειτουργεί ως είσοδος (`machine.Pin.IN`) ή έξοδος (`machine.Pin.OUT`). Ως εκ τούτου, σε αυτήν την περίπτωση η κλήση συνάρτησης θα μοιάζει με `machine.Pin(13,`

machine.Pin.OUT). Το αντικείμενο που επιστρέφεται από τη συνάρτηση Pin εκχωρείται στη μεταβλητή led_green που δημιουργήθηκε.

```
smart_waste_bin.py ×
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. Στο επόμενο βήμα τοποθετούμε έναν βρόχο, ο οποίος θα περιέχει το κύριο μέρος του προγράμματος:

```
smart_waste_bin.py ×
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. Στο επόμενο βήμα ανάβουμε το κόκκινο LED στέλνοντας την τιμή 1 στον ακροδέκτη GP11. Η συνάρτηση value() χρησιμοποιείται για να ορίσει την τιμή στην ακίδα, η οποία παίρνει την τιμή 0 ή 1 ως όρισμα.

```
smart_waste_bin.py * ×
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. Τώρα το πρόγραμμα πρέπει να περιμένει 1 δευτερόλεπτο για να δούμε το αποτέλεσμα του φωτισμού της διόδου. Για το σκοπό αυτό θα χρησιμοποιήσουμε τη βιβλιοθήκη *utime*, η οποία περιέχει συναρτήσεις για καθυστερήσεις. Αρχικά, πρέπει να προσθέσουμε τη βιβλιοθήκη:

```
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     |
```

6. Η συνάρτηση *sleep* από τη βιβλιοθήκη *utime* μας επιτρέπει να καθυστερήσουμε το πρόγραμμα για έναν συγκεκριμένο αριθμό δευτερολέπτων. Ας ορίσουμε μια καθυστέρηση 1 δευτερολέπτου αφότου ανάψει η διόδος:

```
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. ώρα ας προσθέσουμε τον υπόλοιπο κώδικα ώστε να ανάψουν τα υπόλοιπα LED με τη σωστή σειρά.

```
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
```

Χάρη σε αυτή τη δραστηριότητα, οι μαθητές θα μάθουν πώς να χρησιμοποιούν LED, τα οποία θα χρησιμοποιήσουμε για να δημιουργήσουμε έναν έξυπνο κάδο απορριμμάτων.

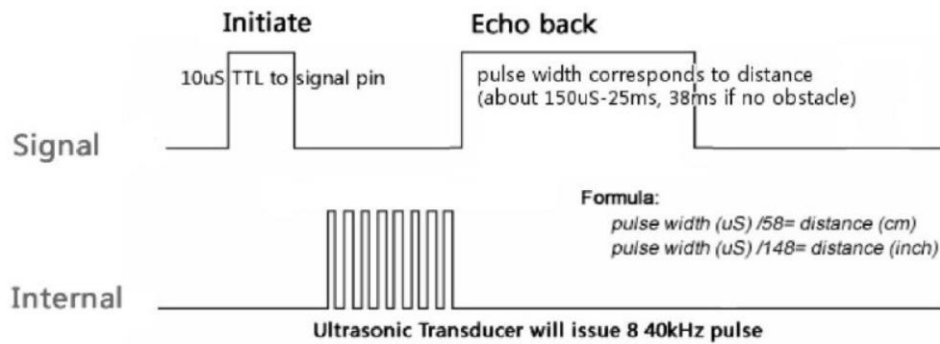
Δραστηριότητα προθέρμανσης 2:

Τώρα ας δημιουργήσουμε ένα πρόγραμμα που θα διαβάσει την απόσταση χρησιμοποιώντας έναν αισθητήρα απόστασης υπερήχων. Για να το κάνουμε αυτό, ακολουθούμε τα εξής βήματα:

1. Αρχικά, ας προσθέσουμε τις απαραίτητες βιβλιοθήκες, π.χ. machine και utime, και ας διαμορφώσουμε τις παραμέτρους των ακίδων: GP18 (trigger) ως έξοδο και GP19 (ηχώ) ως είσοδο.

```
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. Στη συνέχεια, στον κύριο βρόχο, ας προσθέσουμε ένα τμήμα κώδικα για να μετρήσουμε την απόσταση από τον αισθητήρα απόστασης υπερήχων. Σύμφωνα με την τεκμηρίωση για τον αισθητήρα HC-SR04 (βλέπε την εικόνα παρακάτω), ρυθμίζουμε πρώτα το χαμηλό σήμα στον ακροδέκτη trigger για μικρό χρονικό διάστημα, π.χ. 2μs. Στη συνέχεια, ρυθμίζουμε το υψηλό σήμα για 10μs. Για να δημιουργήσουμε καθυστερήσεις σε μικροδευτερόλεπτα, χρησιμοποιούμε τη συνάρτηση: utime.sleep_us(). Στο επόμενο βήμα, ρυθμίζουμε το χαμηλό σήμα στον ακροδέκτη trigger.



```
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)
```

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

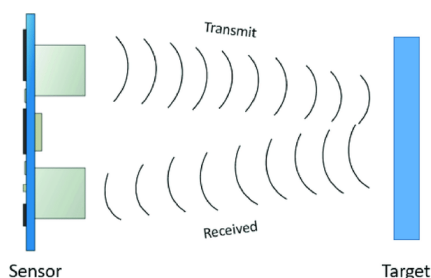
3. Τώρα πρέπει να μετρήσουμε πόσο διήρκεσε το υψηλό σήμα στον ακροδέκτη ηχούς, επειδή η διάρκεια του σήματος στον ακροδέκτη ηχούς σχετίζεται με την απόσταση. Για να το κάνουμε αυτό, θα χρησιμοποιήσουμε τη συνάρτηση `utime.ticks_us()`, η οποία μετρά πόσος χρόνος, σε µs, έχει περάσει από την έναρξη του προγράμματος. Αρχικά, θα δημιουργήσουμε έναν βρόχο `while` που θα εκτελείται όσο το σήμα είναι χαμηλό. Μέσα, θα τοποθετήσουμε τη συνάρτηση `tick_us()`. Με αυτόν τον τρόπο, θα λάβουμε πληροφορίες για το πότε το σήμα ήταν χαμηλό για τελευταία φορά.

```
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. Ομοίως, θα μετρήσουμε πότε το σήμα ήταν τελευταία φορά υψηλό στον ακροδέκτη ηχούς:

```
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()
```

Η διαφορά μεταξύ του χρόνου της τελευταίας εμφάνισης του υψηλού και του χαμηλού σήματος είναι η διάρκεια του υψηλού παλμού στον ακροδέκτη ηχούς. Πώς, όμως, μεταφράζουμε τη διάρκεια του παλμού σε απόσταση; Ας δούμε το παρακάτω σχήμα, που παρουσιάζει την ιδέα της μέτρησης της απόστασης με έναν αισθητήρα απόστασης υπερήχων.



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

Στην αρχή εκπέμπεται ένα ηχητικό κύμα, το οποίο ανακλάται από το αντικείμενο και επιστρέφει στον αισθητήρα. Επομένως, στον μετρούμενο χρόνο t , το κύμα διανύει τη διπλάσια απόσταση μεταξύ του αισθητήρα και του αντικειμένου και κινείται με ταχύτητα περίπου 340 m/s (ταχύτητα του ήχου στον αέρα). Επομένως, μπορούμε να γράψουμε την εξίσωση για την ταχύτητα:

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

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

Ως εκ τούτου, η λαμβανόμενη διάρκεια παλμού θα πρέπει να διαιρεθεί με το 58 για να ληφθεί η απόσταση σε εκατοστά. Για να εμφανίσουμε μια τιμή στο τερματικό του Thonny editor, πρέπει να χρησιμοποιήσουμε τη συνάρτηση *print*. Η συνάρτηση *str* μετατρέπει μια μεταβλητή κινητής υποδιαστολής σε συμβολοσειρά, η οποία είναι απαραίτητη για την εμφάνιση της τιμής στο τερματικό. Το σύμβολο *+* μας επιτρέπει να συνδυάσουμε δικό μας κείμενο με μεταβλητές:

```
smart_waste_bin.py x
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 x
d=10.86207
d=11.10345
d=11.34483
d=11.31034
d=11.31034
```

Στο σημείο αυτό οι μαθητές μπορούν να χρησιμοποιήσουν όλα τα απαραίτητα στοιχεία για την κατασκευή ενός έξυπνου κάδου απορριμμάτων. Ας περάσουμε, λοιπόν, στο έργο.

Έργο έξυπνου κάδου απορριμμάτων:

Κατά τη δημιουργία ενός έργου έξυπνου κάδου απορριμμάτων, θα ξεκινήσουμε τροποποιώντας το πρόγραμμα από τη δραστηριότητα προθέρμανσης 2. Για να γίνει αυτό, θα προσθέσουμε τρεις διόδους LED:

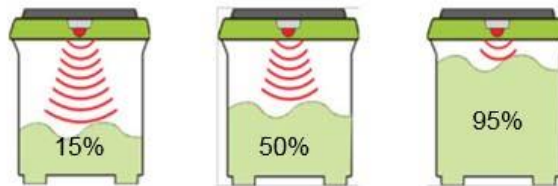
Στη συνέχεια, ας δημιουργήσουμε ένα μεταβλητό βάθος που θα αντιστοιχεί στο βάθος του κάδου απορριμμάτων. Ας ορίσουμε προσωρινά την τιμή στα 100 cm και σε επόμενο βήμα θα την

```
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)
```


προσαρμόσουμε στο πραγματικό βάθος του κάδου απορριμμάτων που θα δημιουργήσουμε:

```
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)
```

Τώρα, ανάλογα με το πόσο γεμάτος είναι ο κάδος, θα ανάψουμε ένα διαφορετικό LED. Όταν είναι γεμάτος κάτω από 50%, θα ανάψει η πράσινη δίοδος LED, όταν είναι γεμάτη πάνω από 50% αλλά λιγότερο από 80%, θα ανάψει η κίτρινη δίοδος και όταν είναι γεμάτη πάνω από 80%, θα ανάψει η κόκκινη δίοδος. Να θυμάστε ότι το 0% γεμάτο αντιστοιχεί στο βάθος του κάδου, δηλαδή σε αυτό το παράδειγμα 100 cm, και το 50% γεμάτο θα είναι όταν διαβάζουμε την απόσταση των 50 cm από τον αισθητήρα, επειδή ο αισθητήρας θα τοποθετηθεί στην κορυφή, στο κάλυμμα του κάδου.



Πηγή:

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 Κατασκευή (χειροτεχνία)

Τώρα ήρθε η ώρα να δημιουργήσουμε το δικό μας έξυπνο δοχείο απορριμμάτων. Για να το κάνουμε αυτό, θα χρησιμοποιήσουμε οποιοδήποτε κουτί αποστολής/παπουτσιών κ.λπ. Ας τοποθετήσουμε ένα breadboard με αισθητήρα υπερήχων στο επάνω κάλυμμα και να προσέξουμε να κατευθύνουμε τον αισθητήρα προς τα κάτω. Ας κάνουμε μια τρύπα για τα σκουπίδια στο μπροστινό μέρος. Μπορούμε να τοποθετήσουμε τις διόδους είτε στο πάνω μέρος του κάδου είτε δίπλα στην τρύπα των σκουπιδιών. Μπορούμε να συνδέσουμε το power bank στο δοχείο απορριμμάτων ή να χρησιμοποιήσουμε το τροφοδοτικό από το USB του υπολογιστή μας. Ένα παράδειγμα υλοποίησης φαίνεται στις παρακάτω φωτογραφίες. Μετά την κατασκευή του έξυπνου δοχείου απορριμμάτων, θα πρέπει να διαβάσουμε ποια απόσταση (βάθος) επιστρέφει ο αισθητήρας όταν το δοχείο απορριμμάτων είναι άδειο και να διορθώσουμε την τιμή στη μεταβλητή βάθους. Διαφορετικά, το έξυπνο δοχείο απορριμμάτων θα εμφανίσει λάθος γέμισμα.



2.2 Επίπεδο 2

Στο επίπεδο 2, θα προσθέσουμε στο σύννεφο (cloud) δεδομένα αποστολής σχετικά με το γέμισμα του δοχείου απορριμμάτων και τη θέση του. Θα εισαγάγουμε χειροκίνητα τη θέση σε μια μεταβλητή στο πρόγραμμα, υποθέτοντας ότι το δοχείο απορριμμάτων θα ανήκει πάντα σε ένα δεδομένο σπίτι/τετράγωνο. Παρακάτω ακολουθεί μια βήμα προς βήμα περιγραφή του προγράμματος.

2.2.1 Προγραμματισμός

Υπάρχουν διαφορετικά σύννεφα (clouds) που μπορούμε να χρησιμοποιήσουμε. Ωστόσο, συνιστούμε το Adafruit IO. Αρχικά, πρέπει να δημιουργήσουμε έναν δωρεάν λογαριασμό στη διεύθυνση <https://io.adafruit.com>. Στη συνέχεια, θέλουμε να στείλουμε δύο τμήματα δεδομένων: πλήρωση κάδου απορριμμάτων και τοποθεσία στο cloud. Για να το κάνουμε αυτό, πρέπει να δημιουργήσουμε δύο ροές (feeds). Οι ροές είναι αντικείμενα που αποθηκεύουν δεδομένα. Για να δημιουργήσουμε μια ροή, μεταβαίνουμε στην καρτέλα "Feeds" και επιλέγουμε το κουμπί "New Feed".



Στη συνέχεια θα εμφανιστεί ένα παράθυρο όπου πρέπει να εισάγουμε το όνομα της ροής, π.χ. Πλήρωση (Filling) ή Τοποθεσία (Place).

2.3

Create a new Feed

Name

Maximum length: 128 characters. Used: 0

Description

Cancel

Create

Ας δημιουργήσουμε δύο ροές δεδομένων. Μόλις το κάνουμε αυτό, θα δούμε τις ροές που δημιουργήσαμε, όπως στο στιγμιότυπο οθόνης παρακάτω:

Shop Learn Blog Forums IO LIVE! AdaBox

Hi, Angelika Tefelska | Account 0

adafruit

Devices Feeds Dashboards Actions Power-Ups

New Device

angtef / Feeds

Help

New Feed

New Group

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

Το επόμενο βήμα είναι να δημιουργήσουμε έναν πίνακα ελέγχου (Dashborad). Για να το κάνουμε αυτό, επιλέγουμε την καρτέλα "Dashboards" και, στη συνέχεια το κουμπί, "New Dashboard":



Θα εμφανιστεί ένα παράθυρο όπου πρέπει να εισάγουμε το όνομα του επιλεγμένου πίνακα εργαλείων (Dashboard). Στη συνέχεια, στη δεξιά πλευρά, επιλέγουμε το σύμβολο ρυθμίσεων και "Create New Block".

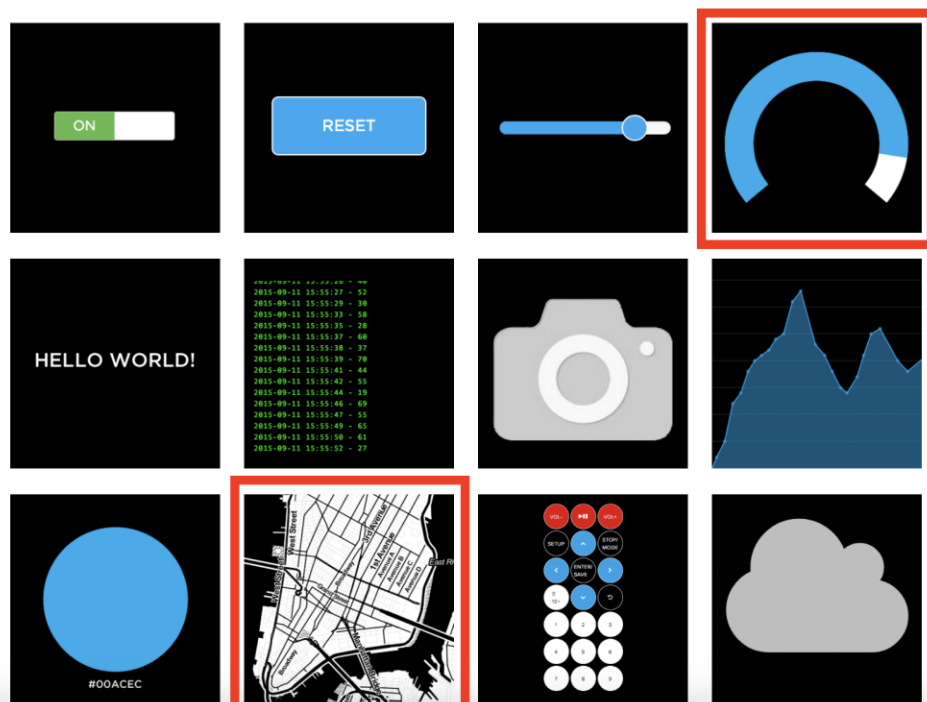


Το Adafruit IO διαθέτει διάφορα μπλοκ διαθέσιμα για την εμφάνιση δεδομένων (πλαίσια κειμένου, μετρητές, γραφήματα, χάρτες κ.λπ.). Στην περίπτωση μας, θα επιλέξουμε μετρητή και χάρτη:

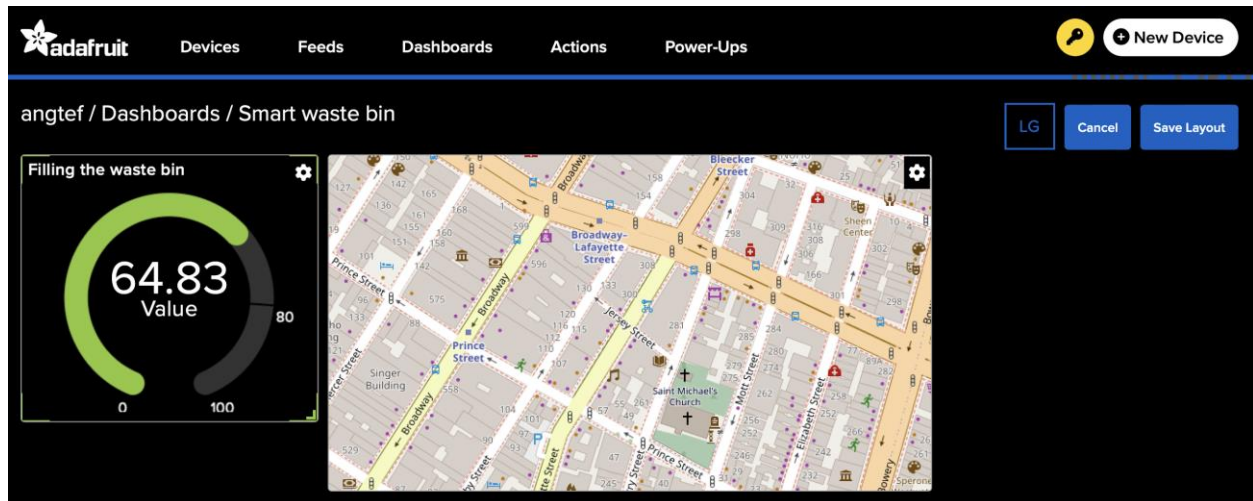
Create a new block



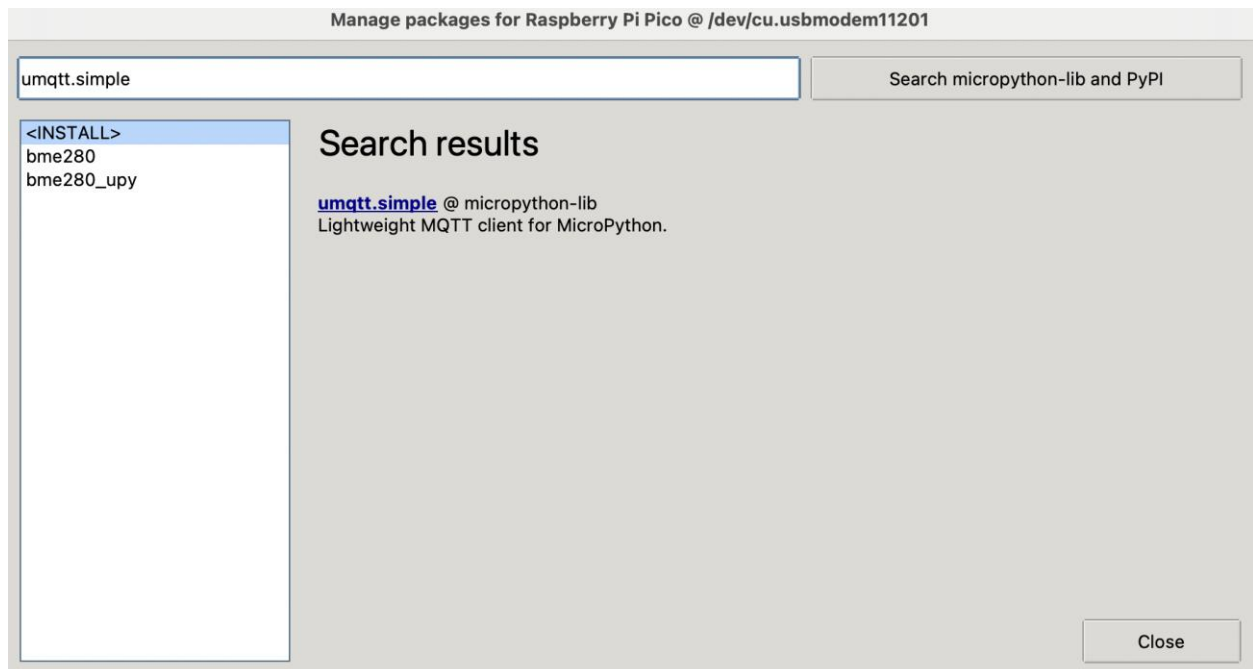
Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.



Στη συνέχεια, θα εμφανιστεί ένα παράθυρο που ρωτά σε ποια ροή θέλουμε να συνδέσουμε το μπλοκ. Επιλέγουμε την ροή που ορίζει την ένδειξη για το γέμισμα του κάδου απορριμμάτων και την τροφοδοσία που καθορίζει τη θέση του στον χάρτη. Στη συνέχεια, επιλέγουμε ξανά το εικονίδιο ρυθμίσεων και επιλέγουμε "Edit Layout". Τακτοποιούμε τα μπλοκ στην οθόνη όπως μας ταιριάζει. Μπορούμε, επίσης, να μεγεθύνουμε και να σμικρύνουμε τα μπλοκ. Ένα παράδειγμα εμφάνισης φαίνεται στο στιγμιότυπο οθόνης παρακάτω:



Τώρα ας πάμε στον επεξεργαστή Thonny και ας εγκαταστήσουμε τη βιβλιοθήκη **umqtt.simple**. Για το σκοπό αυτό, επιλέγουμε «Tools» και μετά «Manage packages». Θα εμφανιστεί ένα παράθυρο όπου θα πρέπει να εισάγουμε το όνομα της βιβλιοθήκης που θέλουμε να εγκαταστήσουμε, στην περίπτωση μας την **umqtt.simple**:



Όταν κάνουμε κλικ στο όνομα της βιβλιοθήκης `umqtt.simple` που βρέθηκε, θα ανοίξει ένα παράθυρο με δεδομένα βιβλιοθήκης και μπορούμε να εγκαταστήσουμε τη βιβλιοθήκη κάνοντας κλικ στο κουμπί Install.

Τώρα μπορούμε να επιστρέψουμε στον κώδικά μας και να προσθέσουμε τα κομμάτια που είναι απαραίτητα για την αποστολή δεδομένων στο cloud. Γι' αυτό, ακολουθούμε τα εξής βήματα:

1. Προσθέτουμε τις απαραίτητες βιβλιοθήκες:

```
smart_waste_bin.py * <untitled> *
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
```

2. Προσθέτουμε ένα κομμάτι κώδικα που θα μας επιτρέψει να συνδεθούμε στο WIFI μας. Εδώ πρέπει να συμπληρώσουμε τις γραμμές 15-18. Πρώτα, πληκτρολογούμε το όνομα του WIFI και μετά τον κωδικό πρόσβασης.

```
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:
```


Οι δύο τελευταίες παράμετροι είναι τα δεδομένα από το Adafruit IO. Επιστρέφουμε στον ιστότοπο Adafruit και κάνουμε κλικ στο σύμβολο του κλειδιού. Τότε, θα εμφανιστούν τα στοιχεία σύνδεσης και το κλειδί μας. Αντιγράφουμε αυτά τα δεδομένα στο πρόγραμμα στις γραμμές 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


angtef

Active Key

aio_lgpq55qYAvCVxMfiJgCW3rr5glWj

REGENERATE KEY





New Device

LG

Cancel

Save Layout

3. Τώρα θα χρησιμοποιήσουμε το MQTT (Message Queuing Telemetry Transport), ένα ελαφρύ πρωτόκολλο επικοινωνίας που βασίζεται στο μοντέλο δημοσίευσης/εγγραφής. Σχεδιάστηκε ειδικά για την αποστολή δεδομένων σε περιβάλλοντα με περιορισμένους πόρους, όπως συσκευές IoT (Internet of Things). Περισσότερες λεπτομέρειες υπάρχουν στον «Τεχνικό οδηγό για το Raspberry Pi Pico και τη microPython». Για το σκοπό αυτό, χρησιμοποιούμε τον παρακάτω κώδικα, αλλάζοντας μόνο τα ονόματα των ρών στις γραμμές 33-34. Το παράδειγμα χρησιμοποιεί ροές με ονόματα: "Filling" και "Location". Αντικαθιστούμε με τα δικά μας. Πρέπει να θυμηθούμε να αφήσουμε το /csv στην περίπτωση Location γιατί όταν χρησιμοποιούμε χάρτες, πρέπει να παρέχουμε δεδομένα σε μορφή csv.

```

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 = "raspberry_pi_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. Τώρα, ας προσθέσουμε συναρτήσεις για αποστολή δεδομένων στο cloud:


```

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

```

```

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)

```

Το τελευταίο βήμα είναι να μεταβιβάσουμε την πληρότητα που μετρήσαμε και την τοποθεσία στη συνάρτηση:

```

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 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"

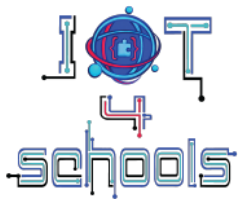
```

Το πρόγραμμα είναι έτοιμο. Το εκτελούμε και μεταβαίνουμε στον πίνακα εργαλείων (dashboard) που δημιουργήθηκε στη σελίδα του Adafruit IO.

Ολόκληρος ο κώδικας μοιάζει με το παρακάτω:

```
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 Συμβουλές και συστάσεις

Το έργο μπορεί να επεκταθεί προς δύο κατευθύνσεις:

1. Προσθέτοντας GPS για να μετρήσουμε την πραγματική θέση των έξυπνων δοχείων απορριμμάτων. Ένα παράδειγμα ανάγνωσης δεδομένων από το GPS υπάρχει στον «Τεχνικό οδηγό για το Raspberry Pi Pico και τη microPython».
2. Διαβάζοντας δεδομένα από το cloud και γράφοντας κώδικα, π.χ. στην Python, για να προσδιορίσουμε τη βέλτιστη διαδρομή για το απορριμματοφόρο. Παραδείγματα κώδικα που μπορούν να χρησιμοποιηθούν για το σκοπό αυτό υπάρχουν εδώ: <https://colab.research.google.com/drive/1aOq9jRh6c6fhaw1ahe0a1-yKVdMnO613?usp=sharing/>