#include <Servo.h> // Include the Servo library

// Define pins for the ultrasonic sensor, servo motor, soil moisture sensor, IR sensor, and buzzer

const int trigPin = 9;

const int echoPin = 10;

const int servoPin = 7;

const int sensorPin = A0; // Analog pin A0 for soil moisture sensor

const int buzzerPin = 13; // Digital pin 13 for buzzer

const int irSensorPin = 8; // IR sensor output pin connected to digital pin 8

const int ledPin = 12; // Optional LED pin for IR sensor status (connected to pin 12)

// Variables to store sensor values

long duration;

int distance;

int sensorValue = 0;

int moistureLevel = 0;

int irSensorValue = HIGH; // Default state of IR sensor (no object detected)

bool objectDetected = false; // Variable to track object detection state

bool doorOpen = false; // Variable to track if the door is open

bool wastePlaced = false; // Variable to check if waste is placed

bool moistureDetected = false; // Flag to check if moisture is detected

bool irWasteDetected = false; // Flag to check if IR sensor detected waste

bool objectRemoved = false; // Flag to check if object has been removed

bool wasteTypeSent = false; // Flag to ensure waste type is sent only once

unsigned long objectRemovedTime = 0; // Timer for delaying the dustbin closing

Servo myServo; // Create a Servo object

void setup() {

// Start serial communication for debugging

Serial.begin(9600);

// Set pin modes

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(sensorPin, INPUT);

pinMode(buzzerPin, OUTPUT);

pinMode(irSensorPin, INPUT);

pinMode(ledPin, OUTPUT);

// Attach the servo motor and set initial position (door closed)

myServo.attach(servoPin);

myServo.write(0);

// Initialize LED to off

digitalWrite(ledPin, LOW);

}

// Function to send data to PC

void sendToPC(bool isOrganic, int moistureValue = 0) {

if (isOrganic) {

Serial.print("1,0,");

Serial.println(moistureValue); // Send organic signal with moisture level

} else {

Serial.println("0,1,0"); // Send inorganic signal with moisture level as 0

}

}

// Dummy function to detect organic waste (Replace with actual detection logic)

bool detectOrganicWaste() {

return moistureDetected; // Example logic: organic waste detected if moisture is present

}

// Dummy function to detect inorganic waste (Replace with actual detection logic)

bool detectInorganicWaste() {

return !moistureDetected && irWasteDetected; // Example logic: inorganic if IR detects waste but no moisture.

}

void loop() {

// Ultrasonic sensor section

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Measure the duration and calculate the distance

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.034 / 2;

// Step 1: Check if an object is detected within 30 cm

if (distance > 0 && distance <= 30) {

if (!objectDetected) {

objectDetected = true; // Object detected

Serial.print("Object detected within 30 cm! Distance: ");

Serial.print(distance);

Serial.println(" cm");

// Step 2: Open the dustbin

myServo.write(90); // Door open position

doorOpen = true; // Mark door as open

wastePlaced = false; // Reset waste placed status

moistureDetected = false; // Reset moisture detected status

irWasteDetected = false; // Reset IR sensor detection status

wasteTypeSent = false; // Reset the flag to allow new waste type detection

Serial.println("Dustbin door opened. Please place the waste.");

}

}

// Step 3: Check the IR sensor and soil moisture sensor only if the door is open

if (doorOpen) {

irSensorValue = digitalRead(irSensorPin); // Read IR sensor value

// Step 4: Check if the waste is placed (IR sensor triggered)

if (irSensorValue == LOW && !irWasteDetected) {

irWasteDetected = true; // Waste detected by IR sensor

Serial.println("Waste detected by IR sensor. Waiting for moisture check.");

// Start checking moisture sensor

unsigned long wastePlacedTime = millis(); // Record the time when waste is placed

// Monitor for 10 seconds or until moisture is detected

while (millis() - wastePlacedTime < 5000) {

sensorValue = analogRead(sensorPin); // Read soil moisture sensor value

// Condition: If moisture is detected (sensor value < dry air reading)

if (sensorValue < 900) {

moistureLevel = map(sensorValue, 1023, 0, 0, 100);

moistureDetected = true; // Mark that moisture is detected

Serial.print("Moisture Level: ");

Serial.println(moistureLevel);

if (moistureLevel > 0) {

Serial.println("Moisture detected! Keeping buzzer on.");

digitalWrite(buzzerPin, HIGH); // Turn on buzzer if moisture is detected

break; // Exit the 10-second countdown early if moisture is detected

}

}

}

// If no moisture is detected after 10 seconds, the IR sensor gives output

if (!moistureDetected) {

Serial.println("No moisture detected within 10 seconds. Waste is dry.");

digitalWrite(ledPin, HIGH); // Turn on LED to indicate waste is dry (or perform other action)

}

}

}

// Waste type detection and sending data to PC

if (!wasteTypeSent) { // Check if waste type has already been sent

bool isOrganic = detectOrganicWaste(); // Replace with actual logic

bool isInorganic = detectInorganicWaste(); // Replace with actual logic

// If organic waste is detected, send organic data to the PC with moisture level

if (isOrganic) {

sendToPC(true, moistureLevel); // Sends "1,0,moistureLevel"

wasteTypeSent = true; // Mark waste type as sent

}

// If inorganic waste is detected, send inorganic data to the PC with moisture level as 0

else if (isInorganic) {

sendToPC(false); // Sends "0,1,0"

wasteTypeSent = true; // Mark waste type as sent

}

}

// Step 5: Check if waste is removed from the moisture sensor and IR sensor

if (irWasteDetected || moistureDetected) {

sensorValue = analogRead(sensorPin); // Continuously read soil moisture sensor

irSensorValue = digitalRead(irSensorPin); // Continuously read IR sensor

// Condition: If waste is removed from both sensors (no moisture and IR sensor HIGH)

if (sensorValue >= 900 && irSensorValue == HIGH) {

Serial.println("Waste removed from both IR and moisture sensors. Resetting system.");

digitalWrite(buzzerPin, LOW); // Turn off the buzzer

digitalWrite(ledPin, LOW); // Turn off the LED

moistureDetected = false; // Reset moisture detected flag

irWasteDetected = false; // Reset IR waste detected flag

wasteTypeSent = false; // Reset waste type sent flag

}

}

// Step 6: Object is removed from detection range

if (objectDetected && distance > 30) {

if (!objectRemoved) {

objectRemoved = true; // Object was just removed

objectRemovedTime = millis(); // Store the time when object is removed

Serial.println("Object removed, starting 10-second timer to close dustbin.");

}

}

// Step 7: Close the door 10 seconds after the object is removed, only if no new object is detected

if (objectRemoved) {

// Recheck the ultrasonic sensor within the 10-second countdown

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.034 / 2;

// If a new object is detected, reset the countdown and keep the door open

if (distance > 0 && distance <= 30) {

Serial.println("New object detected, resetting 10-second timer.");

objectRemoved = false; // Reset removal flag to keep the door open

}

else if (millis() - objectRemovedTime >= 10000) {

Serial.println("10 seconds passed, closing dustbin.");

myServo.write(0); // Close the dustbin

doorOpen = false; // Mark door as closed

objectDetected = false; // Reset object detected state

objectRemoved = false; // Reset removal flag

wasteTypeSent = false; // Reset the flag to allow new waste type detection

digitalWrite(buzzerPin, LOW); // Ensure the buzzer is off

digitalWrite(ledPin, LOW); // Turn off the LED

}

}

delay(100); // Small delay between readings

}

from sklearn.ensemble import RandomForestClassifier  
import joblib  
  
*# Example training data: moisture levels and corresponding waste types  
# X\_train: Moisture levels (input feature)  
# y\_train: Waste types (target labels)*X\_train = [[10], [0], [55], [75], [30], [60]] *# Moisture levels*y\_train = ['Organic', 'Inorganic', 'Organic', 'Organic', 'Inorganic', 'Organic'] *# Waste type labels  
  
# Store the unique moisture levels used during training*trained\_moisture\_levels = set([x[0] for x in X\_train])  
  
*# Initialize and train the classifier*model = RandomForestClassifier(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
*# Save the trained model and the list of trained moisture levels*joblib.dump((model, trained\_moisture\_levels), 'waste\_classifier\_model.pkl')  
  
*# Output the trained moisture levels for reference*print("Trained on the following moisture levels:", trained\_moisture\_levels)

import joblib  
import serial  
import csv  
import time  
import numpy as np  
import os  
import pandas as pd *# Import pandas for CSV file operations  
  
# Load pre-trained ML model (replace with your model file path)*model, trained\_moisture\_levels = joblib.load('waste\_classifier\_model.pkl')  
  
*# Set up the serial connection (ensure the correct COM port is used)*arduino = serial.Serial('COM3', 9600) *# Change COM3 to the correct port for your Arduino  
  
# File path for the CSV file*csv\_file\_path = 'waste\_data.csv'  
  
*# Load existing CSV data into a pandas DataFrame, if the file exists*if os.path.exists(csv\_file\_path):  
 data = pd.read\_csv(csv\_file\_path)  
else:  
 *# Create an empty DataFrame with the necessary columns if the file does not exist* data = pd.DataFrame(columns=['Date', 'Time', 'Predicted Waste Type', 'Moisture Level', 'Moisture Match Status'])  
  
*# Function to check if a moisture level matches any previous entries*def check\_moisture\_status(moisture\_level, data):  
 if moisture\_level == 0: *# No moisture detected* return "No Moisture Detected"  
 elif moisture\_level in data['Moisture Level'].values:  
 return "Moisture Level Matched"  
 else:  
 return "New Moisture Level"  
  
*# Open or create the CSV file to log the data*with open(csv\_file\_path, 'a', newline='') as csvfile: *# Open in append mode ('a') to avoid overwriting* csvwriter = csv.writer(csvfile)  
  
 *# Write the header if the file is new (i.e., doesn't already exist)* if data.empty: *# If the DataFrame is empty, it means the file is new* csvwriter.writerow(['Date', 'Time', 'Predicted Waste Type', 'Moisture Level', 'Moisture Match Status'])  
 csvfile.flush() *# Ensure the header is written immediately* while True:  
 try:  
 *# Read a line from the serial input* line = arduino.readline().decode('utf-8').strip()  
 print(f"Received: {line}") *# Debugging print to see what is received  
  
 # Check if the line contains the expected comma-separated values* if ',' in line:  
 try:  
 *# Split the line into organic, inorganic, and moisture values* organic, inorganic, moisture\_level = line.split(',')  
  
 *# Ensure that moisture level is properly converted to an integer* moisture\_level = int(moisture\_level)  
  
 *# Use the ML model to predict waste type (organic/inorganic)* features = np.array([[moisture\_level]])  
 predicted\_waste\_type = model.predict(features)[0] *# 'Organic' or 'Inorganic'  
  
 # Get the current date and time* current\_date = time.strftime('%Y-%m-%d') *# Extract date* current\_time = time.strftime('%H:%M:%S') *# Extract time  
  
 # Determine the moisture match status using pandas DataFrame* moisture\_status = check\_moisture\_status(moisture\_level, data)  
  
 *# Log the new data entry in the CSV file* csvwriter.writerow([current\_date, current\_time, predicted\_waste\_type, moisture\_level, moisture\_status])  
  
 *# Update the DataFrame with the new entry* new\_row = {  
 'Date': current\_date,  
 'Time': current\_time,  
 'Predicted Waste Type': predicted\_waste\_type,  
 'Moisture Level': moisture\_level,  
 'Moisture Match Status': moisture\_status  
 }  
 data = pd.concat([data, pd.DataFrame([new\_row])], ignore\_index=True)  
  
 *# Flush the file to ensure data is written immediately* csvfile.flush()  
  
 *# Display the logged information in the console* print(f"Logged: {current\_date} {current\_time}, Predicted Waste Type: {predicted\_waste\_type}, "  
 f"Moisture Level: {moisture\_level}, Status: {moisture\_status}")  
  
 except ValueError:  
 *# Handle cases where the data cannot be split or converted correctly* print(f"Error: Unable to process the line: {line}")  
 else:  
 print(f"Unexpected data format: {line}")  
  
 except Exception as e:  
 print(f"Error: {e}")