Σταθερές (Constants) - Γραμμές 5-17

Καθολικές Μεταβλητές (Global Variables) - Γραμμές 18-21

Αρχικοποίηση Αντικειμένων Υλικού (Hardware Object Initialization) - Γραμμές 23-31

Συναρτήσεις (Functions):

**set\_motor\_speed(in1\_pwm, in2\_pwm, speed\_percent):**

Ορίζει την ταχύτητα και την κατεύθυνση ενός κινητήρα.

Παίρνει ως ορίσματα τα δύο αντικείμενα PWM που ελέγχουν τον κινητήρα και μια ποσοστιαία ταχύτητα (speed\_percent) από -100 (πλήρης ταχύτητα προς τα πίσω) έως 100 (πλήρης ταχύτητα προς τα εμπρός).

Μετατρέπει το ποσοστό ταχύτητας σε τιμή κύκλου λειτουργίας PWM (duty).

Αν speed\_percent > 0, ο κινητήρας κινείται εμπρός (το in1\_pwm ενεργοποιείται, το in2\_pwm μηδενίζεται).

Αν speed\_percent < 0, ο κινητήρας κινείται προς τα πίσω (το in1\_pwm μηδενίζεται, το in2\_pwm ενεργοποιείται).

Αν speed\_percent == 0, ο κινητήρας σταματά (και τα δύο PWM μηδενίζονται).

**stop\_motors(brake=False, brake\_duration\_ms=50, brake\_power\_percent=30):**

Σταματά και τους δύο κινητήρες.

Αν η παράμετρος brake είναι True, εφαρμόζει ένα σύντομο "φρενάρισμα" κινούμενος τους κινητήρες στιγμιαία προς την αντίθετη κατεύθυνση (-abs(brake\_power\_percent)) για brake\_duration\_ms χιλιοστά του δευτερολέπτου, πριν τους σταματήσει τελείως.

Τελικά, καλεί την set\_motor\_speed για να μηδενίσει την ταχύτητα και των δύο κινητήρων.

**read\_raw\_sensor\_values():**

Διαβάζει τις ακατέργαστες (raw) τιμές από όλους τους αισθητήρες γραμμής.

Χρησιμοποιεί τη μέθοδο adc.read() για κάθε κανάλι του MCP3008 που αντιστοιχεί σε αισθητήρα (σύμφωνα με τη λίστα SENSOR\_MCP\_CHANNELS).

Επιστρέφει μια λίστα με τις ακατέργαστες τιμές (0-1023).

**get\_normalized\_sensor\_values():**

Παίρνει τις ακατέργαστες τιμές από την read\_raw\_sensor\_values().

Κανονικοποιεί αυτές τις τιμές σε ένα εύρος από 0.0 έως 1.0, χρησιμοποιώντας τις τιμές βαθμονόμησης sensor\_min\_readings (για μαύρο) και sensor\_max\_readings (για λευκό).

Η κανονικοποίηση γίνεται έτσι ώστε η τιμή 1.0 να αντιστοιχεί περίπου σε μαύρο (έντονη αντανάκλαση από τη γραμμή) και η τιμή 0.0 σε λευκό.

Περιλαμβάνει λογική για την περίπτωση που η διαφορά μεταξύ min και max είναι πολύ μικρή.

Επιστρέφει τόσο τις ακατέργαστες όσο και τις κανονικοποιημένες τιμές.

**save\_calibration\_data(min\_b, max\_w):**

Αποθηκεύει τις τιμές βαθμονόμησης (ελάχιστες για μαύρο, μέγιστες για λευκό) σε ένα αρχείο JSON (CALIBRATION\_FILENAME).

Χρησιμοποιεί τη βιβλιοθήκη ujson για τη σειριοποίηση των δεδομένων.

Περιλαμβάνει χειρισμό σφαλμάτων σε περίπτωση που η αποθήκευση αποτύχει.

**load\_calibration\_data():**

Προσπαθεί να φορτώσει τα δεδομένα βαθμονόμησης από το αρχείο JSON.

Χρησιμοποιεί τη βιβλιοθήκη ujson για την αποσειριοποίηση.

Ελέγχει αν τα φορτωμένα δεδομένα έχουν τη σωστή μορφή και αν ο αριθμός των τιμών ταιριάζει με το NUM\_SENSORS.

Αν η φόρτωση είναι επιτυχής και τα δεδομένα έγκυρα, ενημερώνει τις καθολικές μεταβλητές sensor\_min\_readings και sensor\_max\_readings και επιστρέφει True.

Σε περίπτωση σφάλματος (π.χ., το αρχείο δεν υπάρχει, τα δεδομένα είναι κατεστραμμένα), εκτυπώνει ένα μήνυμα και επιστρέφει False.

**perform\_interactive\_calibration(duration\_sec=20):**

Εκτελεί μια διαδικασία διαδραστικής βαθμονόμησης των αισθητήρων.

Για μια καθορισμένη διάρκεια (duration\_sec), διαβάζει συνεχώς τις τιμές των αισθητήρων.

Καταγράφει τις ελάχιστες (min\_b\_cal) και μέγιστες (max\_w\_cal) τιμές που ανιχνεύει κάθε αισθητήρας κατά τη διάρκεια αυτής της περιόδου (υποτίθεται ότι ο χρήστης κινεί το ρομπότ πάνω από μαύρες και λευκές επιφάνειες).

Μετά τη λήξη του χρόνου, ενημερώνει τις καθολικές μεταβλητές sensor\_min\_readings και sensor\_max\_readings με τις τιμές που καταγράφηκαν.

Παίζει έναν ήχο (δύο μπιπ) από τον βομβητή για να σηματοδοτήσει το τέλος της βαθμονόμησης.

Εκτυπώνει προειδοποιήσεις αν για κάποιον αισθητήρα η μέγιστη τιμή είναι μικρότερη ή ίση της ελάχιστης (ένδειξη προβληματικής βαθμονόμησης).

Αποθηκεύει τα νέα δεδομένα βαθμονόμησης χρησιμοποιώντας την save\_calibration\_data().

**calculate\_line\_position(norm\_sensor\_values):**

Υπολογίζει τη θέση της γραμμής σε σχέση με το κέντρο του ρομπότ, χρησιμοποιώντας τις κανονικοποιημένες τιμές των αισθητήρων.

Χρησιμοποιεί μια λίστα "βαρών" (weights = [-2, -1, 0, 1, 2]). Οι ακριανοί αισθητήρες έχουν μεγαλύτερο βάρος.

Υπολογίζει έναν σταθμισμένο μέσο όρο: (άθροισμα(βάρος[i] \* τιμή\_αισθητήρα[i])) / (άθροισμα(τιμή\_αισθητήρα[i])).

Η τιμή που επιστρέφεται είναι το "σφάλμα" (error).Μια τιμή κοντά στο 0 σημαίνει ότι η γραμμή είναι στο κέντρο.Μια αρνητική τιμή σημαίνει ότι η γραμμή είναι αριστερά.Μια θετική τιμή σημαίνει ότι η γραμμή είναι δεξιά.

Αν κανένας αισθητήρας δεν ανιχνεύει έντονα τη γραμμή, προσπαθεί να "μαντέψει" τη θέση της γραμμής με βάση το προηγούμενο σφάλμα (last\_error), για να βοηθήσει το ρομπότ να ξαναβρεί τη γραμμή αν την χάσει.

**turn\_off\_long\_beep\_cb(timer\_object\_passed\_in):**

Μια συνάρτηση επανάκλησης (callback) που προορίζεται να χρησιμοποιηθεί από ένα αντικείμενο Timer.

Απενεργοποιεί τον buzzer θέτοντας τον κύκλο λειτουργίας του PWM του σε 0.

**play\_f1\_start\_sound():**

Παίζει μια ακολουθία ήχων που μιμείται την αντίστροφη μέτρηση και την εκκίνηση ενός αγώνα Formula 1.

Χρησιμοποιεί τον buzzer\_pwm για να παράγει τους ήχους, αλλάζοντας τη συχνότητα και τον κύκλο λειτουργίας.

Για τον τελευταίο, παρατεταμένο ήχο, χρησιμοποιεί ένα αντικείμενο Timer (long\_beep\_timer) για να καλέσει την turn\_off\_long\_beep\_cb μετά από συγκεκριμένο χρονικό διάστημα, ώστε ο ήχος να σταματήσει αυτόματα.

**main\_loop():**

Αυτή είναι η κύρια συνάρτηση που εκτελεί τη λογική του ρομπότ που ακολουθεί τη γραμμή.

Πρώτα, προσπαθεί να φορτώσει δεδομένα βαθμονόμησης. Αν δεν υπάρχουν έγκυρα δεδομένα, εκτελεί την perform\_interactive\_calibration().

Παίζει τον ήχο εκκίνησης (play\_f1\_start\_sound()).

Περιμένει λίγο να τελειώσει ο ήχος.

Μπαίνει σε έναν ατέρμονα βρόχο (while True) μέσα σε ένα μπλοκ try...finally:

* + - **Μέσα στον βρόχο:**
      1. Διαβάζει και κανονικοποιεί τις τιμές των αισθητήρων (get\_normalized\_sensor\_values()).
      2. Ελέγχει αν όλοι οι αισθητήρες ανιχνεύουν μαύρο (all\_sensors\_on\_black). Αν ναι (π.χ., έφτασε σε μια μεγάλη μαύρη περιοχή ή στο τέλος της γραμμής), σταματά τους κινητήρες με φρενάρισμα και βγαίνει από τον βρόχο.
      3. Υπολογίζει τη θέση της γραμμής / το σφάλμα (calculate\_line\_position()).
      4. **Εφαρμόζει τον αλγόριθμο PID:**
         * p\_term = KP \* error (Αναλογικός όρος)
         * integral += error (Συσσωρεύει το σφάλμα)
         * Περιορίζει την τιμή του integral (anti-windup).
         * i\_term = KI \* integral (Ολοκληρωτικός όρος)
         * derivative = error - last\_error (Υπολογίζει την αλλαγή στο σφάλμα)
         * d\_term = KD \* derivative (Παραγωγικός όρος)
         * last\_error = error (Αποθηκεύει το τρέχον σφάλμα για την επόμενη επανάληψη)
         * correction = p\_term + i\_term + d\_term (Η συνολική διόρθωση που πρέπει να εφαρμοστεί στην ταχύτητα των κινητήρων).
      5. Υπολογίζει την επιθυμητή ταχύτητα για τον αριστερό και τον δεξιό κινητήρα (left\_speed\_command, right\_speed\_command) προσθέτοντας/αφαιρώντας τη correction από τη BASE\_SPEED.
      6. Περιορίζει τις τιμές ταχύτητας των κινητήρων στο εύρος [-100, 100].
      7. Ορίζει τις νέες ταχύτητες στους κινητήρες χρησιμοποιώντας την set\_motor\_speed().
    - **except KeyboardInterrupt:** Αν ο χρήστης πατήσει Ctrl+C, εκτυπώνει ένα μήνυμα και το πρόγραμμα θα προχωρήσει στο finally.
    - **finally:** Αυτό το μπλοκ εκτελείται πάντα, είτε ο βρόχος τερματιστεί κανονικά, είτε λόγω εξαίρεσης (όπως KeyboardInterrupt).
      1. Σταματά τους κινητήρες με φρενάρισμα.
      2. Απενεργοποιεί τον βομβητή.
      3. Απενεργοποιεί τον χρονοδιακόπτη long\_beep\_timer αν είχε δημιουργηθεί.