Σκοπός του εργαστηρίου είναι να προγραμματίσουμε τον σεισμογράφο και να παρατηρήσουμε τα δεδομένα που καταγράφονται.

Βήμα 1: Απλός Σεισμογράφος

1. **Γράφουμε το πρόγραμμα. Θα βρείτε αναλυτικότερη επεξήγηση γραμμή-γραμμή στο τέλος του αρχείο, στο Παράρτημα.**

Αυτό το πρόγραμμα καταγράφει δεδομένα από τον **επιταχυνσιόμετρο** του Micro:bit για **60 δευτερόλεπτα** και τα αποθηκεύει σε ένα αρχείο καταγραφής. Τα δεδομένα περιλαμβάνουν τις συνιστώσες επιτάχυνσης x, y, z και το **μέγεθος της επιτάχυνσης** (magnitude), που υπολογίζεται από αυτές τις συνιστώσες.

Αναλυτικά, το πρόγραμμα:

I) **Ρυθμίζει την καταγραφή δεδομένων**:
   a. Ορίζει τις ετικέτες των δεδομένων (x, y, z, `magnitude`).
   b. Ορίζει τη διάρκεια καταγραφής (60 δευτερόλεπτα) και τη συχνότητα δειγματοληψίας (100 Hz, δηλαδή 1 δείγμα κάθε 10 ms).

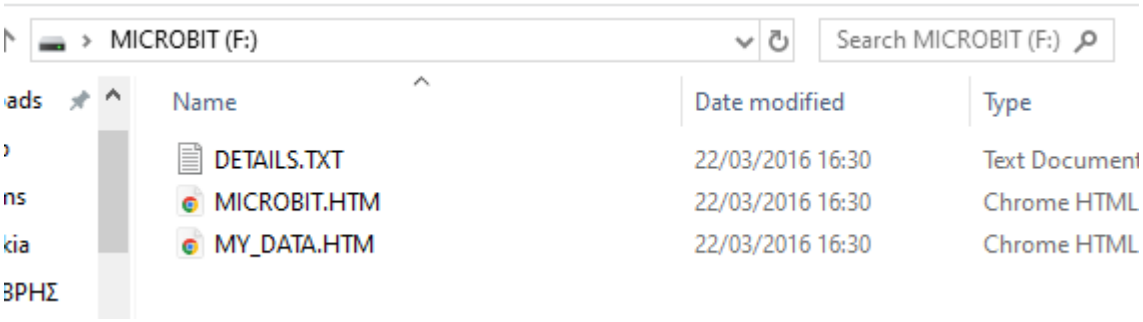
II) **Υπολογίζει το μέγεθος της επιτάχυνσης**:
   a. Χρησιμοποιεί τον τύπο $\text{magnitude} = \sqrt{x^2 + y^2 + z^2}$ για να βρει τη συνολική επιτάχυνση που ασκείται στη συσκευή, εκφρασμένη σε μονάδες g.

III) **Καταγράφει δεδομένα**:
   a. Διαβάζει τις τιμές x, y, z από τον επιταχυνσιόμετρο.
   b. Υπολογίζει το μέγεθος της επιτάχυνσης.
   c. Αποθηκεύει όλα αυτά τα δεδομένα σε ένα αρχείο καταγραφής με τη βοήθεια της βιβλιοθήκης `log`.

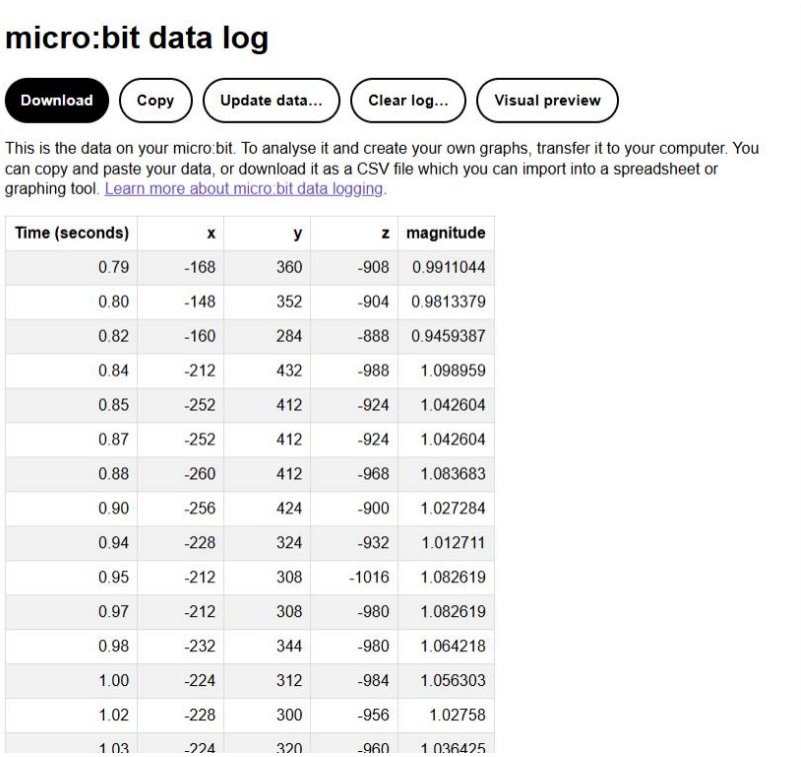
IV) **Επαναλαμβάνει τη διαδικασία**:
   a. Συνεχίζει να καταγράφει δεδομένα κάθε 10 ms για 60 δευτερόλεπτα.

V) **Ολοκληρώνει τη λειτουργία**:
   a. Όταν περάσουν τα 60 δευτερόλεπτα, εμφανίζει το μήνυμα "Done!" για να δείξει ότι η καταγραφή ολοκληρώθηκε.

2. **Κατεβάζουμε το πρόγραμμα στο Micro:bit, δεν βγάζουμε το καλώδιο (!) και προσπαθούμε να κρατήσουμε το Micro:bit σχετικά σταθερό, περιμένουμε μέχρι στη σειριακή να μας δείξει Done!**
3. **Βγάζουμε το καλώδιο από το Micro:bit και το ξαναβάζουμε! Θα δούμε στον φάκελο του Micro:bit που ανοίγει ότι υπάρχει ένα έξτρα αρχείο MY_DATA.HTM**



Σε αυτό το αρχείο είναι αποθηκευμένα τα δεδομένα που καταγράψαμε! Το πατάμε και θα ανοίξει στον browser σαν ιστοσελίδα.



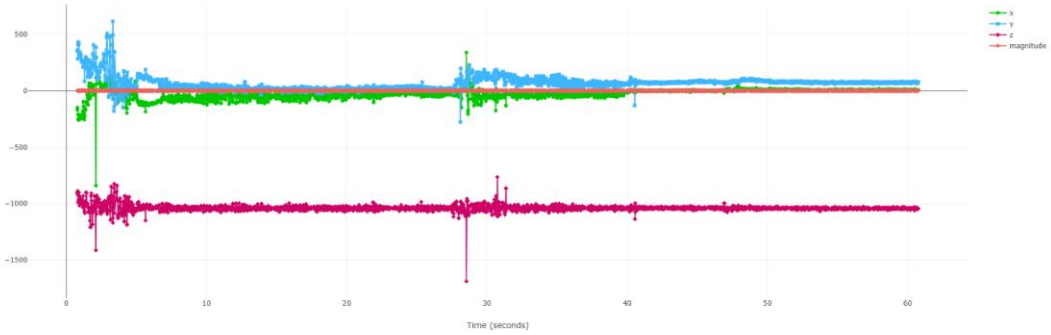
micro:bit data log

Download | Copy | Update data... | Clear log... | Visual preview

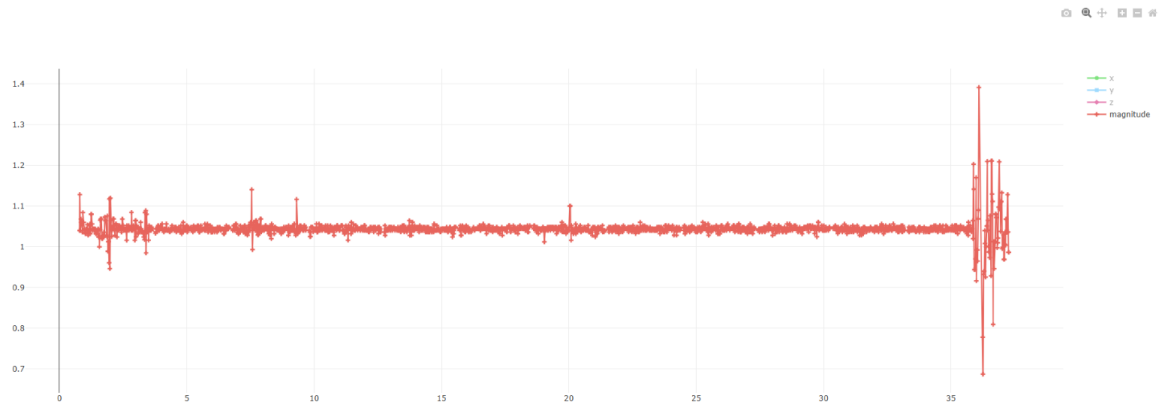
This is the data on your micro:bit. To analyse it and create your own graphs, transfer it to your computer. You can copy and paste your data, or download it as a CSV file which you can import into a spreadsheet or graphing tool. Learn more about micro:bit data logging.

Time (seconds)	x	y	z	magnitude
0.79	-168	360	-908	0.9911044
0.80	-148	352	-904	0.9813379
0.82	-160	284	-888	0.9459387
0.84	-212	432	-988	1.098959
0.85	-252	412	-924	1.042604
0.87	-252	412	-924	1.042604
0.88	-260	412	-968	1.083683
0.90	-256	424	-900	1.027284
0.94	-228	324	-932	1.012711
0.95	-212	308	-1016	1.082619
0.97	-212	308	-980	1.082619
0.98	-232	344	-980	1.064218
1.00	-224	312	-984	1.056303
1.02	-228	300	-956	1.02758
1.03	-224	320	-960	1.036425

4. **Αναλύουμε με τα παιδιά τα δεδομένα που καταγράψαμε! Μπορούμε να τα δούμε και σε μορφή διαγράμματος με το Visual Preview!**



5. **Μπορούμε να δούμε μόνο το Magnitude κάνοντας κλικ σε όλα τα υπόλοιπα labels εκτός από αυτό.**



Το Magnitude που μετράμε είναι ουσιαστικά η επιτάχυνση σε **g** (δυνάμεις βαρύτητας).

Όταν το Micro:bit είναι σταθερό:

- Ο άξονας **z** μετρά περίπου **1g** λόγω της βαρύτητας της Γης.
- Οι άξονες **x** και **y** είναι κοντά στο **0g**, εφόσον το Micro:bit είναι επίπεδο.

Σε σταθερή θέση, το μέγεθος της επιτάχυνσης θα είναι κοντά σε **1g.**

Θα παρατηρήσουμε ότι οι τιμές μας θα έχουν μεγαλύτερες αυξομειώσεις από το θεωρητικό αποτέλεσμα, καθώς το επιταχυνσιόμετρο είναι πολύ ευαίσθητο. Αυτό μπορούμε να το φτιάξουμε εφαρμόζοντας κάποιες τεχνικές στατιστικής.

Σημαντική Σημείωση! Αυτό που μετράει το πρόγραμμα **δεν είναι Ρίχτερ!** Η κλίμακα Ρίχτερ μετρά το **μέγεθος των σεισμών**, δηλαδή την ενέργεια που απελευθερώνεται από το επίκεντρο ενός σεισμού. Είναι λογαριθμική κλίμακα!

Βήμα 2: Βελτίωση των δεδομένων

1. **Θα δοκιμάσουμε μια τεχνική για την βελτίωση των δεδομένων μας, το averaging!**

Averaging

Η μέθοδος μέσου όρου μειώνει τον "θόρυβο" στα δεδομένα επιτάχυνσης, υπολογίζοντας έναν **κινούμενο μέσο όρο** των τιμών του μεγέθους. Αυτό βοηθά στην εξομάλυνση μικρών διακυμάνσεων και παρέχει πιο σταθερές μετρήσεις.

2. **Προσθέτουμε τις παρακάτω μεταβλητές στο πρόγραμμα:**

```
# Initialize variables for averaging
previous_magnitude = 0
alpha = 0.7  # Smoothing factor (0.0 to 1.0, higher = smoother)
```

3. **Προσθέτουμε τη παρακάτω συνάρτηση στο πρόγραμμα:**

```
def calculate_smoothed_magnitude(x, y, z):
    global previous_magnitude
    # Calculate magnitude
    magnitude = math.sqrt(x**2 + y**2 + z**2) / 1024  # Normalize to 'g'

    # Apply moving average (Exponential Smoothing)
    smoothed_magnitude = alpha * magnitude + (1 - alpha) * previous_magnitude
    previous_magnitude = smoothed_magnitude

    # Return smoothed magnitude (scaled to 0-9 for simplicity)
    return (smoothed_magnitude * 10) % 10
```

Προσέχουμε να τη καλέσουμε στη main αντί της απλής συνάρτησης!

```
'magnitude': calculate_smoothed_magnitude(x, y, z)
```

Τι κάνει η συνάρτηση:

1. **Υπολογισμός του μεγέθους:**

Υπολογίζεται το μέγεθος της επιτάχυνσης από τις συνιστώσες x, y, z

2. **Εφαρμογή εκθετικής εξομάλυνσης (Exponential Smoothing):**

Υπολογίζεται ο εξομαλυσμένος μέσος όρος:

```
smoothed_magnitude = alpha * magnitude + (1 - alpha) * previous_magnitude
```

- Το alpha καθορίζει πόσο γρήγορα το φίλτρο προσαρμόζεται σε νέες τιμές:

  Υψηλό alpha: δίνει μεγαλύτερη έμφαση στις νέες τιμές.

  Χαμηλό alpha: δίνει μεγαλύτερη έμφαση στις προηγούμενες τιμές.

3. **Ενημέρωση της προηγούμενης τιμής:**

Η τρέχουσα εξομαλυσμένη τιμή αποθηκεύεται για χρήση στον επόμενο υπολογισμό

```
previous_magnitude = smoothed_magnitude
```

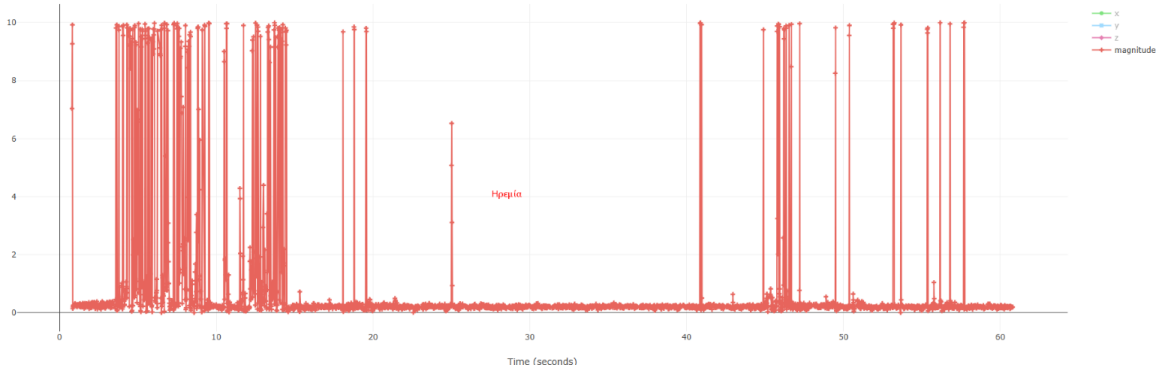
4. **Κλίμακα για απλότητα:**

Η εξομαλυσμένη τιμή επιστρέφεται σε μια κλίμακα από 0-9:

```
return (smoothed_magnitude * 10) % 10
```

5. **Κατεβάζουμε το πρόγραμμα στο Micro:bit, δεν βγάζουμε το καλώδιο (!) και προσπαθούμε να κρατήσουμε το Micro:bit σχετικά σταθερό, περιμένουμε μέχρι στη σειριακή να μας δείξει Done!**
6. **Βγάζουμε το καλώδιο από το Micro:bit και το ξαναβάζουμε! Ανοίγουμε ξανά το αρχείο MY_DATA.HTM**

Παρατηρούμε τα νέα μας δεδομένα και τα συγκρίνουμε με τα προηγούμενα.



Πλεονεκτήματα:

Μείωση θορύβου: Εξαλείφει μικρές διακυμάνσεις που μπορεί να προέρχονται από δονήσεις ή ανακρίβειες του αισθητήρα.

Σταθερές μετρήσεις: Παρέχει πιο ομαλά δεδομένα για ανάλυση ή εμφάνιση.

Επέκταση (Εάν μείνει χρόνος)

Μπορούμε να δοκιμάσουμε άλλη μια τεχνική, το threshold.

Η τεχνική **Thresholding** χρησιμοποιείται για να αγνοούνται μικρές, ασήμαντες μεταβολές στις μετρήσεις, παραβλέποντας τιμές που βρίσκονται μέσα σε ένα καθορισμένο εύρος (π.χ. ±0.1g). Έτσι, καταγράφονται μόνο οι σημαντικές αλλαγές.

1. **Προσθήκη μεταβλητής:**

```
threshold = 0.1  # Ρύθμιση ανάλογα με την ευαισθησία
```

- Το **threshold** καθορίζει το εύρος τιμών που θεωρούνται αμελητέες.
- Π.χ., αν το threshold είναι **0.1**, αγνοούνται αλλαγές μεταξύ **0.9g** και **1.1g**.

2. **Αλλαγή προγράμματος στη main:**

```
# Apply thresholding
 if abs(magnitude - 1.0) > threshold:
```

```
# Log data
log.add({
  'x': accelerometer.get_x(),
  'y': accelerometer.get_y(),
  'z': accelerometer.get_z(),
  'magnitude': calculate_magnitude(x, y, z)
})
```

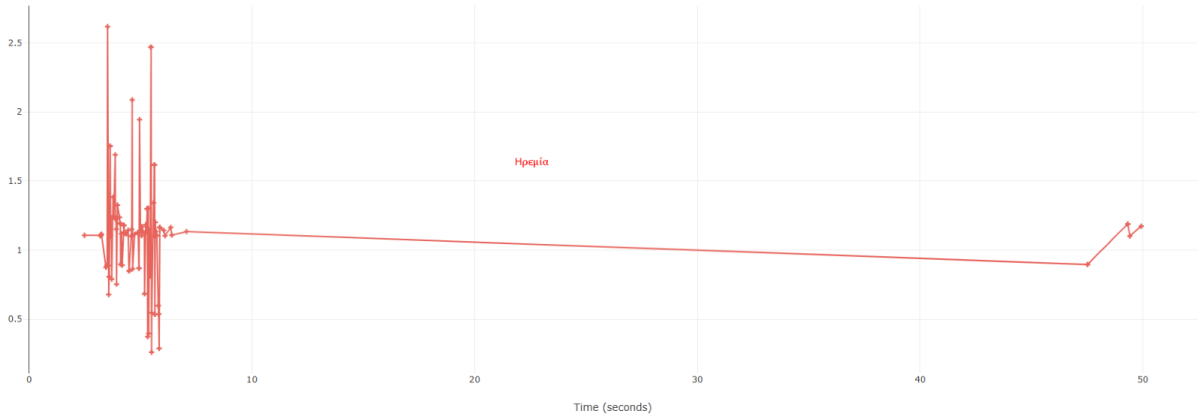
- Ελέγχεται αν η διαφορά μεταξύ του **`magnitude`** και της στατικής επιτάχυνσης της βαρύτητας (**1.0g**) είναι μεγαλύτερη από το κατώφλι.
- Αν η διαφορά είναι μικρότερη από το κατώφλι, αγνοείται η μέτρηση.
- **Καλούμε την απλή συνάρτηση για τον υπολογισμό του magnitude. Για να μπορέσουμε να δούμε τη διαφορά ανάμεσα στις δύο τεχνικές.**

Χαρακτηριστικά και Επιπτώσεις

1. **Ευαισθησία κατωφλίου**:
    a. **Μικρό κατώφλι** (π.χ., 0.05):
        i. Ανιχνεύει ακόμα και πολύ μικρές αλλαγές.
        ii. Κατάλληλο για λεπτομερείς μετρήσεις, αλλά μπορεί να αυξήσει τον "θόρυβο".
    b. **Μεγάλο κατώφλι** (π.χ., 0.2):
        i. Φιλτράρει περισσότερες μικρές αλλαγές.
        ii. Χρήσιμο για την ανίχνευση μόνο μεγάλων γεγονότων.
2. **Χρήσεις**:
    a. Φιλτράρισμα μικρών διακυμάνσεων που προκαλούνται από τυχαίες κινήσεις ή ανακρίβειες του αισθητήρα.
    b. Καταγραφή μόνο των σημαντικών γεγονότων, όπως δονήσεις ή κλίσεις.

3. **Κατεβάζουμε το πρόγραμμα στο Micro:bit, δεν βγάζουμε το καλώδιο (!) και προσπαθούμε να κρατήσουμε το Micro:bit σχετικά σταθερό, περιμένουμε μέχρι στη σειριακή να μας δείξει Done!**
4. **Βγάζουμε το καλώδιο από το Micro:bit και το ξαναβάζουμε! Ανοίγουμε ξανά το αρχείο MY_DATA.HTM**

Παρατηρούμε τα νέα μας δεδομένα και τα συγκρίνουμε με τα προηγούμενα.



Πώς μπορούν οι μαθητές να πειραματιστούν

1. **Ρύθμιση του alpha στο Averaging**:
   Δοκιμάστε διαφορετικά επίπεδα εξομάλυνσης για να δείτε πώς αντιδρά το σύστημα σε γρήγορες ή αργές αλλαγές.
2. **Τροποποίηση του threshold στο Thresholding**:
   Παρατηρήστε πώς η αύξηση ή η μείωση του κατωφλίου επηρεάζει την ευαισθησία στις δονήσεις.
3. **Συνδυασμός και των δύο προσεγγίσεων**:
   Εφαρμόστε ταυτόχρονα εξομάλυνση και κατώφλι για να δείτε πώς αλληλεπιδρούν μεταξύ τους.

Παράρτημα - Επεξήγηση Βασικού Προγράμματος

Εισαγωγές και βιβλιοθήκες:

```
from microbit import *
import math
```

- Εισάγει τη βιβλιοθήκη math, η οποία περιλαμβάνει μαθηματικές συναρτήσεις, όπως η sqrt (τετραγωνική ρίζα).

```
import time
import log
```

- Εισάγει τη βιβλιοθήκη log, που χρησιμοποιείται για την καταγραφή δεδομένων.

Ρύθμιση ετικετών για το αρχείο καταγραφής:

```
log.set_labels('x', 'y', 'z', 'magnitude')
```

- Ορίζει τις ετικέτες (labels) που θα χρησιμοποιηθούν για την καταγραφή δεδομένων. Εδώ οι ετικέτες είναι οι συνιστώσες x, y, z και το μέγεθος της επιτάχυνσης (magnitude).

Ρυθμίσεις προγράμματος:

```
# Configuration
SAMPLE_RATE = 10  # 10 ms delay = 100 Hz sampling
```

- Ορίζει τη συχνότητα δειγματοληψίας. Η καθυστέρηση είναι 10 ms, που αντιστοιχεί σε 100 δείγματα ανά δευτερόλεπτο.

```
LOG_DURATION = 60  # Log for 60 seconds
```

- Ορίζει τη διάρκεια καταγραφής δεδομένων στα 60 δευτερόλεπτα.

Συνάρτηση υπολογισμού μεγέθους επιτάχυνσης:

```
# Function to calculate magnitude of acceleration
def calculate_magnitude(x, y, z):
    return math.sqrt(x**2 + y**2 + z**2) / 1000  # Convert to g-force
```

- Υπολογίζει το μέγεθος της επιτάχυνσης με βάση τις συνιστώσες x, y, z. Η διαίρεση με το 1000 μετατρέπει τις μονάδες από milli-g σε g.

```
start_time = time.ticks_ms()
```

- Καταγράφει την αρχική χρονική στιγμή σε χιλιοστά του δευτερολέπτου.

Κύρια επαναληπτική δομή:

- Επαναλαμβάνεται όσο η διαφορά του τρέχοντος χρόνου από την αρχική χρονική στιγμή είναι μικρότερη από τη διάρκεια καταγραφής (60 δευτερόλεπτα).

```
while time.ticks_diff(time.ticks_ms(), start_time) < LOG_DURATION * 1000:
```

- Διαβάζει τις συνιστώσες x, y, z της επιτάχυνσης από τον επιταχυνσιόμετρο.

```
    x = accelerometer.get_x()
    y = accelerometer.get_y()
    z = accelerometer.get_z()
```

- Καταγράφει τις τιμές των x, y, z και το μέγεθος της επιτάχυνσης που υπολογίζεται από τη συνάρτηση calculate_magnitude.

```
# Log data
log.add({
  'x': accelerometer.get_x(),
  'y': accelerometer.get_y(),
  'z': accelerometer.get_z(),
  'magnitude': calculate_magnitude(x, y, z)
})
```

- Προσθέτει καθυστέρηση 10 ms πριν από την επόμενη επανάληψη, σύμφωνα με τη συχνότητα δειγματοληψίας.

```
sleep(SAMPLE_RATE)
```

- Εκτυπώνει μήνυμα "Done!" όταν ολοκληρωθεί η διάρκεια καταγραφής.

```
print("Done!")
```