Αυτό το πρόγραμμα δημιουργεί έναν αυτοσχέδιο σεισμογράφο με τη χρήση της Micro:bit και ενός servo, καταγράφοντας και αποτυπώνοντας την επιτάχυνση σε κίνηση.

### **Εισαγωγή βιβλιοθηκών**

**from** **microbit** **import** \*   
**import** **math**   
**import** **time**   
**import** **log**

* **from microbit import \***: Εισάγει τις λειτουργίες της Micro:bit, όπως το accelerometer, τις θύρες εισόδου/εξόδου (π.χ. pin0), κ.λπ.
* **import math**: Παρέχει μαθηματικές συναρτήσεις, όπως sqrt για την τετραγωνική ρίζα.
* **import time**: Χρησιμοποιείται για την παρακολούθηση του χρόνου.
* **import log**: Επιτρέπει την καταγραφή δεδομένων από τη Micro:bit.

### **Ετικέτες καταγραφής**

log.set\_labels('x', 'y', 'z', 'magnitude')

* Δημιουργεί ετικέτες για τα δεδομένα που θα καταγράφονται: οι άξονες x, y, z του επιταχυνσιόμετρου και το magnitude (μέτρο επιτάχυνσης).

### **Ρυθμίσεις Servo**

# =========== Servo ===========  
SERVO\_MIN\_PULSE = **25** # theoretical 50, measured 23  
SERVO\_MAX\_PULSE = **125** # theoretical 100, measured 127

* Ορίζουν τα ελάχιστα και μέγιστα παλμικά πλάτη (σε μικροδευτερόλεπτα) για τον έλεγχο της γωνίας του servo.

servoPin = pin0  
servoPin.set\_analog\_period(**20**)

* **servoPin**: Ορίζει την έξοδο του servo στη θύρα pin0.
* **set\_analog\_period(20)**: Ρυθμίζει την περίοδο του PWM σήματος σε 20 ms.

# Define servo limits for drawing  
SERVO\_MIN\_ANGLE = **60** # Minimum angle for pen movement  
SERVO\_MAX\_ANGLE = **120** # Maximum angle for pen movement

* Ορίζουν τα όρια της γωνίας κίνησης του servo (σε μοίρες), ώστε να περιοριστεί η κίνηση του στυλό.

### **Συνάρτηση για τον έλεγχο του Servo**

**def** **SetServo**(servoPin, angle):  
 **if** angle < **0**:  
 angle = **0**  
 **elif** angle > **180**:  
 angle = **180**  
   
 pulse = (SERVO\_MAX\_PULSE - SERVO\_MIN\_PULSE) \* angle / **180** + SERVO\_MIN\_PULSE  
 servoPin.write\_analog(pulse))

* **SetServo**: Ρυθμίζει τη γωνία του servo.
* Ελέγχει αν η γωνία είναι εντός των επιτρεπτών ορίων (0°–180°).

Υπολογίζει το κατάλληλο παλμικό πλάτος για την επιθυμητή γωνία και το εφαρμόζει μέσω PWM.

### **Ρυθμίσεις και αρχικοποίηση**

# Configuration   
SAMPLE\_RATE = **500** # 10 ms delay = 100 Hz sampling   
LOG\_DURATION = **60** # Log for 60 seconds

* **SAMPLE\_RATE**: Ορίζει τη συχνότητα δειγματοληψίας σε χιλιοστά του δευτερολέπτου (500 ms = 2 Hz).
* **LOG\_DURATION**: Καθορίζει τη διάρκεια καταγραφής δεδομένων σε δευτερόλεπτα.

# Initialize variables for averaging  
previous\_magnitude = **0**  
alpha = **0.7** # Smoothing factor (0.0 to 1.0, higher = smoother)

# Threshold value  
threshold = **0.5** # Adjust based on sensitivity needs

* **previous\_magnitude**: Αποθηκεύει την προηγούμενη τιμή επιτάχυνσης για ομαλή μετάβαση.
* **alpha**: Παράγοντας εξομάλυνσης για το φίλτρο εκθετικής εξομάλυνσης.
* **threshold**: Καθορίζει το όριο ευαισθησίας για την καταγραφή δεδομένων.

### **Συναρτήσεις υπολογισμού 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   
  
**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**

* Υπολογίζει την ομαλή τιμή του μέτρου επιτάχυνσης χρησιμοποιώντας εκθετική εξομάλυνση.
* Περιορίζει την τιμή σε ένα εύρος 0–9 για απλότητα.

### **Κανονικοποίηση γωνίας για το Servo**

# Function to normalize magnitude to servo angle (0-180 degrees)  
**def** **normalize\_to\_servo\_angle**(magnitude, min\_value=**0**, max\_value=**10**):  
 # Clamp magnitude to the expected range  
 magnitude = max(min\_value, min(max\_value, magnitude))  
  
 # Normalize to the limited servo range (SERVO\_MIN\_ANGLE to SERVO\_MAX\_ANGLE)  
 **return** int((magnitude - min\_value) / (max\_value - min\_value) \* (SERVO\_MAX\_ANGLE - SERVO\_MIN\_ANGLE) + SERVO\_MIN\_ANGLE)

* Κανονικοποιεί την τιμή του magnitude (0–10) σε γωνία του servo (60°–120°).

### **Εκκίνηση και κύριος βρόχος**

start\_time = time.ticks\_ms()   
  
# Initialize servo to center position  
SetServo(servoPin, (SERVO\_MIN\_ANGLE + SERVO\_MAX\_ANGLE) // **2**))

* Ορίζει την αρχή του χρόνου καταγραφής και τοποθετεί το servo στη μέση της περιοχής κίνησης.

**while** time.ticks\_diff(time.ticks\_ms(), start\_time) < LOG\_DURATION \* **1000**:   
 x = accelerometer.get\_x()   
 y = accelerometer.get\_y()   
 z = accelerometer.get\_z()

* Ξεκινά τον κύριο βρόχο που εκτελείται για 60 δευτερόλεπτα.

Λαμβάνει τις τιμές επιτάχυνσης από τους άξονες x, y, z.

magnitude = calculate\_smoothed\_magnitude(x, y, z)   
 **print**(magnitude)  
 # Normalize magnitude to servo angle  
 servo\_angle = normalize\_to\_servo\_angle(magnitude)  
 **print**(servo\_angle)  
 SetServo(servoPin, servo\_angle) # servo pin and angle

* Υπολογίζει την ομαλή τιμή του magnitude, την κανονικοποιεί σε γωνία και ρυθμίζει το servo.

# 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\_smoothed\_magnitude(x, y, z)   
 })

* Καταγράφει τα δεδομένα αν το μέτρο της επιτάχυνσης ξεπερνά το όριο threshold.

sleep(SAMPLE\_RATE)

* Καθυστερεί την επόμενη μέτρηση για 500 ms.

**print**("Done!")

* Εμφανίζει μήνυμα ολοκλήρωσης μετά το πέρας του κύριου βρόχου.

# Μαθηματική Επεξήγηση Συναρτήσεις Servo

#### **SetServo(servoPin, angle)**

Αυτή η συνάρτηση μετατρέπει τη γωνία (0–180 μοίρες) σε παλμούς PWM, ώστε ο servo να μετακινηθεί στη σωστή θέση.

* **Εύρος παλμών (pulse width):**
  + Οι παλμοί που απαιτούνται για τον servo κυμαίνονται μεταξύ του SERVO\_MIN\_PULSE και του SERVO\_MAX\_PULSE.
  + Ο τύπος για τον υπολογισμό του παλμού είναι:



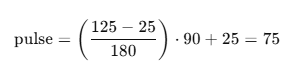
* + Αυτός ο τύπος κανονικοποιεί τη γωνία από το εύρος *0–1800–180*0–180 στο αντίστοιχο εύρος παλμών που μπορεί να χειριστεί ο servo.
* **Περιορισμός γωνίας:**
  + Αν η γωνία είναι μικρότερη από 0 ή μεγαλύτερη από 180, περιορίζεται στο αντίστοιχο όριο:



#### **Παράδειγμα:**

Αν angle=90, τότε:

* SERVO\_MIN\_PULSE=25, SERVO\_MAX\_PULSE=125
* Υπολογίζουμε:



* Ο servo λαμβάνει έναν παλμό 75, που αντιστοιχεί στη γωνία 90°.

#### **normalize\_to\_servo\_angle(magnitude, min\_value=0, max\_value=10)**

Αυτή η συνάρτηση κανονικοποιεί το μέγεθος (magnitude) που υπολογίζεται από τον επιταχυνσιόμετρο στο εύρος γωνιών του servo (60–120 μοίρες).

**Κανονικοποίηση μεγέθους:**

* Το μέγεθος περιορίζεται στο εύρος [min\_value,max\_value]:



**Μετατροπή σε γωνία:**

* Ο τύπος μετατροπής είναι:



* Αυτός ο τύπος κανονικοποιεί το μέγεθος από το εύρος [min\_value,max\_value] στο αντίστοιχο εύρος γωνιών [SERVO\_MIN\_ANGLE,SERVO\_MAX\_ANGLE].

#### **Παράδειγμα:**

Αν magnitude=5, max\_value=10, τότε:

* SERVO\_MIN\_ANGLE=60, SERVO\_MAX\_ANGLE=120
* Υπολογίζουμε:



Ο servo μετακινείται στη γωνία 90°.

### **Συνοψίζοντας:**

1. **SetServo:** Μετατρέπει τη γωνία [0,180] σε παλμούς [25,125].
2. **normalize\_to\_servo\_angle:** Μετατρέπει το μέγεθος [0,10] σε γωνίες [60,120].
3. Οι υπολογισμοί διασφαλίζουν ότι ο servo λειτουργεί εντός ασφαλών ορίων και αντιδρά στις αλλαγές του επιταχυνσιόμετρου με ομαλό τρόπο.