## **1. Πρόγραμμα Αποστολέα (Transmitter)**

Αυτό το πρόγραμμα εκτελείται στο **πρώτο Micro:bit** και είναι υπεύθυνο για:

* Ανάγνωση των δεδομένων από τον **επιταχυνσιόμετρο**.
* Υπολογισμό του **μεγέθους της επιτάχυνσης (magnitude)**.
* Αποστολή της τιμής του **magnitude** μέσω **ραδιοσυχνότητας** (radio) στο δεύτερο Micro:bit.

**from** **microbit** **import** \* # Εισάγουμε τη βιβλιοθήκη του Micro:bit  
**import** **math** # Χρησιμοποιείται για υπολογισμούς της ρίζας  
**import** **radio** # Για ασύρματη επικοινωνία μέσω ραδιοκυμάτων  
  
radio.on() # Ενεργοποιούμε τη ραδιοεπικοινωνία  
radio.config(group=**1**) # Ορίζουμε το κανάλι επικοινωνίας (τα δυο micro:bit πρέπει να έχουν το ίδιο)  
  
alpha = **0.7** # Συντελεστής εξομάλυνσης (για πιο σταθερές τιμές)  
previous\_magnitude = **0** # Αρχική τιμή του προηγούμενου magnitude

🔹 **Γιατί χρησιμοποιούμε το radio.config(group=**1**);**

Για να διασφαλίσουμε ότι μόνο τα Micro:bit που έχουν τον ίδιο αριθμό καναλιού επικοινωνούν μεταξύ τους. Μπορούμε να αλλάξουμε το κανάλι αν θέλουμε να χρησιμοποιήσουμε πολλά ζευγάρια Micro:bit στον ίδιο χώρο.

**def** **calculate\_smoothed\_magnitude**(x, y, z):  
 **global** previous\_magnitude  
 magnitude = math.sqrt(x\*\***2** + y\*\***2** + z\*\***2**) / **1024** # Κανονικοποίηση σε "g"  
 smoothed\_magnitude = alpha \* magnitude + (**1** - alpha) \* previous\_magnitude  
 previous\_magnitude = smoothed\_magnitude  
 **return** int((smoothed\_magnitude \* **10**) % **10**) # Μετατροπή σε κλίμακα 0-9

✅ **Τι κάνει η συνάρτηση αυτή;**

1. Υπολογίζει το **μέγεθος της επιτάχυνσης** χρησιμοποιώντας το **Πυθαγόρειο θεώρημα**:



1. Χρησιμοποιεί **εκθετική εξομάλυνση (Exponential Smoothing)** για να **ελαχιστοποιήσει τις απότομες αλλαγές** και να έχουμε πιο ομαλά δεδομένα.
2. Επιστρέφει μια τιμή μεταξύ **0 και 9**, ώστε να είναι εύκολη η αποστολή και επεξεργασία.

**while** True:  
 x = accelerometer.get\_x()  
 y = accelerometer.get\_y()  
 z = accelerometer.get\_z()  
  
 magnitude = calculate\_smoothed\_magnitude(x, y, z)  
   
 radio.send(str(magnitude)) # Στέλνουμε την τιμή μέσω radio  
 **print**("Sent:", magnitude) # Για έλεγχο μέσω του σειριακού  
  
 sleep(**500**) # Αναμονή 500ms για να μην στέλνει υπερβολικά συχνά

🔹 **Βήματα:**

1. Διαβάζει τα δεδομένα από τον **επιταχυνσιόμετρο**.
2. Υπολογίζει το **magnitude** και το μετατρέπει σε μια τιμή από 0-9.
3. **Στέλνει** αυτή την τιμή μέσω **ραδιοσυχνοτήτων** στο άλλο Micro:bit.
4. **Εμφανίζει** τη μεταβλητή magnitude στην κονσόλα για έλεγχο.
5. **Περιμένει 500ms** πριν ξανατρέξει.

8

## **2. Πρόγραμμα Δέκτη (Receiver)**

Αυτό το πρόγραμμα εκτελείται στο **δεύτερο Micro:bit** και είναι υπεύθυνο για:

* **Λήψη** των δεδομένων που στέλνει ο αποστολέας.
* **Μετατροπή της τιμής** που λαμβάνει σε **γωνία του σερβοκινητήρα**.
* **Κίνηση του σερβοκινητήρα** ανάλογα με την τιμή του magnitude.

**from** **microbit** **import** \* # Εισάγουμε τη βιβλιοθήκη του Micro:bit  
**import** **radio** # Για ασύρματη επικοινωνία μέσω ραδιοκυμάτων  
  
radio.on() # Ενεργοποιούμε τη ραδιοεπικοινωνία  
radio.config(group=**1**) # Ορίζουμε το κανάλι επικοινωνίας (τα δυο micro:bit πρέπει να έχουν το ίδιο)

SERVO\_MIN\_PULSE = **25**  
SERVO\_MAX\_PULSE = **125**  
SERVO\_MIN\_ANGLE = **60**  
SERVO\_MAX\_ANGLE = **120**  
  
servoPin = pin0  
servoPin.set\_analog\_period(**20**)

🔹 **Πώς ελέγχουμε τον σερβοκινητήρα;**

* Το **PWM (Pulse Width Modulation)** χρησιμοποιείται για να ορίσουμε τη γωνία του σερβοκινητήρα.
* Οι **ελάχιστες και μέγιστες γωνίες** έχουν οριστεί μεταξύ **60° και 120°** ώστε να είναι σε περιορισμένο εύρος.

**def** **SetServo**(servoPin, angle):  
 angle = max(**0**, min(**180**, angle)) # Περιορίζουμε τη γωνία στο 0-180  
 pulse = (SERVO\_MAX\_PULSE - SERVO\_MIN\_PULSE) \* angle / **180** + SERVO\_MIN\_PULSE  
 servoPin.write\_analog(pulse)

✅ **Πώς λειτουργεί;**

* Παίρνει μια **γωνία (angle)** μεταξύ **0° και 180°**.
* Υπολογίζει το αντίστοιχο **PWM σήμα** που πρέπει να σταλεί στον σερβοκινητήρα.

**def** **normalize\_to\_servo\_angle**(magnitude, min\_value=**0**, max\_value=**10**):  
 magnitude = max(min\_value, min(max\_value, magnitude))  
 **return** int((magnitude - min\_value) / (max\_value - min\_value) \* (SERVO\_MAX\_ANGLE - SERVO\_MIN\_ANGLE) + SERVO\_MIN\_ANGLE)

✅ **Μετατροπή από magnitude σε γωνία**

* Αν το **magnitude** είναι **0 → γωνία 60°**.
* Αν το **magnitude** είναι **10 → γωνία 120°**.

**while** True:  
 message = radio.receive() # Περιμένουμε να λάβουμε μήνυμα  
 **if** message:  
 **try**:  
 magnitude = int(message) # Μετατροπή του μηνύματος σε ακέραιο  
 **print**("Received:", magnitude)  
  
 servo\_angle = normalize\_to\_servo\_angle(magnitude)  
 SetServo(servoPin, servo\_angle) # Κίνηση του σερβοκινητήρα  
  
 **except** **ValueError**:  
 **pass** # Αν λάβουμε άκυρο μήνυμα, το αγνοούμε  
  
 sleep(**500**) # Αναμονή πριν την επόμενη λήψη

🔹 **Βήματα:**

1. Το **Micro:bit ακούει** για δεδομένα μέσω **ραδιοσυχνοτήτων**.
2. Αν λάβει μήνυμα, το **μετατρέπει** σε αριθμό.
3. Μετατρέπει την τιμή του magnitude σε **γωνία** του servo.
4. **Κινεί τον σερβοκινητήρα** στην κατάλληλη γωνία.

### **Τελικό Αποτέλεσμα**

🚀 **Το πρώτο Micro:bit ανιχνεύει κραδασμούς και στέλνει τα δεδομένα.**

🤖 **Το δεύτερο Micro:bit κινεί τον σερβοκινητήρα, αναπαριστώντας τη σεισμική δραστηριότητα.**