Department of Electronics and Telecommunication Engineering

University of Moratuwa

EN2160 - Electronic Design Realization

Product Sorting Conveyor Belt
Design Documentation

Nawarathna D.M.G.B - 210415N

Contents

Introduction 2

Setting Up the Software Environment – 12/02/2024 to 18/02/2024 3

Installing Python and OpenCV 3

Software Implementation – 19/02/2024 to 03/03/2024 5

Hardware Testing and Selection – 04/03/2024 to 24/03/2024 7

ESP32 Cam Module 7

Power Supply 13

Schematic Designs – 25/03/2024 to 07/04/2024 14

PCB Design – 08/04/2024 to 14/04/2024 18

2D Design 18

3D Design 20

SolidWorks Design – 15/04/2024 to 28/04/2024 21

Final Steps and Conclusion 23

References 24

Appendix 25

Daily Log Entry 25

Code in Previous Submission (Using Arduino Libraries) 26

Introduction

This document presents the design and development process of the **"Product Sorting Conveyor Belt"** system. The primary objective of this project is to create an automated conveyor belt system capable of sorting products based on predefined criteria. This solution aims to enhance efficiency, accuracy, and speed in industrial and manufacturing environments, thereby reducing human error and labor costs. The project requires extensive software development, analyzing conceptual designs, careful hardware selections and well-planned PCB & Solidworks design. This documentation will discuss all the important details of each of the above aspects.

Setting Up the Software Environment – 12/02/2024 to 18/02/2024

Once the project idea was finalized, the software feasibility of the project has to be determined. After some thorough research it was clear that using openCV python library we could do image processing and calculate the dimensions of the products in the image sent. Therefore as the first step we should install python and openCV library to our server computer (in this case I used my own laptop).

Installing Python and OpenCV

1. First of all figure out whether you have already installed python on your computer. Enter the following prompt in the cmd.

   ```
   Python --version
   ```

   If it's already installed, it'll generate a message with the Python version available. In my computer I had already installed **Python 3.10.2** version. If you haven't already installed python, then follow the below link to install it.

   https://www.geeksforgeeks.org/download-and-install-python-3-latest-version/

2. Next, before installing OpenCV library we have to install pip which is the package manager of python. To see whether you have already installed pip, enter the following prompt in the cmd.

   ```
   pip --version
   ```

   If it's already installed, it'll generate a message with the pip version available. If it is not installed then follow the below link to install pip.

   https://www.geeksforgeeks.org/how-to-install-pip-on-windows/

3. Now that all the prerequisites are fulfilled we can install OpenCV library. It can be directly downloaded and install with the use of pip just by entering the following prompt in the cmd.

   ```
   pip install opencv-python
   ```

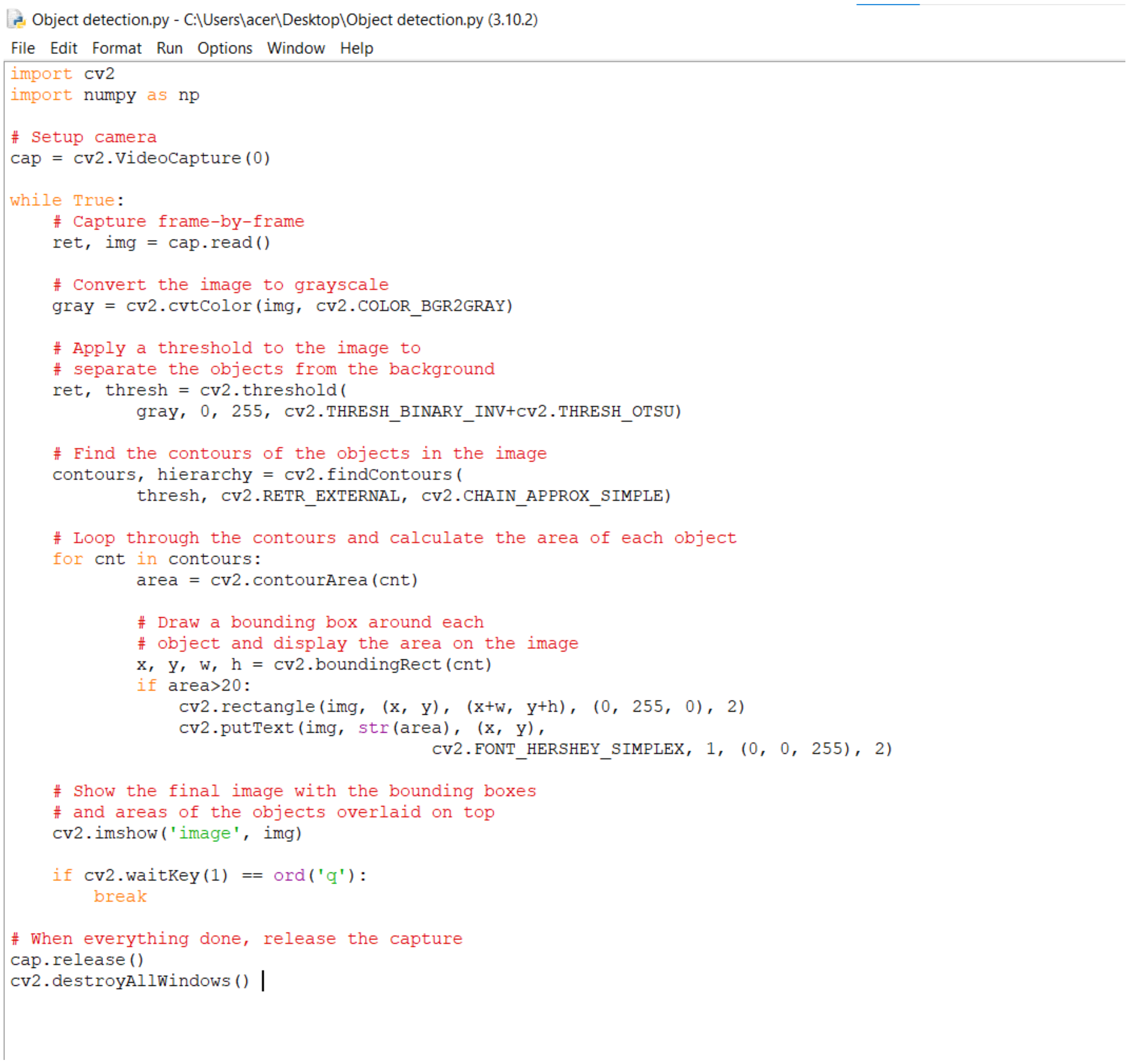
   After a while it'll finish installing and to check whether it's successfully installed we can do a version check using the following prompt.

   ```
   python
   >>>import cv2
   >>>print(cv2.__version__)
   ```

Next, we can move onto the software implementation.

Software Implementation – 19/02/2024 to 03/03/2024

After setting the software environment go to the python IDLE and use the following code.


Object detection.py - C:\Users\acer\Desktop\Object detection.py (3.10.2)
File Edit Format Run Options Window Help

```
import cv2
import numpy as np

# Setup camera
cap = cv2.VideoCapture(0)

while True:
    # Capture frame-by-frame
    ret, img = cap.read()

    # Convert the image to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Apply a threshold to the image to
    # separate the objects from the background
    ret, thresh = cv2.threshold(
            gray, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

    # Find the contours of the objects in the image
    contours, hierarchy = cv2.findContours(
            thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Loop through the contours and calculate the area of each object
    for cnt in contours:
            area = cv2.contourArea(cnt)

            # Draw a bounding box around each
            # object and display the area on the image
            x, y, w, h = cv2.boundingRect(cnt)
            if area>20:
                cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
                cv2.putText(img, str(area), (x, y),
                                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    # Show the final image with the bounding boxes
    # and areas of the objects overlaid on top
    cv2.imshow('image', img)

    if cv2.waitKey(1) == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Figure 1

Since, we haven’t implemented any hardware yet, this code uses the web camera to get the live feed (or for testing purposes we can just input an image saved in our computer as well).

OpenCV is a trained model which can separately identify objects. Above code uses the OpenCV library and creates contours around objects. Then using its functions we can get the co-ordinates of the objects **x & y** and also the dimensions of the object **width & height**.

Run the above code and aim your web camera to a certain object. You will get the output as follows.

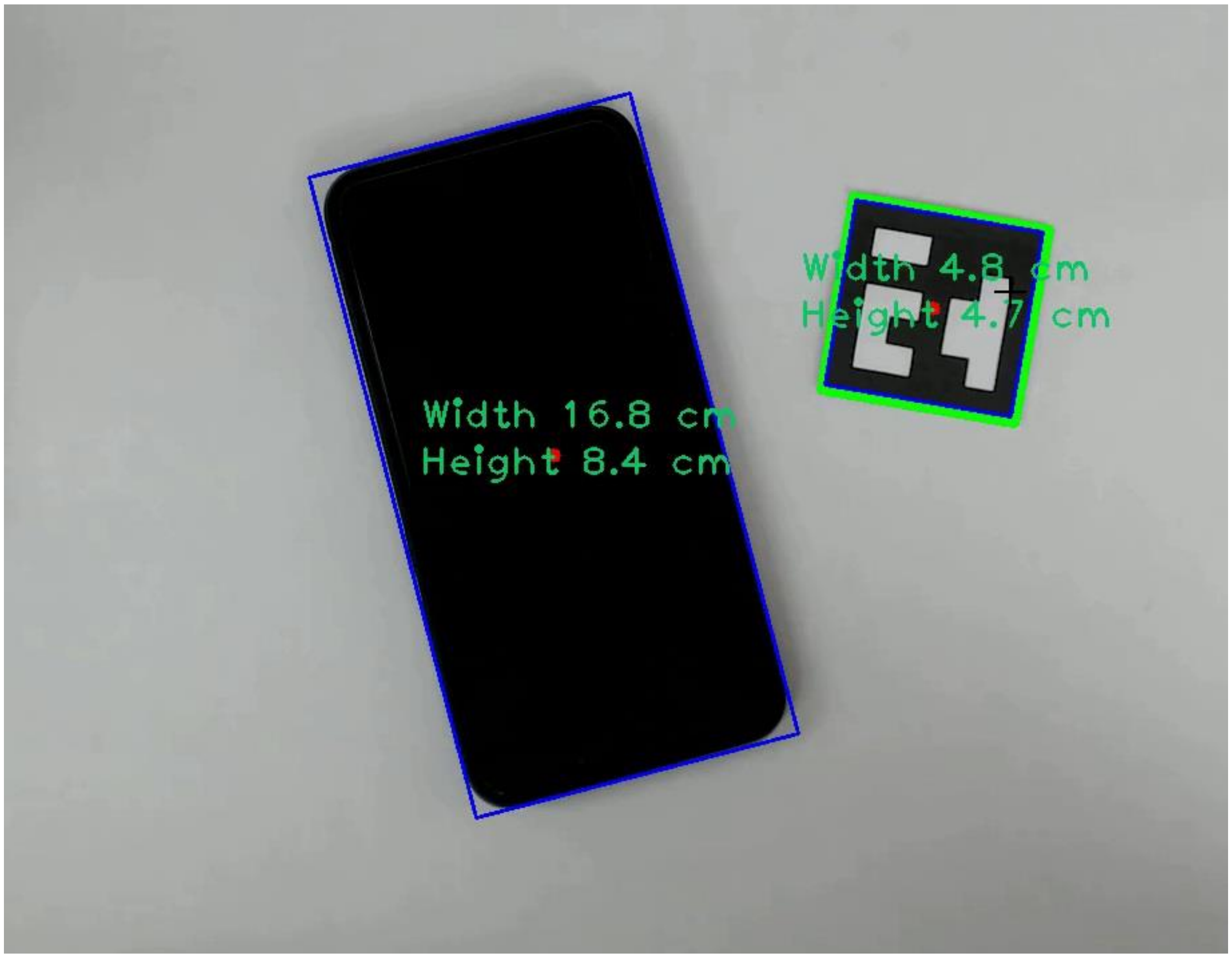


Figure 2

However, one disadvantage of this library is that depending on the height from object to the camera the readings will differ. But since we keep the camera at certain height in the product sorter, we can use a constant K to convert the measured readings to accurate measurements. We can adjust this constant through the software.

Hardware Testing and Selection – 04/03/2024 to 24/03/2024

There are four main hardware requirements for this project. That is the camera, micro-controller with Wi-Fi, stepper motor (for the arm) and a power supply unit. For the camera and the micro-controller we have chosen the ESP32 Cam Module for testing purposes (for the PCB design we will have to get the micro-controller and the camera separately).

ESP32 Cam Module

Advantages	**Disadvantages**
Cost-Effective	Limited Processing Power
Integrated Wi-Fi and Bluetooth	Low RAM
Versatility	Camera Quality
Compact Size	No Onboard USB Interface
Camera Support (OV2640)	Limited GPIO Availability
Development Support	Temperature Sensitivity
Programming Flexibility	

Table 1

However considering these pros and cons, and the requirements for the project, ESP32 Cam is a technically and financially feasible product to be included in the prototype.

Also since there is no onboard USB interface, we will have to use a separate FTDI module to upload the code. The connection between the ESP32 Cam module and the FTDI module is shown in the below image.

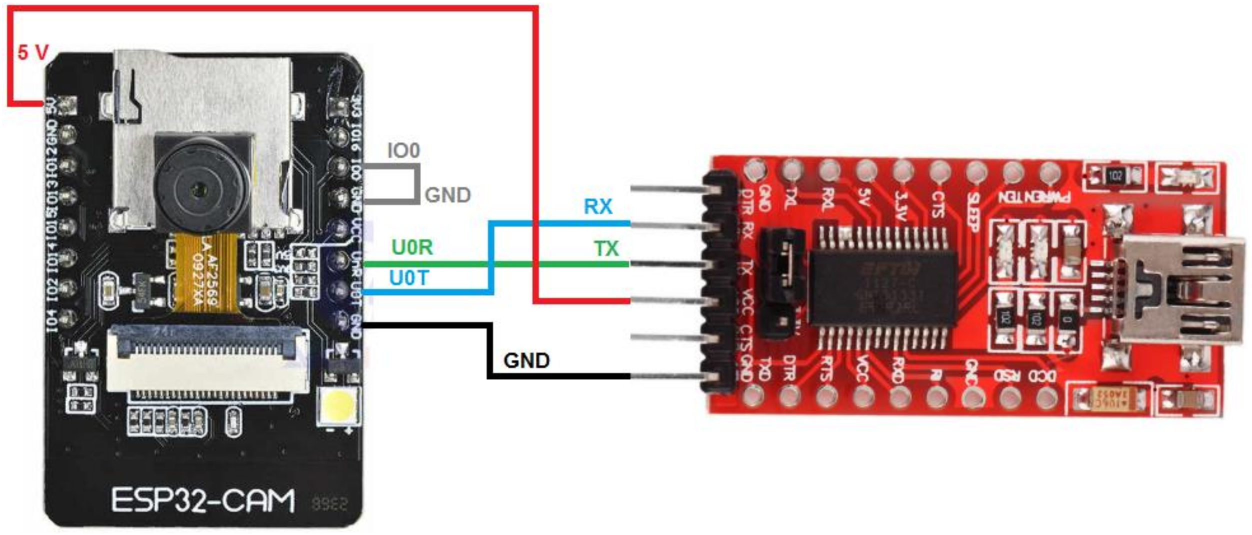
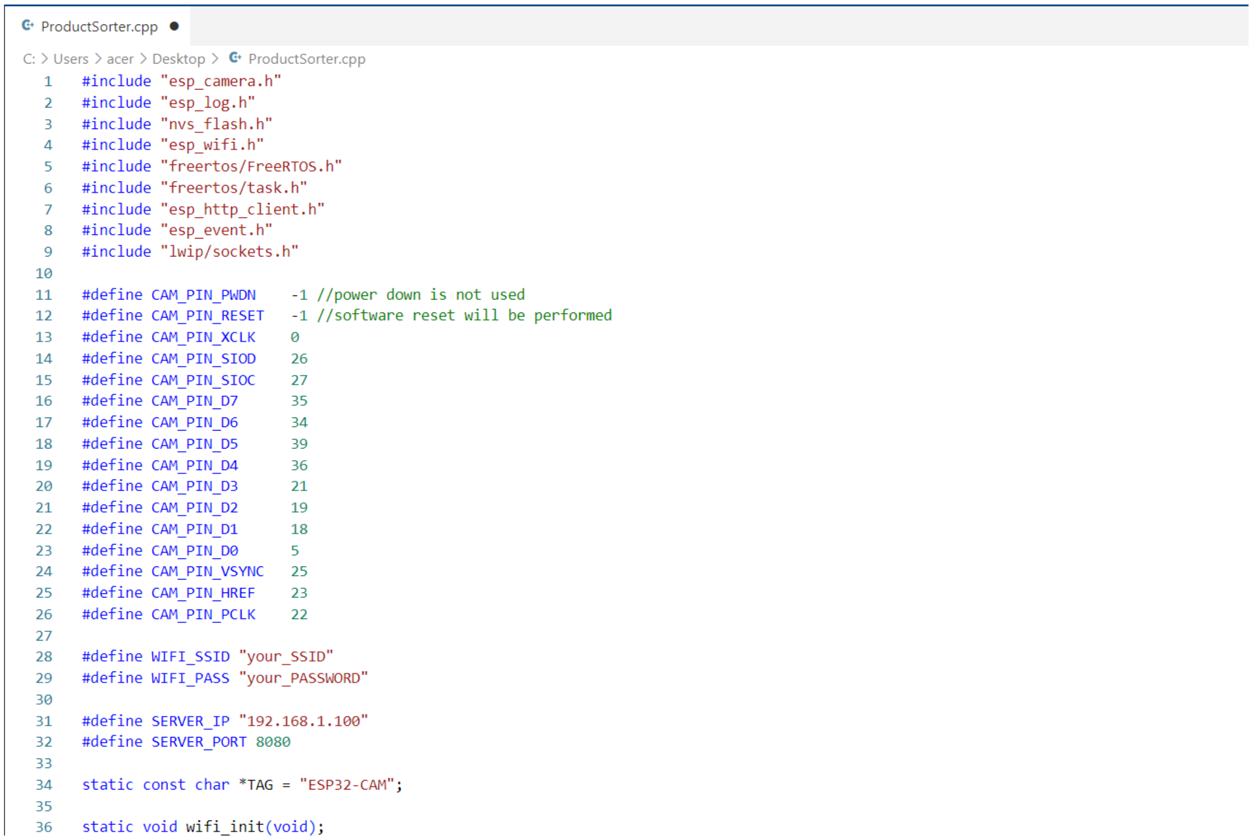
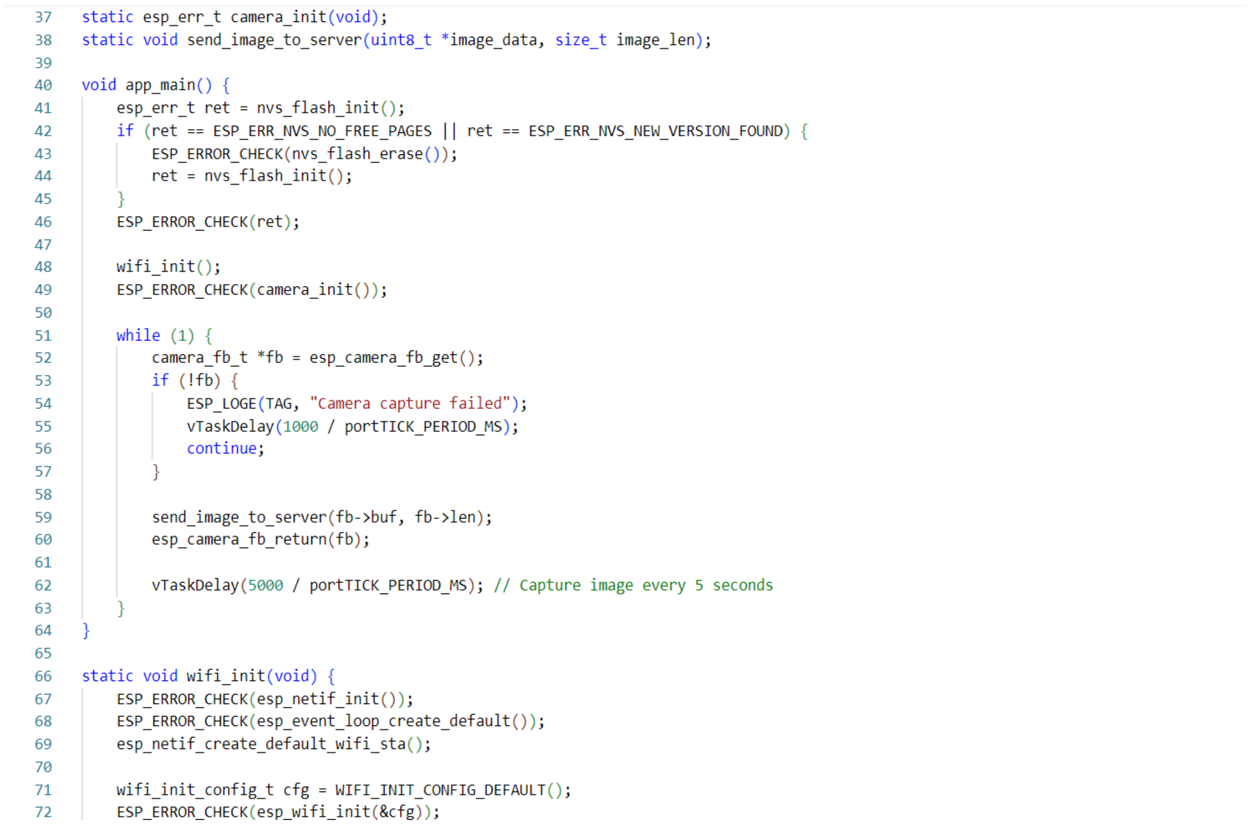


Figure 3

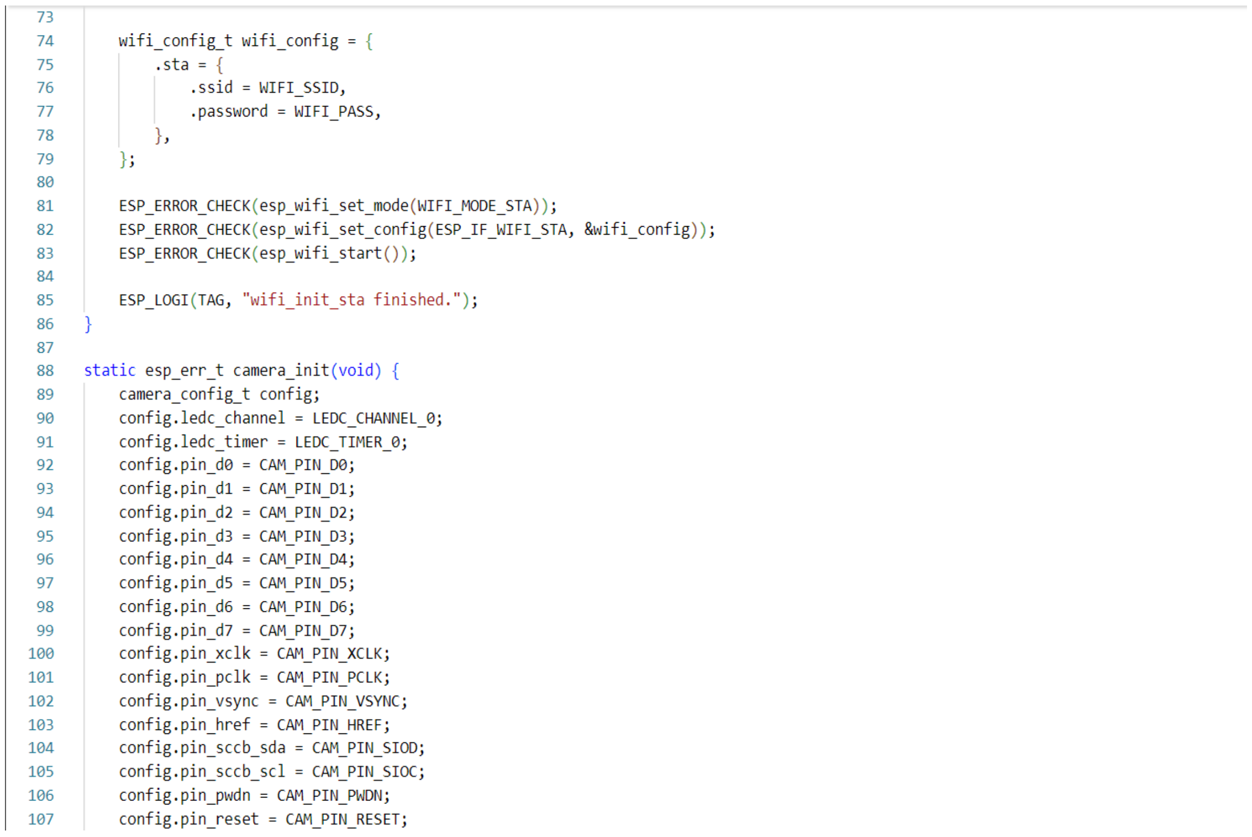
The code to be uploaded to ESP32 is shown below. It uses the ESP32 as a Wi-Fi access point, creates its own IP address and uploads the images to that web server.



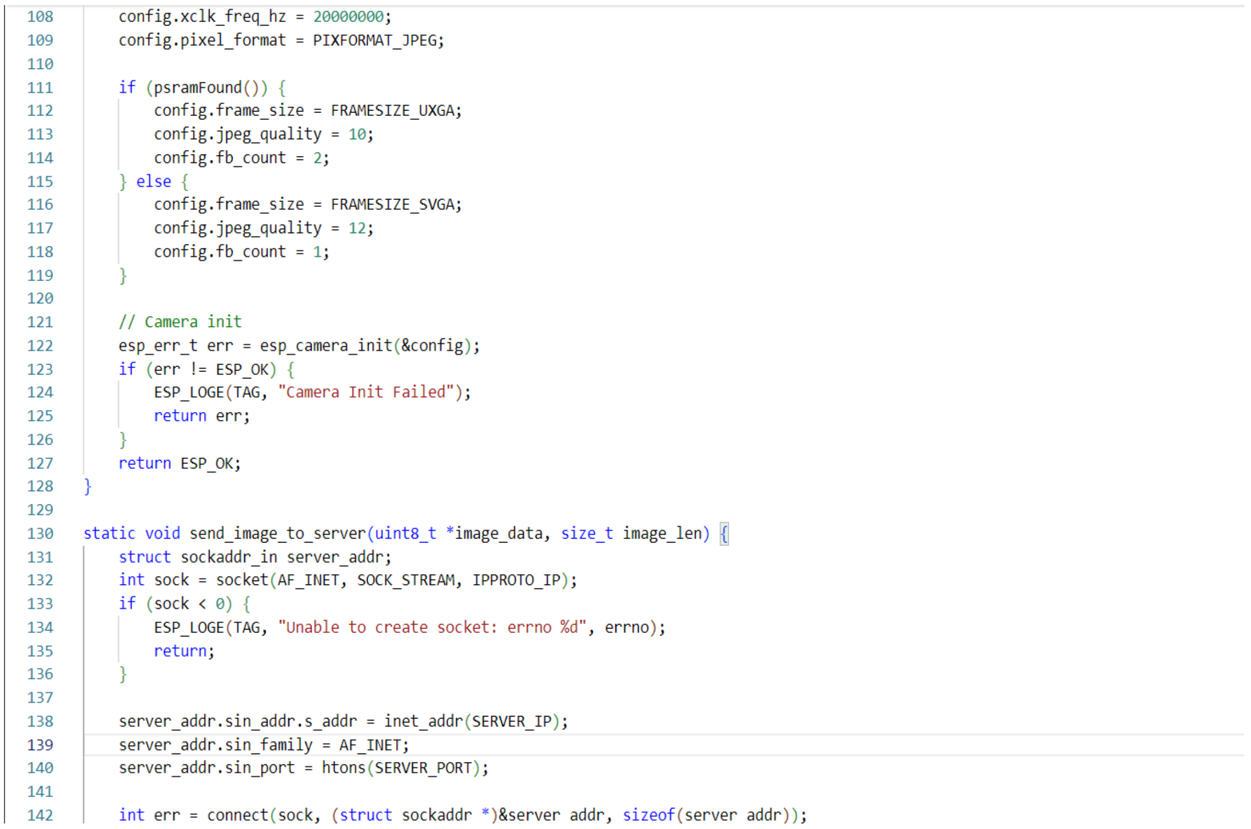
```
#include "esp_camera.h"
#include "esp_log.h"
#include "nvs_flash.h"
#include "esp_wifi.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_http_client.h"
#include "esp_event.h"
#include "lwip/sockets.h"

#define CAM_PIN_PWDN    -1 //power down is not used
#define CAM_PIN_RESET   -1 //software reset will be performed
#define CAM_PIN_XCLK    0
#define CAM_PIN_SIOD    26
#define CAM_PIN_SIOC    27
#define CAM_PIN_D7      35
#define CAM_PIN_D6      34
#define CAM_PIN_D5      39
#define CAM_PIN_D4      36
#define CAM_PIN_D3      21
#define CAM_PIN_D2      19
#define CAM_PIN_D1      18
#define CAM_PIN_D0      5
#define CAM_PIN_VSYNC   25
#define CAM_PIN_HREF    23
#define CAM_PIN_PCLK    22

#define WIFI_SSID "your_SSID"
#define WIFI_PASS "your_PASSWORD"

#define SERVER_IP "192.168.1.100"
#define SERVER_PORT 8080

static const char *TAG = "ESP32-CAM";

static void wifi_init(void);
```



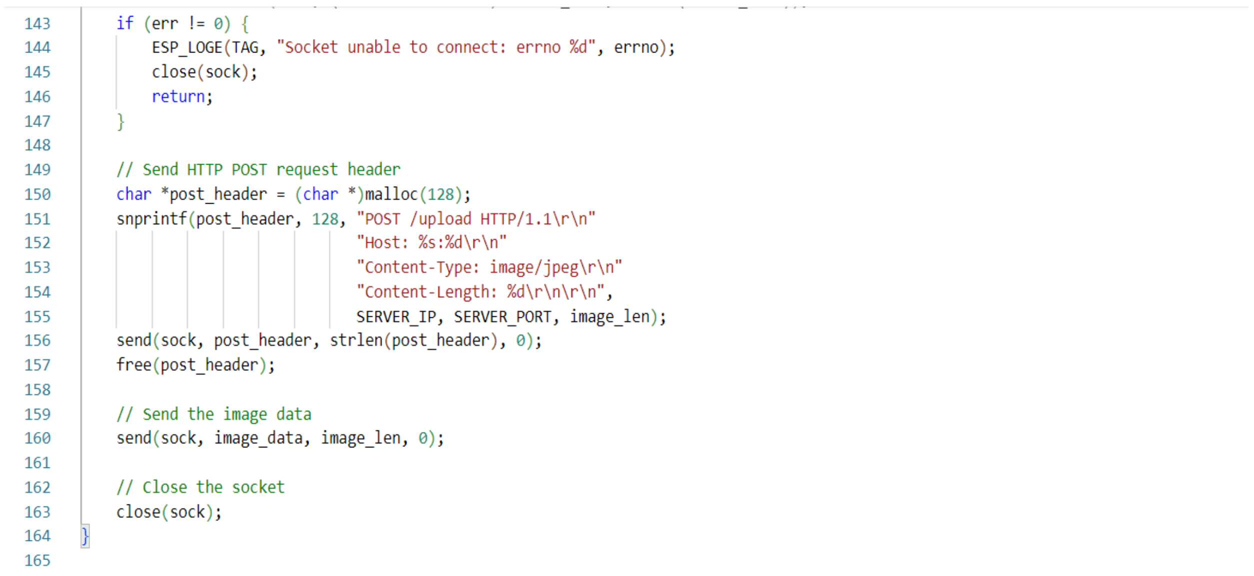
```c
static esp_err_t camera_init(void);
static void send_image_to_server(uint8_t *image_data, size_t image_len);

void app_main() {
    esp_err_t ret = nvs_flash_init();
    if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
        ESP_ERROR_CHECK(nvs_flash_erase());
        ret = nvs_flash_init();
    }
    ESP_ERROR_CHECK(ret);

    wifi_init();
    ESP_ERROR_CHECK(camera_init());

    while (1) {
        camera_fb_t *fb = esp_camera_fb_get();
        if (!fb) {
            ESP_LOGE(TAG, "Camera capture failed");
            vTaskDelay(1000 / portTICK_PERIOD_MS);
            continue;
        }

        send_image_to_server(fb->buf, fb->len);
        esp_camera_fb_return(fb);

        vTaskDelay(5000 / portTICK_PERIOD_MS); // Capture image every 5 seconds
    }
}

static void wifi_init(void) {
    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_create_default_wifi_sta();

    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));
```



```c

    wifi_config_t wifi_config = {
        .sta = {
            .ssid = WIFI_SSID,
            .password = WIFI_PASS,
        },
    };

    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
    ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_config));
    ESP_ERROR_CHECK(esp_wifi_start());

    ESP_LOGI(TAG, "wifi_init_sta finished.");
}

static esp_err_t camera_init(void) {
    camera_config_t config;
    config.ledc_channel = LEDC_CHANNEL_0;
    config.ledc_timer = LEDC_TIMER_0;
    config.pin_d0 = CAM_PIN_D0;
    config.pin_d1 = CAM_PIN_D1;
    config.pin_d2 = CAM_PIN_D2;
    config.pin_d3 = CAM_PIN_D3;
    config.pin_d4 = CAM_PIN_D4;
    config.pin_d5 = CAM_PIN_D5;
    config.pin_d6 = CAM_PIN_D6;
    config.pin_d7 = CAM_PIN_D7;
    config.pin_xclk = CAM_PIN_XCLK;
    config.pin_pclk = CAM_PIN_PCLK;
    config.pin_vsync = CAM_PIN_VSYNC;
    config.pin_href = CAM_PIN_HREF;
    config.pin_sccb_sda = CAM_PIN_SIOD;
    config.pin_sccb_scl = CAM_PIN_SIOC;
    config.pin_pwdn = CAM_PIN_PWDN;
    config.pin_reset = CAM_PIN_RESET;
```



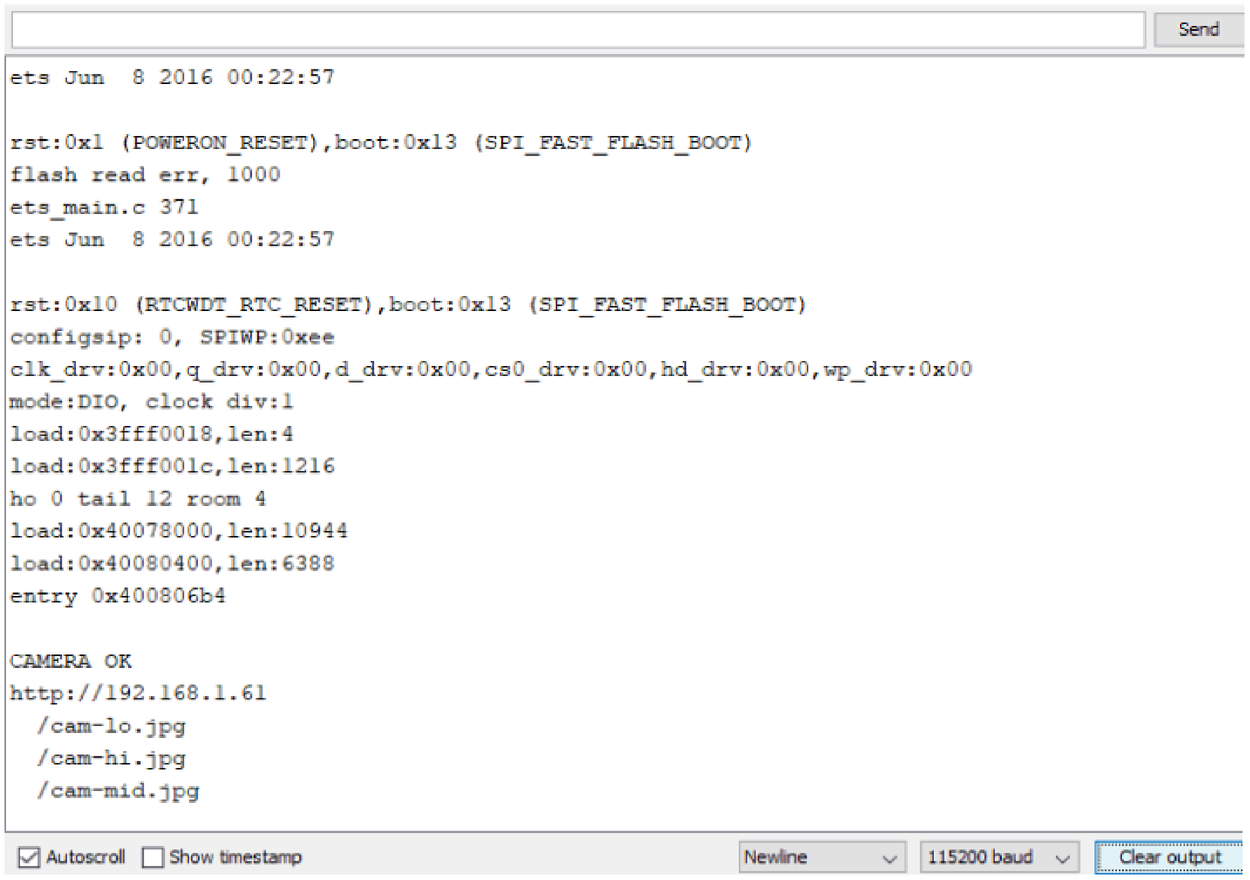
```c
    config.xclk_freq_hz = 20000000;
    config.pixel_format = PIXFORMAT_JPEG;

    if (psramFound()) {
        config.frame_size = FRAMESIZE_UXGA;
        config.jpeg_quality = 10;
        config.fb_count = 2;
    } else {
        config.frame_size = FRAMESIZE_SVGA;
        config.jpeg_quality = 12;
        config.fb_count = 1;
    }

    // Camera init
    esp_err_t err = esp_camera_init(&config);
    if (err != ESP_OK) {
        ESP_LOGE(TAG, "Camera Init Failed");
        return err;
    }
    return ESP_OK;
}

static void send_image_to_server(uint8_t *image_data, size_t image_len) {
    struct sockaddr_in server_addr;
    int sock = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
    if (sock < 0) {
        ESP_LOGE(TAG, "Unable to create socket: errno %d", errno);
        return;
    }

    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(SERVER_PORT);

    int err = connect(sock, (struct sockaddr *)&server_addr, sizeof(server_addr));
```



```c
    if (err != 0) {
        ESP_LOGE(TAG, "Socket unable to connect: errno %d", errno);
        close(sock);
        return;
    }

    // Send HTTP POST request header
    char *post_header = (char *)malloc(128);
    snprintf(post_header, 128, "POST /upload HTTP/1.1\r\n"
                               "Host: %s:%d\r\n"
                               "Content-Type: image/jpeg\r\n"
                               "Content-Length: %d\r\n\r\n",
                               SERVER_IP, SERVER_PORT, image_len);
    send(sock, post_header, strlen(post_header), 0);
    free(post_header);

    // Send the image data
    send(sock, image_data, image_len, 0);

    // Close the socket
    close(sock);
}
```

Figure 4

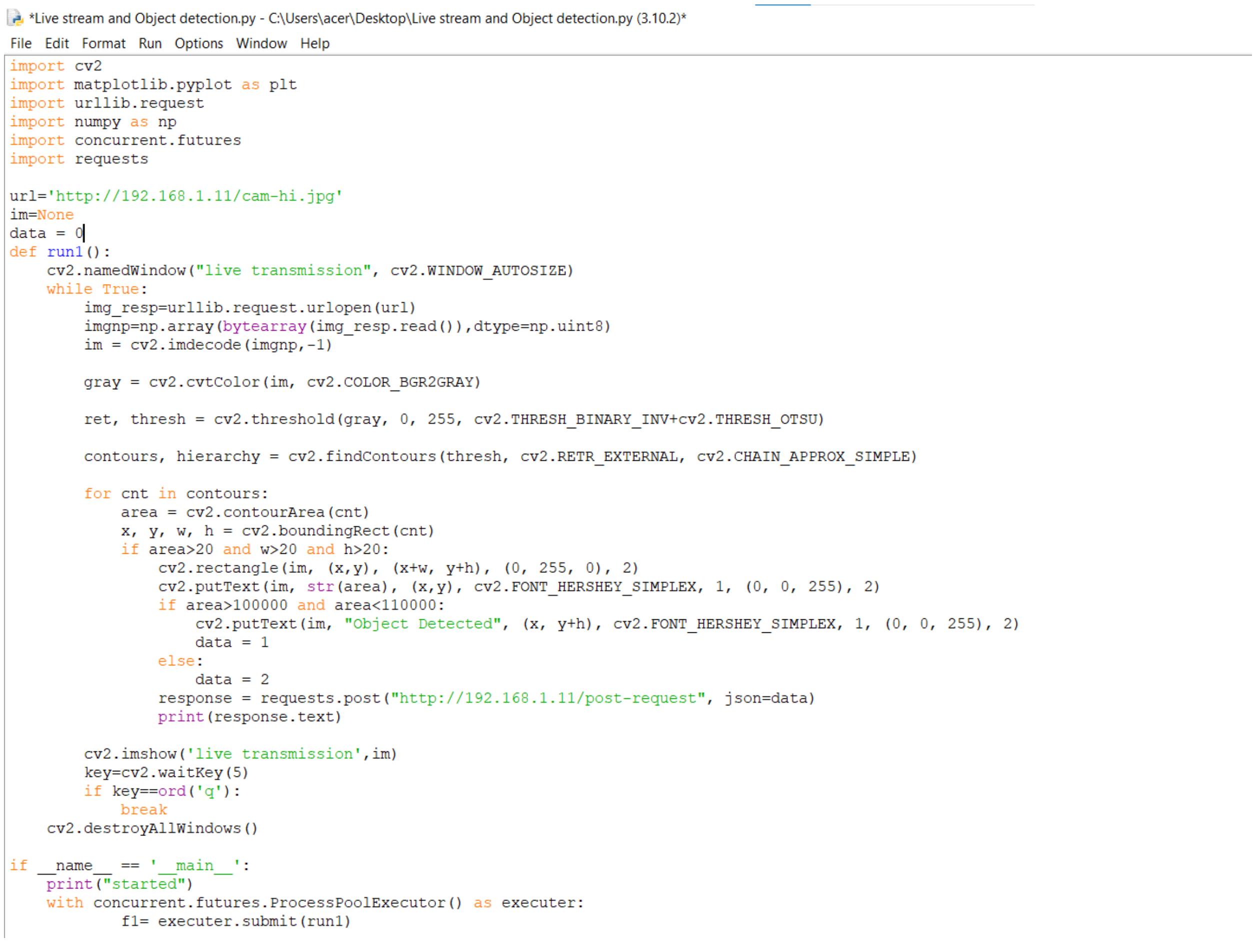
Once the code is uploaded the serial monitor will display something like this.



```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
flash read err, 1000
ets_main.c 371
ets Jun  8 2016 00:22:57

rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:10944
load:0x40080400,len:6388
entry 0x400806b4

CAMERA OK
http://192.168.1.61
  /cam-lo.jpg
  /cam-hi.jpg
  /cam-mid.jpg
```

Figure 5

Here the live feed from the camera has been sent to the IP address – 192.168.1.61 in three different resolutions. Now we have to change the python code in the software in order to retrieve that live feed from the given IP address. The changed python code is shown below.


Live stream and Object detection.py - C:\Users\acer\Desktop\Live stream and Object detection.py (3.10.2)
File Edit Format Run Options Window Help

```
import cv2
import matplotlib.pyplot as plt
import urllib.request
import numpy as np
import concurrent.futures
import requests

url='http://192.168.1.11/cam-hi.jpg'
im=None
data = 0
def run1():
    cv2.namedWindow("live transmission", cv2.WINDOW_AUTOSIZE)
    while True:
        img_resp=urllib.request.urlopen(url)
        imgnp=np.array(bytearray(img_resp.read()),dtype=np.uint8)
        im = cv2.imdecode(imgnp,-1)

        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

        ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

        contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        for cnt in contours:
            area = cv2.contourArea(cnt)
            x, y, w, h = cv2.boundingRect(cnt)
            if area>20 and w>20 and h>20:
                cv2.rectangle(im, (x,y), (x+w, y+h), (0, 255, 0), 2)
                cv2.putText(im, str(area), (x,y), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
                if area>100000 and area<110000:
                    cv2.putText(im, "Object Detected", (x, y+h), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
                    data = 1
                else:
                    data = 2
                response = requests.post("http://192.168.1.11/post-request", json=data)
                print(response.text)

        cv2.imshow('live transmission',im)
        key=cv2.waitKey(5)
        if key==ord('q'):
            break
    cv2.destroyAllWindows()

if __name__ == '__main__':
    print("started")
    with concurrent.futures.ProcessPoolExecutor() as executer:
            f1= executer.submit(run1)
```

Figure 6

The output will be similar to the one shown under software implementation. Additionally, here it will display whether the object is detected or not under given width and height constraints.

Once the object is detected the stepper motor will be activated and the mechanical arm attached to it will push the product, hence product sortation will be completed. The stepper motor used for this is 28BYJ-48 5V stepper motor along with the ULN2003A driver module.

Power Supply

Power supply is the other main component of the project. First of all since this product is meant to be used in the product manufacturing industries, we have to convert the 230V AC current input into 5V DC current using a converter. When choosing the converter power constraints are essential. However, since the micro-controller, the camera and other small electronic components only require power amounts of milli Watt, we can simply use a 230V AC – 5V DC 3Watt converter to supply enough power for all needs in the circuit.

Furthermore, the OV2640 camera which is been used in the ESP32 Cam module requires few voltage inputs such as 5V, 3.3V, 2.8V and 1.2V. In order to cater for those needs we have used following voltage regulators in the power supply.

Voltage Level	**Regulator Used**
5V	Direct from the converter
3.3V	AMS1117-3.3V
2.8V	LM317T Adjustable Voltage Regulator
1.2V	LM317T Adjustable Voltage Regulator

Table 2

Schematic Designs – 25/03/2024 to 07/04/2024

Once all the Hardware component selections and testing are done it's time to design the schematics for the PCB. This will mainly include the schematics for the ESP32 Cam module and the power supply including the regulators. The schematics have been designed in a hierarchical approach where the ESP32 Cam module, power supply and PSRAM schematics will come under the main schematic as follows.

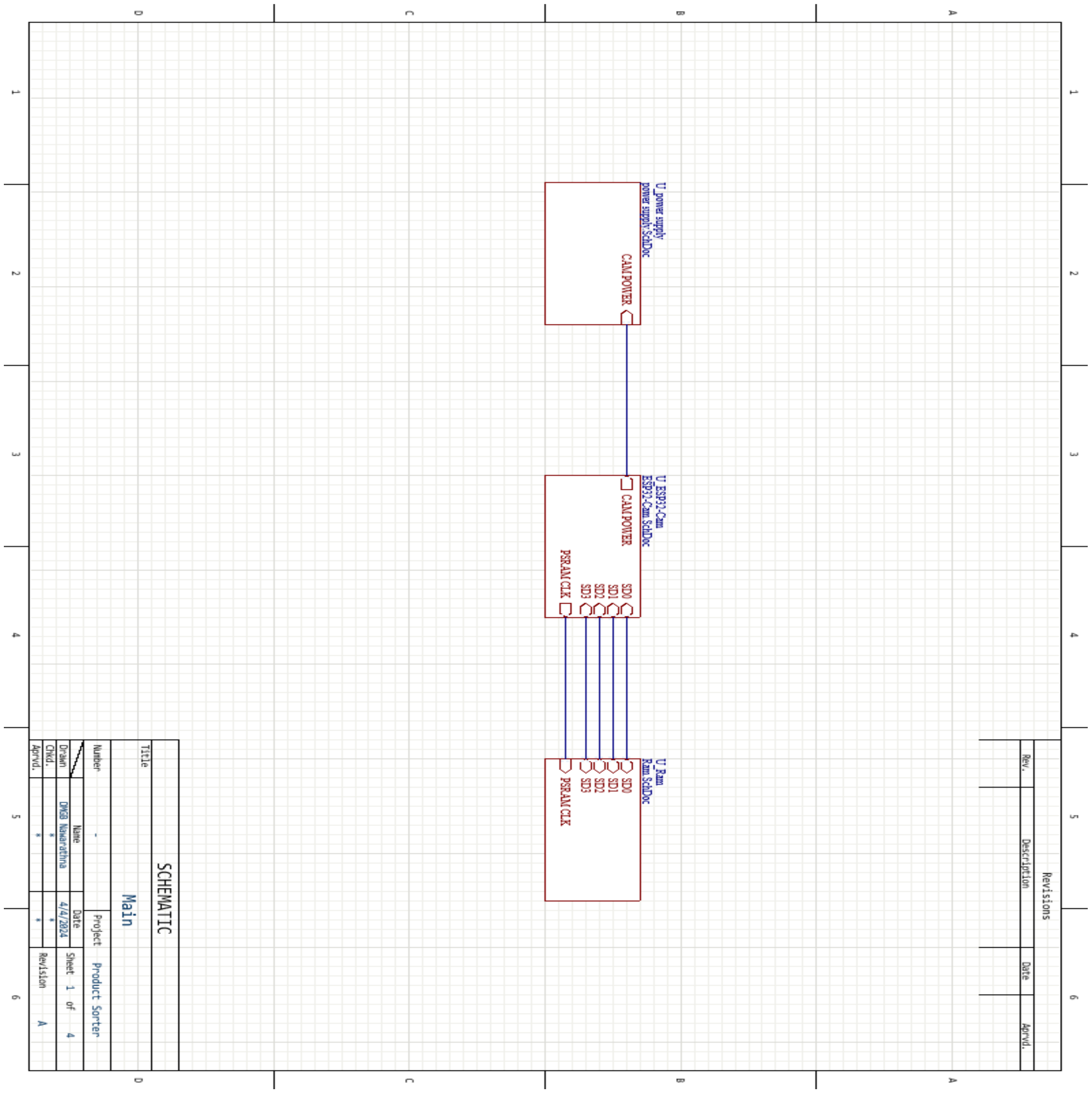


Figure 7

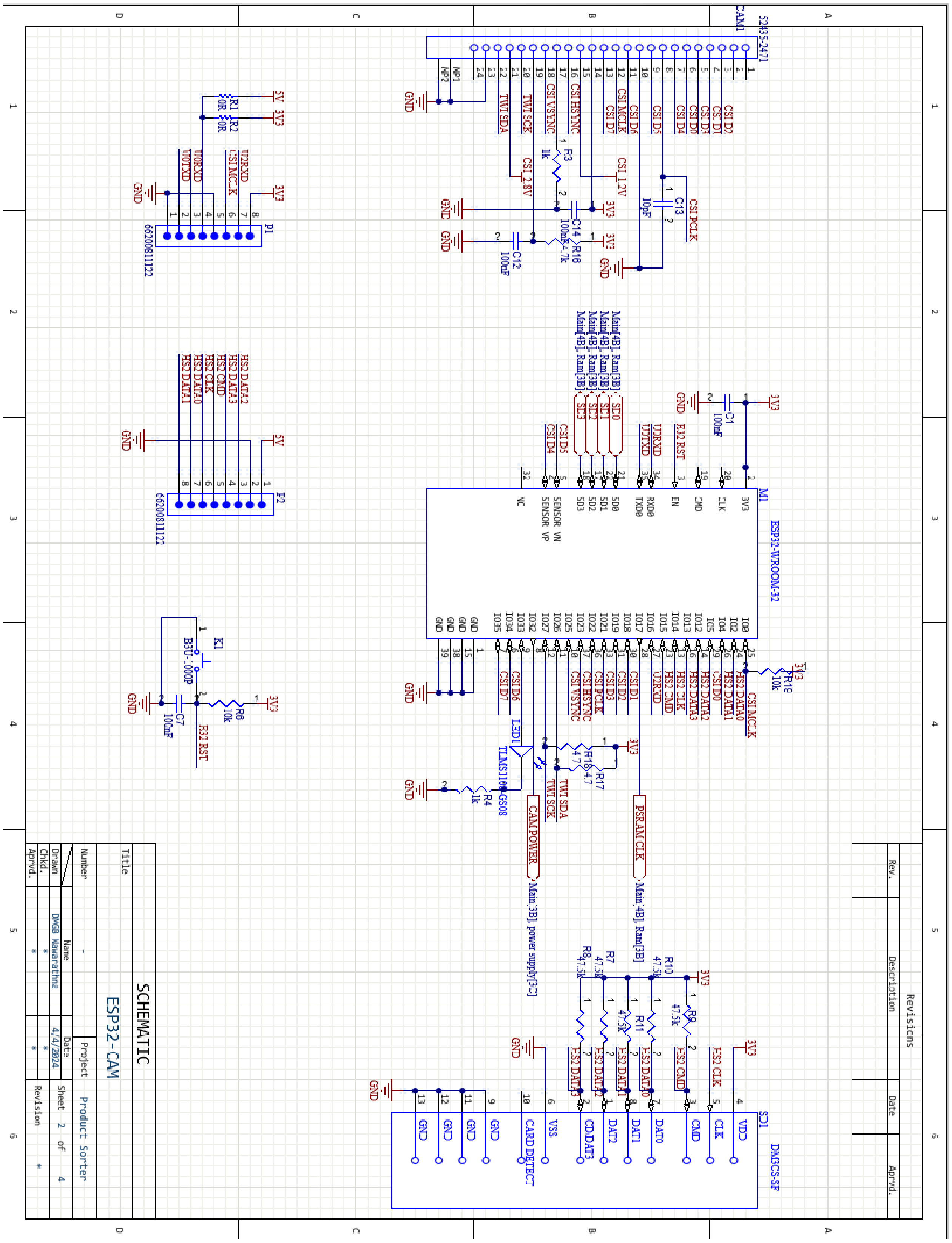

SCHEMATIC
ESP32-CAM
Project Product Sorter
Drawn DMGB Nawarathna
Date 4/4/2024
Sheet 2 of 4
ESP32-WROOM-32
52435-2471
CAM1
DM3CS-SF
SD1
6620081112
P1
P2
B3U-1000P
K1
TLMS1100-GS08
LED1
CAM POWER
Main[3B], power supply[3C]
PSRAM CLK
Main[4B], Ram[3B]
Revisions

Figure 8

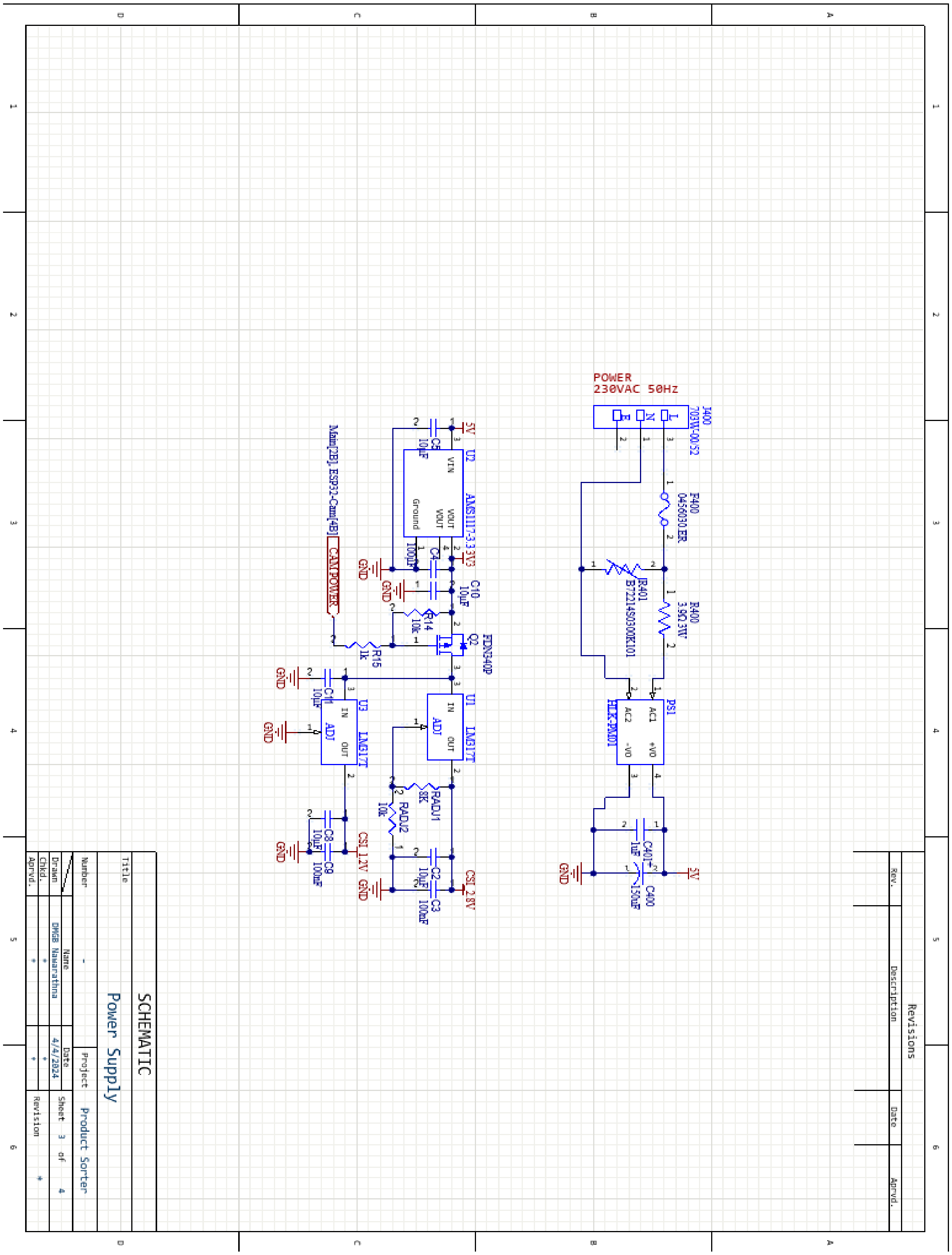

POWER
230VAC 50Hz
J400
703W-00/52
F400
0456030.ER
R400
3.9Ω 3W
R401
B72214S0300K101
PS1
HLK-PM01
AC1
AC2
+VO
-VO
C400
150uF
C401
1uF
5V
GND
U2
AMS1117-3.3
VIN
VOUT
Ground
3V3
C5
10uF
C4
100nF
C10
10uF
R14
10k
Q2
FDN340P
R15
1k
CAM POWER
Main[2B], ESP32-Cam[4B]
U1
LM317T
IN
ADJ
OUT
RADJ1
8K
RADJ2
10k
C2
10uF
C3
100nF
CSI 2.8V
U3
LM317T
C11
10uF
C8
10uF
C9
100nF
CSI 1.2V
SCHEMATIC
Title
Power Supply
Number
Project
Product Sorter
Drawn
DMGB Nawarathna
Date
4/4/2024
Sheet 3 of 4
Chkd.
Aprvd.
Revision
Revisions
Rev.
Description
Date
Aprvd.

Figure 9

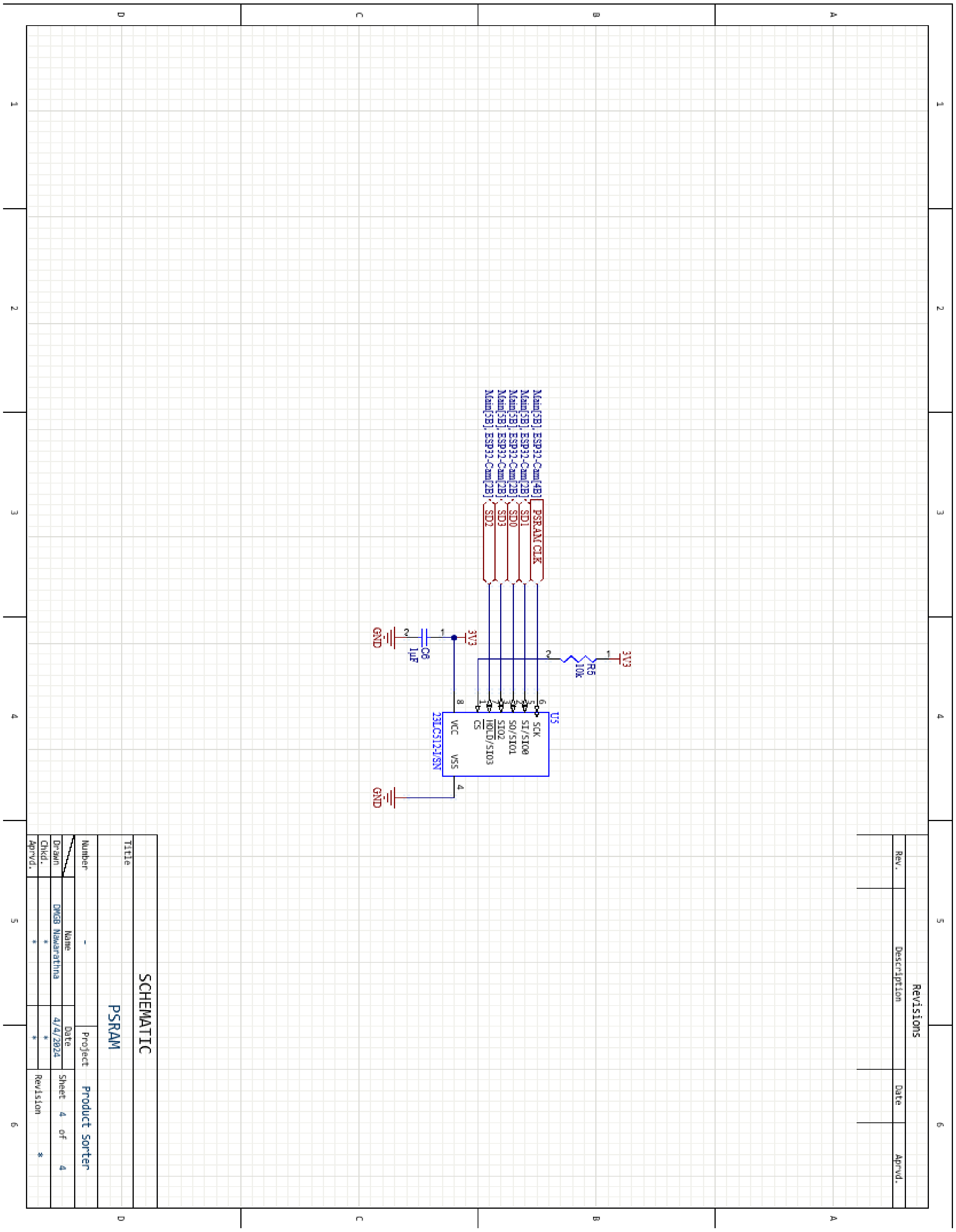

Main[SB].ESP32-Cam[4B] PSRAM CLK
Main[SB].ESP32-Cam[2B] SD1
Main[SB].ESP32-Cam[2B] SD0
Main[SB].ESP32-Cam[2B] SD3
Main[SB].ESP32-Cam[2B] SD2
3V3
C6 1uF
GND
R5 10k
U5
SCK
SI/SIO0
SO/SIO1
SIO2
HOLD/SIO3
CS
VCC
VSS
23LC512-I/SN
Title
SCHEMATIC
PSRAM
Number
Project
Product Sorter
Drawn
DMGB Nawarathna
4/4/2024
Sheet 4 of 4
Revision
Revisions
Rev.
Description
Date
Aprvd.

Figure 10

PCB Design – 08/04/2024 to 14/04/2024

Once the schematics are finalized using a PCB designing software (Altium) we extract the components into the PCB, placed them carefully and routed them. The finalized 2D and 3D PCBs are shown below.

2D Design

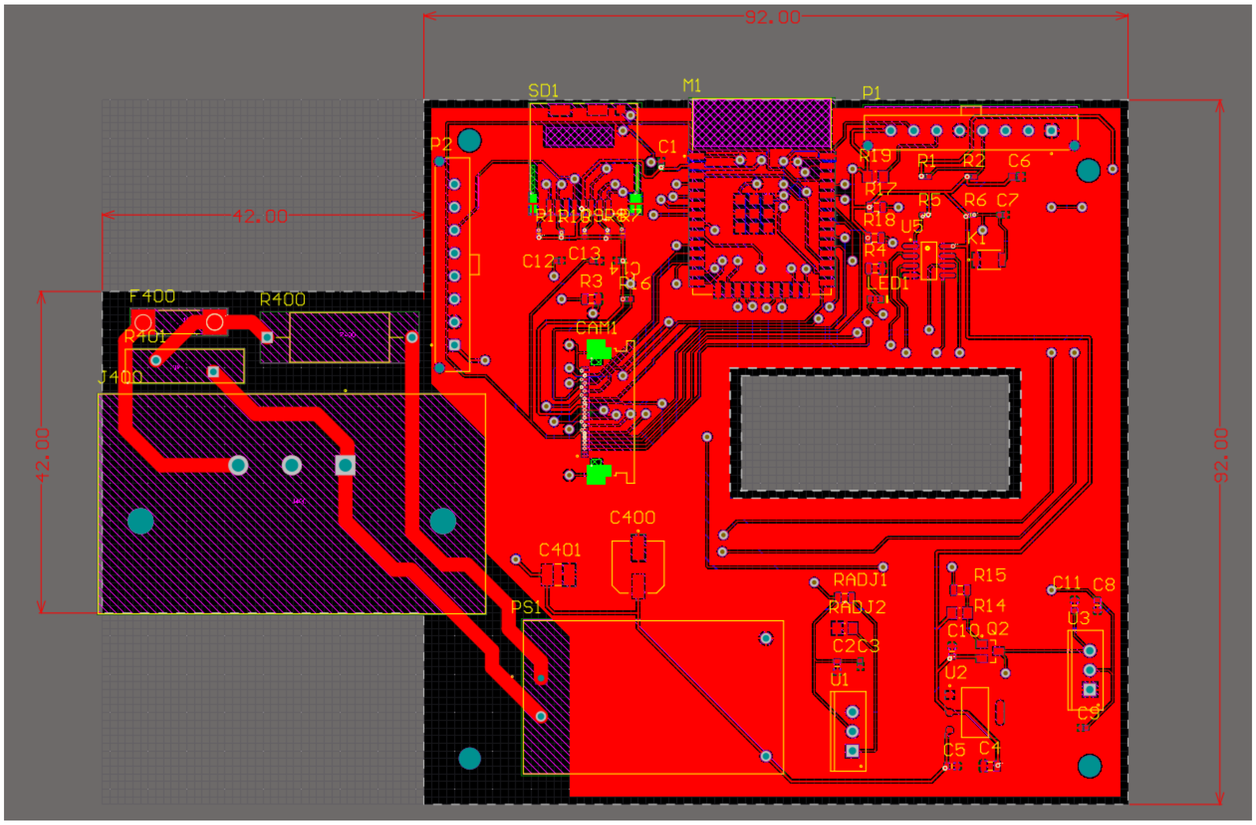


Figure 11

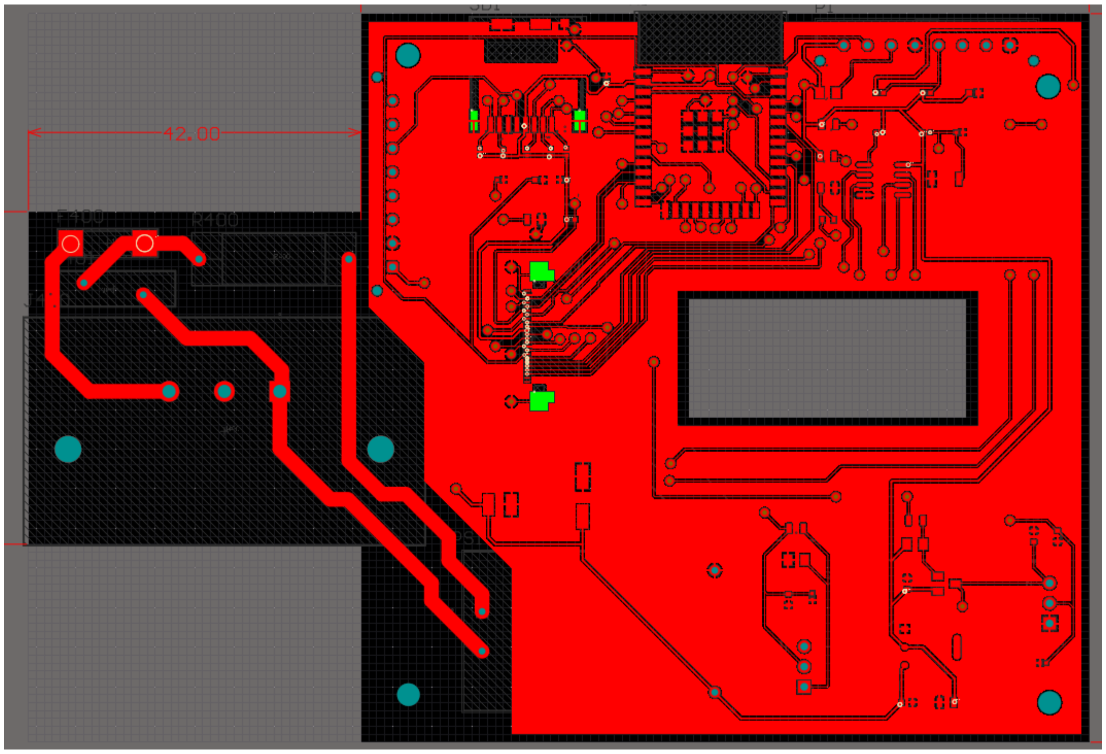


Figure 12

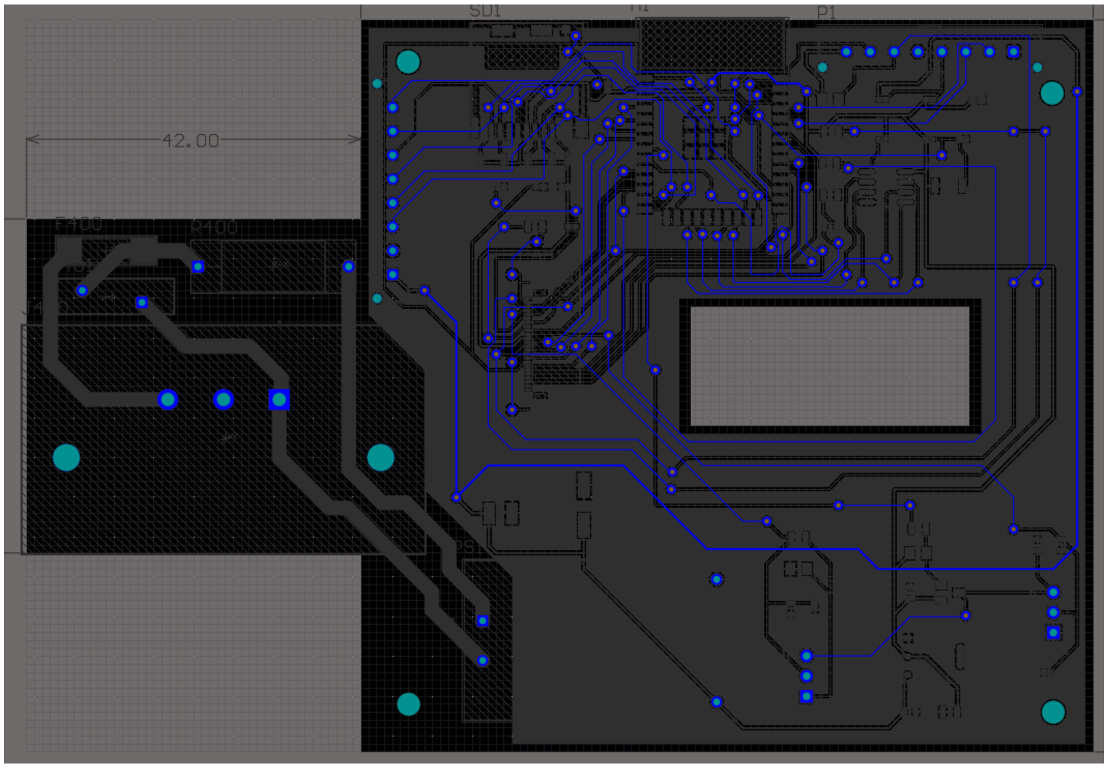


Figure 13

3D Design

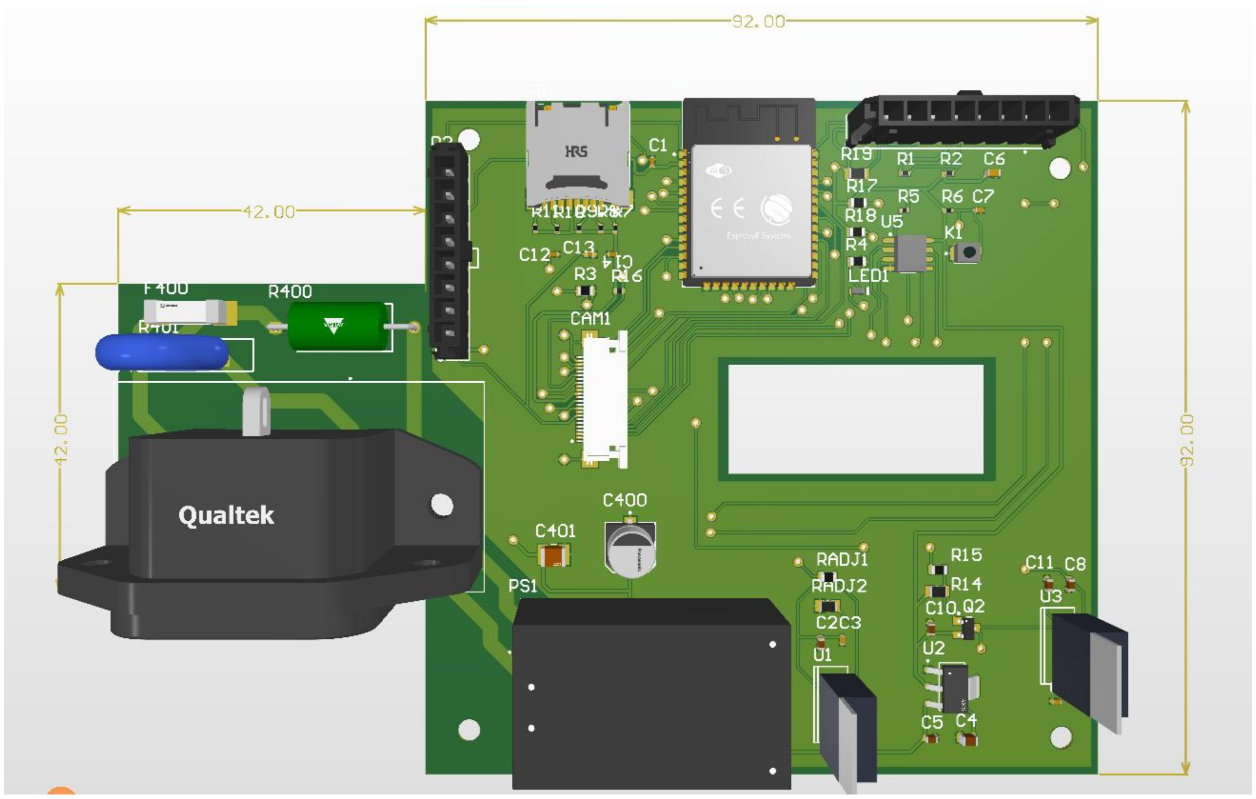


Figure 14

Once the design is finalized we adjusted the PCB constraints according to JLCPCB constraints and ensured there are no errors for the PCB print. Next, the design was sent for the printing process.

SolidWorks Design – 15/04/2024 to 28/04/2024

While the PCB is being printed, we designed the enclosure. The main idea of the enclosure is that it should hold the camera above the conveyor belt. Therefore, it should have a tower kind of structure which will stand on one side of the conveyor belt holding the camera above it. Further, it should have a bar perpendicular to the tower. That bar will carry the stepper motor which will operate the mechanical arm and push the product on the conveyor belt when necessary. The design is shown below.

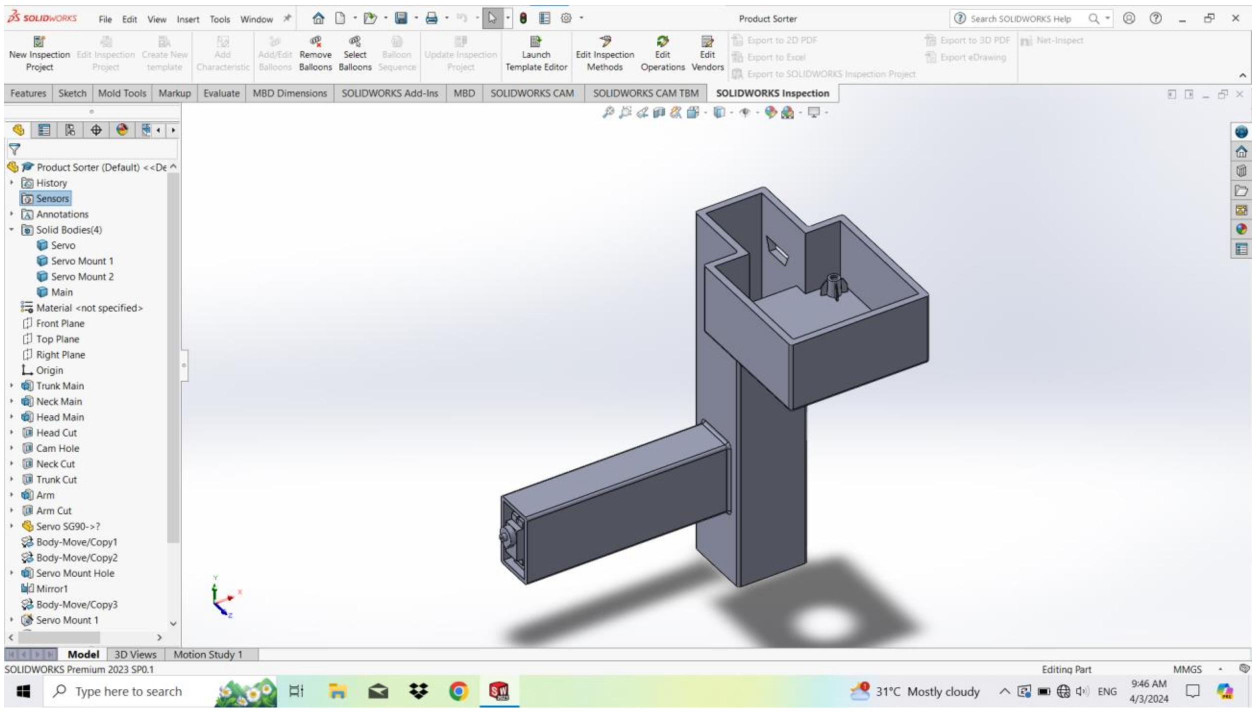


Figure 15

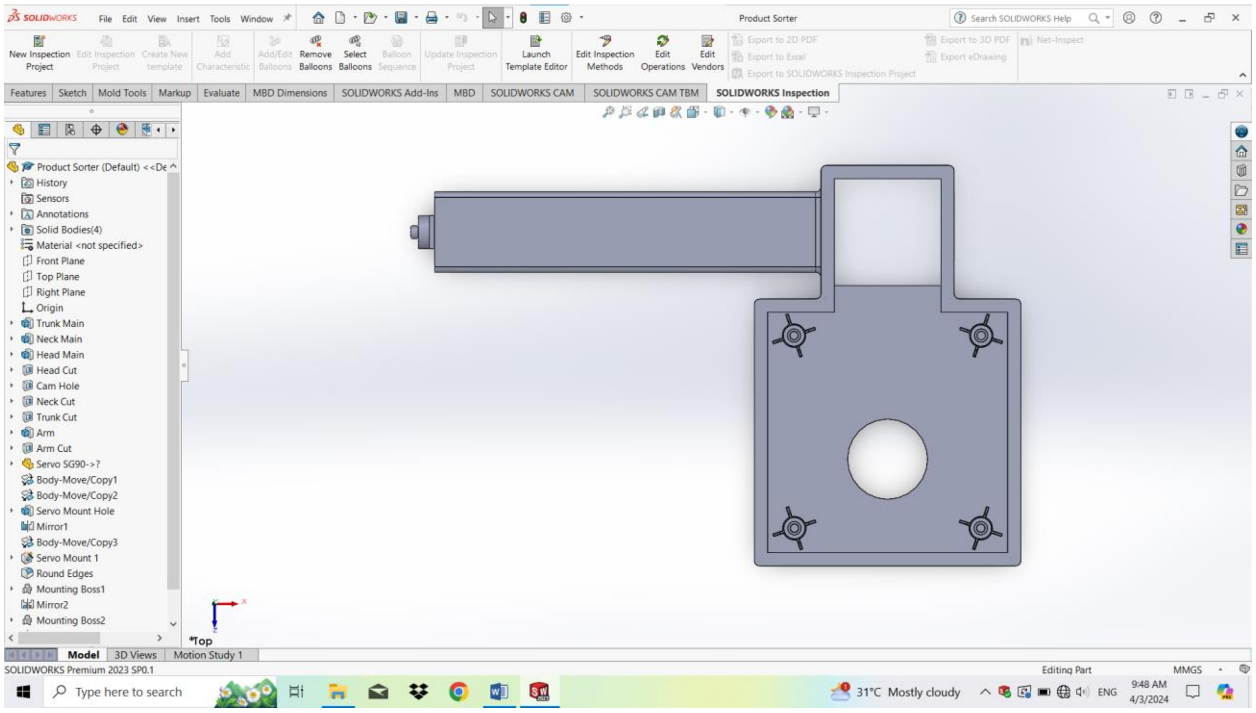


Figure 16

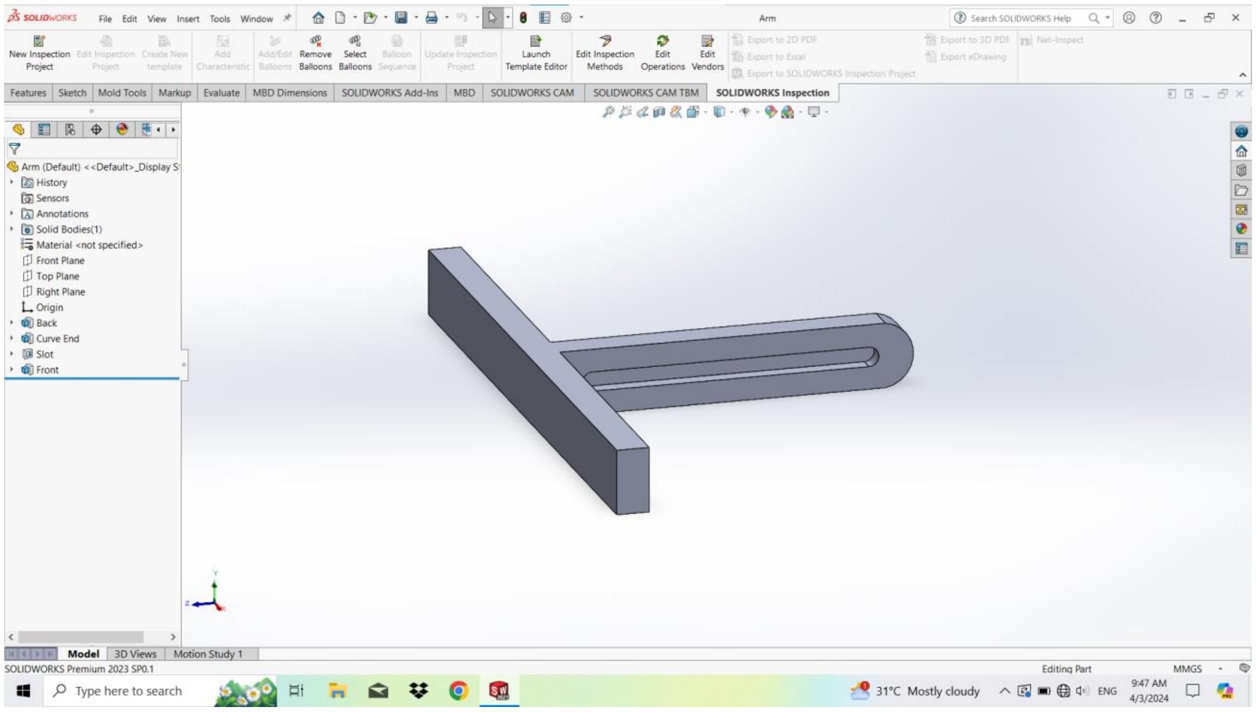


Figure 17

Once the solidworks design is completed and finalized, it was sent for the printing process as well.

Final Steps and Conclusion

Once the printed PCB is received, it was soldered and tested. Next the completed PCB was integrated with the printed enclosure and the final tests were run.

In summary, the journey towards developing the Product Sorting Conveyor Belt using Computer Vision technology has been characterized by meticulous planning and innovative design. Guided by stakeholder needs and a commitment to excellence, significant progress has been made from conceptualization to final design.

As we advance to prototype development and integration, we remain focused on validating performance and refining the system based on user feedback. Ultimately, this project represents a leap forward in industrial sorting technology, driven by innovation and customer satisfaction. With confidence in the approach, I anticipate delivering a solution that will redefine product sorting in industrial environments.

References

1. "How to Install OpenCV for Python on Windows?" geeksforgeeks.org, https://www.geeksforgeeks.org/how-to-install-opencv-for-python-in-windows/ (accessed on 14/02/2024).

2. "ESP32 Cam Object Detection & Identification with OpenCV" how2electronics.com, https://how2electronics.com/esp32-cam-based-object-detection-identification-with-opencv/ (accessed on 24/02/2024).

3. "ESP32-CAM AI-Thinker Pinout Guide: GPIOs Usage Explained" randomnerdtutorials.com, https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/ (accessed on 09/03/2024).

4. Espressif Systems. *ESP32-S2-WROOM-1, WROOM-1U v1.3.* Datasheet. 2023. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (accessed on 17/03/2024).

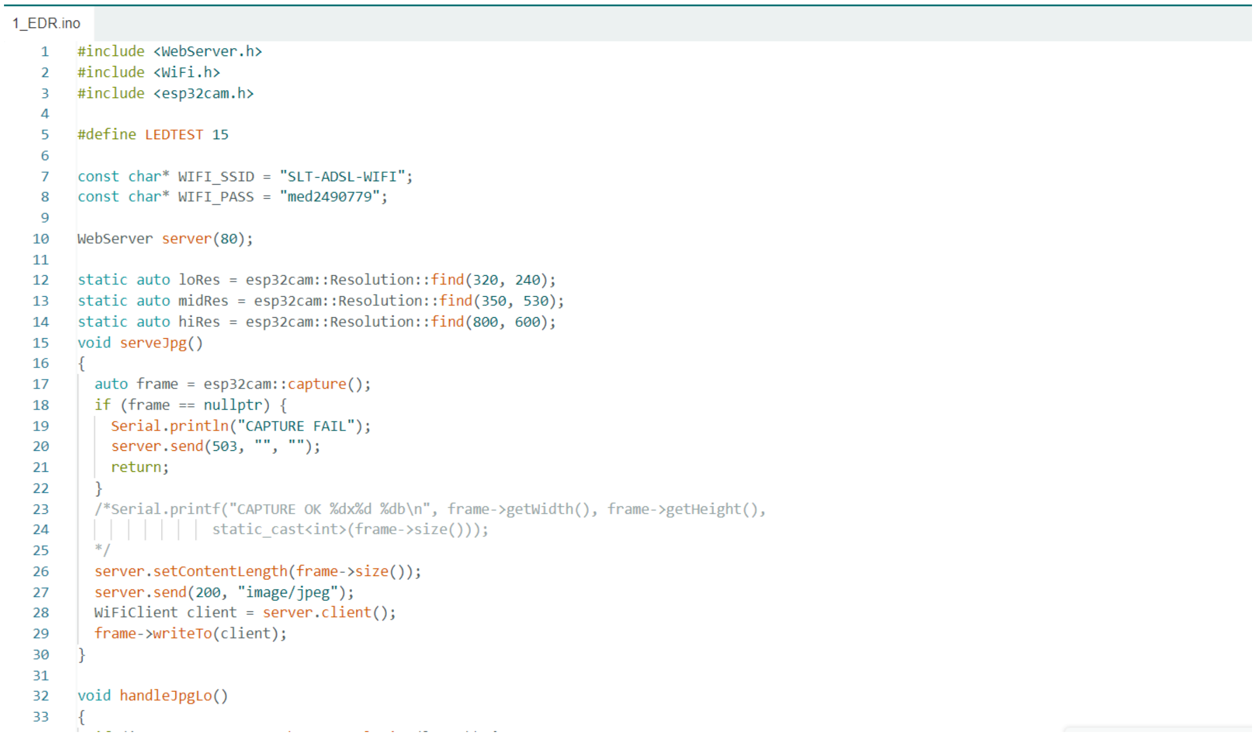
Appendix

Daily Log Entry

Date	Entry
14/02/2024	Installed OpenCV and setting up the environment
20/02/2024 – 02/03/2024	Software implemented using python and OpenCV
08/03/2024	Came up with three conceptual designs
10/03/2024	Purchased hardware modules for testing
14/03/2024 – 18/03/2024	Hardware implementation and coding using arduino
20/03/2024	Evaluated all conceptual designs and finalized one preliminary design
20/03/2024 – 22/03/2024	Changed the software code in order to adjust with the hardware implementation
26/03/2024	Started drawing schematics
04/04/2024	Finalized the schematic designs and initiated the PCB design
08/04/2024	Finalized the PCB design
16/04/2024	Started designing the enclosure using Solidworks
24/04/2024	Sent the PCB for printing
28/04/2024	Finalized the Solidworks design
03/05/2024	Retrieved the printed PCB
07/05/2024	Purchased components required for the PCB
08/06/2024	3D printed the enclosure
14/06/2024	Soldered the PCB components

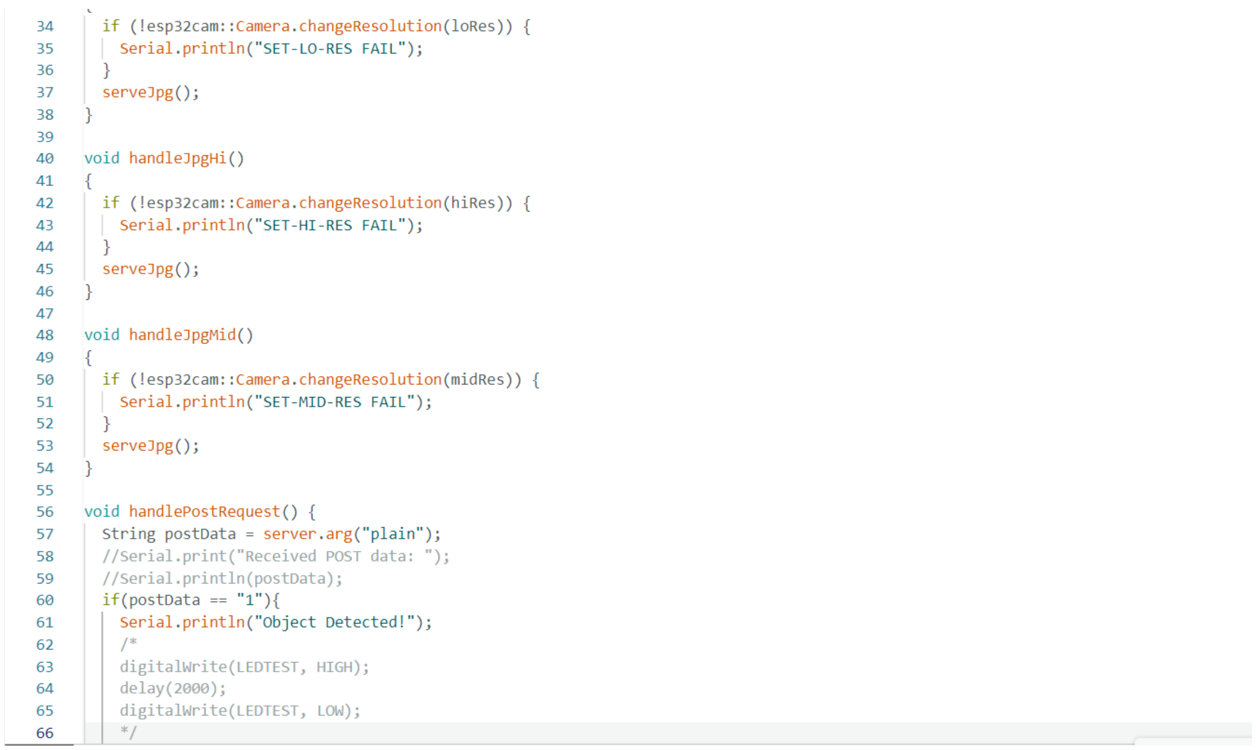
Table 3

Code in Previous Submission (Using Arduino Libraries)

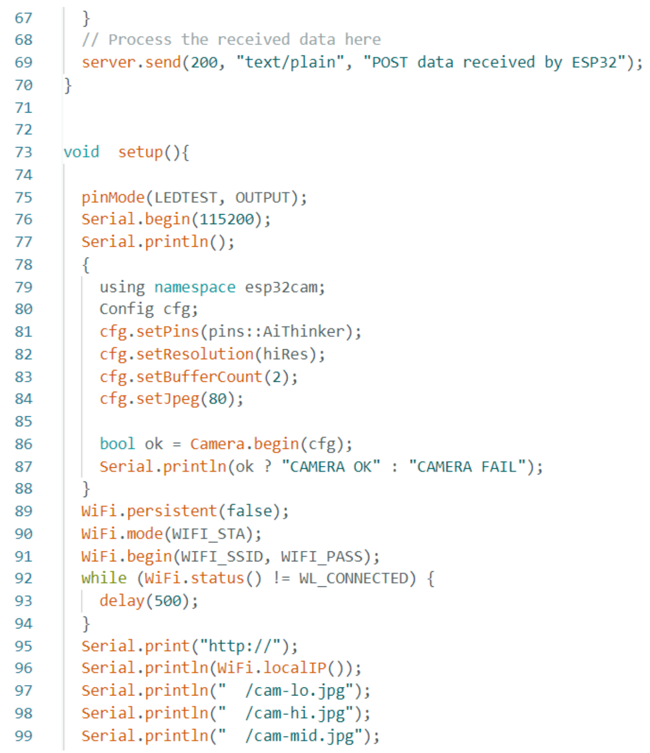


1_EDR.ino

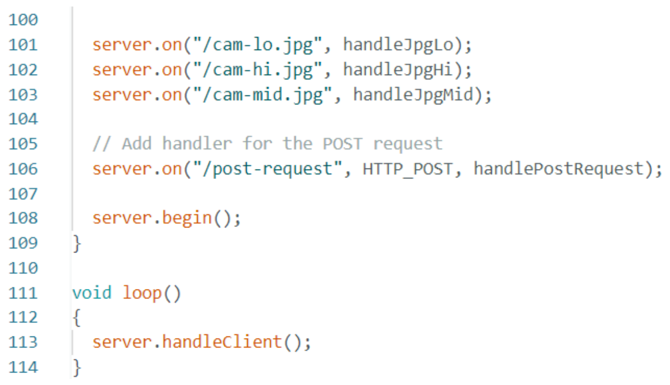
```
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>

#define LEDTEST 15

const char* WIFI_SSID = "SLT-ADSL-WIFI";
const char* WIFI_PASS = "med2490779";

WebServer server(80);

static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);
void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  /*Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
                static_cast<int>(frame->size()));
  */
  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}

void handleJpgLo()
{
```



```
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}

void handleJpgHi()
{
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}

void handleJpgMid()
{
  if (!esp32cam::Camera.changeResolution(midRes)) {
    Serial.println("SET-MID-RES FAIL");
  }
  serveJpg();
}

void handlePostRequest() {
  String postData = server.arg("plain");
  //Serial.print("Received POST data: ");
  //Serial.println(postData);
  if(postData == "1"){
    Serial.println("Object Detected!");
    /*
    digitalWrite(LEDTEST, HIGH);
    delay(2000);
    digitalWrite(LEDTEST, LOW);
    */
```



```
  }
  // Process the received data here
  server.send(200, "text/plain", "POST data received by ESP32");
}


void  setup(){

  pinMode(LEDTEST, OUTPUT);
  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
  }
  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.print("http://");
  Serial.println(WiFi.localIP());
  Serial.println("  /cam-lo.jpg");
  Serial.println("  /cam-hi.jpg");
  Serial.println("  /cam-mid.jpg");
```



```

  server.on("/cam-lo.jpg", handleJpgLo);
  server.on("/cam-hi.jpg", handleJpgHi);
  server.on("/cam-mid.jpg", handleJpgMid);

  // Add handler for the POST request
  server.on("/post-request", HTTP_POST, handlePostRequest);

  server.begin();
}

void loop()
{
  server.handleClient();
}
```