



**University of Science and Technology Chittagong (USTC)**

Faculty of Science, Engineering & Technology  
Department of Computer Science & Engineering

**Lab Report - 6, 7 & 8**

**Course code : CSE 324**

**Course Title : Artificial Intelligence and Expert Systems Lab**

**Team Name - The Elite**

**Submitted by :**

Abhilash Chowdhury

Unique ID : 0022210005101010

Roll No : 22010110

Reg No : 888

Semester : 6<sup>th</sup>

Batch : 38<sup>th</sup>

**Submitted to:**

Debabarata Mallick

[ Lecturer ]

FSET, USTC.

# Analysis of Arduino Code

## Why Did We Use Arduino Code in our Project :

The Arduino code is essential for automating and controlling the smart bin ai by integrating sensors (ultrasonic, IR, and soil moisture) and actuators like the servo motor and buzzer. It enables the dustbin to detect objects, identify waste types (organic or inorganic), and operate the lid autonomously. The code processes real-time sensor inputs, making decisions like opening the lid when waste is detected and closing it once the object is removed. It also communicates with a PC to send waste data for further analysis. Overall, the Arduino code makes the dustbin efficient, user-friendly, and capable of functioning without manual intervention.

## How It Works :

### STEP 1 : Object Detection (Ultrasonic Sensor)

The ultrasonic sensor measures the distance to objects in front of the dustbin. If an object is detected within 30 cm, the system considers that something is approaching the bin.

```
// 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 : Opening the Dustbin (Servo Motor)

Once the ultrasonic sensor detects an object, the servo motor moves the dustbin door to the open position (90°). This allows the user to place waste into the bin.

```
// 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 : Waste Detection (IR Sensor and Soil Moisture Sensor)

After the door is open, the IR sensor monitors for waste. Once waste is placed, it checks for the presence of moisture using the soil moisture sensor. If moisture is detected, the system assumes the waste is organic.

```
// 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 : Moisture Detection and Buzzer/LED Alerts

If moisture is detected, the system classifies the waste as organic, turns on the buzzer, and logs the moisture level. If no moisture is found, the waste is considered dry (inorganic), and the LED is turned on to indicate dry waste.

```
// 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 : Waste Removal Detection (IR and Soil Moisture Sensor)

After the waste is placed, the system continuously monitors both the IR and soil moisture sensors. If the waste is removed from the bin (IR sensor is high, and moisture sensor detects no moisture), the system resets the states.

```

// 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 Removal from Detection Range

After placing the waste, if the object (e.g., user's hand) is removed from the detection range of the ultrasonic sensor, a 10-second countdown begins to close the dustbin.

```

// 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 : Closing the Dustbin (Servo Motor)

After 10 seconds without any new objects being detected, the servo motor moves the door to the closed position (0°), sealing the bin. If a new object is detected within this time, the countdown resets to keep the door open.

```
// 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
}
```



# Implementation of Random Forest

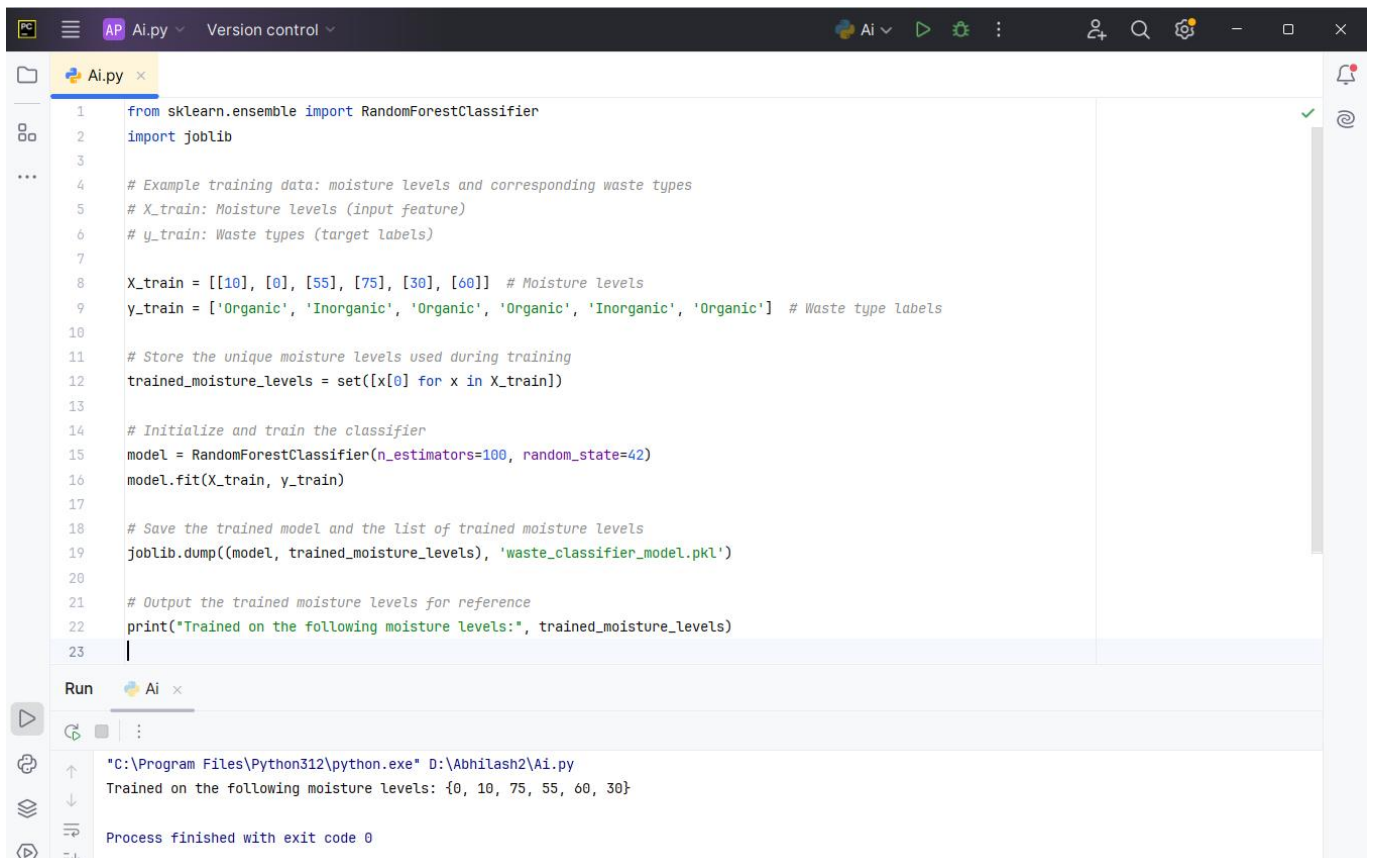
## Why Did We Use Random Forest in our Project

- Random Forest is an ensemble machine learning algorithm that builds multiple decision trees during training.
- Each decision tree makes a prediction, and the majority vote (or average in regression tasks) across all trees is taken as the final output.
- For our project, the input is the moisture level of the waste, and the output is whether the waste is classified as organic or inorganic.

## How It Works

### Training the Model :

- We have training data consisting of moisture levels and corresponding waste types. These values are used to train the model.
- The moisture levels (X\_train) are the input features, and the waste types (y\_train) are the target labels.
- A Random Forest Classifier is trained on this data to predict the waste type based on new moisture levels.



```
1 from sklearn.ensemble import RandomForestClassifier
2 import joblib
3
4 # Example training data: moisture levels and corresponding waste types
5 # X_train: Moisture levels (input feature)
6 # y_train: Waste types (target labels)
7
8 X_train = [[10], [0], [55], [75], [30], [60]] # Moisture levels
9 y_train = ['Organic', 'Inorganic', 'Organic', 'Organic', 'Inorganic', 'Organic'] # Waste type labels
10
11 # Store the unique moisture levels used during training
12 trained_moisture_levels = set([x[0] for x in X_train])
13
14 # Initialize and train the classifier
15 model = RandomForestClassifier(n_estimators=100, random_state=42)
16 model.fit(X_train, y_train)
17
18 # Save the trained model and the list of trained moisture levels
19 joblib.dump((model, trained_moisture_levels), 'waste_classifier_model.pkl')
20
21 # Output the trained moisture levels for reference
22 print("Trained on the following moisture levels:", trained_moisture_levels)
23 |
```

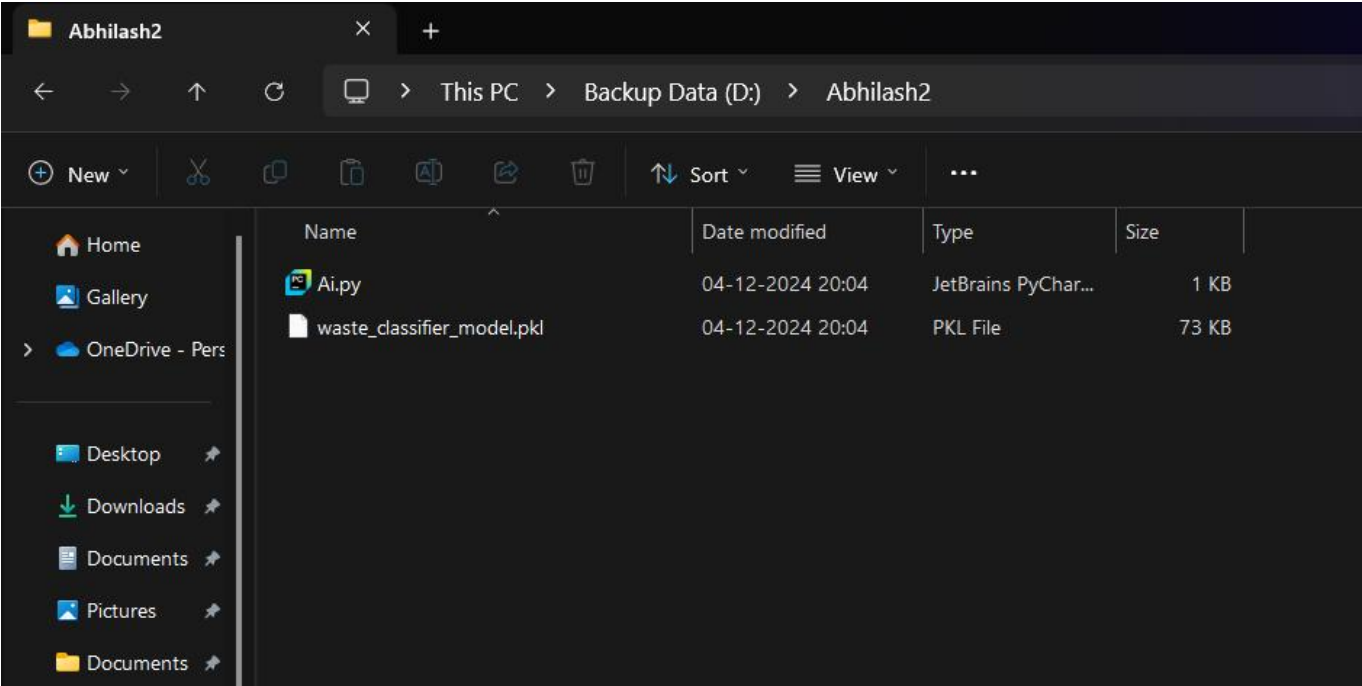
Run

```
"C:\Program Files\Python312\python.exe" D:\Abhilash2\Ai.py
Trained on the following moisture levels: {0, 10, 75, 55, 60, 30}

Process finished with exit code 0
```

## Saving the Model :

- After training, the model is saved using joblib so that it can be loaded and used later to classify new moisture levels in real-time as the Arduino provides them.



# Implementation of Machine Learning

## Why Did We Use ML in our Project :

We use Machine Learning (ML) in our smart dustbin project to enhance the efficiency and accuracy of waste classification. By leveraging ML, the system can analyze data from various sensors, such as the soil moisture sensor and IR sensor, to identify patterns and classify waste more intelligently. An ML model can learn to distinguish between organic and inorganic waste based on sensor readings and other inputs, improving detection accuracy.

The implementation of ML allows the dustbin to adapt and make more informed decisions, enabling continuous improvement in waste identification. This helps in better sorting of materials, leading to more effective waste management, reduced environmental impact, and optimized recycling processes. By utilizing ML, our project moves toward a more advanced and responsive waste management system.

## How It Works :

### Data Preprocessing :

- The Arduino sends data in the format `moisture_level`.
- This data is read via the serial connection and split into individual components.

```
11
12 # Set up the serial connection (ensure the correct COM port is used)
13 arduino = serial.Serial('COM3', 9600) # Change COM3 to the correct port for your Arduino
14
```

- For our ML model, the primary input is likely the `moisture_level`, which the model will use to predict whether the waste is organic or inorganic.

## Using the Pre-trained ML Model :

- You have loaded a pre-trained model (`waste_classifier_model.pkl`) using `joblib.load`. This model takes moisture levels as input and predicts the waste type (organic or inorganic).

```
8
9 # Load pre-trained ML model (replace with your model file path)
10 model, trained_moisture_levels = joblib.load('waste_classifier_model.pkl')
11
```

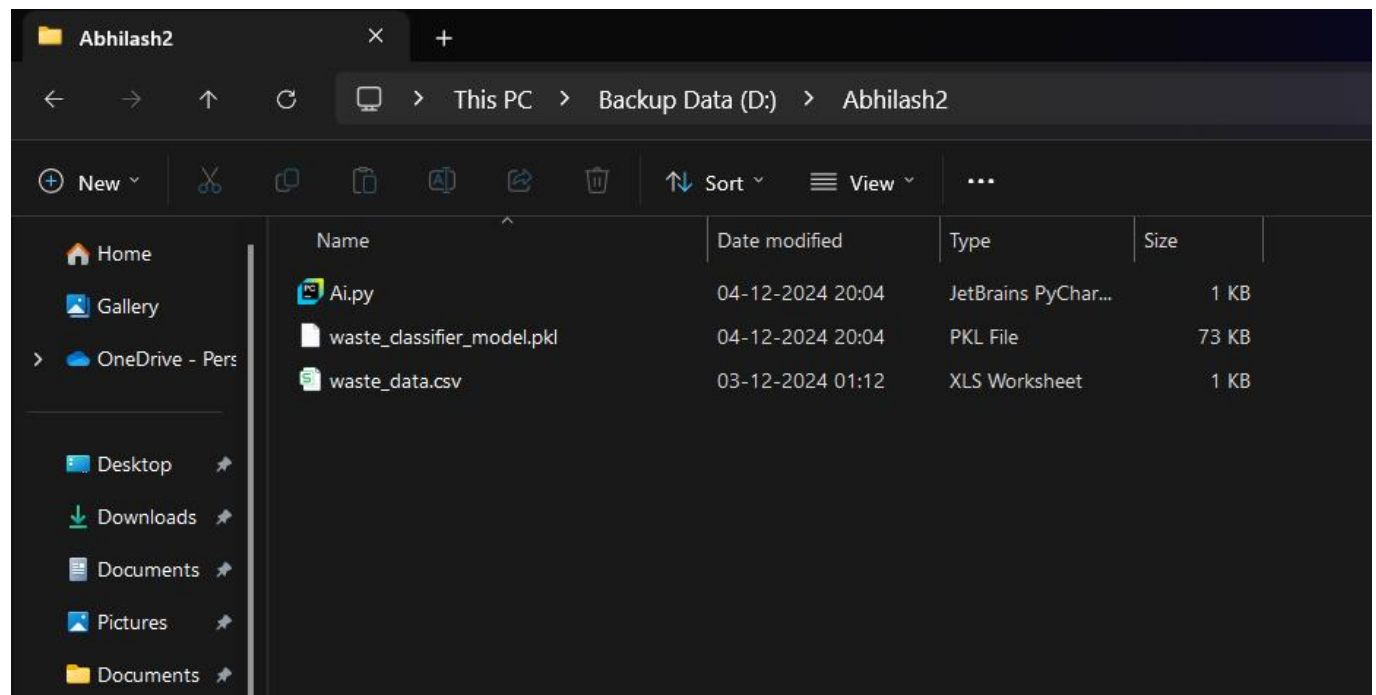


- We will pass the moisture level to the model and let it predict the waste type.

```
42
43 while True:
44     try:
45         # Read a line from the serial input
46         line = arduino.readline().decode('utf-8').strip()
47         print(f"Received: {line}") # Debugging print to see what is received
48
49
50     # Check if the line contains the expected comma-separated values
51     if ',' in line:
52         try:
53             # Split the line into organic, inorganic, and moisture values
54             organic, inorganic, moisture_level = line.split(',')
55
56             # Ensure that moisture level is properly converted to an integer
57             moisture_level = int(moisture_level)
58
59             # Use the ML model to predict waste type (organic/inorganic)
60             features = np.array([[moisture_level]])
61             predicted_waste_type = model.predict(features)[0] # 'Organic' or 'Inorganic'
62
63             # Get the current date and time
64             current_date = time.strftime('%Y-%m-%d') # Extract date
65             current_time = time.strftime('%H:%M:%S') # Extract time
66
67             # Determine the moisture match status using pandas DataFrame
68             moisture_status = check_moisture_status(moisture_level, data)
69
70             # Log the new data entry in the CSV file
71             csvwriter.writerow([current_date, current_time, predicted_waste_type, moisture_level, moisture_status])
72
73             # Update the DataFrame with the new entry
74             new_row = {
75                 'Date': current_date,
76                 'Time': current_time,
77                 'Predicted Waste Type': predicted_waste_type,
78                 'Moisture Level': moisture_level,
79                 'Moisture Match Status': moisture_status
80             }
81             data = pd.concat([data, pd.DataFrame([new_row])], ignore_index=True)
82
83             # Flush the file to ensure data is written immediately
84             csvfile.flush()
85
86             # Display the logged information in the console
87             print(f"Logged: {current_date} {current_time}, Predicted Waste Type: {predicted_waste_type}, "
88                   f"Moisture Level: {moisture_level}, Status: {moisture_status}")
89
90         except ValueError:
91             # Handle cases where the data cannot be split or converted correctly
92             print(f"Error: Unable to process the line: {line}")
93     else:
94         print(f"Unexpected data format: {line}")
95
96 except Exception as e:
97     print(f"Error: {e}")
```

### Logging Data :

- After predicting the waste type using the model, the predicted type, along with the moisture level, will be logged into a CSV file.



- The system also checks if the current moisture level matches any previously recorded levels (to track if the same waste type has been classified before).

[illegible]