**Project Report**

**On**

**Defect Detection System**

***Submitted in partial fulfillment of the***

***Requirements for the award of the degree***

***Of***

BACHELORS OF ENGINEERING

in

MECHATRONICS ENGINEERING

**Submitted By :- Faculty Supervisor :-**

**Jayanth.k (18BEM1027) Asst Prof. Inderpreet**

**Raghavendra.k (18BEM1050)**

**Garima Ittan (18BEM1051)**

**Adeel Beg (18BEM1059)**



**UNIVERSITY INSTITUTE OF ENGINEERING**

**CHANDIGARH UNIVERSITY**

**GHARUAN, MOHALI, PUNJAB, INDIA-140413**

**2021**

**CONTENTS**

**Sr.No : Title : Page No :**

1. Introduction  **2**
2. Literature Survey **3**
3. Methodology :- **5-6**
4. Working
5. Flowchart
6. Circuit Diagram
7. Material Used **7**
8. Components Description **7-9**
9. Work Done **9**
10. Approximate Cost **9**
11. Work Plan & Place **10**
12. Final Output Of Project **10**
13. References **10**
14. Project Pictures **11**
15. **Introduction :-**

Every Company needs Maintains Its Product quality as it is the most important thing for their brand value in the market if any defective or damaged product is bought by consumers brand image among public goes and it effects any company very high and sales go down and company looses its profits and its not good for companies future.

To Stop all this from happening every company make sure that product sent out of production line is not defective and so they adapt for system after producing the product products will be sent For Defect Detection After passing that test they will sent to packing if product is damaged it will be taken out of production line, For detecting the Defect In Product the process varies depending on the product as each item/product is different from one another.

As Defect Detection is part of Quality Control and Assurance to customer, This system is widely used in field of industrial automation so that process is fast and can be productive at the same time this helps in mass production industries where it is hard to employee Humans for Defect Identification Of high amount of products at a time and also it delays the packing process and turnover falls down, By using System Automation Into the production line this can make work faster and company’s output increases, Industrial Automation is good for any company because it increases productivity and no need for regular maintenance and it is one time investment to produce any product and can be easily variable to change product design by flexible manufacturing system as many industries use Defect Detection there are few ways to detect defect item based of product and precision of detecting it they are, By Using Laser Scanning System In this method product is scanned by a laser Scanner From every angle and compares with the data that is already programmed into system if there is any defect in product the system detects it by comparing the data and takes it out of production line above mentioned technique is mainly used in detecting the defects in products which are huge in size and also having different depths and heights at each side more of an irregular shaped ,

this types needed to compared from original dimensions of data that is of product and another process is using Deep Learning in our project we are going to use this method in our model integrated into ESP-32 Module and there as camera mounted to it for this process module is pretrained with images so that it can detect the defect product from good product in this we first input the model with multiple images of Defective products and when camera capture the image of each product in line it matches with the records after product image is processed if Defective/Good this system works as efficient as we give the multiple input data of images for detecting more efficiently

In Production line Camera captures each product in line and analyzed it with Deep Learning Program many companies use this method to detect Defects In Color of the product and detecting basic defects in product like cracks etc.. as this is low cost consumes less power same any other production line after identifying the Defective Piece product will be taken out before it is sent for packing the products. Computer vision technologies have been widely implemented in the defect detection. However, most of the existing detection methods generally require images with high quality, and they can only process code characters on simple backgrounds with high contrast. In this paper, a defect detection approach based on deep learning has been proposed to efficiently perform defect detection of code characters on complex backgrounds with a high accuracy. Specifically, image processing algorithms and data enhancement techniques were utilized to generate a large number of defect samples to construct a large data set featuring a balanced positive and negative sample ratio. The object detection network called BBE was build based on the core module of Efficient Net. Experimental results show that the mAP of the model and the accuracy reach 0.9961 and 0.9985, respectively. Individual character detection results were screened by setting relevant quality inspection standards to evaluate the overall quality of the code characters, the results of which have verified the effectiveness of the proposed method for industrial production. Its accuracy and speed are high with high robustness and transferability to other similar defect detection tasks. To the best of our knowledge, this report describes the first time that the BBE has been applied to defect inspections for real plastic container industry.

Currently, optical and machine vision technologies are intensively used in the inkjet inspection machine to detect and recognize the code characters for the product quality check. These methods feature fast speeds, low cost and high accuracy. The commonly used defect detection methods in inkjet inspection machines are generally traditional machine vision methods. For these methods, the product's code image is firstly collected by the industrial camera, and single characters are located and segmented by image processing algorithms, such as binarization. Afterward, the segmented characters are sent to the recognizer for recognition. Then, the quality of the code characters is evaluated by the comparison between the recognized characters and the real characters.

Therefore, the defect detection of the code characters can be summarized as a category of text detection [[1](https://www.aimspress.com/article/doi/10.3934/mbe.2021189?viewType=HTML#b1)]. To date, it becomes a very popular research field, and mature text detection methods have been developed to get good detection results. Typically, a feature extraction model composed of Gabor filter and Sobel operator to extract the character region and the K-means algorithm was used to distinguish the character regions from the background regions . Then, a two-stage character segmentation framework has been developed with an accuracy of 99.92% in the vehicle license plate recognition, in which the AdaBoost algorithm was used to train a cascade classifier to locate key character regions and the position information of the key characters was utilized to predict the remaining characters [. Putro et al. proposed a real-time text detection method, in which the Sobel operator was used to perform edge detection on the image and the K-means was used to extract text from the image background . To find text strings from arbitrary natural scene images, two methods have been proposed based on morphological operations with structure-based partitioning and morphology-based grouping

Due to the rapid development of artificial intelligence recently, various solutions have been provided for the industrial automation. Deep learning can extract effective deep features from images by virtue of the convolutional neural networks, which has already shown great potentiality in tasks such as text detection. Tan et al.  proposed a character recognition method based on the corner detection and the convolutional neural network, which were implemented to mark candidate regions of text and recognize the separated text, respectively.

Based on HCCR-Google Net, a simple and efficient character recognition network has been designed to construct a complete network architecture for oblique text correction, horizontal text line detection, character segmentation and character recognition . Also, an end-to-end deep neural network has been proposed, in which the local information, global structural features and contextual prompt information for region-oriented suggestions were utilized to identify text instances . Almost at the same time, another text detection method was developed to implement a custom faster RCNN and ResNet 50 convolutional neural network to locate the 320 × 240 text images and utilize a custom regression residual neural network to predict the text direction . A new framework, which can run on mobile or embedded systems, realized the text line recognition based on two independent artificial neural networks and dynamic programming . Cao et al. proposed a text detection method based on CTPN and presented an improved YOLO v3 network , the accuracy increased by 9.8% by changing the regression object from a single character to a fixed-width text and applying a stitching strategy to construct text lines based on the relation matrix. There is also a text structure feature extraction method for Chinese language based on the text structure component detector (TSCD) layer and residual network .

The objective of this paper is to develop an accurate and efficient detection system for the code characters by integrating the deep learning framework and the advanced object detection algorithm. The research subjects are the code characters on the beverage product package since the product lines in this field have become increasingly large with the development of economy and there is a strong need for the automatic quality inspection. In this paper, the code character pictures were directly from the industrial production line, in which the inkjet printer was used to print the codes of production date and the industrial camera was subsequently used to capture the area of codes. As shown in , the backgrounds of the collected code images are so complicated that it is very challenging for traditional detection methods to perform accurate character segmentation. To clearly demonstrate the key features of the proposed defect detection system and the difference from the traditional detection method, its flow chart has been depicted in . The overall detection system can be divided into three parts, namely the preprocessing of the code images, the design of the code defect detection algorithm and the design of the code defect detection software, as illustrated in .

In the experiments, a large number of image processing techniques were used to process the collected code images and construct the data set required. The convolutional neural network constructed the detection algorithm to extract the deep features of the code characters on complex backgrounds, and the detection model based on BBE has been trained with Python and Tensor flow to obtain the detector. Also, a friendly detect software has been developed based Python and PyQt5 to visually display the detection results and facilitate the operations for engineering personnel.

In Our Project we are going to show how an defect detection system works as product is made and comes out of production after all processing its turn to process through “Defect Detection System” This is the process we are going to show in our model where product will be passing by conveyor belt and undergoes the Defect Detection System Where an camera captures a picture of product and then analyze it Through Deep Learning we have trained the model with images of “Plate” As an example and our model detects whether the plate under line is “Broken/Good” when system detects that product is broken then it activates servo motor which is fixed by line to take out the Defective/Broken Product from line and then it is sent for packing the product

Here the image is captured and processed by one Module “ESP-32 CAM” it has processor which runs the program for analyzing the image which is trained using EDGE IMUPLSE, Conveyor Belt system is made from DC Motor Rotating the belt to move the product, after Passing the Defect Test then all the products are sent for packing this is done using Robotic Arm placing the Product On Cart in order and packed and further shipped.

Before capturing the image the product needs to placed exactly below camera for capturing the image this done by using the IR Sensor Placed by the side of conveyor belt which it triggers the DC Motor To Stop and Hence Conveyor Belt stops at a certain time that Product is under the Camera Module, This Triggering process is also programmed within the ESP-32 module that is used for Processing the defective once.

1. **Literature Survey :-**

Here Are few review papers which we have reviewed & gives the importance and need of this Project :-

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Sr.No | **Author(s)** | **Type/Title of study** |
| 1] | H. Schweitzer, W.B. James, and W. Feng | "Very fast template matching." |
| 2] | H. Zhang, S. Shen, and I.A. McAllister | "Camera blemish defects detection." |
| 3] | M.S. Kim | "Lens shading correction device and method in image sensor." |
| 4] | D.Y. Kim, Y.H. Noh | "Method and apparatus for compensating image sensor lens shading." |
| 5] | CW. Huang, | "System and method for detecting defects in camera modules." |

* In all Above Mentioned Projects were not using any Deep Learning method,



* They were just defect detection system or packing system not combined
* Few of them were using Pneumatic Actuator to remove the defective product
* They were not using ESP-32 Module
* They were using pneumatic Arm for picking & packing system.

1. **Methodology :-**
2. **Working :-**

Computer vision technologies have been widely implemented in the defect detection. However, most of the existing detection methods generally require images with high quality, and they can only process code characters on simple backgrounds with high contrast. In this paper, a defect detection approach based on deep learning has been proposed to efficiently perform defect detection of code characters on complex backgrounds with a high accuracy. Specifically, image processing algorithms and data enhancement techniques were utilized to generate a large number of defect samples to construct a large data set featuring a balanced positive and negative sample ratio. The object detection network called BBE was build based on the core module of Efficient Net. Experimental results show that the mAP of the model and the accuracy reach 0.9961 and 0.9985, respectively. Individual character detection results were screened by setting relevant quality inspection standards to evaluate the overall quality of the code characters, the results of which have verified the effectiveness of the proposed method for industrial production. Its accuracy and speed are high with high robustness and transferability to other similar defect detection tasks. To the best of our knowledge, this report describes the first time that the BBE has been applied to defect inspections for real plastic container industry.

Currently, optical and machine vision technologies are intensively used in the inkjet inspection machine to detect and recognize the code characters for the product quality check. These methods feature fast speeds, low cost and high accuracy. The commonly used defect detection methods in inkjet inspection machines are generally traditional machine vision methods. For these methods, the product's code image is firstly collected by the industrial camera, and single characters are located and segmented by image processing algorithms, such as binarization. Afterward, the segmented characters are sent to the recognizer for recognition. Then, the quality of the code characters is evaluated by the comparison between the recognized characters and the real characters.

Therefore, the defect detection of the code characters can be summarized as a category of text detection [[1](https://www.aimspress.com/article/doi/10.3934/mbe.2021189?viewType=HTML#b1)]. To date, it becomes a very popular research field, and mature text detection methods have been developed to get good detection results. Typically, a feature extraction model composed of Gabor filter and Sobel operator to extract the character region and the K-means algorithm was used to distinguish the character regions from the background regions . Then, a two-stage character segmentation framework has been developed with an accuracy of 99.92% in the vehicle license plate recognition, in which the AdaBoost algorithm was used to train a cascade classifier to locate key character regions and the position information of the key characters was utilized to predict the remaining characters [. Putro et al. proposed a real-time text detection method, in which the Sobel operator was used to perform edge detection on the image and the K-means was used to extract text from the image background . To find text strings from arbitrary natural scene images, two methods have been proposed based on morphological operations with structure-based partitioning and morphology-based grouping

Due to the rapid development of artificial intelligence recently, various solutions have been provided for the industrial automation. Deep learning can extract effective deep features from images by virtue of the convolutional neural networks, which has already shown great potentiality in tasks such as text detection. Tan et al.  proposed a character recognition method based on the corner detection and the convolutional neural network, which were implemented to mark candidate regions of text and recognize the separated text, respectively. Based on HCCR-Google Net, a simple and efficient character recognition network has been designed to construct a complete network architecture for oblique text correction, horizontal text line detection, character segmentation and character recognition . Also, an end-to-end deep neural network has been proposed, in which the local information, global structural features and contextual prompt information for region-oriented suggestions were utilized to identify text instances . Almost at the same time, another text detection method was developed to implement a custom faster RCNN and ResNet 50 convolutional neural network to locate the 320 × 240 text images and utilize a custom regression residual neural network to predict the text direction . A new framework, which can run on mobile or embedded systems, realized the text line recognition based on two independent artificial neural networks and dynamic programming . Cao et al. proposed a text detection method based on CTPN and presented an improved YOLO v3 network , the accuracy increased by 9.8% by changing the regression object from a single character to a fixed-width text and applying a stitching strategy to construct text lines based on the relation matrix. There is also a text structure feature extraction method for Chinese language based on the text structure component detector (TSCD) layer and residual network .

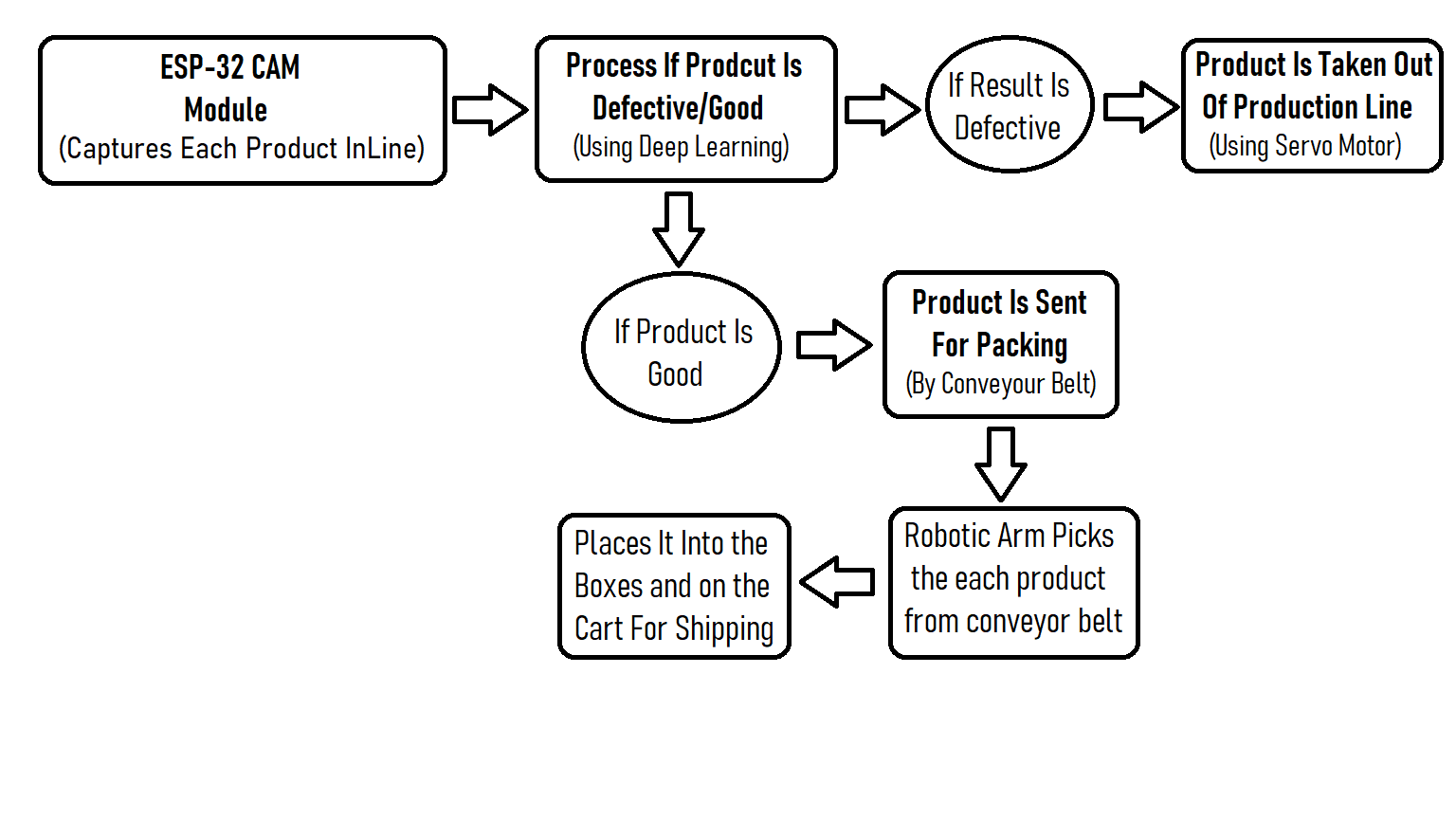
The objective of this paper is to develop an accurate and efficient detection system for the code characters by integrating the deep learning framework and the advanced object detection algorithm. The research subjects are the code characters on the beverage product package since the product lines in this field have become increasingly large with the development of economy and there is a strong need for the automatic quality inspection. In this paper, the code character pictures were directly from the industrial production line, in which the inkjet printer was used to print the codes of production date and the industrial camera was subsequently used to capture the area of codes. As shown in , the backgrounds of the collected code images are so complicated that it is very challenging for traditional detection methods to perform accurate character segmentation. To clearly demonstrate the key features of the proposed defect detection system and the difference from the traditional detection method, its flow chart has been depicted in . The overall detection system can be divided into three parts, namely the preprocessing of the code images, the design of the code defect detection algorithm and the design of the code defect detection software, as illustrated in . In the experiments, a large number of image processing techniques were used to process the collected code images and construct the data set required. The convolutional neural network constructed the detection algorithm to extract the deep features of the code characters on complex backgrounds, and the detection model based on BBE has been trained with Python and Tensor flow to obtain the detector. Also, a friendly detect software has been developed based Python and PyQt5 to visually display the detection results and facilitate the operations for engineering personnel.

In Our Project we are going to show how an defect detection system works as product is made and comes out of production after all processing its turn to process through “Defect Detection System” This is the process we are going to show in our model where product will be passing by conveyor belt and undergoes the Defect Detection System Where an camera captures a picture of product and then analyze it Through Deep Learning we have trained the model with images of “Plate” As an example and our model detects whether the plate under line is “Broken/Good” when system detects that product is broken then it activates servo motor which is fixed by line to take out the Defective/Broken Product from line and then it is sent for packing the product

Here the image is captured and processed by one Module “ESP-32 CAM” it has processor which runs the program for analyzing the image which is trained using EDGE IMUPLSE, Conveyor Belt system is made from DC Motor Rotating the belt to move the product, after Passing the Defect Test then all the products are sent for packing this is done using Robotic Arm placing the Product On Cart in order and packed and further shipped.

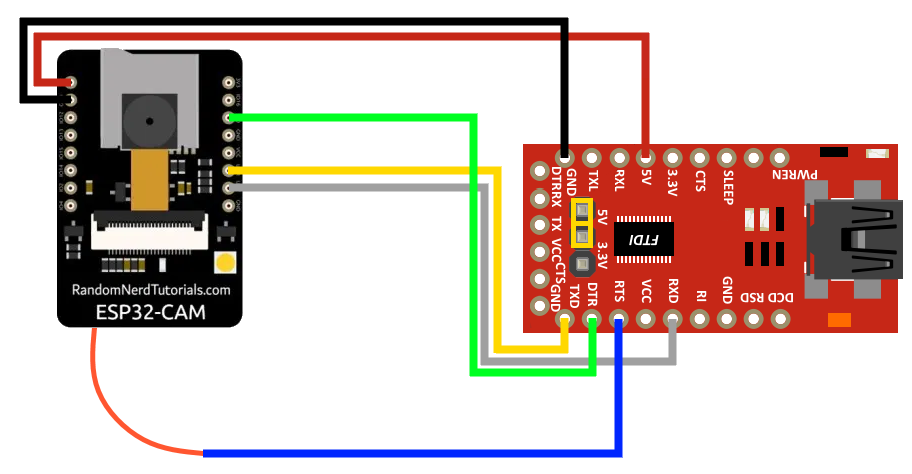
Before capturing the image the product needs to placed exactly below camera for capturing the image this done by using the IR Sensor Placed by the side of conveyor belt which it triggers the DC Motor To Stop and Hence Conveyor Belt stops at a certain time that Product is under the Camera Module, This Triggering process is also programmed within the ESP-32 module that is used for Processing the defective once.

**ii. Flow Chart :-**

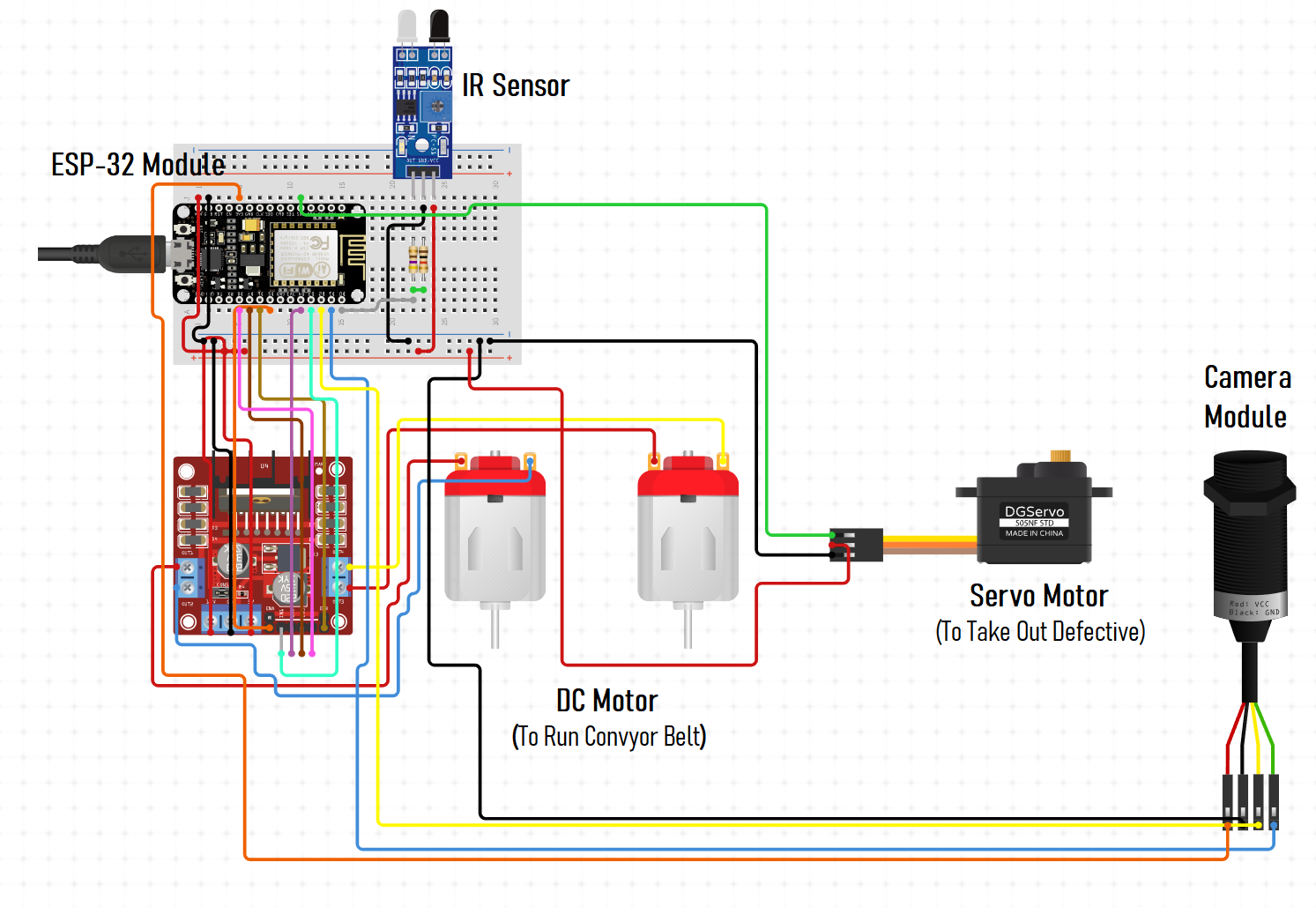
****

**Fig-1:Flow Chart**

1. **Circuit Diagram :-**



**Fig-2: ESP-32 – FTDI Programming**

****

**Fig-3: Schematic Diagram**

1. **Material Used :-**
2. ESP-32 CAM Module
3. TCRT5000 IR Sensor
4. Servo Motor
5. DC Motor
6. L293D Motor Driver
7. Connecting Wires
8. **Description Of Components :-**
9. **ESP-32 CAM Module :-**

The **ESP32**-**CAM** is a small size, low power consumption camera module based on **ESP32**. It comes with an OV2640 camera and provides onboard TF card slot. The **ESP32**-**CAM** can be widely used in intelligent IoT applications such as wireless video monitoring, WiFi image upload, QR identification, and so on. It Has Bluetooth Connectivity Also which make this b module more specs small size & low power consumption

The AI-Thinker ESP32-CAM module comes with an ESP32-S chip, a very small size OV2640 camera and a micro SD card slot. Micro SD card slot can be used to store images taken from the camera or to store files. This ESP32-CAM module can be widely used in various IoT applications. It can be used as a face detection system in offices, schools and other private areas and can also be used as wireless monitoring, QR wireless identification, and many other IoT applications.

The ESP32-CAM module can be programmed with ESP-IDF or with Arduino IDE. ESP32-CAM module also has several GPIO pins to connect the external hardware. The ESP32-CAM doesn’t have a USB connector, so to program the module you need an FTDI board.

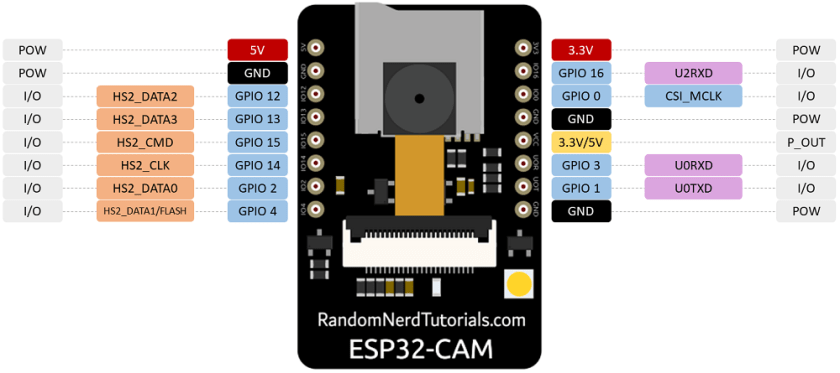


Features:

* The smallest 802.11b/g/n Wi-Fi BT SoC module
* Low power 32-bit CPU, can also serve the application processor
* Up to 160MHz clock speed, summary computing power up to 600 DMIPS
* Built-in 520 KB SRAM, external 4MPSRAM
* Supports UART/SPI/I2C/PWM/ADC/DAC
* Support OV2640 and OV7670 cameras, built-in flash lamp
* Support image Wi-Fi upload
* Support TF card
* Supports multiple sleep modes
* Embedded Lwip and FreeRTOS
* Supports STA/AP/STA+AP operation mode
* Support Smart Config/AirKiss technology
* Support for serial port local and remote firmware upgrades (FOTA)

Specifications:

* SPI Flash: Default 32Mbit
* RAM: 520KB SRAM+ 4M PSRAM
* Support TF Card: Max. 4G
* Support Interface: UART, SPI, I2C, PWM
* Image Output Format: JPEG, BMP, GRAYSCALE
* IO Port: 9
* Power Supply Range: 5V

****

**Fig-4: ESP-32 CAM Module Pinout**

1. **TCRT5000 :-**

The **TCRT5000** is an IR **sensor** unit. It has both a Photodiode and a Phototransistor coupled in its package. ... This **sensor** can be used to detect the presence of object or any other reflective surface in front it, also with some level of programming it can also calculate the distance of the object in front it. It runs on 3.3v input voltage

SPECIFICATIONS:

1. IR sensor with transistor output.

2. Operating Voltage:5V.

3. Diode forward Current: 60mA.

4. Output: Analog or digital data.

5. Transistor collector current: 100mA.

6. Operating temperature: -25°C to +85°C

The TCRT500 sensor is just a combination of a photodiode and a photo transistor. The photo diode is powered through a current limiting resistor of value 220R and the Transistor is also grounded through a 10K resistor as shown in the below picture. The transistor does not have a base pin because the biasing of the transistor is controlled by the amount of IR light it receives. So basically the IR light from the photo diode hits an object/surface and returns back to the photo transistor to bias it.

FEATURES

1. Package type: leaded.

2. Detector type: photo-transistor.

3. Peak operating distance: 2.5 mm.

4. Operating range within > 20 % relative.

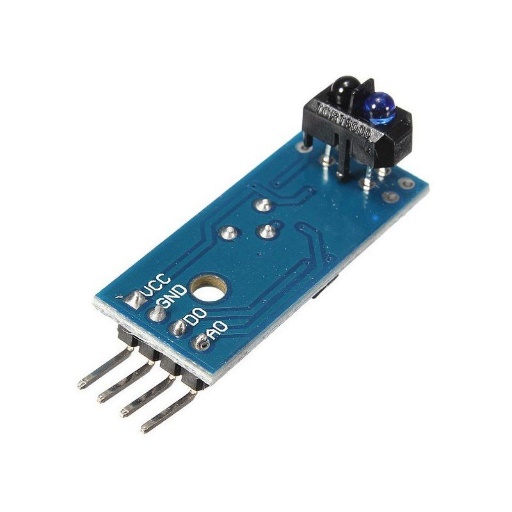
5. collector current: 0.2 mm to 15 mm.

6. Typical output current under test: IC = 1 mA.

7. Daylight blocking filter.

8. Emitter wavelength: 950 nm.

9. Lead (Pb)-free soldering released

****

**Fig-5: TCRT5000 IR Sensor**

1. **Servo Motor :-**

Micro **Servo Motor SG90** is a tiny and lightweight server **motor** with high output power. **Servo** can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any **servo** code, hardware or library to control these **servos**.

**Servo Motor Wire Configuration**

|  |  |  |
| --- | --- | --- |
| **Wire Number** | **Wire Colour** | **Description** |
| 1 | Brown | Ground wire connected to the ground of system |
| 2 | Red | Powers the motor typically +5V is used |
| 3 | Orange | PWM signal is given in through this wire to drive the motor |

**SG-90 Features**

* Operating Voltage is +5V typically
* Torque: 2.5kg/cm
* Operating speed is 0.1s/60°
* Gear Type: Plastic
* Rotation : 0°-180°
* Weight of motor : 9gm
* Package includes gear horns and screws

****

**Fig-6: Servo Motor SG90**

1. **DC Motor :-**

A direct current (DC) motor is a type of electric machine that converts electrical energy into mechanical energy. DC motors take electrical power through direct current, and convert this energy into mechanical rotation. The output torque and speed depends upon both the electrical input and the design of the motor.

DC Motor Characteristics:

* 1. Runs on DC power or AC line voltage with a rectifier.
  2. Operating speeds of 1,000 to 5,000 rpm.
  3. 60-75% efficiency rate.
  4. High starting torque.
  5. Low no-load speeds.



**Fig-7: DC Motor**

1. **L293D Motor Driver :-**

L293D IC is a typical Motor Driver IC which allows the DC motor to drive on any direction. This IC consists of 16-pins which are used to control a set of two DC motors instantaneously in any direction. It means, by using a L293D IC we can control two DC motors. As well, this IC can drive small and quiet big motors.

It is a basic motor driver integrated chip (IC) that enables us to drive a DC motor in either direction and also control the speed of the motor. The L293D is a 16 pin IC, with 8 pins on each side, allowing us to control the motor. It means that we can use a single L293D to run up to two DC motors. L293D consist of two H-bridge circuit. H-bridge is the simplest circuit for changing polarity across the load connected to it.

There are 2 OUTPUT pins, 2 INPUT pins, and 1 ENABLE pin for driving each motor. It is designed to drive inductive loads such as solenoids, relays, DC motors, and bipolar stepper motors, as well as other high-current/high-voltage loads.

IN1, IN2, and IN3, IN4 are input pins used for providing a control signal from the controller to run the motor in different directions.

If input logic at IN1, IN2 is (1,0) the motor rotates in one direction.

If input logic at IN1, IN2 is (0,1) the motor rotates in the other direction.

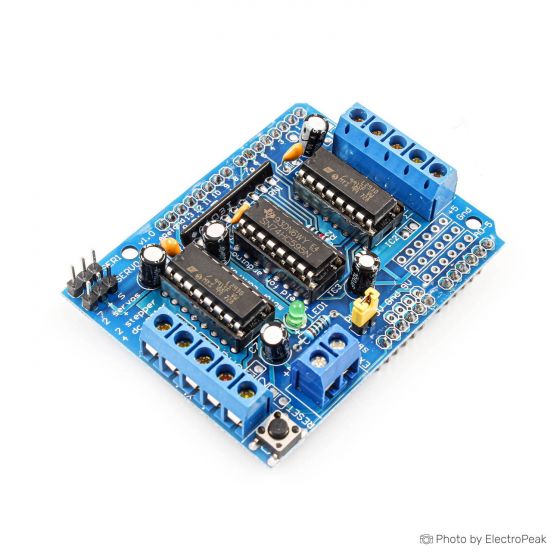
EN1 and EN2 are enable pins. Connect 5v DC to EN1 and EN2 pin to operate the motor at its normal speed

If speed control is needed, then give PWM output at pin EN1 and En2 from the microcontroller.

Power for the motor. If 12V DC gear motor is used then apply 12V.

Features :

* Wide Supply-Voltage Range: 4.5 V to 36 V
* Separate Input-Logic Supply
* Internal ESD Protection
* High-Noise-Immunity Inputs
* Output Current 600 mA Per Channel
* Peak Output Current 1.2 A Per Channel
* Output Clamp Diodes for Inductive Transient Suppression
* Operation Temperature 0°C to 70°C.
* Automatic thermal shutdown is available



**Fig-7: Servo Motor SG90**

**Source Code :-**

import os

import cv2

import numpy as np

################################

# --- Variables ---

################################

IMAGE\_SIZE = (500, 500)

# -- Threshold Details --

# \*THRESHOLD\_VALUE need to INCREASED if no contour detected,

# \*if there're INACCURATE CONTOUR NUMBERS the value need to DECREASED

THRESHOLD\_VALUE = 110

MAX\_VALUE = 255

# -- Invert Threshold Details --

INV\_THRESHOLD\_VALUE = 50

INV\_MAX\_VALUE = 255

# -- Canny Details --

THRESHOLD1 = 100

THRESHOLD2 = 70

# --contour properties--

CON\_COLOR = (0, 0, 255)

CON\_THICKNESS = 1

# -- Image Stack properties--

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

GREEN = (0, 255, 0)

RED = (0, 0, 255)

STACK\_IMG\_SIZE = (200, 200)

################################

while True:

files = os.listdir('images')

print("======================================")

print("= Available Images =")

print("======================================")

for i in files:

print('-> {}\t '.format(i), end='')

if files.index(i) % 3 == 0 and files.index(i) != 0:

print('\n')

print("\n======================================")

# need to select image name with the extension (ex: img1.jpeg)

file = input("Select a file from the directory(q- quit): ").strip()

# quit program

if file == 'q' or file == 'Q':

break

PATH = 'images/' + file

# Image Path

imageOri = cv2.imread(PATH)

try:

# converts to grayscale

image = cv2.cvtColor(imageOri, cv2.COLOR\_BGR2GRAY)

except:

print("Invalid Input! Please select correct file(ex: 'imgSample.jpg')")

else:

# resize image

image = cv2.resize(image, IMAGE\_SIZE)

imageOri = cv2.resize(imageOri, IMAGE\_SIZE)

image = cv2.GaussianBlur(image, (3, 3), 0)

# Threshold the image so that your black markings are black on a white background.

ret, thresh\_basic = cv2.threshold(image, THRESHOLD\_VALUE, MAX\_VALUE, cv2.THRESH\_BINARY)

# show thresholded image - DEBUGGING

cv2.imshow("Thresh basic", thresh\_basic)

#thresh\_addapt = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 115, 1)

# cv2.imshow("Thresh Adapt", thresh\_addapt)

# Taking a matrix of size 5 as the kernel

kernel = np.ones((5, 5), np.uint8)

# Morphological operations-Erodes away the boundaries of foreground object

# Use morphology to clean up extraneous markings.

img\_erosion = cv2.erode(thresh\_basic, kernel, iterations=1)

#####################

# The invert the thresholded image,

# so that the black markings are white on a black background and then find the external contours of those.

ret, thresh\_inv = cv2.threshold(img\_erosion, INV\_THRESHOLD\_VALUE, INV\_MAX\_VALUE, cv2.THRESH\_BINARY\_INV)

# show inverted threshold image - DEBUGGING

#cv2.imshow("INV", thresh\_inv)

#####################

# Find Canny edges

edged = cv2.Canny(img\_erosion, THRESHOLD1, THRESHOLD2)

# show canny edges - DEBUGGING

#cv2.imshow('Canny', edged)

#cv2.waitKey(0)

# Find Contours

# findContours alters the image

contours, hierarchy = cv2.findContours(thresh\_inv, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)

# ++++++++++++++++++

# -- Image Stack --

# ++++++++++++++++++

font = cv2.FONT\_HERSHEY\_SIMPLEX

imageRz = cv2.resize(image, STACK\_IMG\_SIZE)

thresh\_basicRz = cv2.resize(thresh\_basic, STACK\_IMG\_SIZE)

img\_erosionRz = cv2.resize(img\_erosion, STACK\_IMG\_SIZE)

thresh\_invRz = cv2.resize(thresh\_inv, STACK\_IMG\_SIZE)

edgedRz = cv2.resize(edged, STACK\_IMG\_SIZE)

imageRz = cv2.putText(imageRz, 'GrayScale', (5, 15), font, 0.5, WHITE, 1, cv2.LINE\_AA)

thresh\_basicRz = cv2.putText(thresh\_basicRz, 'ThresholdBasic', (5, 15), font,

0.5, WHITE, 1,cv2.LINE\_AA)

img\_erosionRz = cv2.putText(img\_erosionRz, 'Morphology-Erosion', (5, 15), font,

0.5, WHITE, 1, cv2.LINE\_AA)

thresh\_invRz = cv2.putText(thresh\_invRz, 'Threshold-mode INV', (5, 15), font,

0.5, BLACK, 1, cv2.LINE\_AA)

edgedRz = cv2.putText(edgedRz, 'Canny Edges', (5, 15), font, 0.5, WHITE, 1, cv2.LINE\_AA)

numpy\_horizontal\_concat = np.concatenate((imageRz, thresh\_basicRz, img\_erosionRz,

thresh\_invRz, edgedRz), axis=1)

cv2.imshow('Filtering...', numpy\_horizontal\_concat)

# +++++++

# get total contours

num\_of\_con = str(len(contours) - 1)

print("Number of Contours found = " + num\_of\_con)

if len(contours) > 1:

print('======================================')

print('= MARKINGS DETECTED =')

print('======================================\n\n')

# show original img

cv2.imshow('Original Image', imageOri)

# draw contours on original img

if int(num\_of\_con) != 0:

for i in range(int(num\_of\_con)):

highlighted\_img = cv2.drawContours(imageOri, contours, i, CON\_COLOR, CON\_THICKNESS)

highlighted\_img = cv2.putText(highlighted\_img, 'Approximately {} defect(s) detected'.

format(num\_of\_con), (5, 15),

font, 0.5, GREEN, 1, cv2.LINE\_AA)

else:

highlighted\_img = cv2.putText(imageOri, 'Unable to detect defects!',

(5, 15), font, 0.5, RED, 2, cv2.LINE\_AA)

# show markings highlighted img

cv2.imshow('Highlighted Defect', highlighted\_img)

# save image containing highlighted defect

cv2.imwrite('Output Images/{}\_DEFECTS\_HIGHLIGHTED.jpg'.format(file.split('.')[0]), highlighted\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

|  |  |  |
| --- | --- | --- |
| **Name** | **UID** | **Work Done** |
| Jayanth | 18BEM1027 | Component Selection, Hardware Making, Testing, Coding |
| Raghavendra | 18BEM1050 | Component Selection, Problem Solving, Component Purchase |
| Garima Ittan | 18BEM1051 | Report Making, Research & Reviewing |
| Adeel Beg | 18BEM1059 | Research, Designing |

1. **Work Done :-**
2. **Approximate Cost :-**

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Model Name** | **Approximate Price** |
| ESP Module | ESP-32 CAM Module | 1000 rs/- |
| IR Module | TCRT5000 | 150 rs/- |
| Servo Motor | SG90 | 200 rs/- |
| DC Motor | 100RPM | 150 rs/- |
| Motor Driver | L293D | 250 rs/- |
| 7805&Connecting Wires | - | 200 rs/- |

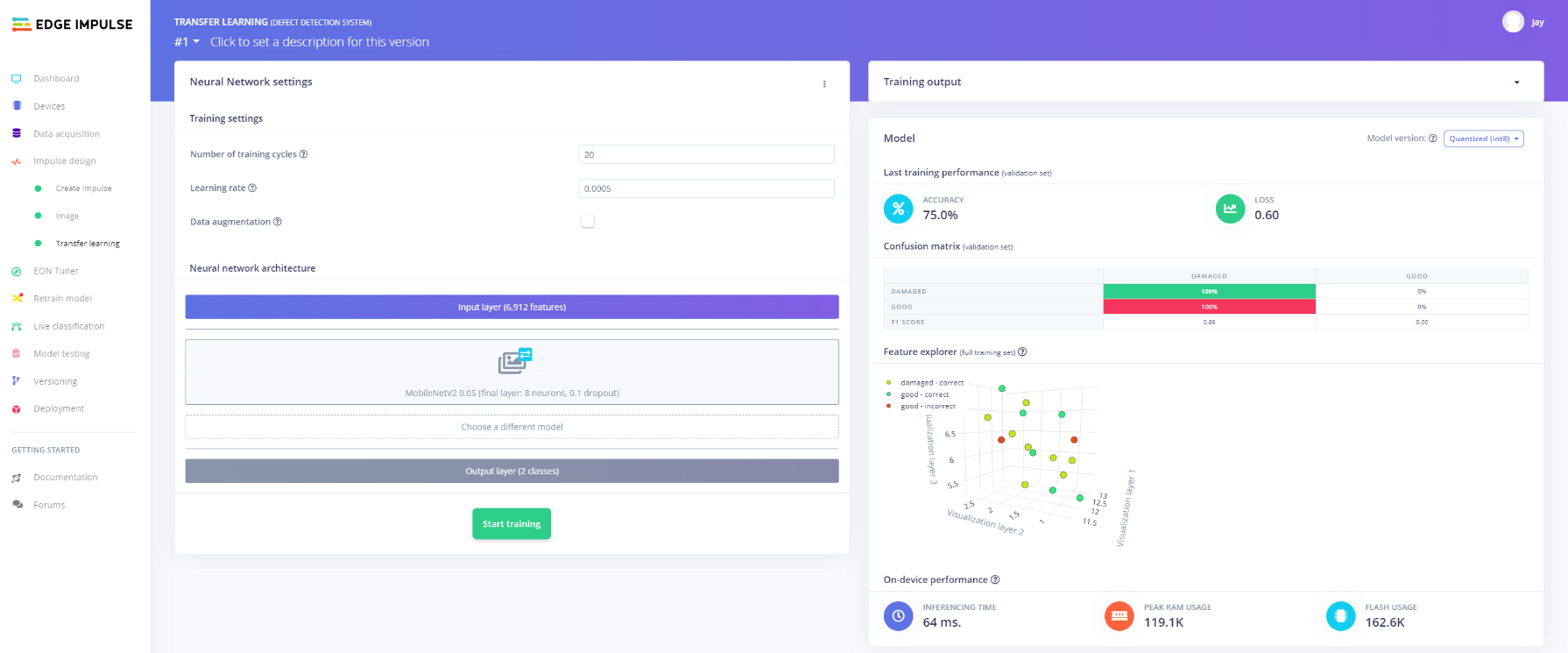
1. **Work Plan :-**

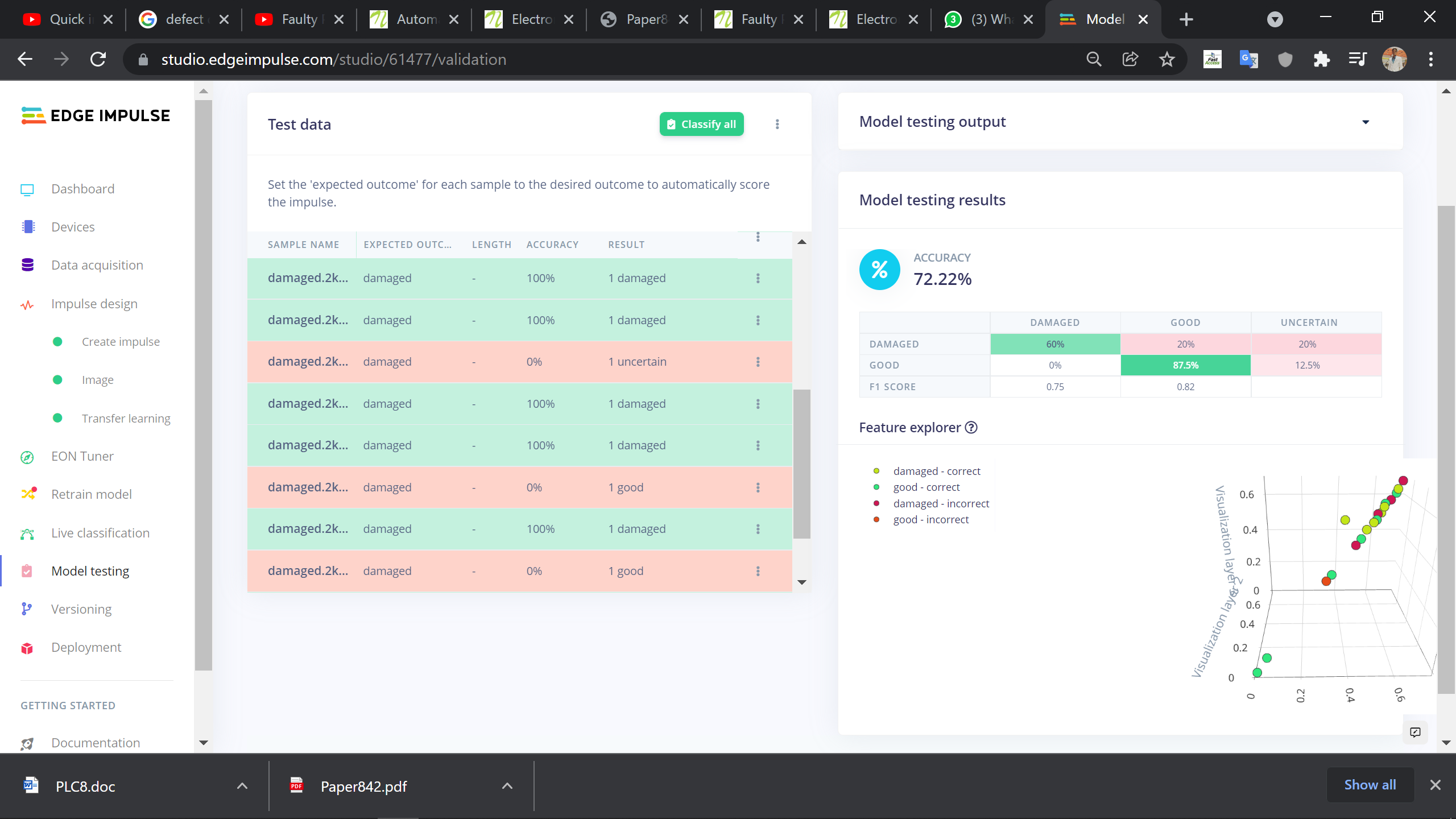
|  |  |  |
| --- | --- | --- |
| **Days** | **Work Done** | **Work Place** |
| 1-3 | Problem Identification & Selection | Home, Web |
| 4-7 | Designing , Listing Components & Placing Order | Home |
| 8-10 | Testing Working Of Components | Home |
| 10-17 | Preparing Coding & Testing Work | Home |
| 18-22 | Clearing Errors & Solving Problems | Home, Web |
| 23-26 | Redesigning & Checking Requirements | Home |
|  | Final Project |  |

1. **Final Output Of Project :-**

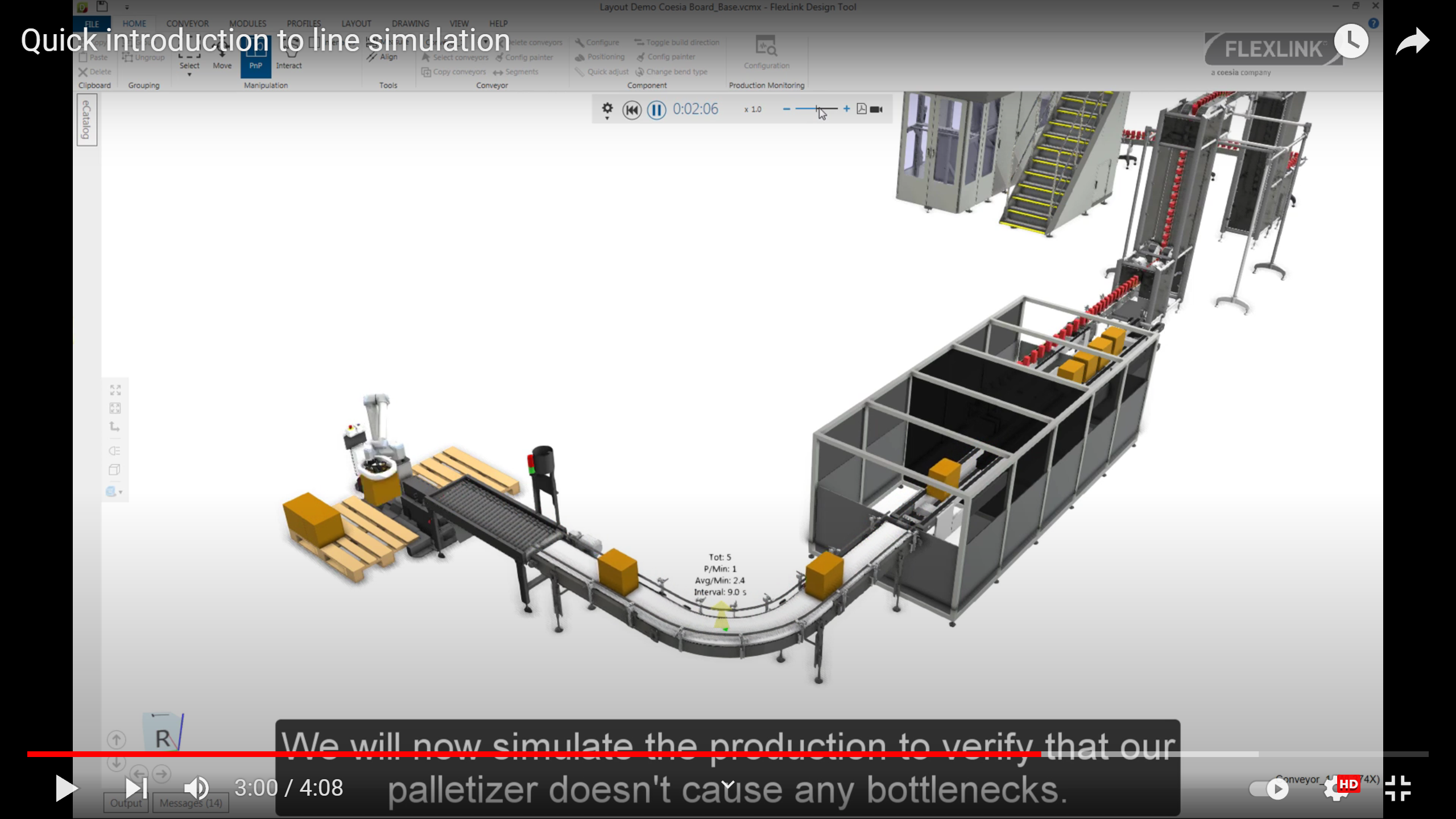
* Final output project is to improve the Quality Control In Industries using ML to make model of Defect Detection & Packing System By using Deep Learning and integrating it into the ESP-32 Module and create working model of Production Line running and taking out the Defective products & packing the good once so that Quality of product out of Factory is compromising companies image.

1. **References :-**
2. H. Schweitzer, W.B. James, and W. Feng, "Very fast template matching." European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2002.
3. H. Zhang, S. Shen, and I.A. McAllister, "Camera blemish defects detection." U.S. Patent No. 8,797,429. 5 Aug. 2014.
4. M.S. Kim, "Lens shading correction device and method in image sensor." U.S. Patent No. 7,834,925. 16 Nov. 2010.
5. D.Y. Kim, Y.H. Noh, "Method and apparatus for compensating image sensor lens shading." U.S. Patent No. 8,023,014. 20 Sep. 2011.
6. CW. Huang, "System and method for detecting defects in camera modules." U.S. Patent No. 7,974,458. 5 Jul. 2011.
7. R. Mehrotra, K.R. Namuduri, and N Ranganathan. "Gabor filter-based edge detection." Pattern recognition, p.1479-1494, December 1992.
8. Y. Sun, X. Li and J. Xiao, "A cascaded mura defect detection method based on mean shift and level set algorithm for active-matrix OLED display panel", J. Soc. Inf. Display, vol. 27, no. 1, pp. 13-20, Jan. 2019.
9. Y. Wang and D. D. L. Chung, "Capacitance-based defect detection and defect location determination for cement-based material", Mater. Struct., vol. 50, no. 6, Dec. 2017.
10. C.-H. Chan and G. K. H. Pang, "Fabric defect detection by Fourier analysis", IEEE Trans. Ind. Appl., vol. 36, no. 5, pp. 1267-1276, 2000.
11. L. H. Siew, R. M. Hodgson and E. J. Wood, "Texture measures for carpet wear assessment", IEEE Trans. Pattern Anal. Mach. Intell., vol. 10, no. 1, pp. 92-105, Jan. 1988.
12. D. L. Yuan, L. P. Lu and Y. M. Song, "Recent studies on automatic fabric defect detection technique", J. Zhengzhou Inst. Light Ind., vol. 20, no. 3, pp. 69-73, 2005.
13. Mr. Shete Yogesh Shreekrushna ,“Fabrication of Faulty Product Detection and Separation System”, RST Pvt. Ltd., Akluj.PaperID: IJSRDV8I60043,Published in: Volume:8,Issue:6,PublicationDate: 01/09/2020,Page(s): 25-30
14. **Project Pictures :-**

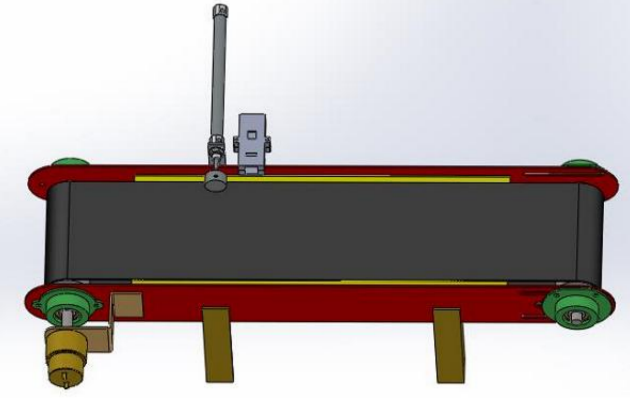
** Fig-8: Training Module**

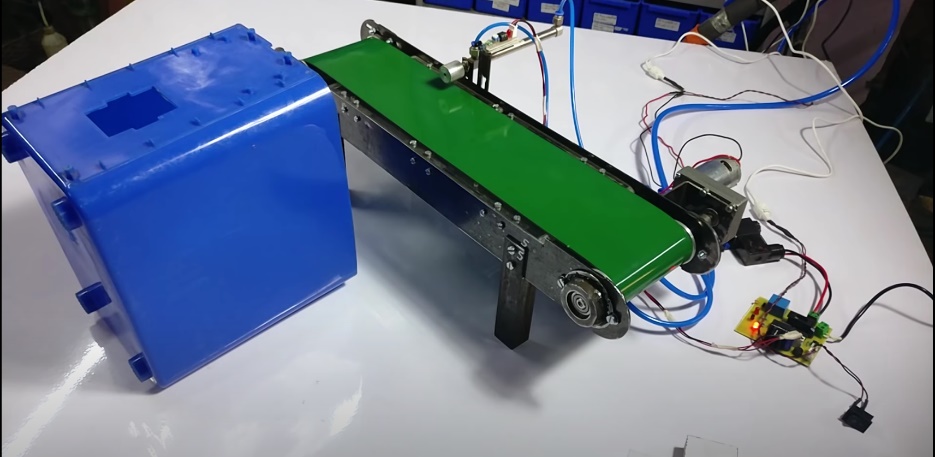


**Fig-9: Training Module**



**Fig :- Project Setup**

****

****

**Model Of Final Project**