

*Heaven's Light is Our Guide*



**Rajshahi University of Engineering & Technology**  
**Department of Electronics & Telecommunication Engineering**

Project Report  
on  
Reverse Vending Machine

Submitted by

---

**Saraf Anzum Shreya**

Roll No. 2004036

Submitted to

---

**Md. Aslam Mollah**

Assistant Professor

Department of ETE

January 16, 2024

# **Contents**

<b>1 . Abstract</b>	<b>1</b>
<b>2 . Acknowledgement</b>	<b>2</b>
<b>3 . Introduction</b>	<b>4</b>
<b>4 . Components and Methodology</b>	<b>5</b>
<b>5 . Working Process</b>	<b>14</b>
<b>6 . Implementation Code</b>	<b>16</b>
<b>7 . Reverse Vending Machine</b>	<b>70</b>
<b>8 . Future Work</b>	<b>71</b>
<b>9 . Discussion and Conclusion</b>	<b>72</b>
<b>10 . References</b>	<b>73</b>

*Heaven's Light is Our Guide*

**RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY, RAJSHAHI**

**Department of Electronics & Telecommunication Engineering**



## **CERTIFICATE**

*This is to certify that the project entitled "Reverse Vending machine" by Saraf Anzum Shreya, Roll: 2004036 has been carried out under my supervision. To the best of my knowledge, this project work is an original one and was not submitted anywhere for any degree or diploma.*

### **Supervisor**

Md. Aslam Mollah  
Assistant Professor  
Department of Electronics & Telecommunication Engineering  
Rajshahi University of Engineering & Technology  
Rajshahi – 6204

## **Abstract**

Reverse vending machine (RVM) is an innovative solution in waste management and sustainability. Reverse vending machines are automated systems designed to accept used beverage containers, such as plastic bottles and provide incentives or rewards to users in return. This project explores the integration of machine learning technology into Reverse Vending Machines (RVMs) as a cutting-edge solution for enhancing waste management. The primary focus is on automating the recognition and collection process of used beverage containers, specifically plastic bottles, within RVMs while providing users with incentives or rewards. Additionally, the project emphasizes the reciprocal relationship between automated recycling and user engagement, with incentives or rewards acting as catalysts for increased participation.

## Acknowledgement

Regarding the project entitled "*Reverse vending Machine*", I wish to offer my profound sense of gratitude and indebtedness to my project supervisor, *Md. Aslam Mollah, Assistant Professor, Department of Electronics & Telecommunication Engineering* who has supported me throughout my project with his patience and knowledge. I shall always remain grateful to him for his valuable guidance, advice, encouragement, cordial and amiable contribution to my project. I also want to thank the closest companions in my life for giving me huge support.

Saraf Anzum Shreya

Roll: 2004036

Department of ETE, RUET

# Introduction

The escalating global concern over environmental sustainability and the pervasive issue of plastic pollution have necessitated innovative solutions in waste management. Among these solutions, Reverse Vending Machines (RVMs) have emerged as a groundbreaking technology aimed at transforming the way we approach plastic bottle recycling. With the rapid increase in single-use plastic consumption, the need for effective recycling methods has never been more pressing. RVMs offer a promising avenue by seamlessly blending automation, technology, and user incentives to encourage the responsible disposal and recycling of used plastic bottles.

Some of the basic operations of a reverse vending machine are given below:

1. **Bottle Recognition:** The machine uses advanced technology, such as machine learning algorithms, to identify beverage container, primarily focusing on plastic bottles.
2. **Deposit Mechanism:** The machine is equipped with a secure deposit area where users place their used beverage containers. This area ensures proper handling and containment.
3. **QR code scanning:** Upon depositing plastic bottle, a QR code containing users account ID is scanned for providing reward.
4. **Rewarding:** Upon scanning the QR code, a certain amount of credit with added to their account which can later be redeemed as usable currency.
5. **Bin full sensing:** When the container or the bin is full, a message is shown alerting about the condition.

Reverse vending machines operate on a simple yet ingenious principle: they autonomously accept and process used beverage containers, predominantly plastic bottles, and reward users for their contributions. The integration of cutting-edge technologies, such as machine learning and advanced sensor systems, has elevated the efficiency and accuracy of these machines, marking a significant step towards a more sustainable and circular economy.

The main objectives of this project are-

- To provide for better management of trash such as plastic bottles.
- To encourage people not to litter by rewarding them.
- To automate the process of bottle collection.
- To automate the process of bottle collection.
- To make profit by recycling the collected plastic bottles.

This introduction provides an overview of the transformative role of Reverse Vending Machines, focusing on their technological foundations, environmental impact, and the positive behavioral shift they bring about in individuals engaged in recycling practices.

## **Components and Methodology**

The project "Reverse Vending Machine" is made of different parts and softwares-

- Machine learning model
- ESP32 camera module (2 pcs)
- Sonar sensor
- 16x2 LCD display and 12C LCD display driver
- Active buzzer
- Servo motor MG995
- ATmega328P Microcontroller
- 16MHz crystal oscillator
- Resistor (10K ohm; 1 pcs)
- capacitor (22uF; 2 pcs)
- IC L7805CV Voltage Regulator (2 pcs)
- Arduino IDE
- Connecting wire
- Battery (3.9v; 2 pcs)
- Battery holder
- Glue gun
- Soldering iron and solder

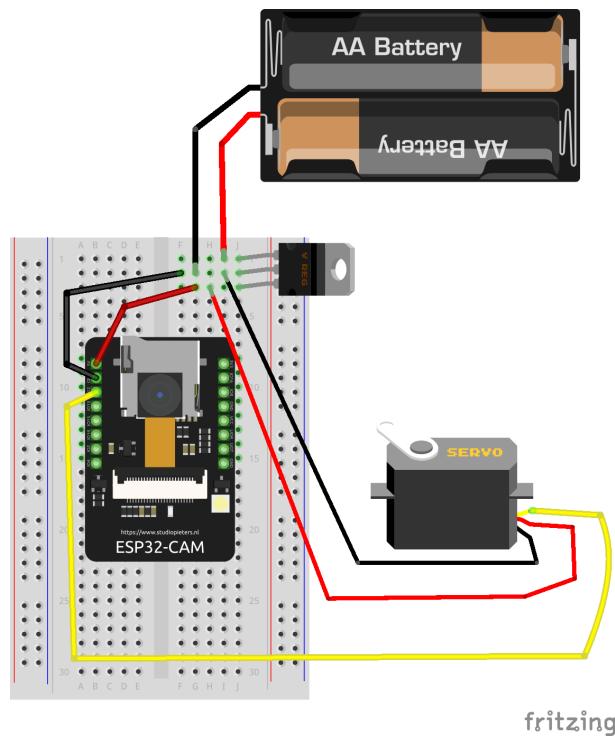


Figure 1: Circuit diagram for bottle detection

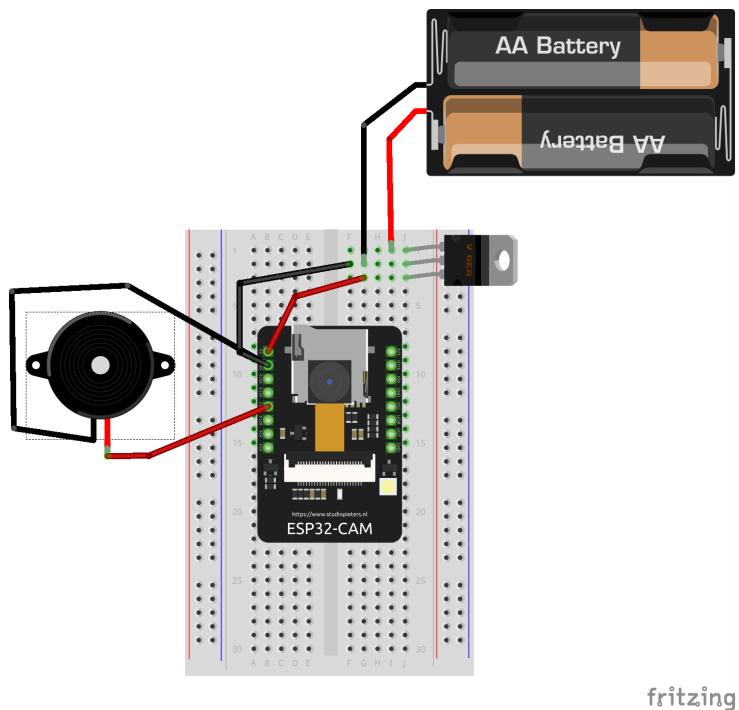


Figure 2: Circuit diagram for QR code scan.

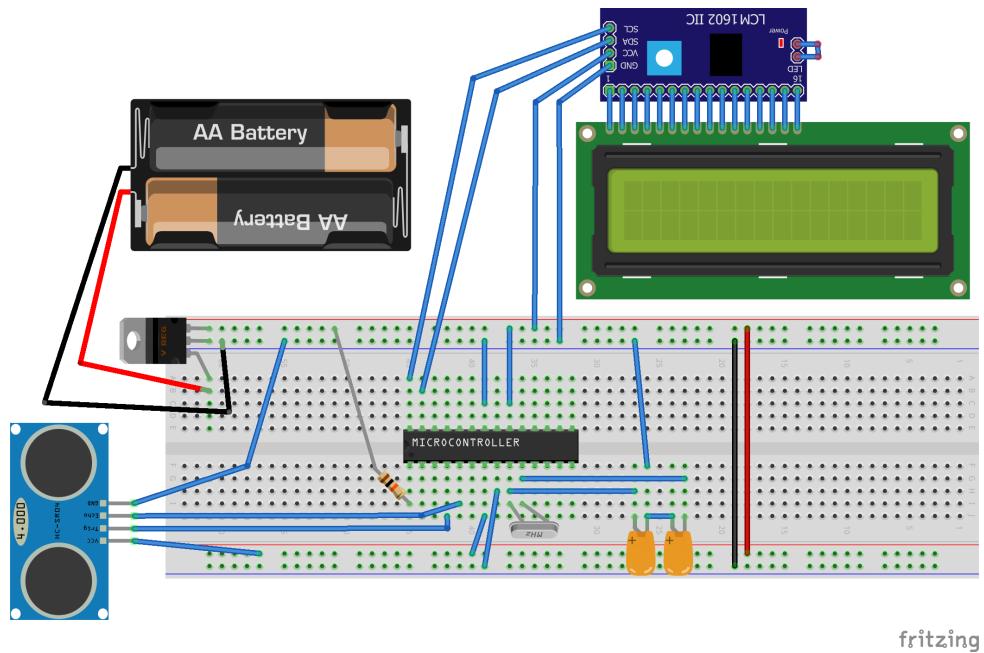


Figure 3: Circuit diagram for bins fullness detection.

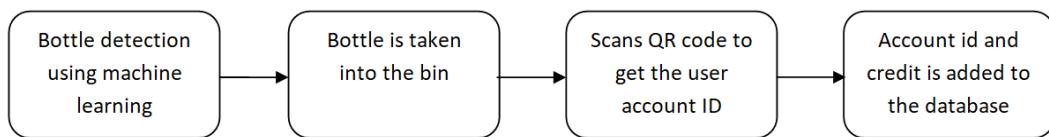


Figure 4: Block diagram for bottle detection and QR code scan.

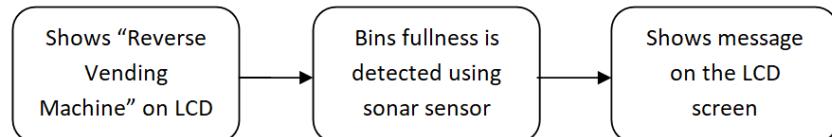


Figure 5: Block diagram for bins fullness detection.

## ESP32-cam

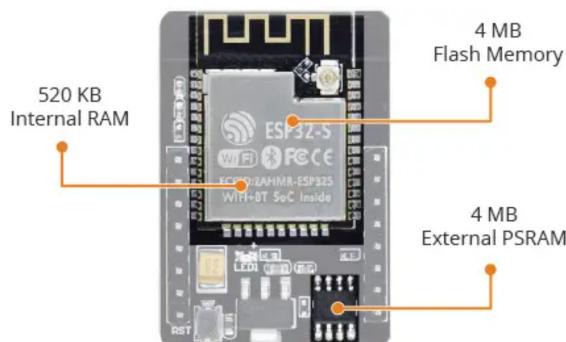
### Hardware overview

The heart of the ESP32-CAM is an ESP32-S System-on-Chip (SoC) from Ai-Thinker. Being an SoC, the ESP32-S chip contains an entire computer—the microprocessor, RAM, storage, and peripherals—on a single chip. While the chip's capabilities are quite impressive, the ESP32-CAM development board adds even more features to the mix.

- **The ESP32-S Processor:** The ESP32-CAM equips the ESP32-S surface-mount printed circuit board module from Ai-Thinker. It is equivalent to Espressif's ESP-WROOM-32 module (same form factor and general specifications).



- **The Memory:** Memory is paramount for complex tasks, so the ESP32-S has a full 520 kilobytes of internal RAM, which resides on the same die as the rest of the chip's components.



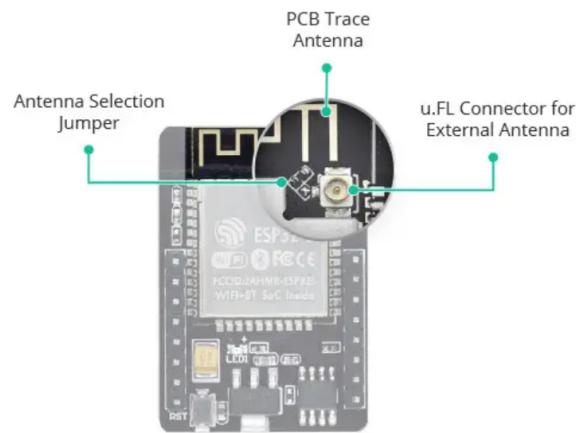
- **The Camera:** The OV2640 camera sensor on the ESP32-CAM is what sets it apart from other ESP32 development boards and makes it ideal for use in video projects like a video doorbell or nanny cam.



- **The Storage:** The addition of a microSD card slot on the ESP32-CAM is a nice bonus. This allows for limitless expansion, making it a great little board for data loggers or image capture.



- **The Antenna:** The ESP32-CAM comes with an on-board PCB trace antenna as well as a u.FL connector for connecting an external antenna. An Antenna Selection jumper (zero-ohm resistor) allows you to choose between the two options.

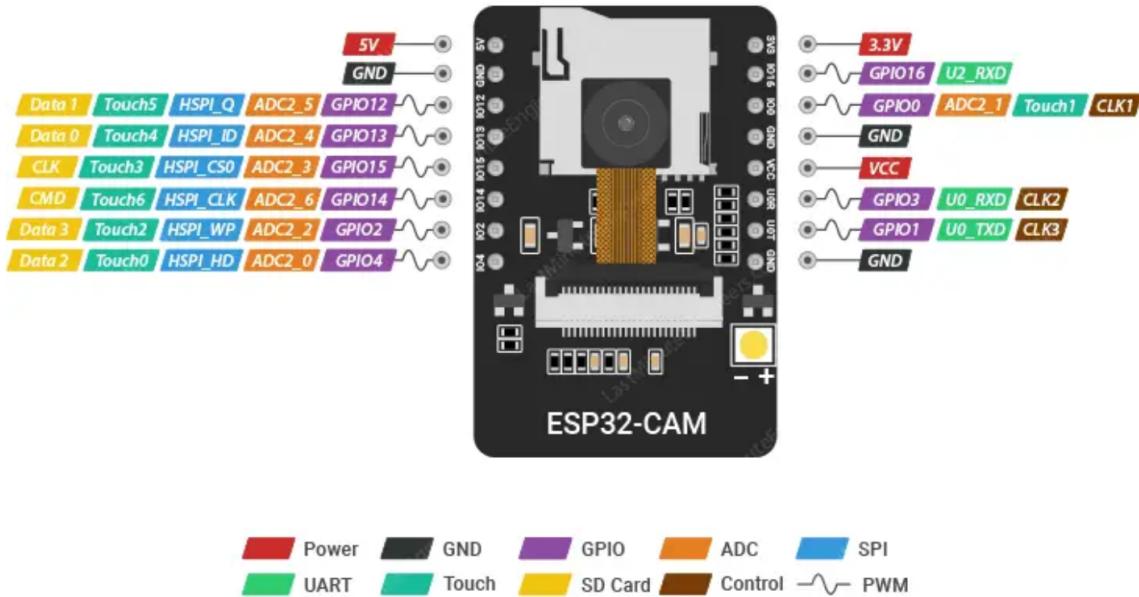


- **LEDs:** The ESP32-CAM has a white square LED. It is intended to be used as a camera flash, but it can also be used for general illumination.



## ESP32-cam pinout

The ESP32-CAM has 16 pins in total. For convenience, pins with similar functionality are grouped together. The pinout is as follows:

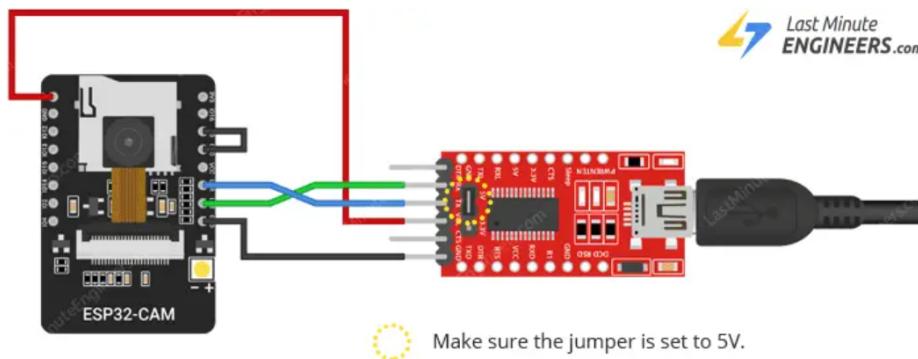


ESP32-CAM Pinout



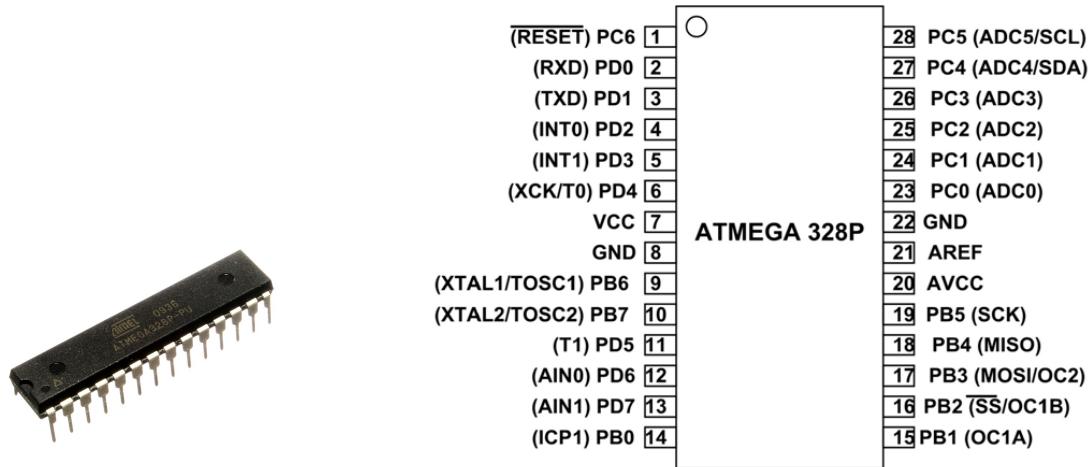
## Programming of ESP32-cam Using FTDI Adopter

If you've decided to use the FTDI adapter, here's how you connect it to the ESP32-CAM module.



## ATmega328P Microcontroller

ATMEGA328P is high performance, low power controller from Microchip. ATMEGA328P is an 8-bit microcontroller based on AVR RISC architecture. It is the most popular of all AVR controllers as it is used in ARDUINO boards.



## Servo Motor

A servo motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. Servo motors are rated in kg/cm (kilogram per centimeter) most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm tells you how much weight your servo motor can lift at a particular distance.



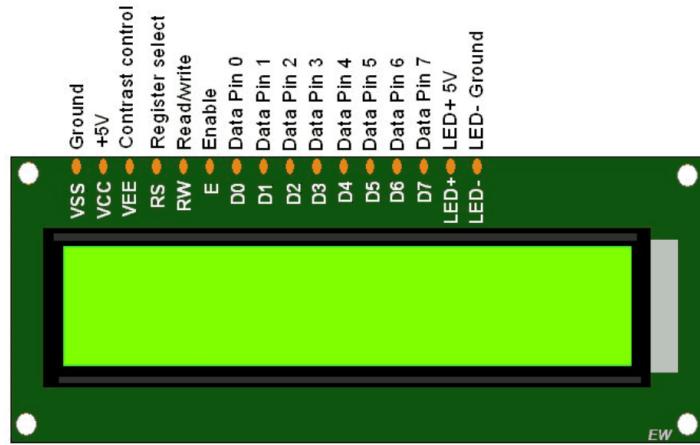
## Sonar Sensor

Ultrasonic sensors are used to detect the distance between the sensor and an object. It does this by emitting ultrasonic sound waves. The sound waves will then reflect off an object and return to the sensor. The wave will be converted into an electrical signal.



## 16x2 LCD display

LCDs (Liquid Crystal Displays) are used in embedded system applications for displaying various parameters and status of the system. LCD 16x2 is a 16-pin device that has 2 rows that can accommodate 16 characters each. LCD 16x2 can be used in 4-bit mode or 8-bit mode. It is also possible to create custom characters. It has 8 data lines and 3 control lines that can be used for control purposes.



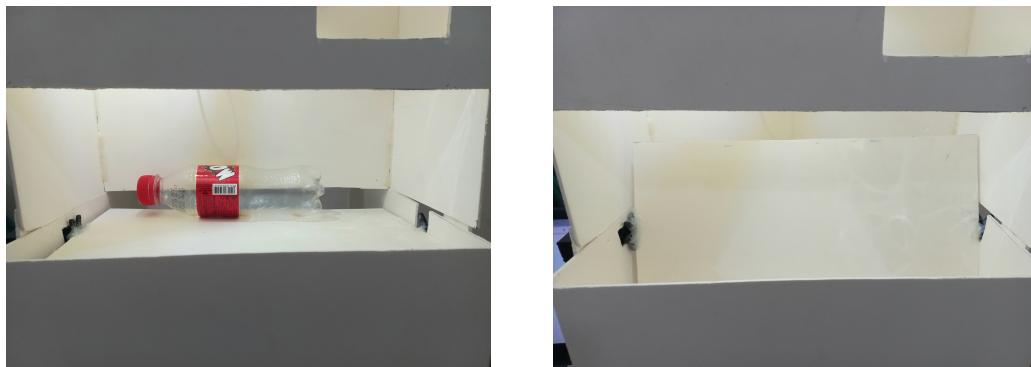
## Working Process

1. After placing the bottle into the compartment ESP-32 cam uses machine learning model[1] to detect the bottle.

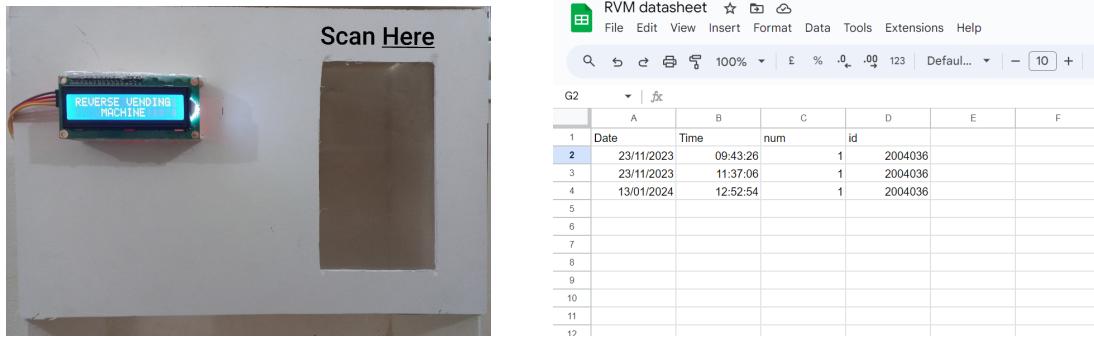


<input type="button" value="Restart"/>	<input type="button" value="Start Detect"/>	
Object	<input type="button" value="bottle"/>	0
ScoreLimit	<input type="button" value="0"/>	
MirrorImage	<input type="button" value="yes"/>	
Resolution	<input type="button" value="QVGA(320x240)"/>	
Flash	<input type="button" value=""/>	
Quality	<input type="button" value=""/>	
Brightness	<input type="button" value=""/>	
Contrast	<input type="button" value=""/>	
Rotate	<input type="button" value="0deg"/>	

When a bottle is detected, the servo motor moves 90 degree to drop the bottle into the bin.



- Upon accepting the bottle a QR code containing users account ID is scanned via 2nd ESP-32 cam. When the QR code is scanned the buzzer alerts the user that the scan is done. Then the credit gets added to their account.



- When the sonar detects anything to it's 10cm range, "The bin is full" message is shown onto the LCD display. Initially the display shows text "Reverse Vending Machine".



# Implementation Code

## CAM1 Code(Bottle Detection)

```
1 const char* ssid      = "monir zx online";
2 const char* password  = "01718146157";    // http://192.168.
3                                         .125.77
4
5 const char* apssid    = "ESP32-CAM";
6 const char* appassword = "12345678";
7
8 #include <WiFi.h>
9 #include <Wire.h>
10 #include <WiFiClientSecure.h>
11 #include "esp_camera.h"
12 #include "soc/soc.h"
13 #include "soc/rtc_cntl_reg.h"
14 #include <ESP32Servo.h>
15 #include <HTTPClient.h>
16
17
18 String ID = "AKfycbyC39YkVjLq7lJavRwuKNP70LZgd0-
19                                         n3eT2j0HcwPdfTN2J-peTtAtYt5RmVYB0D9I"; //-->
20                                         spreadsheet script ID
21 const char* host = "script.google.com";
22
23 Servo myservo; // create servo object to control a servo
24 // twelve servo objects can be created on most boards
25
26 String Feedback="";
```

```

25
26 String Command="" , cmd="" , P1="" , P2="" , P3="" , P4="" , P5="" , P6=
    "" , P7="" , P8="" , P9="" ;
27
28 byte ReceiveState=0 , cmdState=1 , strState=1 , questionstate=0 ,
    equalstate=0 , semicolonstate=0 ;
29
30 #define CAMERA_MODEL_AI_THINKER
31 #define PWDN_GPIO_NUM      32
32 #define RESET_GPIO_NUM     -1
33 #define XCLK_GPIO_NUM       0
34 #define SIOD_GPIO_NUM      26
35 #define SIOC_GPIO_NUM      27
36
37 #define Y9_GPIO_NUM         35
38 #define Y8_GPIO_NUM         34
39 #define Y7_GPIO_NUM         39
40 #define Y6_GPIO_NUM         36
41 #define Y5_GPIO_NUM         21
42 #define Y4_GPIO_NUM         19
43 #define Y3_GPIO_NUM         18
44 #define Y2_GPIO_NUM         5
45 #define VSYNC_GPIO_NUM      25
46 #define HREF_GPIO_NUM       23
47 #define PCLK_GPIO_NUM       22
48
49 #define LED_BUILTIN 4
50
51 WiFiServer server(80);
52

```

```

53
54 void update_google_sheet()
55 {
56     // Serial.print("connecting to ");
57     // Serial.println(host);
58
59     // Use WiFiClient class to create TCP connections
60     WiFiClientSecure client;
61     const int httpPort = 443; // 80 is for HTTP / 443 is for
62     // HTTPS !
63
64
65     client.setInsecure(); // this is the magical line that
66     // makes everything work
67
68 }
69
70 //-----Processing
71 // data and sending data
72 String url = "/macros/s/" + ID + "/exec?num=";
73
74 url += 1;
75 // Serial.print("Requesting URL: ");
76 // Serial.println(url);
77
78 // This will send the request to the server
79 client.print(String("GET ") + url + " HTTP/1.1\r\n" +
80             "Host: " + host + "\r\n" +

```

```

80                         "Connection: close\r\n\r\n");
81
82     // Serial.println();
83     // Serial.println("closing connection");
84     delay(90000);
85 }
86
87
88 void motor(){
89     digitalWrite(LED_BUILTIN, LOW);
90     myservo.write(90);           // tell servo to go to
91     position in variable 'pos'
92     Serial.println("motor moved 1");
93     delay(2000);
94
95     myservo.write(0);
96     Serial.println("motor moved 2");
97     update_google_sheet();
98
99 }
100
101 void ExecuteCommand()
102 {
103     // Serial.println("");
104     // Serial.println("Command: "+Command);
105     if (cmd!="getstill") {
106         Serial.println("cmd= "+cmd+" ,P1= "+P1+" ,P2= "+P2+" ,
107         P3= "+P3+" ,P4= "+P4+" ,P5= "+P5+" ,P6= "+P6+" ,P7= "+P
108         7+" ,P8= "+P8+" ,P9= "+P9);

```

```

107     Serial.println("");
108     if(P1=="bottle") {
109         motor();
110     }
111 }
112
113 if (cmd=="your cmd") {
114     // You can do anything.
115     // Feedback=<font color=\"red\">Hello World</font>;
116 }
117 else if (cmd=="ip") {
118     Feedback="AP IP: "+WiFi.softAPIP().toString();
119     Feedback+=" , ";
120     Feedback+="STA IP: "+WiFi.localIP().toString();
121 }
122 else if (cmd=="mac") {
123     Feedback="STA MAC: "+WiFi.macAddress();
124 }
125 else if (cmd=="resetwifi") {
126     WiFi.begin(P1.c_str(), P2.c_str());
127     Serial.print("Connecting to ");
128     Serial.println(P1);
129     long int StartTime=millis();
130     while (WiFi.status() != WL_CONNECTED)
131     {
132         delay(500);
133         if ((StartTime+5000) < millis()) break;
134     }
135     Serial.println("");
136     Serial.println("STAIP: "+WiFi.localIP().toString());

```

```

137     Feedback="STAIP: "+WiFi.localIP().toString();
138 }
139 else if (cmd=="restart") {
140     ESP.restart();
141 }
142 else if (cmd=="digitalwrite") {
143     ledcDetachPin(P1.toInt());
144     pinMode(P1.toInt(), OUTPUT);
145     digitalWrite(P1.toInt(), P2.toInt());
146 }
147 else if (cmd=="analogwrite") {
148     if (P1=="4") {
149         ledcAttachPin(4, 4);
150         ledcSetup(4, 5000, 8);
151         ledcWrite(4,P2.toInt());
152     }
153     else {
154         ledcAttachPin(P1.toInt(), 5);
155         ledcSetup(5, 5000, 8);
156         ledcWrite(5,P2.toInt());
157     }
158 }
159 else if (cmd=="flash") {
160     ledcAttachPin(4, 4);
161     ledcSetup(4, 5000, 8);
162
163     int val = P1.toInt();
164     ledcWrite(4,val);
165 }
166 else if (cmd=="framesize") {

```

```

167     sensor_t * s = esp_camera_sensor_get();
168
169     if (P1=="QQVGA")
170
171         s->set_framesize(s, FRAMESIZE_QQVGA);
172
173     else if (P1=="HQVGA")
174
175         s->set_framesize(s, FRAMESIZE_HQVGA);
176
177     else if (P1=="QVGA")
178
179         s->set_framesize(s, FRAMESIZE_QVGA);
180
181     else if (P1=="CIF")
182
183         s->set_framesize(s, FRAMESIZE_CIF);
184
185     else if (P1=="VGA")
186
187         s->set_framesize(s, FRAMESIZE_VGA);
188
189     else if (P1=="SVGA")
190
191         s->set_framesize(s, FRAMESIZE_SVGA);
192
193     else if (P1=="XGA")
194
195         s->set_framesize(s, FRAMESIZE_XGA);
196
197     else if (P1=="SXGA")
198
199         s->set_framesize(s, FRAMESIZE_SXGA);
200
201     else if (P1=="UXGA")
202
203         s->set_framesize(s, FRAMESIZE_UXGA);
204
205     else
206
207         s->set_framesize(s, FRAMESIZE_QVGA);
208
209     }
210
211     else if (cmd=="quality") {
212
213         sensor_t * s = esp_camera_sensor_get();
214
215         int val = P1.toInt();
216
217         s->set_quality(s, val);
218
219     }
220
221     else if (cmd=="contrast") {
222
223         sensor_t * s = esp_camera_sensor_get();
224
225         int val = P1.toInt();
226
227     }

```

```

197     s->set_contrast(s, val);
198 }
199 else if (cmd=="brightness") {
200     sensor_t * s = esp_camera_sensor_get();
201     int val = P1.toInt();
202     s->set_brightness(s, val);
203 }
204 else if (cmd=="serial") {
205     Serial.println(P1);
206 }
207 else if (cmd=="detectCount") {
208     Serial.println(P1+" = "+P2);
209 }
210 else if (cmd=="tcp") {
211     String domain=P1;
212     int port=P2.toInt();
213     String request=P3;
214     int wait=P4.toInt();           // wait = 0 or 1
215
216     if ((port==443)|| (domain.indexOf("https")==0)|| (domain
217 .indexOf("HTTPS")==0))
218         Feedback=tcp_https(domain,request,port,wait);
219     else
220         Feedback=tcp_http(domain,request,port,wait);
221 }
222 else if (cmd=="linenotify") {      //message=xxx&
223     stickerPackageId=xxx&stickerId=xxx
224     String token = P1;
225     String request = P2;
226     Feedback=LineNotify(token,request,1);

```

```

225     if (Feedback.indexOf("status") != -1) {
226
227         int s=Feedback.indexOf("{");
228
229         Feedback=Feedback.substring(s);
230
231         int e=Feedback.indexOf("}");
232
233         Feedback=Feedback.substring(0,e);
234
235         Feedback.replace("\\"", "");
236
237         Feedback.replace("{", "");
238
239         Feedback.replace("}", "");
240
241     }
242
243 }
244
245 void setup() {
246
247     WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
248
249     Serial.begin(115200);
250
251     Serial.setDebugOutput(true);
252
253     // Serial.println();
254
255
256     pinMode(LED_BUILTIN, OUTPUT); //Specify that LED pin is
257     output

```

```
253     myservo.attach(12);  
254  
255     camera_config_t config;  
256     config.ledc_channel = LEDC_CHANNEL_0;  
257     config.ledc_timer = LEDC_TIMER_0;  
258     config.pin_d0 = Y2_GPIO_NUM;  
259     config.pin_d1 = Y3_GPIO_NUM;  
260     config.pin_d2 = Y4_GPIO_NUM;  
261     config.pin_d3 = Y5_GPIO_NUM;  
262     config.pin_d4 = Y6_GPIO_NUM;  
263     config.pin_d5 = Y7_GPIO_NUM;  
264     config.pin_d6 = Y8_GPIO_NUM;  
265     config.pin_d7 = Y9_GPIO_NUM;  
266     config.pin_xclk = XCLK_GPIO_NUM;  
267     config.pin_pclk = PCLK_GPIO_NUM;  
268     config.pin_vsync = VSYNC_GPIO_NUM;  
269     config.pin_href = HREF_GPIO_NUM;  
270     config.pin_sscb_sda = SIOD_GPIO_NUM;  
271     config.pin_sscb_scl = SIOC_GPIO_NUM;  
272     config.pin_pwdn = PWDN_GPIO_NUM;  
273     config.pin_reset = RESET_GPIO_NUM;  
274     config.xclk_freq_hz = 20000000;  
275     config.pixel_format = PIXFORMAT_JPEG;  
276     //init with high specs to pre-allocate larger buffers  
277     if(psramFound()){  
278         config.frame_size = FRAMESIZE_UXGA;  
279         config.jpeg_quality = 10; //0-63 lower number means  
higher quality  
280         config.fb_count = 2;  
281     } else {
```

```

282     config.frame_size = FRAMESIZE_SVGA;
283     config.jpeg_quality = 12; //0-63 lower number means
284     higher quality
285
286     config.fb_count = 1;
287 }
288
289 // camera init
290
291 esp_err_t err = esp_camera_init(&config);
292
293 if (err != ESP_OK) {
294     Serial.printf("Camera init failed with error 0x%x",
295     err);
296     delay(1000);
297     ESP.restart();
298 }
299
300
301 //drop down frame size for higher initial frame rate
302 sensor_t * s = esp_camera_sensor_get();
303 s->set_framesize(s, FRAMESIZE_QVGA); //UXGA|SXGA|XGA|
304 SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
305
306 ledcAttachPin(4, 4);
307 ledcSetup(4, 5000, 8);
308
309 WiFi.mode(WIFI_AP_STA);
310
311
312 //WiFi.config(IPAddress(192, 168, 201, 100), IPAddress(1
313 92, 168, 201, 2), IPAddress(255, 255, 255, 0));
314
315 WiFi.begin(ssid, password);
316
317 delay(1000);

```

```

308     Serial.println("");
309     Serial.print("Connecting to ");
310     Serial.println(ssid);
311
312     long int StartTime=millis();
313     while (WiFi.status() != WL_CONNECTED)
314     {
315         delay(500);
316         if ((StartTime+10000) < millis()) break;
317     }
318
319     if (WiFi.status() == WL_CONNECTED) {
320         WiFi.softAP((WiFi.localIP().toString()+"_"+(String)
321             apssid).c_str(), appassword);
322         Serial.println("");
323         Serial.println("STAIP address: ");
324         Serial.println(WiFi.localIP());
325
326         for (int i=0;i<5;i++) {
327             ledcWrite(4,10);
328             delay(200);
329             ledcWrite(4,0);
330             delay(200);
331         }
332     else {
333         WiFi.softAP((WiFi.softAPIP().toString()+"_"+(String)
334             apssid).c_str(), appassword);
335
336         for (int i=0;i<2;i++) {

```

```

336     ledcWrite(4, 10);
337
338     delay(1000);
339
340     ledcWrite(4, 0);
341
342     delay(1000);
343
344     }
345
346     //WiFi.softAPConfig(IPAddress(192, 168, 4, 1), IPAddress
347     //                      (192, 168, 4, 1), IPAddress(255, 255, 255, 0));
348     Serial.println("");
349     Serial.println("APIP address: ");
350     Serial.println(WiFi.softAPIP());
351
352     pinMode(4, OUTPUT);
353     digitalWrite(4, LOW);
354
355     server.begin();
356
357     }
358
359     static const char PROGMEM INDEX_HTML[] = R"rawliteral(
360
361         <!DOCTYPE html>
362
363         <head>
364
365             <meta charset="utf-8">
366
367             <meta name="viewport" content="width=device-width,
368                 initial-scale=1">
369
370             <script src="https://ajax.googleapis.com/ajax/libs/
371                 jquery/1.8.0/jquery.min.js"></script>
372
373             <script src="https://cdn.jsdelivr.net/npm/@tensorflow/
374                 tfjs@1.3.1/dist/tf.min.js"> </script>
375
376             <script src="https://cdn.jsdelivr.net/npm/@tensorflow-

```

```

    models/coco-ssd@2.1.0"> </script>
362 </head><body>
363 <img id="ShowImage" src="" style="display:none">
364 <canvas id="canvas" width="0" height="0"></canvas>
365 <table>
366 <tr>
367     <td><input type="button" id="restart" value="Restart">
368     </td>
369     <td colspan="2"><input type="button" id="getStill" value="Start Detect" style="display:none"></td>
370 </tr>
371 <tr>
372     <td>Object</td>
373     <td colspan="2">
374         <select id="object" onchange="count.innerHTML='';">
375             <option value="bottle">bottle</option>
376
377             </select>
378         </td>
379         <td><span id="count" style="color:red"><span></td>
380     </tr>
381     <tr>
382         <td>ScoreLimit</td>
383         <td colspan="2">
384             <select id="score">
385                 <option value="1.0">1</option>
386                 <option value="0.9">0.9</option>
387                 <option value="0.8">0.8</option>

```

```

388         <option value="0.7">0.7</option>
389         <option value="0.6">0.6</option>
390         <option value="0.5">0.5</option>
391         <option value="0.4">0.4</option>
392         <option value="0.3">0.3</option>
393         <option value="0.2">0.2</option>
394         <option value="0.1">0.1</option>
395         <option value="0" selected="selected">0</option>
396     </select>
397   </td>
398 </tr>
399 <tr>
400   <td>MirrorImage</td>
401   <td colspan="2">
402     <select id="mirrorimage">
403       <option value="1">yes</option>
404       <option value="0">no</option>
405     </select>
406   </td>
407 </tr>
408 <tr>
409   <td>Resolution</td>
410   <td colspan="2">
411     <select id="framesize">
412       <option value="UXGA">UXGA (1600x1200)</option>
413       <option value="SXGA">SXGA (1280x1024)</option>
414       <option value="XGA">XGA (1024x768)</option>
415       <option value="SVGA">SVGA (800x600)</option>
416       <option value="VGA">VGA (640x480)</option>
417       <option value="CIF">CIF (400x296)</option>

```

```

418         <option value="QVGA" selected="selected">QVGA (320
419             x240)</option>
420
421             <option value="HQVGA">HQVGA (240x176)</option>
422             <option value="QQVGA">QQVGA (160x120)</option>
423
424         </select>
425     </td>
426
427 </tr>
428
429 <tr>
430
431     <td>Flash</td>
432
433     <td colspan="2"><input type="range" id="flash" min="0"
434             max="255" value="0"></td>
435
436 </tr>
437
438 <tr>
439
440     <td>Quality</td>
441
442     <td colspan="2"><input type="range" id="quality" min="
443             10" max="63" value="10"></td>
444
445 </tr>
446
447 <tr>
448
449     <td>Brightness</td>
450
451     <td colspan="2"><input type="range" id="brightness"
452             min="-2" max="2" value="0"></td>
453
454 </tr>
455
456 <tr>
457
458     <td>Contrast</td>
459
460     <td colspan="2"><input type="range" id="contrast" min=
461             "-2" max="2" value="0"></td>
462
463 </tr>
464
465 <tr><td>Rotate</td>
466
467     <td align="left" colspan="2">
468
469         <select onchange="document.getElementById('canvas').value = this.value">

```

```

        style.transform='rotate(' +this.value+')' ;">
    443         <option value="0deg">0deg</option>
    444         <option value="90deg">90deg</option>
    445         <option value="180deg">180deg</option>
    446         <option value="270deg">270deg</option>
    447     </select>
    448 </td>
    449 </tr>
    450 </table>
    451 <div id="result" style="color:red"><div>
    452 </body>
    453 </html>
    454
    455 <script>
    456     var getStill = document.getElementById('getStill');
    457     var ShowImage = document.getElementById('ShowImage');
    458     var canvas = document.getElementById("canvas");
    459     var context = canvas.getContext("2d");
    460     var object = document.getElementById('object');
    461     var score = document.getElementById("score");
    462     var mirrorimage = document.getElementById("mirrorimage");
    463
    464     var count = document.getElementById('count');
    465     var result = document.getElementById('result');
    466     var flash = document.getElementById('flash');
    467     var lastValue="";
    468     var myTimer;
    469     var restartCount=0;
    470     var Model;
    471
    472     getStill.onclick = function (event) {

```

```

471         clearInterval(myTimer);
472
473         myTimer = setInterval(function(){error_handle()
474     ;},5000);
475
476         ShowImage.src=location.origin+'/?getstill='+Math.
477         random();
478
479     }
480
481     function error_handle() {
482
483         restartCount++;
484
485         clearInterval(myTimer);
486
487         if (restartCount<=2) {
488
489             result.innerHTML = "Get still error. <br>Restart
490             ESP32-CAM "+restartCount+" times.";
491
492             myTimer = setInterval(function(){getStill.click()
493         ;},10000);
494
495         }
496
497         else
498
499             result.innerHTML = "Get still error. <br>Please
500             close the page and check ESP32-CAM.";
501
502     }
503
504     ShowImage.onload = function (event) {
505
506         clearInterval(myTimer);
507
508         restartCount=0;
509
510         canvas.setAttribute("width", ShowImage.width);
511
512         canvas.setAttribute("height", ShowImage.height);
513
514         if (mirrorimage.value==1) {
515
516             context.translate((canvas.width + ShowImage.width)
517             / 2, 0);
518
519             context.scale(-1, 1);
520
521             context.drawImage(ShowImage, 0, 0, ShowImage.width
522             , ShowImage.height);

```

```

494         context.setTransform(1, 0, 0, 1, 0, 0);
495     }
496     else
497         context.drawImage(ShowImage, 0, 0, ShowImage.width,
498                         ShowImage.height);
499     if (Model)
500         DetectImage();
501     }
502
503     restart.onclick = function (event) {
504         fetch(location.origin+'/?restart=stop');
505     }
506     framesize.onclick = function (event) {
507         fetch(document.location.origin+'/?framesize='+this.
508               value+';stop');
509     }
510     flash.onchange = function (event) {
511         fetch(location.origin+'/?flash=' + this.value+';stop')
512     ;
513     }
514     quality.onclick = function (event) {
515         fetch(document.location.origin+'/?quality=' + this.
516               value+';stop');
517     }
518     brightness.onclick = function (event) {
519         fetch(document.location.origin+'/?brightness=' + this.
520               value+';stop');
521     }
522     contrast.onclick = function (event) {

```

```

519         fetch(document.location.origin+'/?contrast='+this.
520               value+';stop');
521
522     function ObjectDetect() {
523         result.innerHTML = "Please wait for loading model.";
524         cocoSsd.load().then(cocoSsd_Model => {
525             Model = cocoSsd_Model;
526             result.innerHTML = "";
527             getStill.style.display = "block";
528             getStill.click();
529         });
530     }
531
532     function DetectImage() {
533         Model.detect(canvas).then(Predictions => {
534             var s = (canvas.width>canvas.height)?canvas.width:
535                   canvas.height;
536
537             //console.log('Predictions: ', Predictions);
538             if (Predictions.length>0) {
539                 result.innerHTML = "";
540                 for (var i=0;i<Predictions.length;i++) {
541                     const x = Predictions[i].bbox[0];
542                     const y = Predictions[i].bbox[1];
543                     const width = Predictions[i].bbox[2];
544                     const height = Predictions[i].bbox[3];
545                     context.lineWidth = Math.round(s/200);
546                     context.strokeStyle = "#00FFFF";

```

```

547         context.beginPath();
548
549         context.rect(x, y, width, height);
550
551         context.stroke();
552
553         context.lineWidth = "2";
554
555         context.fillStyle = "red";
556
557         context.font = Math.round(s/30) + "px Arial";
558
559         context.fillText(Predictions[i].class, x, y);
560
561         //context.fillText(i, x, y);
562
563         result.innerHTML+= "[ "+i+" ] "+Predictions[i]
564         .class+", "+Math.round(Predictions[i].score*100)+"%, "
565         +Math.round(x)+", "+Math.round(y)+", "+Math.round(width)
566         +", "+Math.round(height)+"<br>";
567
568
569         if (Predictions[i].class==object.value&&
570             Predictions[i].score>=score.value) {
571
572             objectCount++;
573
574         }
575
576     }
577
578     count.innerHTML = objectCount;
579
580 }
581
582 else {
583
584     result.innerHTML = "Unrecognizable";
585
586     count.innerHTML = "0";
587
588 }
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
999

```

```

572         $.ajax({url: document.location.origin+'/?
573 detectCount='+object.value+';'+String(objectCount)+';
574 stop', async: false});
575     }
576     //}
577     try {
578         document.createEvent("TouchEvent");
579         setTimeout(function(){getStill.click();},250);
580     }
581     catch(e) {
582         setTimeout(function(){getStill.click();},150);
583     }
584 }
585 function getFeedback(target) {
586     var data = $.ajax({
587         type: "get",
588         dataType: "text",
589         url: target,
590         success: function(response)
591         {
592             result.innerHTML = response;
593         },
594         error: function(exception)
595         {
596             result.innerHTML = 'fail';
597         }
598     });
599 window.onload = function () { ObjectDetect(); }

```

```

600      </script>
601  ) rawliteral";
602
603
604
605 void loop() {
606   Feedback="" ; Command="" ; cmd="" ; P1="" ; P2="" ; P3="" ; P4="" ; P5
607   ="" ; P6="" ; P7="" ; P8="" ; P9="";
608   ReceiveState=0 , cmdState=1 , strState=1 , questionstate=0 ,
609   equalstate=0 , semicolonstate=0 ;
610
611   WiFiClient client = server.available();
612
613   if (client) {
614     String currentLine = "";
615
616     while (client.connected()) {
617       if (client.available()) {
618         char c = client.read();
619
620         getCommand(c);
621
622         if (c == '\n') {
623           if (currentLine.length() == 0) {
624
625             if (cmd=="getstill") {
626
627               camera_fb_t * fb = NULL ;

```

```
628     fb = esp_camera_fb_get();
629
630     if(!fb) {
631
632         // Serial.println("Camera capture failed")
633
634
635         delay(1000);
636
637         ESP.restart();
638
639     }
640
641
642     client.println("HTTP/1.1 200 OK");
643
644     client.println("Access-Control-Allow-Origin:");
645
646     *");
647
648     client.println("Access-Control-Allow-Headers");
649
650     : Origin, X-Requested-With, Content-Type, Accept");
651
652     client.println("Access-Control-Allow-Methods");
653
654     : GET,POST,PUT,DELETE,OPTIONS");
655
656     client.println("Content-Type: image/jpeg");
657
658     client.println("Content-Disposition: form-
659     data; name=\"imageFile\"; filename=\"picture.jpg\"");
660
661     client.println("Content-Length: " + String(
662     fb->len));
663
664     client.println("Connection: close");
665
666     client.println();
667
668
669     uint8_t *fbBuf = fb->buf;
670
671     size_t fbLen = fb->len;
672
673     for (size_t n=0;n<fbLen;n=n+1024) {
674
675         if (n+1024<fbLen) {
676
677             client.write(fbBuf, 1024);
678
679             fbBuf += 1024;
680
681         }
682
683     }
684
685
686     fb = esp_camera_fb_get();
687
688     if(!fb) {
689
690         // Serial.println("Camera capture failed")
691
692         delay(1000);
693
694         ESP.restart();
695
696     }
697
698
699     client.println("HTTP/1.1 200 OK");
700
701     client.println("Access-Control-Allow-Origin:");
702
703     *");
704
705     client.println("Access-Control-Allow-Headers");
706
707     : Origin, X-Requested-With, Content-Type, Accept");
708
709     client.println("Access-Control-Allow-Methods");
710
711     : GET,POST,PUT,DELETE,OPTIONS");
712
713     client.println("Content-Type: image/jpeg");
714
715     client.println("Content-Disposition: form-
716     data; name=\"imageFile\"; filename=\"picture.jpg\"");
717
718     client.println("Content-Length: " + String(
719     fb->len));
720
721     client.println("Connection: close");
722
723     client.println();
724
725
726     uint8_t *fbBuf = fb->buf;
727
728     size_t fbLen = fb->len;
729
730     for (size_t n=0;n<fbLen;n=n+1024) {
731
732         if (n+1024<fbLen) {
733
734             client.write(fbBuf, 1024);
735
736             fbBuf += 1024;
737
738         }
739
740     }
741
742
743     fb = esp_camera_fb_get();
744
745     if(!fb) {
746
747         // Serial.println("Camera capture failed")
748
749         delay(1000);
750
751         ESP.restart();
752
753     }
754
755
756     client.println("HTTP/1.1 200 OK");
757
758     client.println("Access-Control-Allow-Origin:");
759
760     *");
761
762     client.println("Access-Control-Allow-Headers");
763
764     : Origin, X-Requested-With, Content-Type, Accept");
765
766     client.println("Access-Control-Allow-Methods");
767
768     : GET,POST,PUT,DELETE,OPTIONS");
769
770     client.println("Content-Type: image/jpeg");
771
772     client.println("Content-Disposition: form-
773     data; name=\"imageFile\"; filename=\"picture.jpg\"");
774
775     client.println("Content-Length: " + String(
776     fb->len));
777
778     client.println("Connection: close");
779
780     client.println();
781
782
783     uint8_t *fbBuf = fb->buf;
784
785     size_t fbLen = fb->len;
786
787     for (size_t n=0;n<fbLen;n=n+1024) {
788
789         if (n+1024<fbLen) {
790
791             client.write(fbBuf, 1024);
792
793             fbBuf += 1024;
794
795         }
796
797     }
798
799
800     fb = esp_camera_fb_get();
801
802     if(!fb) {
803
804         // Serial.println("Camera capture failed")
805
806         delay(1000);
807
808         ESP.restart();
809
810     }
811
812
813     client.println("HTTP/1.1 200 OK");
814
815     client.println("Access-Control-Allow-Origin:");
816
817     *");
818
819     client.println("Access-Control-Allow-Headers");
820
821     : Origin, X-Requested-With, Content-Type, Accept");
822
823     client.println("Access-Control-Allow-Methods");
824
825     : GET,POST,PUT,DELETE,OPTIONS");
826
827     client.println("Content-Type: image/jpeg");
828
829     client.println("Content-Disposition: form-
830     data; name=\"imageFile\"; filename=\"picture.jpg\"");
831
832     client.println("Content-Length: " + String(
833     fb->len));
834
835     client.println("Connection: close");
836
837     client.println();
838
839
840     uint8_t *fbBuf = fb->buf;
841
842     size_t fbLen = fb->len;
843
844     for (size_t n=0;n<fbLen;n=n+1024) {
845
846         if (n+1024<fbLen) {
847
848             client.write(fbBuf, 1024);
849
850             fbBuf += 1024;
851
852         }
853
854     }
855
856
857     fb = esp_camera_fb_get();
858
859     if(!fb) {
860
861         // Serial.println("Camera capture failed")
862
863         delay(1000);
864
865         ESP.restart();
866
867     }
868
869
870     client.println("HTTP/1.1 200 OK");
871
872     client.println("Access-Control-Allow-Origin:");
873
874     *");
875
876     client.println("Access-Control-Allow-Headers");
877
878     : Origin, X-Requested-With, Content-Type, Accept");
879
880     client.println("Access-Control-Allow-Methods");
881
882     : GET,POST,PUT,DELETE,OPTIONS");
883
884     client.println("Content-Type: image/jpeg");
885
886     client.println("Content-Disposition: form-
887     data; name=\"imageFile\"; filename=\"picture.jpg\"");
888
889     client.println("Content-Length: " + String(
890     fb->len));
891
892     client.println("Connection: close");
893
894     client.println();
895
896
897     uint8_t *fbBuf = fb->buf;
898
899     size_t fbLen = fb->len;
900
901     for (size_t n=0;n<fbLen;n=n+1024) {
902
903         if (n+1024<fbLen) {
904
905             client.write(fbBuf, 1024);
906
907             fbBuf += 1024;
908
909         }
910
911     }
912
913
914     fb = esp_camera_fb_get();
915
916     if(!fb) {
917
918         // Serial.println("Camera capture failed")
919
920         delay(1000);
921
922         ESP.restart();
923
924     }
925
926
927     client.println("HTTP/1.1 200 OK");
928
929     client.println("Access-Control-Allow-Origin:");
930
931     *");
932
933     client.println("Access-Control-Allow-Headers");
934
935     : Origin, X-Requested-With, Content-Type, Accept");
936
937     client.println("Access-Control-Allow-Methods");
938
939     : GET,POST,PUT,DELETE,OPTIONS");
940
941     client.println("Content-Type: image/jpeg");
942
943     client.println("Content-Disposition: form-
944     data; name=\"imageFile\"; filename=\"picture.jpg\"");
945
946     client.println("Content-Length: " + String(
947     fb->len));
948
949     client.println("Connection: close");
950
951     client.println();
952
953
954     uint8_t *fbBuf = fb->buf;
955
956     size_t fbLen = fb->len;
957
958     for (size_t n=0;n<fbLen;n=n+1024) {
959
960         if (n+1024<fbLen) {
961
962             client.write(fbBuf, 1024);
963
964             fbBuf += 1024;
965
966         }
967
968     }
969
970
971     fb = esp_camera_fb_get();
972
973     if(!fb) {
974
975         // Serial.println("Camera capture failed")
976
977         delay(1000);
978
979         ESP.restart();
980
981     }
982
983
984     client.println("HTTP/1.1 200 OK");
985
986     client.println("Access-Control-Allow-Origin:");
987
988     *");
989
990     client.println("Access-Control-Allow-Headers");
991
992     : Origin, X-Requested-With, Content-Type, Accept");
993
994     client.println("Access-Control-Allow-Methods");
995
996     : GET,POST,PUT,DELETE,OPTIONS");
997
998     client.println("Content-Type: image/jpeg");
999
1000    client.println("Content-Disposition: form-
1001    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1002
1003    client.println("Content-Length: " + String(
1004    fb->len));
1005
1006    client.println("Connection: close");
1007
1008    client.println();
1009
1010
1011    uint8_t *fbBuf = fb->buf;
1012
1013    size_t fbLen = fb->len;
1014
1015    for (size_t n=0;n<fbLen;n=n+1024) {
1016
1017        if (n+1024<fbLen) {
1018
1019            client.write(fbBuf, 1024);
1020
1021            fbBuf += 1024;
1022
1023        }
1024
1025    }
1026
1027
1028    fb = esp_camera_fb_get();
1029
1030    if(!fb) {
1031
1032        // Serial.println("Camera capture failed")
1033
1034        delay(1000);
1035
1036        ESP.restart();
1037
1038    }
1039
1040
1041    client.println("HTTP/1.1 200 OK");
1042
1043    client.println("Access-Control-Allow-Origin:");
1044
1045    *");
1046
1047    client.println("Access-Control-Allow-Headers");
1048
1049    : Origin, X-Requested-With, Content-Type, Accept");
1050
1051    client.println("Access-Control-Allow-Methods");
1052
1053    : GET,POST,PUT,DELETE,OPTIONS");
1054
1055    client.println("Content-Type: image/jpeg");
1056
1057    client.println("Content-Disposition: form-
1058    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1059
1060    client.println("Content-Length: " + String(
1061    fb->len));
1062
1063    client.println("Connection: close");
1064
1065    client.println();
1066
1067
1068    uint8_t *fbBuf = fb->buf;
1069
1070    size_t fbLen = fb->len;
1071
1072    for (size_t n=0;n<fbLen;n=n+1024) {
1073
1074        if (n+1024<fbLen) {
1075
1076            client.write(fbBuf, 1024);
1077
1078            fbBuf += 1024;
1079
1080        }
1081
1082    }
1083
1084
1085    fb = esp_camera_fb_get();
1086
1087    if(!fb) {
1088
1089        // Serial.println("Camera capture failed")
1090
1091        delay(1000);
1092
1093        ESP.restart();
1094
1095    }
1096
1097
1098    client.println("HTTP/1.1 200 OK");
1099
1100    client.println("Access-Control-Allow-Origin:");
1101
1102    *");
1103
1104    client.println("Access-Control-Allow-Headers");
1105
1106    : Origin, X-Requested-With, Content-Type, Accept");
1107
1108    client.println("Access-Control-Allow-Methods");
1109
1110    : GET,POST,PUT,DELETE,OPTIONS");
1111
1112    client.println("Content-Type: image/jpeg");
1113
1114    client.println("Content-Disposition: form-
1115    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1116
1117    client.println("Content-Length: " + String(
1118    fb->len));
1119
1120    client.println("Connection: close");
1121
1122    client.println();
1123
1124
1125    uint8_t *fbBuf = fb->buf;
1126
1127    size_t fbLen = fb->len;
1128
1129    for (size_t n=0;n<fbLen;n=n+1024) {
1130
1131        if (n+1024<fbLen) {
1132
1133            client.write(fbBuf, 1024);
1134
1135            fbBuf += 1024;
1136
1137        }
1138
1139    }
1140
1141
1142    fb = esp_camera_fb_get();
1143
1144    if(!fb) {
1145
1146        // Serial.println("Camera capture failed")
1147
1148        delay(1000);
1149
1150        ESP.restart();
1151
1152    }
1153
1154
1155    client.println("HTTP/1.1 200 OK");
1156
1157    client.println("Access-Control-Allow-Origin:");
1158
1159    *");
1160
1161    client.println("Access-Control-Allow-Headers");
1162
1163    : Origin, X-Requested-With, Content-Type, Accept");
1164
1165    client.println("Access-Control-Allow-Methods");
1166
1167    : GET,POST,PUT,DELETE,OPTIONS");
1168
1169    client.println("Content-Type: image/jpeg");
1170
1171    client.println("Content-Disposition: form-
1172    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1173
1174    client.println("Content-Length: " + String(
1175    fb->len));
1176
1177    client.println("Connection: close");
1178
1179    client.println();
1180
1181
1182    uint8_t *fbBuf = fb->buf;
1183
1184    size_t fbLen = fb->len;
1185
1186    for (size_t n=0;n<fbLen;n=n+1024) {
1187
1188        if (n+1024<fbLen) {
1189
1190            client.write(fbBuf, 1024);
1191
1192            fbBuf += 1024;
1193
1194        }
1195
1196    }
1197
1198
1199    fb = esp_camera_fb_get();
1200
1201    if(!fb) {
1202
1203        // Serial.println("Camera capture failed")
1204
1205        delay(1000);
1206
1207        ESP.restart();
1208
1209    }
1210
1211
1212    client.println("HTTP/1.1 200 OK");
1213
1214    client.println("Access-Control-Allow-Origin:");
1215
1216    *");
1217
1218    client.println("Access-Control-Allow-Headers");
1219
1220    : Origin, X-Requested-With, Content-Type, Accept");
1221
1222    client.println("Access-Control-Allow-Methods");
1223
1224    : GET,POST,PUT,DELETE,OPTIONS");
1225
1226    client.println("Content-Type: image/jpeg");
1227
1228    client.println("Content-Disposition: form-
1229    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1230
1231    client.println("Content-Length: " + String(
1232    fb->len));
1233
1234    client.println("Connection: close");
1235
1236    client.println();
1237
1238
1239    uint8_t *fbBuf = fb->buf;
1240
1241    size_t fbLen = fb->len;
1242
1243    for (size_t n=0;n<fbLen;n=n+1024) {
1244
1245        if (n+1024<fbLen) {
1246
1247            client.write(fbBuf, 1024);
1248
1249            fbBuf += 1024;
1250
1251        }
1252
1253    }
1254
1255
1256    fb = esp_camera_fb_get();
1257
1258    if(!fb) {
1259
1260        // Serial.println("Camera capture failed")
1261
1262        delay(1000);
1263
1264        ESP.restart();
1265
1266    }
1267
1268
1269    client.println("HTTP/1.1 200 OK");
1270
1271    client.println("Access-Control-Allow-Origin:");
1272
1273    *");
1274
1275    client.println("Access-Control-Allow-Headers");
1276
1277    : Origin, X-Requested-With, Content-Type, Accept");
1278
1279    client.println("Access-Control-Allow-Methods");
1280
1281    : GET,POST,PUT,DELETE,OPTIONS");
1282
1283    client.println("Content-Type: image/jpeg");
1284
1285    client.println("Content-Disposition: form-
1286    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1287
1288    client.println("Content-Length: " + String(
1289    fb->len));
1290
1291    client.println("Connection: close");
1292
1293    client.println();
1294
1295
1296    uint8_t *fbBuf = fb->buf;
1297
1298    size_t fbLen = fb->len;
1299
1300    for (size_t n=0;n<fbLen;n=n+1024) {
1301
1302        if (n+1024<fbLen) {
1303
1304            client.write(fbBuf, 1024);
1305
1306            fbBuf += 1024;
1307
1308        }
1309
1310    }
1311
1312
1313    fb = esp_camera_fb_get();
1314
1315    if(!fb) {
1316
1317        // Serial.println("Camera capture failed")
1318
1319        delay(1000);
1320
1321        ESP.restart();
1322
1323    }
1324
1325
1326    client.println("HTTP/1.1 200 OK");
1327
1328    client.println("Access-Control-Allow-Origin:");
1329
1330    *");
1331
1332    client.println("Access-Control-Allow-Headers");
1333
1334    : Origin, X-Requested-With, Content-Type, Accept");
1335
1336    client.println("Access-Control-Allow-Methods");
1337
1338    : GET,POST,PUT,DELETE,OPTIONS");
1339
1340    client.println("Content-Type: image/jpeg");
1341
1342    client.println("Content-Disposition: form-
1343    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1344
1345    client.println("Content-Length: " + String(
1346    fb->len));
1347
1348    client.println("Connection: close");
1349
1350    client.println();
1351
1352
1353    uint8_t *fbBuf = fb->buf;
1354
1355    size_t fbLen = fb->len;
1356
1357    for (size_t n=0;n<fbLen;n=n+1024) {
1358
1359        if (n+1024<fbLen) {
1360
1361            client.write(fbBuf, 1024);
1362
1363            fbBuf += 1024;
1364
1365        }
1366
1367    }
1368
1369
1370    fb = esp_camera_fb_get();
1371
1372    if(!fb) {
1373
1374        // Serial.println("Camera capture failed")
1375
1376        delay(1000);
1377
1378        ESP.restart();
1379
1380    }
1381
1382
1383    client.println("HTTP/1.1 200 OK");
1384
1385    client.println("Access-Control-Allow-Origin:");
1386
1387    *");
1388
1389    client.println("Access-Control-Allow-Headers");
1390
1391    : Origin, X-Requested-With, Content-Type, Accept");
1392
1393    client.println("Access-Control-Allow-Methods");
1394
1395    : GET,POST,PUT,DELETE,OPTIONS");
1396
1397    client.println("Content-Type: image/jpeg");
1398
1399    client.println("Content-Disposition: form-
1400    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1401
1402    client.println("Content-Length: " + String(
1403    fb->len));
1404
1405    client.println("Connection: close");
1406
1407    client.println();
1408
1409
1410    uint8_t *fbBuf = fb->buf;
1411
1412    size_t fbLen = fb->len;
1413
1414    for (size_t n=0;n<fbLen;n=n+1024) {
1415
1416        if (n+1024<fbLen) {
1417
1418            client.write(fbBuf, 1024);
1419
1420            fbBuf += 1024;
1421
1422        }
1423
1424    }
1425
1426
1427    fb = esp_camera_fb_get();
1428
1429    if(!fb) {
1430
1431        // Serial.println("Camera capture failed")
1432
1433        delay(1000);
1434
1435        ESP.restart();
1436
1437    }
1438
1439
1440    client.println("HTTP/1.1 200 OK");
1441
1442    client.println("Access-Control-Allow-Origin:");
1443
1444    *");
1445
1446    client.println("Access-Control-Allow-Headers");
1447
1448    : Origin, X-Requested-With, Content-Type, Accept");
1449
1450    client.println("Access-Control-Allow-Methods");
1451
1452    : GET,POST,PUT,DELETE,OPTIONS");
1453
1454    client.println("Content-Type: image/jpeg");
1455
1456    client.println("Content-Disposition: form-
1457    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1458
1459    client.println("Content-Length: " + String(
1460    fb->len));
1461
1462    client.println("Connection: close");
1463
1464    client.println();
1465
1466
1467    uint8_t *fbBuf = fb->buf;
1468
1469    size_t fbLen = fb->len;
1470
1471    for (size_t n=0;n<fbLen;n=n+1024) {
1472
1473        if (n+1024<fbLen) {
1474
1475            client.write(fbBuf, 1024);
1476
1477            fbBuf += 1024;
1478
1479        }
1480
1481    }
1482
1483
1484    fb = esp_camera_fb_get();
1485
1486    if(!fb) {
1487
1488        // Serial.println("Camera capture failed")
1489
1490        delay(1000);
1491
1492        ESP.restart();
1493
1494    }
1495
1496
1497    client.println("HTTP/1.1 200 OK");
1498
1499    client.println("Access-Control-Allow-Origin:");
1500
1501    *");
1502
1503    client.println("Access-Control-Allow-Headers");
1504
1505    : Origin, X-Requested-With, Content-Type, Accept");
1506
1507    client.println("Access-Control-Allow-Methods");
1508
1509    : GET,POST,PUT,DELETE,OPTIONS");
1510
1511    client.println("Content-Type: image/jpeg");
1512
1513    client.println("Content-Disposition: form-
1514    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1515
1516    client.println("Content-Length: " + String(
1517    fb->len));
1518
1519    client.println("Connection: close");
1520
1521    client.println();
1522
1523
1524    uint8_t *fbBuf = fb->buf;
1525
1526    size_t fbLen = fb->len;
1527
1528    for (size_t n=0;n<fbLen;n=n+1024) {
1529
1530        if (n+1024<fbLen) {
1531
1532            client.write(fbBuf, 1024);
1533
1534            fbBuf += 1024;
1535
1536        }
1537
1538    }
1539
1540
1541    fb = esp_camera_fb_get();
1542
1543    if(!fb) {
1544
1545        // Serial.println("Camera capture failed")
1546
1547        delay(1000);
1548
1549        ESP.restart();
1550
1551    }
1552
1553
1554    client.println("HTTP/1.1 200 OK");
1555
1556    client.println("Access-Control-Allow-Origin:");
1557
1558    *");
1559
1560    client.println("Access-Control-Allow-Headers");
1561
1562    : Origin, X-Requested-With, Content-Type, Accept");
1563
1564    client.println("Access-Control-Allow-Methods");
1565
1566    : GET,POST,PUT,DELETE,OPTIONS");
1567
1568    client.println("Content-Type: image/jpeg");
1569
1570    client.println("Content-Disposition: form-
1571    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1572
1573    client.println("Content-Length: " + String(
1574    fb->len));
1575
1576    client.println("Connection: close");
1577
1578    client.println();
1579
1580
1581    uint8_t *fbBuf = fb->buf;
1582
1583    size_t fbLen = fb->len;
1584
1585    for (size_t n=0;n<fbLen;n=n+1024) {
1586
1587        if (n+1024<fbLen) {
1588
1589            client.write(fbBuf, 1024);
1590
1591            fbBuf += 1024;
1592
1593        }
1594
1595    }
1596
1597
1598    fb = esp_camera_fb_get();
1599
1600    if(!fb) {
1601
1602        // Serial.println("Camera capture failed")
1603
1604        delay(1000);
1605
1606        ESP.restart();
1607
1608    }
1609
1610
1611    client.println("HTTP/1.1 200 OK");
1612
1613    client.println("Access-Control-Allow-Origin:");
1614
1615    *");
1616
1617    client.println("Access-Control-Allow-Headers");
1618
1619    : Origin, X-Requested-With, Content-Type, Accept");
1620
1621    client.println("Access-Control-Allow-Methods");
1622
1623    : GET,POST,PUT,DELETE,OPTIONS");
1624
1625    client.println("Content-Type: image/jpeg");
1626
1627    client.println("Content-Disposition: form-
1628    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1629
1630    client.println("Content-Length: " + String(
1631    fb->len));
1632
1633    client.println("Connection: close");
1634
1635    client.println();
1636
1637
1638    uint8_t *fbBuf = fb->buf;
1639
1640    size_t fbLen = fb->len;
1641
1642    for (size_t n=0;n<fbLen;n=n+1024) {
1643
1644        if (n+1024<fbLen) {
1645
1646            client.write(fbBuf, 1024);
1647
1648            fbBuf += 1024;
1649
1650        }
1651
1652    }
1653
1654
1655    fb = esp_camera_fb_get();
1656
1657    if(!fb) {
1658
1659        // Serial.println("Camera capture failed")
1660
1661        delay(1000);
1662
1663        ESP.restart();
1664
1665    }
1666
1667
1668    client.println("HTTP/1.1 200 OK");
1669
1670    client.println("Access-Control-Allow-Origin:");
1671
1672    *");
1673
1674    client.println("Access-Control-Allow-Headers");
1675
1676    : Origin, X-Requested-With, Content-Type, Accept");
1677
1678    client.println("Access-Control-Allow-Methods");
1679
1680    : GET,POST,PUT,DELETE,OPTIONS");
1681
1682    client.println("Content-Type: image/jpeg");
1683
1684    client.println("Content-Disposition: form-
1685    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1686
1687    client.println("Content-Length: " + String(
1688    fb->len));
1689
1690    client.println("Connection: close");
1691
1692    client.println();
1693
1694
1695    uint8_t *fbBuf = fb->buf;
1696
1697    size_t fbLen = fb->len;
1698
1699    for (size_t n=0;n<fbLen;n=n+1024) {
1700
1701        if (n+1024<fbLen) {
1702
1703            client.write(fbBuf, 1024);
1704
1705            fbBuf += 1024;
1706
1707        }
1708
1709    }
1710
1711
1712    fb = esp_camera_fb_get();
1713
1714    if(!fb) {
1715
1716        // Serial.println("Camera capture failed")
1717
1718        delay(1000);
1719
1720        ESP.restart();
1721
1722    }
1723
1724
1725    client.println("HTTP/1.1 200 OK");
1726
1727    client.println("Access-Control-Allow-Origin:");
1728
1729    *");
1730
1731    client.println("Access-Control-Allow-Headers");
1732
1733    : Origin, X-Requested-With, Content-Type, Accept");
1734
1735    client.println("Access-Control-Allow-Methods");
1736
1737    : GET,POST,PUT,DELETE,OPTIONS");
1738
1739    client.println("Content-Type: image/jpeg");
1740
1741    client.println("Content-Disposition: form-
1742    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1743
1744    client.println("Content-Length: " + String(
1745    fb->len));
1746
1747    client.println("Connection: close");
1748
1749    client.println();
1750
1751
1752    uint8_t *fbBuf = fb->buf;
1753
1754    size_t fbLen = fb->len;
1755
1756    for (size_t n=0;n<fbLen;n=n+1024) {
1757
1758        if (n+1024<fbLen) {
1759
1760            client.write(fbBuf, 1024);
1761
1762            fbBuf += 1024;
1763
1764        }
1765
1766    }
1767
1768
1769    fb = esp_camera_fb_get();
1770
1771    if(!fb) {
1772
1773        // Serial.println("Camera capture failed")
1774
1775        delay(1000);
1776
1777        ESP.restart();
1778
1779    }
1780
1781
1782    client.println("HTTP/1.1 200 OK");
1783
1784    client.println("Access-Control-Allow-Origin:");
1785
1786    *");
1787
1788    client.println("Access-Control-Allow-Headers");
1789
1790    : Origin, X-Requested-With, Content-Type, Accept");
1791
1792    client.println("Access-Control-Allow-Methods");
1793
1794    : GET,POST,PUT,DELETE,OPTIONS");
1795
1796    client.println("Content-Type: image/jpeg");
1797
1798    client.println("Content-Disposition: form-
1799    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1800
1801    client.println("Content-Length: " + String(
1802    fb->len));
1803
1804    client.println("Connection: close");
1805
1806    client.println();
1807
1808
1809    uint8_t *fbBuf = fb->buf;
1810
1811    size_t fbLen = fb->len;
1812
1813    for (size_t n=0;n<fbLen;n=n+1024) {
1814
1815        if (n+1024<fbLen) {
1816
1817            client.write(fbBuf, 1024);
1818
1819            fbBuf += 1024;
1820
1821        }
1822
1823    }
1824
1825
1826    fb = esp_camera_fb_get();
1827
1828    if(!fb) {
1829
1830        // Serial.println("Camera capture failed")
1831
1832        delay(1000);
1833
1834        ESP.restart();
1835
1836    }
1837
1838
1839    client.println("HTTP/1.1 200 OK");
1840
1841    client.println("Access-Control-Allow-Origin:");
1842
1843    *");
1844
1845    client.println("Access-Control-Allow-Headers");
1846
1847    : Origin, X-Requested-With, Content-Type, Accept");
1848
1849    client.println("Access-Control-Allow-Methods");
1850
1851    : GET,POST,PUT,DELETE,OPTIONS");
1852
1853    client.println("Content-Type: image/jpeg");
1854
1855    client.println("Content-Disposition: form-
1856    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1857
1858    client.println("Content-Length: " + String(
1859    fb->len));
1860
1861    client.println("Connection: close");
1862
1863    client.println();
1864
1865
1866    uint8_t *fbBuf = fb->buf;
1867
1868    size_t fbLen = fb->len;
1869
1870    for (size_t n=0;n<fbLen;n=n+1024) {
1871
1872        if (n+1024<fbLen) {
1873
1874            client.write(fbBuf, 1024);
1875
1876            fbBuf += 1024;
1877
1878        }
1879
1880    }
1881
1882
1883    fb = esp_camera_fb_get();
1884
1885    if(!fb) {
1886
1887        // Serial.println("Camera capture failed")
1888
1889        delay(1000);
1890
1891        ESP.restart();
1892
1893    }
1894
1895
1896    client.println("HTTP/1.1 200 OK");
1897
1898    client.println("Access-Control-Allow-Origin:");
1899
1900    *");
1901
1902    client.println("Access-Control-Allow-Headers");
1903
1904    : Origin, X-Requested-With, Content-Type, Accept");
1905
1906    client.println("Access-Control-Allow-Methods");
1907
1908    : GET,POST,PUT,DELETE,OPTIONS");
1909
1910    client.println("Content-Type: image/jpeg");
1911
1912    client.println("Content-Disposition: form-
1913    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1914
1915    client.println("Content-Length: " + String(
1916    fb->len));
1917
1918    client.println("Connection: close");
1919
1920    client.println();
1921
1922
1923    uint8_t *fbBuf = fb->buf;
1924
1925    size_t fbLen = fb->len;
1926
1927    for (size_t n=0;n<fbLen;n=n+1024) {
1928
1929        if (n+1024<fbLen) {
1930
1931            client.write(fbBuf, 1024);
1932
1933            fbBuf += 1024;
1934
1935        }
1936
1937    }
1938
1939
1940    fb = esp_camera_fb_get();
1941
1942    if(!fb) {
1943
1944        // Serial.println("Camera capture failed")
1945
1946        delay(1000);
1947
1948        ESP.restart();
1949
1950    }
1951
1952
1953    client.println("HTTP/1.1 200 OK");
1954
1955    client.println("Access-Control-Allow-Origin:");
1956
1957    *");
1958
1959    client.println("Access-Control-Allow-Headers");
1960
1961    : Origin, X-Requested-With, Content-Type, Accept");
1962
1963    client.println("Access-Control-Allow-Methods");
1964
1965    : GET,POST,PUT,DELETE,OPTIONS");
1966
1967    client.println("Content-Type: image/jpeg");
1968
1969    client.println("Content-Disposition: form-
1970    data; name=\"imageFile\"; filename=\"picture.jpg\"");
1971
1972    client.println("Content-Length: " + String(
1973    fb->len));
1974
1975    client.println("Connection: close");
1976
1977    client.println();
1978
1979
1980    uint8_t *fbBuf = fb->buf;
1981
1982    size_t fbLen = fb->len;
1983
1984    for (size_t n=0;n<fbLen;n=n+1024) {
1985
1986        if (n+1024<fbLen) {
1987
1988            client.write(fbBuf, 1024);
1989
1990            fbBuf += 1024;
1991
1992        }
1993
1994    }
1995
1996
1997    fb = esp_camera_fb_get();
1998
1999    if(!fb) {
2000
2001        // Serial.println("Camera capture failed")
2002
2003        delay(1000);
2004
2005        ESP.restart();
2006
2007    }
2008
2009
2010    client.println("HTTP/1.1 200 OK");
2011
2012    client.println("Access-Control-Allow-Origin:");
2013
2014    *");
2015
2016    client.println("Access-Control-Allow-Headers");
2017
2018    : Origin, X-Requested-With, Content-Type, Accept");
2019
2020    client.println("Access-Control-Allow-Methods");
2021
2022    : GET,POST,PUT,DELETE,OPTIONS");
2023
2024    client.println("Content-Type: image/jpeg");
2025
2026    client.println("Content-Disposition: form-
2027    data; name=\"imageFile\"; filename=\"picture.jpg\"");
2028
2029    client.println("Content-Length: " + String(
2030    fb->len));
2031
2032    client.println("Connection: close");
2033
2034    client.println
```

```

652     else if (fbLen%1024>0) {
653
654         size_t remainder = fbLen%1024;
655
656         client.write(fbBuf, remainder);
657
658     }
659
660     esp_camera_fb_return(fb);
661
662     pinMode(4, OUTPUT);
663     digitalWrite(4, LOW);
664
665     else {
666
667         client.println("HTTP/1.1 200 OK");
668
669         client.println("Access-Control-Allow-Headers
: Origin, X-Requested-With, Content-Type, Accept");
670
671         client.println("Access-Control-Allow-Methods
: GET,POST,PUT,DELETE,OPTIONS");
672
673         client.println("Content-Type: text/html;
charset=utf-8");
674
675         client.println("Access-Control-Allow-Origin:
*");
676
677         client.println("Connection: close");
678
679         client.println();
680
681         String Data="";
682
683         if (cmd!="")
684
685             Data = Feedback;
686
687         else {
688
689             Data = String((const char *)INDEX_HTML);

```

```

678 }
679     int Index;
680     for (Index = 0; Index < Data.length(); Index
= Index+1000) {
681         client.print(Data.substring(Index, Index+1
000));
682     }
683
684     client.println();
685 }
686
687     Feedback="";
688     break;
689 } else {
690     currentLine = "";
691 }
692 }
693 else if (c != '\r') {
694     currentLine += c;
695 }
696
697 if ((currentLine.indexOf("/") != -1) && (currentLine.
indexOf(" HTTP") != -1)) {
698     if (Command.indexOf("stop") != -1) {      http://192.
168.xxx.xxx/?cmd=aaa;bbb;ccc;stop
699         client.println();
700         client.println();
701         client.stop();
702     }
703     currentLine="";

```

```

704     Feedback="";
705     ExecuteCommand();
706 }
707 }
708 }
709 delay(1);
710 client.stop();
711 }
712
713 }
714
715 String tcp_http(String domain, String request, int port, byte
716 wait)
717 {
718     WiFiClient client_tcp;
719
720     if (client_tcp.connect(domain.c_str(), port))
721     {
722         Serial.println("GET " + request);
723         client_tcp.println("GET " + request + " HTTP/1.1");
724         client_tcp.println("Host: " + domain);
725         client_tcp.println("Connection: close");
726         client_tcp.println();
727
728         String getResponse="", Feedback="";
729         boolean state = false;
730         int waitTime = 3000; // timeout 3 seconds
731         long startTime = millis();
732         while ((startTime + waitTime) > millis())
733         {

```

```

733     while (client_tcp.available())
734     {
735         char c = client_tcp.read();
736         if (state==true) Feedback += String(c);
737         if (c == '\n')
738         {
739             if (getResponse.length()==0) state=true;
740             getResponse = "";
741         }
742         else if (c != '\r')
743             getResponse += String(c);
744         if (wait==1)
745             startTime = millis();
746     }
747     if (wait==0)
748     if ((state==true)&&(Feedback.length()!= 0))
749     break;
750     client_tcp.stop();
751     return Feedback;
752 }
753 else
754     return "Connection failed";
755 }

756
757 String sendCapturedImageToLineNotify(String token)
758 {
759     String getAll="", getBody = "";
760
761     camera_fb_t * fb = NULL;

```

```

762     fb = esp_camera_fb_get();
763
764     if(!fb) {
765
765         Serial.println("Camera capture failed");
766
767         delay(1000);
768
769         ESP.restart();
770
771         return "";
772
773     }
774
775
776     WiFiClientSecure client_tcp;
777
778     client_tcp.setInsecure(); //run version 1.0.5 or above
779
780     Serial.println("Connect to notify-api.line.me");
781
782     if (client_tcp.connect("notify-api.line.me", 443)) {
783
784         Serial.println("Connection successful");
785
786
787         String message = "Welcome to Taiwan.";
788
789         String head = "--Taiwan\r\nContent-Disposition: form-
790         data; name=\"message\"; \r\n\r\n" + message + "\r\n--"
791
792         Taiwan\r\nContent-Disposition: form-data; name="
793
794         imageFile"; filename="esp32-cam.jpg"\r\nContent-Type
795         : image/jpeg\r\n\r\n";
796
797         String tail = "\r\n--Taiwan--\r\n";
798
799
800         uint16_t imageLen = fb->len;
801
802         uint16_t extraLen = head.length() + tail.length();
803
804         uint16_t totalLen = imageLen + extraLen;
805
806
807         client_tcp.println("POST /api/notify HTTP/1.1");
808
809         client_tcp.println("Connection: close");

```

```

788     client_tcp.println("Host: notify-api.line.me");
789     client_tcp.println("Authorization: Bearer " + token);
790     client_tcp.println("Content-Length: " + String(
791         totalLen));
792     client_tcp.println("Content-Type: multipart/form-data;
793         boundary=Taiwan");
794     client_tcp.println();
795     client_tcp.print(head);
796
797     uint8_t *fbBuf = fb->buf;
798     size_t fbLen = fb->len;
799     for (size_t n=0;n<fbLen;n=n+1024) {
800         if (n+1024<fbLen) {
801             client_tcp.write(fbBuf, 1024);
802             fbBuf += 1024;
803         }
804         else if (fbLen%1024>0) {
805             size_t remainder = fbLen%1024;
806             client_tcp.write(fbBuf, remainder);
807         }
808     }
809     client_tcp.print(tail);
810     esp_camera_fb_return(fb);
811
812     int waitTime = 10000;    // timeout 10 seconds
813     long startTime = millis();
814     boolean state = false;
815     while ((startTime + waitTime) > millis())
{

```

```

816     Serial.print(".");
817     delay(100);
818     while (client_tcp.available())
819     {
820         char c = client_tcp.read();
821         if (state==true)getBody += String(c);
822         if (c == '\n')
823         {
824             if (getAll.length()==0) state=true;
825             getAll = "";
826         }
827         else if (c != '\r')
828             getAll += String(c);
829         startTime = millis();
830     }
831     if (getBody.length()>0) break;
832 }
833 client_tcp.stop();
834 //Serial.println(getAll);
835 Serial.println(getBody);
836 }
837 else {
838     getAll="Connected to notify-api.line.me failed.";
839     getBody="Connected to notify-api.line.me failed.";
840     Serial.println("Connected to notify-api.line.me failed
841 .");
842 }
843 return getBody;
844 }

```

## CAM2 Code(QR code Scan)

```
1      const char* ssid = "monir zx online";
2  const char* password = "01718146157";
3
4
5  const char* apssid = "esp32-cam1";
6  const char* appassword = "12345678";
7
8  String Feedback="";
9  String Command="" , cmd="" , P1="" , P2="" , P3="" , P4="" , P5="" , P6=
  "" , P7="" , P8="" , P9="" ;
10
11 byte ReceiveState=0 , cmdState=1 , strState=1 , questionstate=0 ,
  equalstate=0 , semicolonstate=0;
12
13 #include <WiFi.h>
14 #include "esp_camera.h"
15 #include <HTTPClient.h>
16 #include "soc/soc.h"
17 #include "soc/rtc_cntl_reg.h"
18 #include "quirc.h"
19 #include <HardwareSerial.h>
20 #include "esp_system.h"
21 String ID =
  AKfycbx0ATD7b1rALzoFIdvE810D53u7oJHPkEKzCCcWP9N5ZNUB5_PsVKW1Ek_Om
  -SNOkuasQ; //--> spreadsheet script ID
22 const char* host = "script.google.com";
23 TaskHandle_t Task;
24
```

```

25 #define PWDN_GPIO_NUM      32
26 #define RESET_GPIO_NUM     -1
27 #define XCLK_GPIO_NUM       0
28 #define SIOD_GPIO_NUM      26
29 #define SIOC_GPIO_NUM      27
30 #define Y9_GPIO_NUM         35
31 #define Y8_GPIO_NUM         34
32 #define Y7_GPIO_NUM         39
33 #define Y6_GPIO_NUM         36
34 #define Y5_GPIO_NUM         21
35 #define Y4_GPIO_NUM         19
36 #define Y3_GPIO_NUM         18
37 #define Y2_GPIO_NUM         5
38 #define VSYNC_GPIO_NUM      25
39 #define HREF_GPIO_NUM       23
40 #define PCLK_GPIO_NUM       22
41 #define BUZZER_PIN          15 // ESP32-cam pin GPIO15
                                connected to Buzzer's pin
42
43 void update_google_sheet(String id)
44 {
45   Serial.print("connecting to ");
46   Serial.println(host);
47
48   // Use WiFiClient class to create TCP connections
49   WiFiClientSecure client;
50   const int httpPort = 443; // 80 is for HTTP / 443 is for
                            HTTPS!
51
52   client.setInsecure(); // this is the magical line that

```

```

    makes everything work

53

54     if (!client.connect(host, httpPort)) { //works!
55         Serial.println("connection failed");
56         return;
57     }

58

59 //-----Processing
60
61
62     data and sending data
63
64     String url = "/macros/s/" + ID + "/exec?num=";
65
66     url += 1;
67     url += "&id=";
68     url += String(id);
69
70     Serial.print("Requesting URL: ");
71     Serial.println(url);

72
73     // This will send the request to the server
74     client.print(String("GET ") + url + " HTTP/1.1\r\n" +
75                 "Host: " + host + "\r\n" +
76                 "Connection: close\r\n\r\n");

77
78     Serial.println();
79     Serial.println("closing connection");
80     digitalWrite(BUZZER_PIN, HIGH); // turn on
81     delay(1500);
82     digitalWrite(BUZZER_PIN, LOW); // turn off
83     Serial.print('A');
84     delay(50000);

```

```

81    }
82
83    struct QRCodeData
84    {
85        bool valid;
86        int dataType;
87        uint8_t payload[1024];
88        int payloadLen;
89    };
90
91    struct quirc *q = NULL;
92    uint8_t *image = NULL;
93    camera_fb_t * fb = NULL;
94    struct quirc_code code;
95    struct quirc_data data;
96    quirc_decode_error_t err;
97    struct QRCodeData qrCodeData;
98    String QRCodeResult = "";
99
100   WiFiServer server(80);
101   WiFiClient client;
102
103   camera_config_t config;
104
105 void setup() {
106     WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
107
108     Serial.begin(115200);
109     Serial.setDebugOutput(true);
110     Serial.println();

```

```

111     pinMode(BUZZER_PIN, OUTPUT);

112

113     config.ledc_channel = LEDC_CHANNEL_0;
114     config.ledc_timer = LEDC_TIMER_0;
115     config.pin_d0 = Y2_GPIO_NUM;
116     config.pin_d1 = Y3_GPIO_NUM;
117     config.pin_d2 = Y4_GPIO_NUM;
118     config.pin_d3 = Y5_GPIO_NUM;
119     config.pin_d4 = Y6_GPIO_NUM;
120     config.pin_d5 = Y7_GPIO_NUM;
121     config.pin_d6 = Y8_GPIO_NUM;
122     config.pin_d7 = Y9_GPIO_NUM;
123     config.pin_xclk = XCLK_GPIO_NUM;
124     config.pin_pclk = PCLK_GPIO_NUM;
125     config.pin_vsync = VSYNC_GPIO_NUM;
126     config.pin_href = HREF_GPIO_NUM;
127     config.pin_sscb_sda = SIOD_GPIO_NUM;
128     config.pin_sscb_scl = SIOC_GPIO_NUM;
129     config.pin_pwdn = PWDN_GPIO_NUM;
130     config.pin_reset = RESET_GPIO_NUM;
131     config.xclk_freq_hz = 10000000;
132     config.pixel_format = PIXFORMAT_GRAYSCALE;
133     config.frame_size = FRAMESIZE_QVGA;
134     config.jpeg_quality = 15;
135     config.fb_count = 1;

136

137     esp_err_t err = esp_camera_init(&config);
138     if (err != ESP_OK) {
139         Serial.printf("Camera init failed with error 0x%x",
140             err);

```

```

140     ESP.restart();
141 }
142
143 sensor_t * s = esp_camera_sensor_get();
144 s->set_framesize(s, FRAMESIZE_QVGA);
145
146
147 ledcAttachPin(4, 4);
148 ledcSetup(4, 5000, 8);
149
150 WiFi.mode(WIFI_AP_STA);
151
152
153
154 for (int i=0;i<2;i++) {
155     WiFi.begin(ssid, password);
156
157     delay(1000);
158     Serial.println("");
159     Serial.print("Connecting to ");
160     Serial.println(ssid);
161
162     long int StartTime=millis();
163     while (WiFi.status() != WL_CONNECTED) {
164         delay(500);
165         if ((StartTime+5000) < millis()) break;
166     }
167
168     if (WiFi.status() == WL_CONNECTED) {
169         WiFi.softAP((WiFi.localIP().toString()+"_"+(String)

```

```

apssid).c_str(), appassword); //  

                           SSIDIP  

170   Serial.println("");  

171   Serial.println("STAIP address: ");  

172   Serial.println(WiFi.localIP());  

173   Serial.println("");  

174  

175   for (int i=0;i<5;i++) {  

176     ledcWrite(4,10);  

177     delay(200);  

178     ledcWrite(4,0);  

179     delay(200);  

180   }  

181   break;  

182 }  

183 }  

184  

185 if (WiFi.status() != WL_CONNECTED) {  

186   WiFi.softAP((WiFi.softAPIP().toString()+"_"+(String)  

apssid).c_str(), appassword);  

187  

188   for (int i=0;i<2;i++) {  

189     ledcWrite(4,10);  

190     delay(1000);  

191     ledcWrite(4,0);  

192     delay(1000);  

193   }  

194 }  

195  

196 Serial.println("");

```

```

197     Serial.println("APIP address: ");
198     Serial.println(WiFi.softAPIP());
199     Serial.println("");
200
201
202     pinMode(4, OUTPUT);
203     digitalWrite(4, LOW);
204
205     server.begin();
206
207     xTaskCreatePinnedToCore(
208         QRCodeReader, /* Task function. */
209         "Task",      /* name of task. */
210         10000,       /* Stack size of task */
211         NULL,        /* parameter of the task */
212         1,           /* priority of the task */
213         &Task,       /* Task handle to keep track of
214         created task */
215         0);          /* pin task to core 0 */
216
217     Serial.print("listenConnection running on core ");
218     Serial.println(xPortGetCoreID());
219 }
220
221 void loop() {
222     listenConnection();
223 }
224
225 void QRCodeReader( void * pvParameters ){
226     Serial.print("QRCodeReader running on core ");

```

```

226     Serial.println(xPortGetCoreID());
227
228     while(1){
229         q = quirc_new();
230         if (q == NULL){
231             Serial.print("can't create quirc object\r\n");
232             continue;
233         }
234
235         fb = esp_camera_fb_get();
236         if (!fb)
237         {
238             Serial.println("Camera capture failed");
239             continue;
240         }
241
242         //Serial.printf("quirc_begin\r\n");
243         quirc_resize(q, fb->width, fb->height);
244         image = quirc_begin(q, NULL, NULL);
245         //Serial.printf("Frame w h len: %d, %d, %d \r\n", fb
246         ->width, fb->height, fb->len);
247         memcpy(image, fb->buf, fb->len);
248         quirc_end(q);
249         //Serial.printf("quirc_end\r\n");
250
251         int count = quirc_count(q);
252         if (count > 0) {
253             Serial.println(count);
254             quirc_extract(q, 0, &code);
255             err = quirc_decode(&code, &data);

```

```

255
256     if  (err){
257         Serial.println("Decoding FAILED");
258         QRCodeResult = "Decoding FAILED";
259     } else {
260         Serial.printf("Decoding successful:\n");
261         dumpData(&data);
262     }
263     Serial.println();
264 }
265
266     esp_camera_fb_return(fb);
267     fb = NULL;
268     image = NULL;
269     quirc_destroy(q);
270 }
271 }
272
273 void dumpData(const struct quirc_data *data)
274 {
275     Serial.printf("Version: %d\n", data->version);
276     Serial.printf("ECC level: %c\n", "MLHQ"[data->ecc_level
277     ]);
278     Serial.printf("Mask: %d\n", data->mask);
279     Serial.printf("Length: %d\n", data->payload_len);
280     Serial.printf("Payload: %s\n", data->payload);
281     QRCodeResult = (const char *)data->payload;
282     Serial.println(QRCodeResult);
283     update_google_sheet(QRCodeResult);

```

```

284     }
285
286
287     static const char PROGMEM INDEX_HTML[] = R"rawliteral(
288         <!DOCTYPE html>
289         <head>
290             <title></title>
291             <meta charset="utf-8">
292             <meta name="viewport" content="width=device-width,
293                 initial-scale=1">
294         </head>
295         <body>
296             <canvas id="canvas" width="320" height="240"></canvas><br>
297             Flash<input type="range" id="flash" min="0" max="255"
298                 value="0">
299             <input type="button" value="Get Still" onclick="getStill()
300                 ;"><br>
301             <div id="result" style="color:red"></div>
302         </body>
303     </html>
304
305
306
307
308     flash.onchange = function() {
309         var query = document.location.origin+ "?flash=" + flash.
310             value;

```

```

310     fetch(query);
311 }
312 function getStill() {
313     var xhr = new XMLHttpRequest();
314     xhr.open("GET", "?getstill", true);
315     xhr.responseType = "arraybuffer";
316
317     xhr.onload = function (oEvent) {
318         var arrayBuffer = xhr.response; // Note: not xhr.
319         responseType
320         if (arrayBuffer) {
321             var byteArray = new Uint8Array(arrayBuffer);
322             var imgData=context.getImageData(0,0,canvas.width,
323             canvas.height);
324             var val = 0;
325             for (var i=0;i

```

```

338   </script>
339 ) rawliteral";
340
341 void listenConnection() {
342   Feedback="" ; Command="" ; cmd="" ; P1="" ; P2="" ; P3="" ; P4="" ; P5
343   ="" ; P6="" ; P7="" ; P8="" ; P9="";
344   ReceiveState=0 , cmdState=1 , strState=1 , questionstate=0 ,
345   equalstate=0 , semicolonstate=0 ;
346
347   if (client) {
348     String currentLine = "";
349
350     while (client.connected()) {
351       if (client.available()) {
352         char c = client.read();
353         getCommand(c);
354         if (c == '\n') {
355           if (currentLine.length() == 0) {
356             if (cmd=="getstill") {
357               getStill();
358             } else {
359               mainPage();
360             }
361             Feedback="";
362             break;
363           } else {
364             currentLine = "";
365           }

```

```

366     } else if (c != '\r') {
367
368         currentLine += c;
369
370     if ((currentLine.indexOf("?") != -1) && (currentLine.
371
372         indexOf(" HTTP") != -1)) {
373
374             if (Command.indexOf("stop") != -1) {
375
376                 client.println();
377
378                 client.println();
379
380                 client.stop();
381
382             delay(1);
383
384             client.stop();
385
386
387         void MainPage() {
388
389             client.println("HTTP/1.1 200 OK");
390
391             client.println("Access-Control-Allow-Headers: Origin,
X-Requested-With, Content-Type, Accept");
392
393             client.println("Access-Control-Allow-Methods: GET,POST
,PUT,DELETE,OPTIONS");
394
395             client.println("Content-Type: text/html; charset=utf-8
");

```

```

392     client.println("Access-Control-Allow-Origin: *");
393     client.println("Connection: close");
394     client.println();
395
396     String Data="";
397     if (cmd!="")
398         Data = Feedback;
399     else {
400         Data = String((const char *)INDEX_HTML);
401     }
402     int Index;
403     for (Index = 0; Index < Data.length(); Index = Index+1
024) {
404         client.print(Data.substring(Index, Index+1024));
405     }
406 }
407
408 void getStill() {
409     camera_fb_t * fb = NULL;
410     fb = esp_camera_fb_get();
411     if(!fb) {
412         Serial.println("Camera capture failed");
413         return;
414     }
415     uint8_t *fbBuf = fb->buf;
416     size_t fbLen = fb->len;
417
418     client.println("HTTP/1.1 200 OK");
419     client.println("Access-Control-Allow-Origin: *");
420     client.println("Access-Control-Allow-Headers: Origin, X-
```

```

    Requested-With, Content-Type, Accept");
421 client.println("Access-Control-Allow-Methods: GET,POST,
422 PUT,DELETE,OPTIONS");
423 client.println("Content-Type: application/octet-stream")
424 ;
425 client.println();
426
427 for (size_t n=0;n<fbLen;n=n+1024) {
428     if (n+1024<fbLen) {
429         client.write(fbBuf, 1024);
430         fbBuf += 1024;
431     }
432     else if (fbLen%1024>0) {
433         size_t remainder = fbLen%1024;
434         client.write(fbBuf, remainder);
435     }
436 }
437 esp_camera_fb_return(fb);
438
439 }

```

## Microcontroller Code

```

1      #include <Wire.h>
2      #include <LiquidCrystal_I2C.h>
3
4 // SDA A4
5 // SCL A5

```

```

6  const int trigPin = 2;
7  const int echoPin = 3;
8  long duration;
9  float distanceCm;
10 float distanceInch;
11 //define sound speed in cm/uS
12 #define SOUND_SPEED 0.034
13 #define CM_TO_INCH 0.393701
14
15 LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address
16 to 0x27 for a 16 chars and 2 line display
17
18
19 int sample_distance() {
20     long duration, distance;
21
22     digitalWrite(trigPin, LOW);
23     delayMicroseconds(10);
24     digitalWrite(trigPin, HIGH);
25     delayMicroseconds(10);
26     digitalWrite(trigPin, LOW);
27
28     duration = pulseIn(echoPin, HIGH);
29     distance = (duration / 2) / 29.155;
30     return distance;
31 }
32
33
34 void setup() {

```

```
35 Serial.begin(115200);
36 pinMode(trigPin, OUTPUT); // Sets the trigPin as an
37 // Output
38 pinMode(echoPin, INPUT); // Sets the echoPin as an Input
39 lcd.init(); // initialize the lcd
40 // Print a message to the LCD.
41 lcd.backlight();
42 lcd.setCursor(0,0);
43 lcd.print("REVERSE VENDING");
44 lcd.setCursor(0,1);
45 lcd.print("      MACHINE      ");
46 }
47 void loop() {
48
49     int distanceCm = sample_distance();
50     //Serial.println(distanceCm);
51
52
53     if (distanceCm<10){
54         lcd.init();
55         lcd.clear();
56         lcd.backlight();
57         lcd.setCursor(0,0);
58         lcd.print("      The bin is      ");
59         lcd.setCursor(0,1);
60         lcd.print("      full      ");
61     }
62     else if (distanceCm>10){
63         lcd.init();
```

```

64     lcd.clear();
65     lcd.backlight();
66     lcd.setCursor(0,0);
67     lcd.print("REVERSE VENDING");
68     lcd.setCursor(0,1);
69     lcd.print("      MACHINE      ");
70 }
71 else if (Serial.available()){
72     char read = Serial.read();
73     Serial.print(read);
74     if (read=='A'){
75         lcd.init();
76         lcd.clear();
77         lcd.backlight();
78         lcd.setCursor(0,0);
79         lcd.print(" Credit is added");
80         lcd.setCursor(0,1);
81         lcd.print(" To your account");
82         delay(5000);
83     }
84 }
85 }
```

### Google Script Code 1 (Data base)

```

1     function doGet(e) {
2     Logger.log( JSON.stringify(e) );
3     var result = 'Ok';
4     if (e.parameter == 'undefined') {
5       result = 'No Parameters';
6     }
7     return ContentService.createTextOutput(result);
8   }
```

```

6    }
7  else {
8    var sheet_id = '1ClNE3e6AVFA15YZ_-
9      vDhdz8hNxUryIkL4sdLqJnZkEQ'; // Spreadsheet ID
10   var sheet = SpreadsheetApp.openById(sheet_id).
11     getActiveSheet();
12   var newRow = sheet.getLastRow();
13   var value1 = sheet.getRange(newRow, 3).getValue();
14   var value2 = sheet.getRange(newRow, 4).getValue();
15   var rowData = [];
16   if (value1 == 1 && value2 == ''){
17     var Curr_Date = new Date();
18     rowData[0] = Curr_Date; // Date in column A
19     var Curr_Time = Utilities.formatDate(Curr_Date, "Asia/
20       Bangkok", 'HH:mm:ss');
21     rowData[1] = Curr_Time; // Time in column B
22     for (var param in e.parameter) {
23       Logger.log('In for loop, param=' + param);
24       var value = stripQuotes(e.parameter[param]);
25       Logger.log(param + ':' + e.parameter[param]);
26       switch (param) {
27
28         case 'num':
29           rowData[2] = value;
30           result = 'OK';
31           break;
32
33         case 'id':
34           rowData[3] = value;
35           result = 'OK';

```

```

33     break;
34
35     default:
36         result = "unsupported parameter";
37     }
38
39 }
40 }
41 Logger.log(JSON.stringify(rowData));
42 var newRange = sheet.getRange(newRow, 1, 1, rowData.length
);
43 newRange.setValues([rowData]);
44 }
45 return ContentService.createTextOutput(result);
46 }
47 function stripQuotes( value ) {
48     return value.replace(/^["]|["]$/g, "");
49 }

```

### Google Script Code 1 (Data base)

```

1     function doGet(e) {
2     Logger.log( JSON.stringify(e) );
3     var result = 'Ok';
4     if (e.parameter == 'undefined') {
5     result = 'No Parameters';
6     }
7     else {
8     var sheet_id = '1ClNE3e6AVFA15YZ_-'
vDhdz8hNxUryIkL4sdLqJnZkEQ'; // Spreadsheet ID

```

```

9  var sheet = SpreadsheetApp.openById(sheet_id).
10    getActiveSheet();
11 var newRow = sheet.getLastRow();
12 var rowData = [];
13 var Curr_Date = new Date();
14 rowData[0] = Curr_Date; // Date in column A
15 var Curr_Time = Utilities.formatDate(Curr_Date, "Asia/
16   Bangkok", 'HH:mm:ss');
17 rowData[1] = Curr_Time; // Time in column B
18 for (var param in e.parameter) {
19   Logger.log('In for loop, param=' + param);
20   var value = stripQuotes(e.parameter[param]);
21   Logger.log(param + ':' + e.parameter[param]);
22   switch (param) {
23     case 'num':
24       rowData[2] = value;
25       result = 'OK';
26       break;
27     default:
28       result = "unsupported parameter";
29   }
30
31 }
32
33 Logger.log(JSON.stringify(rowData));
34 var newRange = sheet.getRange(newRow, 1, 1, rowData.length
35 );
36 newRange.setValues([rowData]);

```

```
36  }
37  return ContentService.createTextOutput(result);
38 }
39 function stripQuotes( value ) {
40   return value.replace(/^["]|["]$/g, "");
41 }
```

## Reverse Vending Machine

The primary goal of reverse vending machines is to promote recycling, reduce litter, and conserve resources. Users place their empty containers into the machine's designated input area. Using machine learning bottle is detected and QR code is scanned to reward the user. Also a display and a sonar sensor is used to indicate bins fullness.



## Future Work

Expanding the use of reverse vending machines (RVMs) to various locations such as public parks, restaurants, schools, and university campuses is an excellent idea. To enhance the user experience and promote widespread adoption, integrating mobile banking services and collaborating with financial institutions like Bkash, Nogod, and Rocket can add significant value. Additionally, incorporating several features and security measures can make the system more user-friendly and secure.

Enhanced Features of Reverse Vending Machines:

- **Mobile Banking Integration:** Users can link their mobile banking accounts (e.g., Bkash, Nogod, Rocket) to the reverse vending machine for seamless reward redemption. The machine can generate QR codes or transaction IDs, which users can scan or enter into their mobile banking apps to receive credits.
- **Flexible Rewards:** Provide users with the option to choose their preferred reward format, including mobile wallet credits, vouchers for partner stores, or direct cash transfers to their linked mobile banking accounts.
- **Educational Interface:** Implement a user-friendly touchscreen interface on the RVM, displaying information about the environmental impact of recycling, the number of containers saved, and the cumulative benefits to the community.
- **Sustainability Scoreboard:** Display a live sustainability scoreboard near the RVM, showcasing the top contributors in recycling within the community. This can serve as positive reinforcement and motivation.
- **Security Measures:** Implement robust security features, including surveillance cameras, tamper-resistant containers, and secure transaction protocols to ensure the safety of users and their transactions.
- **Weather-Resistant Design:** Construct RVMs with durable, weather-resistant materials to withstand outdoor conditions in parks and other open areas.

## **Discussion and Conclusion**

As technology continues to evolve, future advancements in RVMs hold the potential to further optimize the recycling process, enhance user engagement, and contribute to broader environmental initiatives. The success of these machines lies not only in their ability to incentivize recycling but also in their role as educational tools, raising awareness and fostering a collective commitment to preserving our planet for future generations. As communities embrace the concept of reverse vending machines, we move closer to creating a more sustainable and environmentally conscious world. In conclusion, reverse vending machines represent a crucial step toward a more sustainable and responsible approach to waste management. By promoting recycling in diverse public spaces and incorporating user-friendly features, these machines effectively engage individuals in the process of environmental conservation. The collaboration with mobile banking services adds a layer of accessibility and convenience, making recycling a rewarding and seamless experience.

## **References**

- [1] [https://github.com/Unic-coder/Arduino-Projects/blob/main/ESP32-Cam-Object\\_Detector/ESP32-Cam-Object\\_Detector.ino](https://github.com/Unic-coder/Arduino-Projects/blob/main/ESP32-Cam-Object_Detector/ESP32-Cam-Object_Detector.ino)
- [2] <https://github.com/alvarowlfx/ESP32QRCodeReader>
- [3] <https://github.com/NishantSahay7/Nodemcu-to-Google-Sheets/blob/master/Script.txt>