# PROJECT REPORT

# Submitted to

# NAANMUDHALVAN NIRAL THIRUVIZHA

# Theme: Solid Waste/ Bio-waste/ E-waste

**Problem Statement No.: 77**

**Problem Statement:**

How might we design a cost-effective, compact, and user-friendly device to help households easily segregate wet and dry waste, improving source segregation, recycling efficiency, and reducing environmental impact at the ward level?

# Mentor Name

# 

# Dr. T. Revathi, HoD/IT

# Team Members

# 

# Vijesh Pethuram K (9517202106056)

# Kanishkaran K (9517202106021)

# Raj Karna G (9517202106252)

# PROJECT DETAILS

**Theme**: Solid Waste/ Bio-waste/ E-waste

**Problem statement**: How might we design a cost-effective, compact, and user-friendly device to help households easily segregate wet and dry waste, improving source segregation, recycling efficiency, and reducing environmental impact at the ward level?

**College Code & College Name**: 9517 - Mepco Schlenk Engineering College, Sivakasi.

**Guide Name, Designation, Mobile No. & Email id**: Dr. T. Revathi, HoD/IT, 9442325608, trevathi@mepcoeng.ac.in.

**Student Team details**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sl.No. | Student Reg.  No. | Name of the  Student | Branch | Mobile No. | email id |
|  | 9517202106056 | Vijesh Pethuram K | Information Technology | 8637480351 | vijesh.moses@gmail.com |
|  | 9517202106021 | Kanishkaran K | Information Technology | 8489834410 | kalidassumithra425@gmail.com |
|  | 9517202106252 | Raj Karna G | Information Technology | 9075950243 | rajkarna65@gmail.com |

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE** **NUMBER** |
| 1 | Abstract | 4 |
| 2 | Problem Description | 5 |
| 3 | Introduction | 6 |
| 4 | Purpose | 7 |
| 5 | Scope | 8 |
| 6 | Software Requirements | 10 |
| 7 | Hardware Requirements | 11 |
| 8 | Circuit Diagram | 12 |
| 9 | Coding | 14 |
| 10 | Prototype | 25 |
| 11 | Conclusion | 26 |

1. **ABSTRACT**

In an era defined by automation and smart technology, this project addresses a critical but often overlooked issue in household waste management: efficient and accurate waste sorting for eco-friendly disposal. Our project merges advanced machine learning with accessible hardware, leveraging the capabilities of TinyML on a compact, customized setup. At the heart of the system is a microcontroller-based solution with a TinyML model optimized for on-device waste classification, enabling rapid, reliable, and cost-effective waste segregation. This waste sorting system uses a DC motor-driven conveyor belt and an IR sensor to detect incoming waste items, triggering the TinyML model to classify each item as biodegradable or non-biodegradable. Once classified, a stepper motor rotates a circular bin divided into two compartments, guiding the waste to the appropriate section for disposal. This automated response not only reduces the need for manual sorting but also prevents cross- contamination, enhancing recycling efficiency. Beyond its technical merit, this project exemplifies the seamless application of embedded AI in everyday life, offering a straightforward, sustainable solution to waste sorting that is both scalable and suitable for small households. Designed for ease of use and efficiency, this innovation aims to relieve users of the complexities of waste separation, fostering responsible disposal practices that contribute to environmental sustainability. Our system thus stands as a meaningful integration of technology into daily routines, offering a reliable and practical approach to sustainable living.

# PROBLEM DESCRIPTION

Waste segregation is essential for effective waste management and environmental sustainability. However, manual sorting of waste into biodegradable and non-biodegradable categories is time-consuming, labor-intensive, and prone to errors. This challenge is particularly relevant in households and small businesses, where limited time and resources can hinder proper sorting practices. Incorrect disposal and cross-contamination in waste streams can reduce recycling efficiency and lead to increased landfill waste, posing a serious environmental threat.

Traditional solutions to this problem often involve manual intervention, which is not only inconvenient but also inefficient, especially when large volumes of waste need sorting. Additionally, commercial waste management systems that utilize advanced technology for sorting are often expensive and not accessible to individual households. This gap between the need for efficient waste segregation and the lack of affordable automated solutions for small- scale use underscores the importance of developing an accessible, automated system.

Our project seeks to address this issue by creating a low-cost, automated waste segregation system that utilizes machine learning for real-time waste classification and sorting. By integrating a TinyML model with simple hardware components—such as an IR sensor, a stepper motor, and a DC motor-driven conveyor belt—this solution provides an efficient and accurate method for sorting waste at the household level. The project aims to enhance user convenience and reduce the environmental impact of improper waste disposal, making responsible waste management practices more attainable for everyday users.

# INTRODUCTION

Efficient waste management is a crucial step toward achieving sustainable environmental practices. Proper segregation of waste into biodegradable and non-biodegradable categories is essential for effective recycling and reducing landfill contributions. However, in many households and small businesses, manual waste sorting remains a time-consuming, labor- intensive process that is often neglected, leading to improper disposal and cross- contamination in waste streams. The lack of affordable, automated solutions for waste segregation at smaller scales further exacerbates the problem, making it difficult for individuals to contribute effectively to sustainable waste practices.

This project introduces an automated waste segregation system that leverages the power of machine learning and embedded hardware to classify and sort waste in real-time. Built around a compact, cost-effective design, the system integrates TinyML technology on a microcontroller, enabling rapid on-device classification of waste items as biodegradable or non-biodegradable. By incorporating a DC motor-driven conveyor belt, an IR sensor, and a stepper motor-controlled sorting bin, the system achieves a seamless workflow: the IR sensor detects waste on the conveyor belt, triggering the TinyML model to capture an image and classify it. Based on the classification result, the stepper motor rotates a circular bin divided into two compartments, directing the waste to the appropriate section.

This automated setup significantly reduces the need for manual sorting, offering a practical, low-cost solution for households and small establishments. It demonstrates the potential of TinyML and embedded systems in everyday applications, bridging the gap between advanced machine learning technologies and routine tasks. By making waste segregation more accessible and reliable, this project promotes sustainable disposal practices, fostering environmental responsibility at the household level.

# PURPOSE

The primary purpose of this project is to develop an automated waste segregation system that efficiently classifies and sorts waste into biodegradable and non-biodegradable categories. By leveraging machine learning through TinyML technology and integrating accessible hardware components, the project aims to achieve the following objectives:

**Enhance** **Waste** **Management** **Efficiency:** Streamline the waste sorting process to reduce the time and effort required for manual segregation, making it easier for households and small businesses to manage their waste responsibly.

**Promote** **Environmental** **Sustainability:** Contribute to sustainable waste management practices by ensuring accurate segregation of waste materials, thereby improving recycling rates and reducing landfill contributions.

**Increase** **Accessibility** **to** **Automation:** Provide an affordable and user-friendly automated solution for waste sorting that can be easily implemented in everyday settings, making advanced waste management technology accessible to a broader audience.

**Reduce** **Cross-Contamination:** Minimize the chances of contamination in recycling streams by accurately sorting waste, which is crucial for maintaining the integrity of recyclable materials.

**Raise** **Awareness** **and** **Responsibility:** Foster a sense of environmental responsibility among users by demonstrating the importance of proper waste segregation and encouraging sustainable practices in their daily lives.

Overall, the project aims to combine technology and practicality to create a reliable solution that simplifies waste management, making it both effective and sustainable for individual users.

# SCOPE

The scope of this project encompasses the design, development, and implementation of an automated waste segregation system, focusing on the following key areas:

## System Design and Integration:

* + Develop a hardware platform that includes a conveyor belt, IR sensor, stepper motor, and a microcontroller (utilizing TinyML).
  + Design a sorting mechanism with a circular bin divided into compartments for biodegradable and non-biodegradable waste.

## Machine Learning Implementation:

* + Implement a TinyML model capable of accurately classifying waste materials based on visual input.
  + Utilize an appropriate dataset to train the model for effective identification and sorting of various waste items.

## Automation and User Interaction:

* + Create an automated workflow where the IR sensor detects incoming waste, triggers the classification process, and activates the sorting mechanism.
  + Develop a user-friendly interface or notification system to inform users about the sorting status or any errors.

## Testing and Validation:

* + Conduct extensive testing of the system to ensure accuracy in waste classification and efficiency in sorting.
  + Validate the performance of the TinyML model in real-world scenarios, adjusting parameters and retraining as necessary to improve accuracy.

## Application and Adaptability:

* + Explore potential applications of the system in various settings, such as homes, small businesses, or community centers.
  + Assess the adaptability of the system to handle different types of waste materials, enhancing its utility in diverse environments.

## Future Enhancements:

* + Identify opportunities for future enhancements, such as integrating additional waste categories, improving the model's accuracy, or expanding the system’s capabilities with IoT features for remote monitoring.

Through this project, the aim is to create a robust, automated waste segregation solution that promotes sustainable waste management practices, ultimately contributing to environmental conservation and awareness.

# SOFTWARE REQUIREMENTS

## TensorFlow and TensorFlow Lite:

* + TensorFlow: For training the machine learning model.
  + TensorFlow Lite: For converting and deploying the model on Arduino.

## Development Environment:

* + Arduino IDE: For writing and uploading code to the Arduino board.
  + Python: Required for training the TensorFlow model and data preprocessing.

## Libraries:

* + Keras: High-level API for building and training the CNN model.
  + TensorFlow Lite Micro: To run TensorFlow Lite models on microcontrollers.

# HARDWARE REQUIREMENTS

## TinyML Model with Arduino Shield and Camera Module:

The TinyML model, integrated with an Arduino shield and a camera module, facilitates on-device inference for real-time waste classification. The Arduino shield provides a seamless connection between the microcontroller and the camera, enabling efficient data processing and classification of waste as either biodegradable or non- biodegradable. This compact setup ensures quick decision-making and simplifies the overall hardware architecture, making it well-suited for embedded applications in waste management.

## Stepper Motor with ULN2003 Driver:

The stepper motor, driven by a ULN2003 driver, provides precise control over the sorting mechanism's movement. This setup allows for accurate rotation to direct waste into the appropriate bins based on the classification results, ensuring effective waste separation.

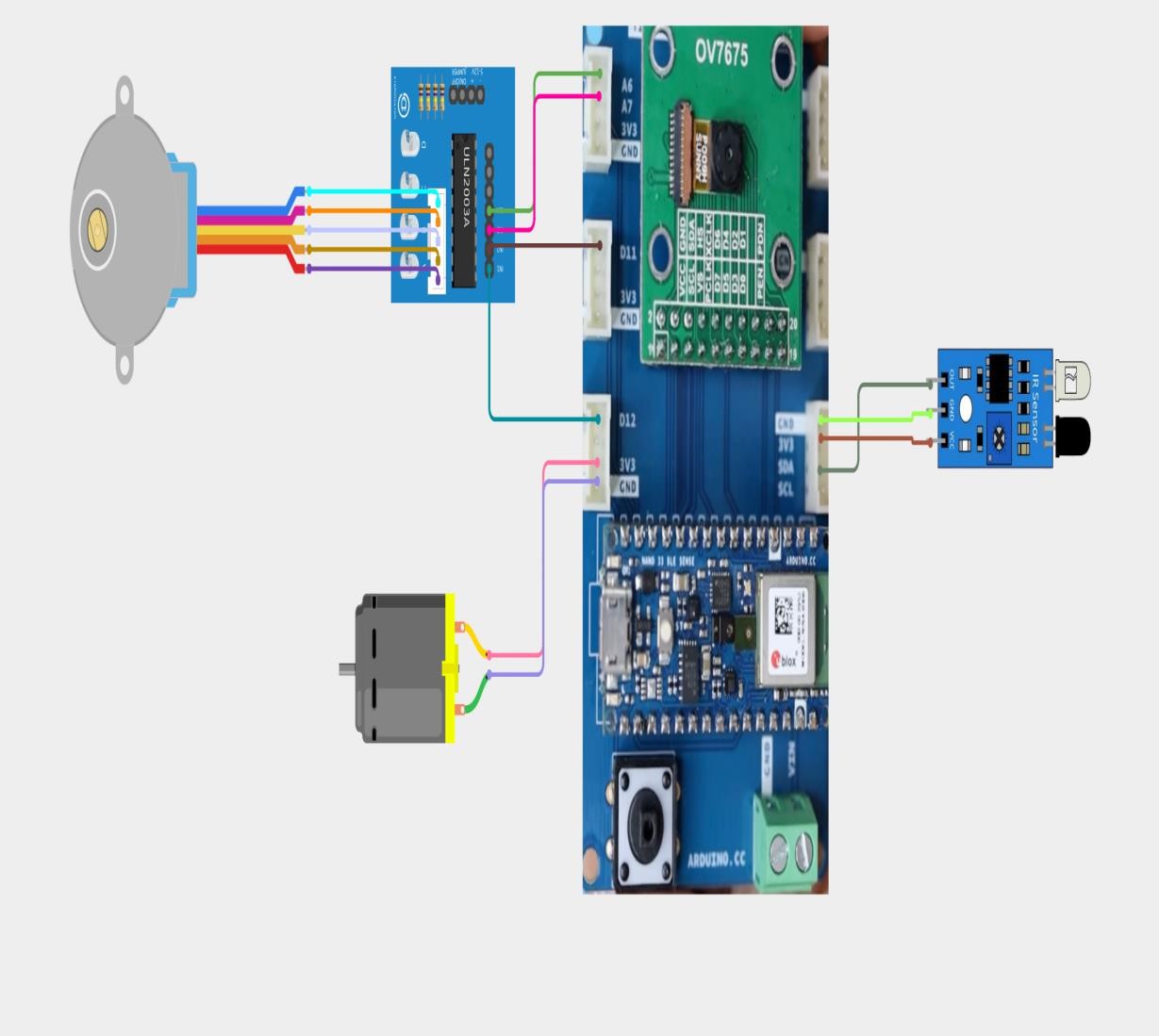
## IR Sensor:

The IR sensor plays a critical role in detecting the presence of waste on the conveyor belt. It triggers the classification process by signaling the TinyML model when an object passes through, ensuring timely and efficient waste identification.

## DC Motor:

The DC motor powers the conveyor belt, transporting waste to the classification area. Its reliable operation is crucial for maintaining the flow of waste, allowing the IR sensor and TinyML model to function seamlessly in the waste management process.

# CIRCUIT DIAGRAMS

****

Conveyor belt motor

Stepper Motor to control bin rotation

Motor Driver

IR Sensor

TinyML Board

Camera Module

Figure 1: Circuit Diagram

# 

Figure 2: Flow Diagram

# CODING

## ARDUINO CODE:

#include "object\_recognition.h" #include "mbed.h"

#include <Arduino\_OV767X.h> #include <TensorFlowLite.h>

#include <tensorflow/lite/micro/all\_ops\_resolver.h> #include <tensorflow/lite/micro/micro\_interpreter.h> #include <tensorflow/lite/micro/micro\_log.h> #include <tensorflow/lite/micro/system\_setup.h>

#include <tensorflow/lite/schema/schema\_generated.h>

#define IR\_SENSOR\_PIN A4 #define STEP\_PIN1 D11 #define STEP\_PIN2 D12 #define STEP\_PIN3 A6 #define STEP\_PIN4 A7

const int stepsFor180Degrees = 260; const int stepDelay = 2;

int sequence[8][4] = {

{HIGH, LOW, LOW, LOW},

{HIGH, HIGH, LOW, LOW},

{LOW, HIGH, LOW, LOW},

{LOW, HIGH, HIGH, LOW},

{LOW, LOW, HIGH, LOW},

{LOW, LOW, HIGH, HIGH},

{LOW, LOW, LOW, HIGH},

{HIGH, LOW, LOW, HIGH}

};

static const char \*label[] PROGMEM = {"Organic", "Recyclable", "unknown"}; static int32\_t bytes\_per\_frame;

static int32\_t bytes\_per\_pixel;

static bool debug\_application = false; String previousPrediction = "Recyclable";

static uint8\_t data[160 \* 120 \* 2]; static int32\_t height\_i = 0;

static int32\_t width\_i = 0; static int32\_t height\_o = 48; static int32\_t width\_o = 48; static float scale\_x = 0.0f; static float scale\_y = 0.0f; static int32\_t stride\_x = 0; static int32\_t stride\_y = 0;

template <typename T> inline T clamp\_0\_255(T x) {

return std::max(std::min(x, static\_cast<T>(255)), static\_cast<T>(0));

}

inline void ycbcr422\_rgb888(int32\_t Y, int32\_t Cb, int32\_t Cr, uint8\_t\* out) { Cr = Cr - 128;

Cb = Cb - 128;

out[0] = clamp\_0\_255((int32\_t)(Y + Cr + (Cr >> 2) + (Cr >> 3) + (Cr >> 5)));

out[1] = clamp\_0\_255((int32\_t)(Y - ((Cb >> 2) + (Cb >> 4) + (Cb >> 5)) - ((Cr >> 1) + (Cr >> 3) + (Cr >> 4)) + (Cr >> 5)));

out[2] = clamp\_0\_255((int32\_t)(Y + Cb + (Cb >> 1) + (Cb >> 2) + (Cb >> 6)));

}

inline uint8\_t bilinear(uint8\_t v00, uint8\_t v01, uint8\_t v10, uint8\_t v11, float xi\_f, float yi\_f) {

const float xi = (int32\_t)std::floor(xi\_f); const float yi = (int32\_t)std::floor(yi\_f); const float wx1 = (xi\_f - xi);

const float wx0 = (1.0f - wx1); const float wy1 = (yi\_f - yi); const float wy0 = (1.0f - wy1);

float res = 0;

res += (v00 \* wx0 \* wy0); res += (v01 \* wx1 \* wy0); res += (v10 \* wx0 \* wy1); res += (v11 \* wx1 \* wy1);

return clamp\_0\_255(res);

}

inline float rescale(float x, float scale, float offset) { return (x \* scale) - offset;

}

inline int8\_t quantize(float x, float scale, float zero\_point) { return (x / scale) + zero\_point;

}

static const tflite::Model\* tflu\_model = nullptr; static tflite::MicroInterpreter\* tflu\_interpreter = nullptr; static TfLiteTensor\* tflu\_i\_tensor = nullptr; static TfLiteTensor\* tflu\_o\_tensor = nullptr;

static constexpr int tensor\_arena\_size = 128000; static uint8\_t \*tensor\_arena = nullptr;

static float tflu\_scale = 0.0f; static int32\_t tflu\_zeropoint = 0;

void tflu\_initialization() { Serial.println("TFLu initialization - start");

tensor\_arena = new uint8\_t[tensor\_arena\_size];

tflu\_model = tflite::GetModel(model\_tflite);

if (tflu\_model->version() != TFLITE\_SCHEMA\_VERSION) { Serial.print(tflu\_model->version());

Serial.println(""); Serial.print(TFLITE\_SCHEMA\_VERSION); Serial.println("");

while(1);

}

tflite::AllOpsResolver tflu\_ops\_resolver;

static tflite::MicroInterpreter static\_interpreter( tflu\_model,

tflu\_ops\_resolver, tensor\_arena, tensor\_arena\_size);

tflu\_interpreter = &static\_interpreter;

tflu\_interpreter->AllocateTensors();

tflu\_i\_tensor = tflu\_interpreter->input(0); tflu\_o\_tensor = tflu\_interpreter->output(0);

const auto\* i\_quant = reinterpret\_cast<TfLiteAffineQuantization\*>(tflu\_i\_tensor-

>quantization.params);

tflu\_scale = i\_quant->scale->data[0]; tflu\_zeropoint = i\_quant->zero\_point->data[0];

Serial.println("TFLu initialization - completed");

}

void setup() { Serial.begin(115200); while (!Serial);

pinMode(IR\_SENSOR\_PIN, INPUT); pinMode(STEP\_PIN1, OUTPUT); pinMode(STEP\_PIN2, OUTPUT); pinMode(STEP\_PIN3, OUTPUT); pinMode(STEP\_PIN4, OUTPUT);

if (!Camera.begin(QQVGA, YUV422, 1)) { Serial.println("Failed to initialize camera!"); while (1);

}

bytes\_per\_pixel = Camera.bytesPerPixel();

bytes\_per\_frame = Camera.width() \* Camera.height() \* bytes\_per\_pixel; tflu\_initialization();

height\_i = Camera.height(); width\_i = Camera.height();

if(debug\_application) { height\_o = Camera.height(); width\_o = Camera.height(); Camera.testPattern();

}

stride\_x = bytes\_per\_pixel;

stride\_y = Camera.width() \* bytes\_per\_pixel;

scale\_x = (float)width\_i / (float)width\_o; scale\_y = (float)height\_i / (float)height\_o;

}

void stepMotor(int steps) { int stepCount = abs(steps);

int direction = steps > 0 ? 1 : -1;

for (int i = 0; i < stepCount; i++) { for (int j = 0; j < 8; j++) {

int index = (j \* direction + 8) % 8; digitalWrite(STEP\_PIN1, sequence[index][0]); digitalWrite(STEP\_PIN2, sequence[index][1]); digitalWrite(STEP\_PIN3, sequence[index][2]); digitalWrite(STEP\_PIN4, sequence[index][3]); delay(stepDelay);

}

}

}

void loop() {

int ir\_sensor\_value = digitalRead(IR\_SENSOR\_PIN);

if (ir\_sensor\_value == 0) {

Camera.readFrame(data); uint8\_t rgb888[3]; if(debug\_application) { Serial.println("<image>"); Serial.println(width\_o); Serial.println(height\_o);

}

int32\_t idx = 0;

for (int32\_t yo = 0; yo < height\_o; yo++) { const float yi\_f = (yo \* scale\_y);

const int32\_t yi = (int32\_t)std::floor(yi\_f);

for(int32\_t xo = 0; xo < width\_o; xo++) { const float xi\_f = (xo \* scale\_x);

const int32\_t xi = (int32\_t)std::floor(xi\_f);

const int32\_t x0 = xi; const int32\_t y0 = yi;

const int32\_t x1 = std::min(xi + 1, width\_i - 1); const int32\_t y1 = std::min(yi + 1, height\_i - 1);

const int32\_t off\_Y00 = x0 \* stride\_x + y0 \* stride\_y; const int32\_t off\_Y01 = x1 \* stride\_x + y0 \* stride\_y; const int32\_t off\_Y10 = x0 \* stride\_x + y1 \* stride\_y; const int32\_t off\_Y11 = x1 \* stride\_x + y1 \* stride\_y;

const int32\_t Y00 = data[off\_Y00]; const int32\_t Y01 = data[off\_Y01]; const int32\_t Y10 = data[off\_Y10]; const int32\_t Y11 = data[off\_Y11];

const int32\_t adj\_cr00 = xi % 2 == 0? 1 : -1;

const int32\_t adj\_cr01 = (xi + 1) % 2 == 0? 1 : -1;

const int32\_t adj\_cr10 = xi % 2 == 0? 1 : -1;

const int32\_t adj\_cr11 = (xi + 1) % 2 == 0? 1 : -1;

const int32\_t off\_Cb = (yi / 2) \* stride\_y + (xi / 2) \* stride\_x; const int32\_t Cb = data[off\_Cb + 1] \* adj\_cr00;

const int32\_t Cr = data[off\_Cb] \* adj\_cr00;

rgb888[0] = ycbcr422\_rgb888(Y00, Cb, Cr); rgb888[1] = ycbcr422\_rgb888(Y01, Cb, Cr); rgb888[2] = ycbcr422\_rgb888(Y10, Cb, Cr); rgb888[3] = ycbcr422\_rgb888(Y11, Cb, Cr);

tflu\_i\_tensor->data.int8[idx++] = quantize(rgb888[0], tflu\_scale, tflu\_zeropoint);

}

}

tflu\_interpreter->Invoke(); uint8\_t max\_index = 0;

float max\_value = tflu\_o\_tensor->data.f[0];

for (int i = 0; i < tflu\_o\_tensor->dims->data[1]; i++) { if (max\_value < tflu\_o\_tensor->data.f[i]) { max\_value = tflu\_o\_tensor->data.f[i];

max\_index = i;

}

}

String currentPrediction = (char\*)label[max\_index]; if (currentPrediction != previousPrediction) { previousPrediction = currentPrediction; stepMotor(stepsFor180Degrees);

}

} }

## MACHINE LEARNING CODE:

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

train\_datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2) train\_generator = train\_datagen.flow\_from\_directory(

'DATASET/TRAIN/',

target\_size=(224, 224), batch\_size=32, class\_mode='binary', subset='training'

)

validation\_generator = train\_datagen.flow\_from\_directory( 'DATASET/TEST/',

target\_size=(224, 224), batch\_size=32, class\_mode='binary', subset='validation'

)

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(224, 224, 3)),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D((2, 2)), Flatten(),

Dense(128, activation='relu'), Dropout(0.5),

Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy']) model.fit(train\_generator, validation\_data=validation\_generator, epochs=2)

converter = tf.lite.TFLiteConverter.from\_keras\_model(model) converter.optimizations = [tf.lite.Optimize.DEFAULT] tflite\_model = converter.convert()

with open('model.tflite', 'wb') as f: f.write(tflite\_model)

def convert\_to\_c\_array(file\_path): with open(file\_path, 'rb') as f:

data = f.read()

output\_file\_path = file\_path.replace('.tflite', '.h') with open(output\_file\_path, 'w') as f:

f.write('#ifndef MODEL\_H\n') f.write('#define MODEL\_H\n\n') f.write('unsigned char model[] = {\n')

for i in range(0, len(data), 12):

line = ', '.join(f'0x{byte:02x}' for byte in data[i:i+12]) + ',\n' f.write(line)

f.write('};\n\n')

f.write(f'unsigned int model\_len = {len(data)};\n') f.write('#endif\n')

print(f'Header file created: {output\_file\_path}')

convert\_to\_c\_array('model.tflite')

train\_datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2) train\_generator = train\_datagen.flow\_from\_directory(

'DATASET/TRAIN/',

target\_size=(64, 64), batch\_size=32, class\_mode='binary', subset='training'

)

validation\_generator = train\_datagen.flow\_from\_directory( 'DATASET/TEST/',

target\_size=(64, 64), batch\_size=32, class\_mode='binary', subset='validation'

)

model = Sequential([

Conv2D(16, (3, 3), activation='relu', input\_shape=(64, 64, 3)),

MaxPooling2D((2, 2)),

Conv2D(32, (3, 3), activation='relu'),

MaxPooling2D((2, 2)), Flatten(),

Dense(64, activation='relu'), Dropout(0.5),

Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy']) model.fit(train\_generator, validation\_data=validation\_generator, epochs=1)

converter = tf.lite.TFLiteConverter.from\_keras\_model(model) converter.optimizations = [tf.lite.Optimize.DEFAULT] tflite\_model = converter.convert()

with open('quantized\_model.tflite', 'wb') as f: f.write(tflite\_model)

def convert\_to\_c\_array(file\_path): with open(file\_path, 'rb') as f:

data = f.read()

output\_file\_path = file\_path.replace('.tflite', '.h') with open(output\_file\_path, 'w') as f:

f.write('#ifndef MODEL\_H\n') f.write('#define MODEL\_H\n\n') f.write('unsigned char model[] = {\n')

for i in range(0, len(data), 12):

line = ', '.join(f'0x{byte:02x}' for byte in data[i:i+12]) + ',\n' f.write(line)

f.write('};\n\n')

f.write(f'unsigned int model\_len = {len(data)};\n') f.write('#endif\n')

print(f'Header file created: {output\_file\_path}') convert\_to\_c\_array('quantized\_model.tflite')

# PROTOTYPE

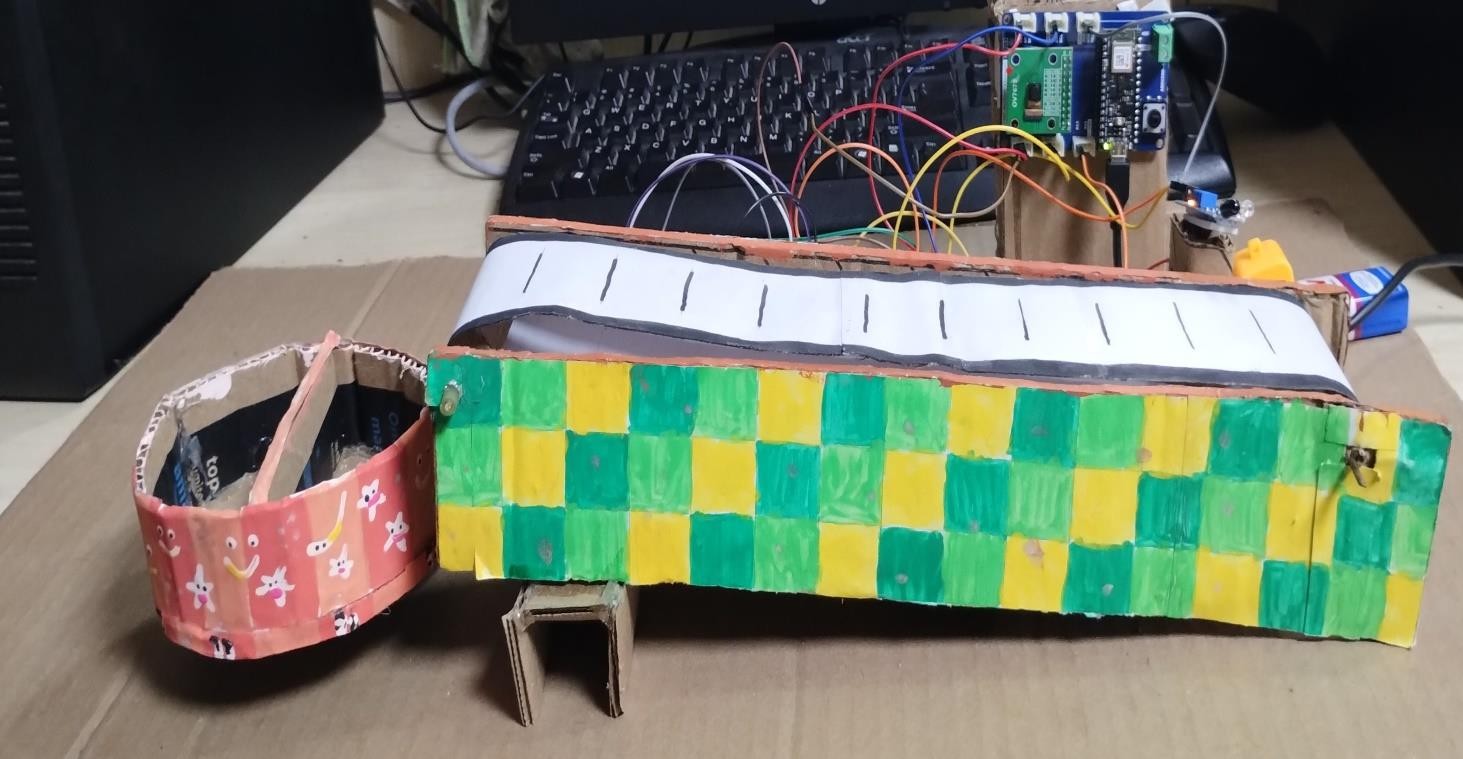
****

Figure 3: shows the side view of the Garbage Separator separating the Waste

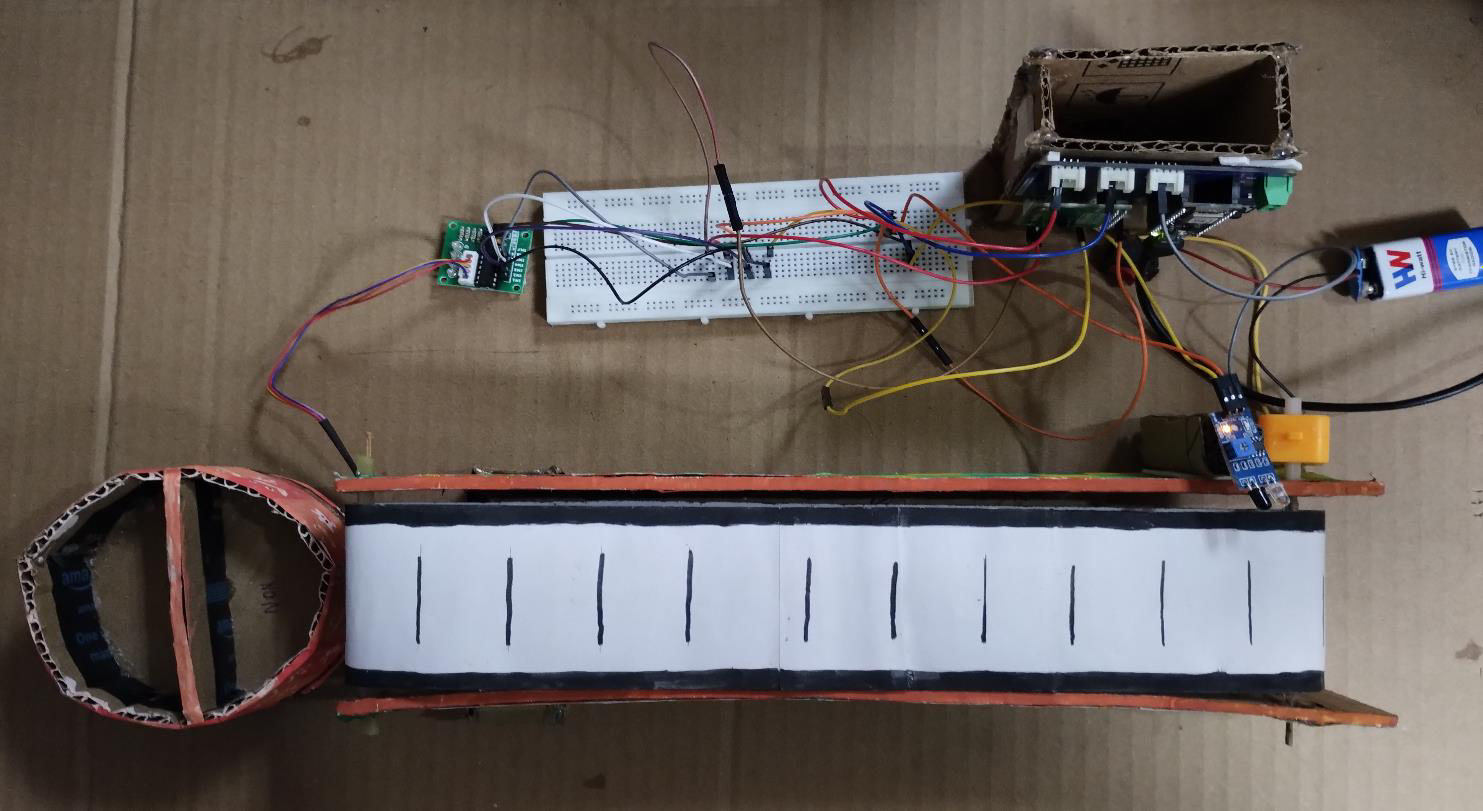


Figure 4: shows the top view of the Garbage Separator consisting of TinyMl and Conveyer Belt.

# CONCLUSION

The project successfully demonstrates an intelligent waste classification system that integrates an IR sensor, a stepper motor, and a TensorFlow Lite model for image recognition. By continuously capturing images and classifying waste as either "Organic" or "Recyclable," the system automates the waste separation process, enhancing efficiency in waste management. This automation promotes environmental sustainability by encouraging proper recycling practices and reducing contamination in recycling streams. The effective control of the stepper motor allows for precise physical separation of waste based on classification predictions, showcasing a practical approach to addressing environmental challenges.

For future enhancements, expanding the classification capabilities to include a wider range of waste categories, such as plastics and metals, could improve overall waste management strategies. Leveraging advanced deep learning models would further boost classification accuracy by utilizing larger datasets. Additionally, integrating smart bins equipped with sensors could optimize waste collection routes based on fill levels. Developing a user interface that educates the public on proper waste disposal practices would foster responsible behavior, while IoT capabilities for remote monitoring could streamline maintenance and collection needs. These enhancements could lead to more effective waste management systems that support sustainability efforts and promote community engagement.