WasteClassification.c

```
#include <Servo.h>


// Pin definitions   const int trigPin = 2;  // Trigger pin for
ultrasonic sensor   const int echoPin = 3;       // Echo pin for
ultrasonic sensor   const int irSensorPin = 4;   // Input pin for
IR sensor   const int ledPin = 9;        // Pin for LED   const int
buzzerPin = 10;    // Pin for buzzer   const int servoPin = 6;
// Pin for servo motor


Servo myServo;


void setup() {
Serial.begin(9600);       // Start serial communication
myServo.attach(servoPin); // Attach the servo to the pin
pinMode(ledPin, OUTPUT);  // Set LED pin as output
pinMode(buzzerPin, OUTPUT);// Set Buzzer pin as output
pinMode(trigPin, OUTPUT);  // Set trigger pin as output
pinMode(echoPin, INPUT);   // Set echo pin as input
pinMode(irSensorPin, INPUT); // Set IR sensor pin as input
}
```

```
void loop() {       if (Serial.available() > 0) {           int classification =
Serial.read() - '0'; // Read the classification from PC


       // Take action based on the classification result
if (classification == 1) {   myServo.write(90); // Move
to a specific position   digitalWrite(ledPin, HIGH); //
Turn on LED   tone(buzzerPin, 1000); // Activate
buzzer           } else if (classification == 2) {
myServo.write(0); // Move to another position
digitalWrite(ledPin, LOW); // Turn off LED
noTone(buzzerPin); // Deactivate buzzer
       } else {
myServo.write(0); // Default position   digitalWrite(ledPin,
LOW); // Turn off LED   noTone(buzzerPin); // Deactivate
buzzer
       }
   }


   // Ultrasonic sensor measurement
long duration, distance;


digitalWrite(trigPin, LOW);  // Set trigger pin low
delayMicroseconds(2);   digitalWrite(trigPin, HIGH); // Set
```

```
trigger pin high   delayMicroseconds(10);        // Keep high for 10
microseconds   digitalWrite(trigPin, LOW);  // Set trigger pin low

   duration = pulseIn(echoPin, HIGH); // Read the echo pin       distance
= (duration * 0.034) / 2; // Convert to distance in cm

   // Check IR sensor       if (digitalRead(irSensorPin) ==
HIGH) {          // If an object is detected by the IR sensor
myServo.write(180); // Move servo to another position
digitalWrite(ledPin, HIGH); // Turn on LED
tone(buzzerPin, 750); // Activate buzzer   delay(1000); //
Delay for 1 second
   } else {
      // No object detected by IR   myServo.write(0); //
Move servo to default position   digitalWrite(ledPin,
LOW); // Turn off LED   noTone(buzzerPin); //
Deactivate buzzer
   }

delay(100); // Delay for stability
}
```

WasteClassification.py

```
import cv2   import
```

```
numpy as np   import

tensorflow as tf


# Load pre-trained model   model =

tf.keras.models.load_model('path_to_your_model.h5')


def process_image(image_path):       # Load and preprocess image   img

= cv2.imread(image_path)   img = cv2.resize(img, (224, 224))  # Resize

to fit the model input size   img = img.astype('float32') / 255  # Normalize

img = np.expand_dims(img, axis=0)  # Add batch dimension        return

img


def classify_image(image_path):   img = process_image(image_path)       prediction

= model.predict(img)   class_index = np.argmax(prediction)  # Get the index of the

highest probability       return class_index


# Assuming you have a way to get the image from the camera

image_path = 'path_to_your_captured_image.jpg'   class_index =

classify_image(image_path)


# Send classification result to Arduino via USB   import serial

arduino = serial.Serial('COM3', 9600)  # Adjust 'COM3' as needed
```

```
arduino.write(str(class
_index).encode())
```