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

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 vanding 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 throughoutmy project with his patience and knowledge. I shall always remain grateful to him for his valuableguidance, advice, encouragement, cordial and amiable contribution to my project. I also want to thank theclosest companionsin 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 recyceling 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

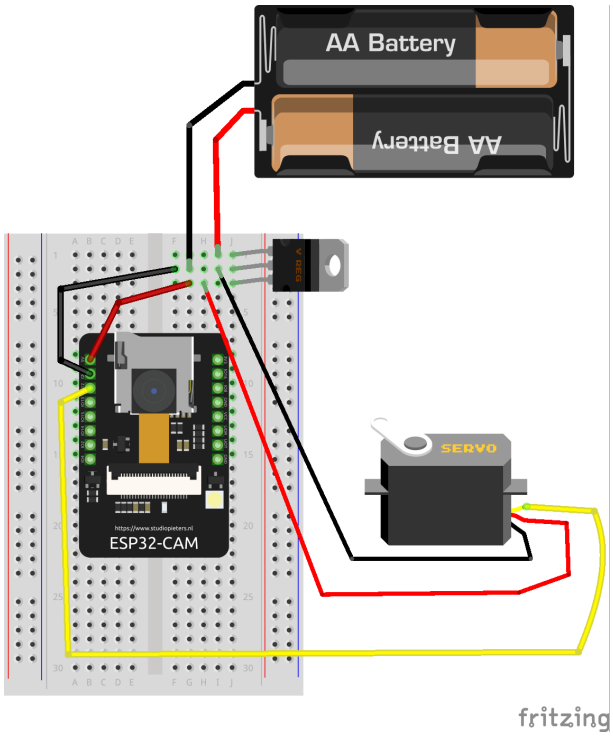


Figure 1: Circuit diagram for bottle detection

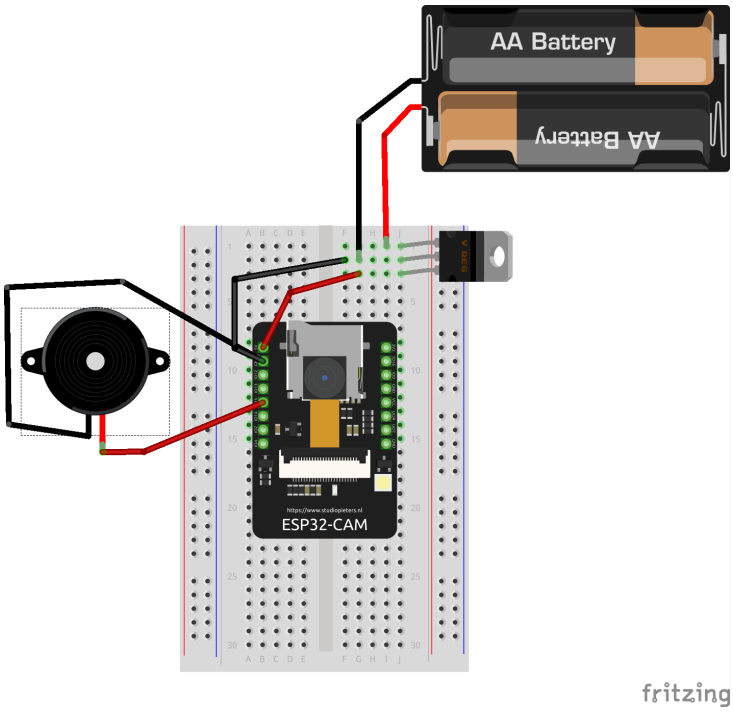


Figure 2: Circuit diagram for QR code scan.

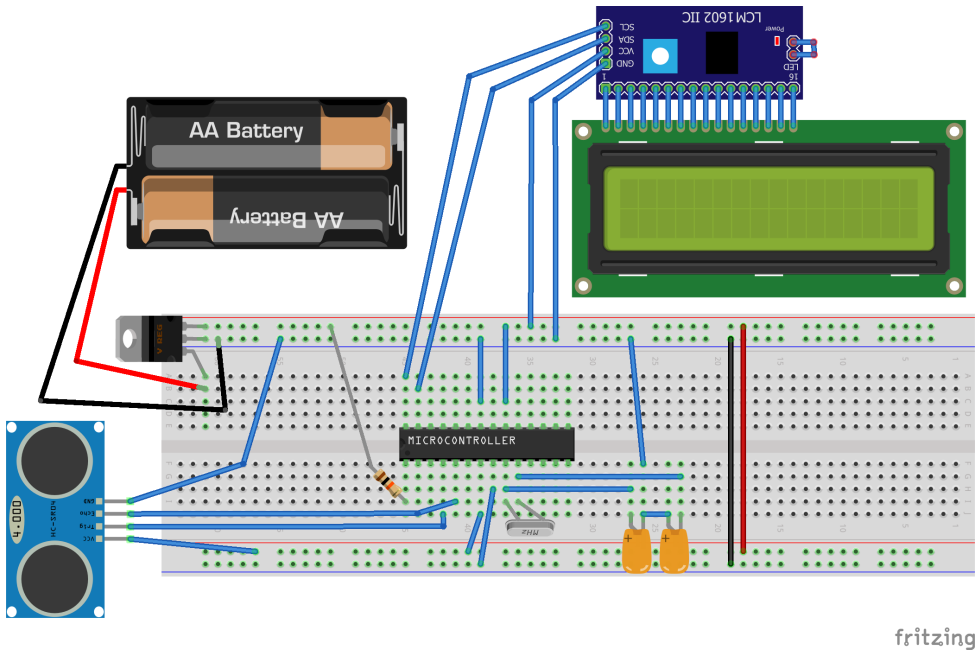


Figure 3: Circuit diagram for bins fullness detection.

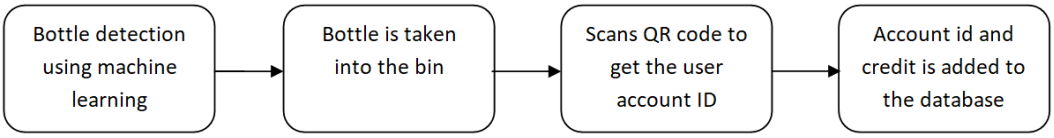


Figure 4: Block diagram for bottle detection and QR code scan.

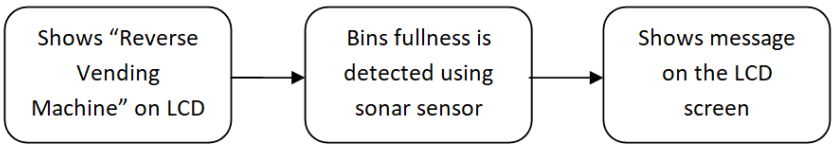


Figure 5: Block diagram for bins fullness detection.

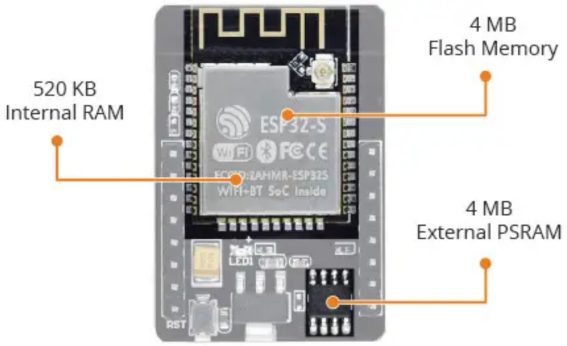
ESP32-cam

Hardware overview

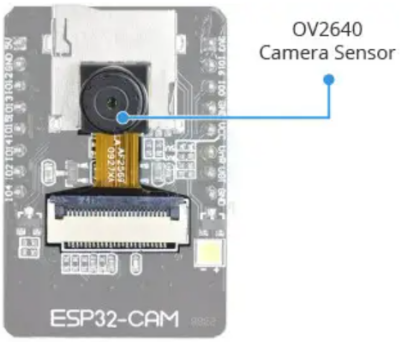
The heart of the ESP32-CAM is an ESP32-S System-on-Chip (SoC) from Ai-Thinker. Being an SoC, the ESP32-S chip contains an entire computer—the microprocessor, RAM, storage, and peripherals—on a single chip. While the chip's capabilities are quite impressive, the ESP32-CAM development board adds even more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

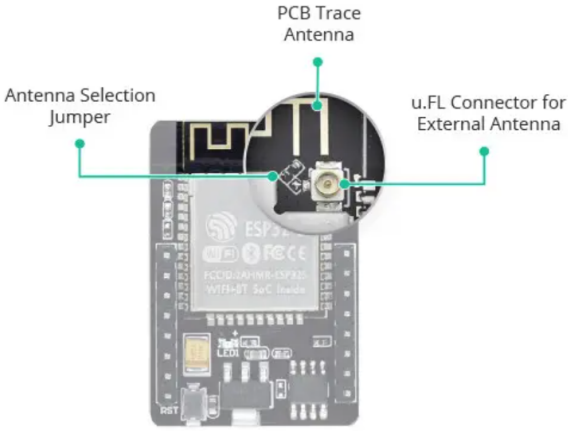


- **The Camera:** The OV2640 camera sensor on the ESP32-CAM is what sets it apart from other ESP32 development boards and makes it ideal for use in video projects like a video doorbell or nanny cam.

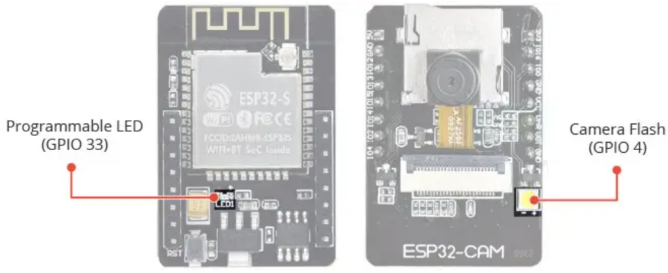


- **The Storage:** The addition of a microSD card slot on the ESP32-CAM is a nice bonus. This allows for limitless expansion, making it a great little board for data loggers or image capture.

- **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.

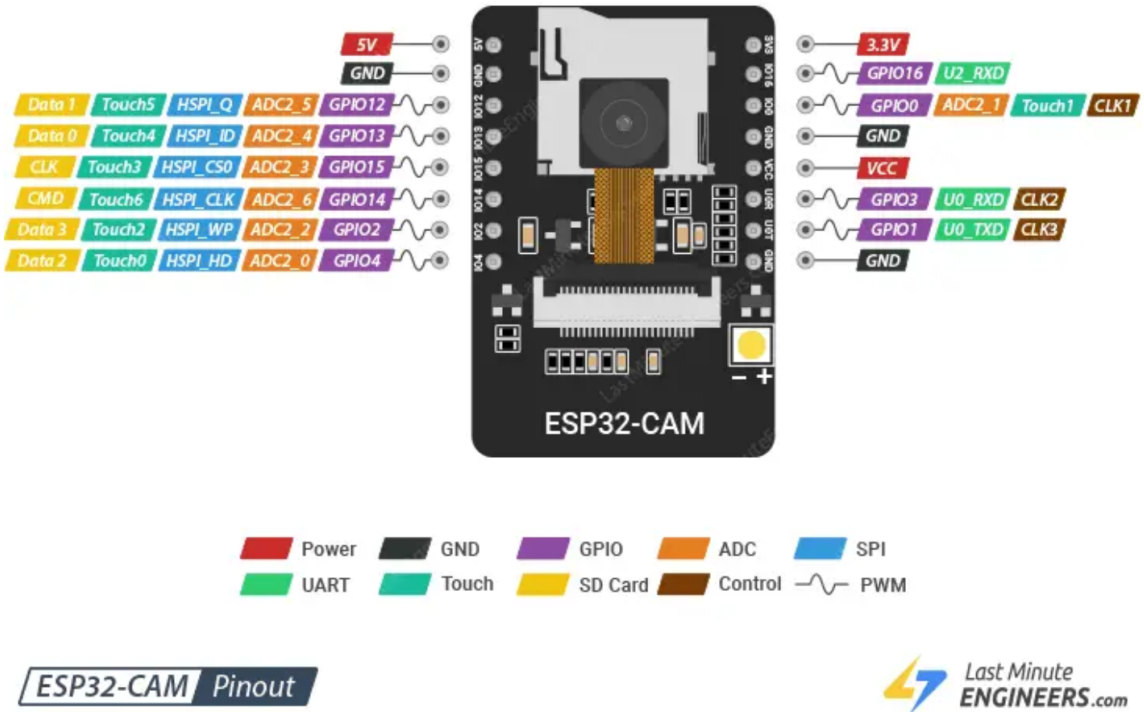


- **LEDs:** The ESP32-CAM has a white square LED. It is intended to be used as a camera flash, but it can also be used for general illumination.



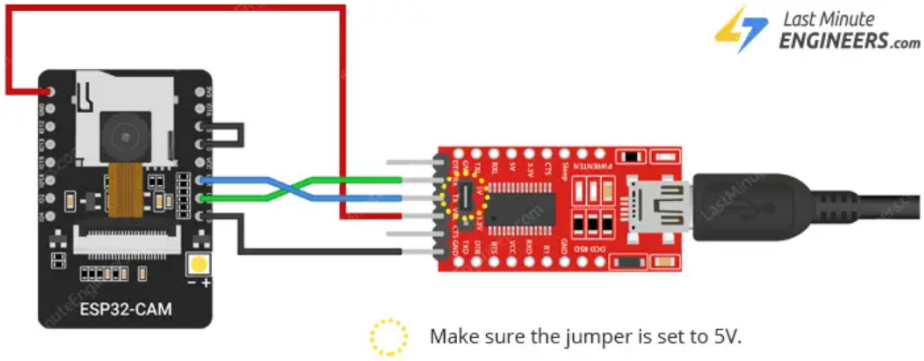
ESP32-cam pinout

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



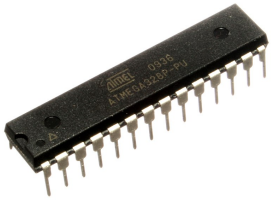
Programming of ESP32-cam Using FTDI Adopter

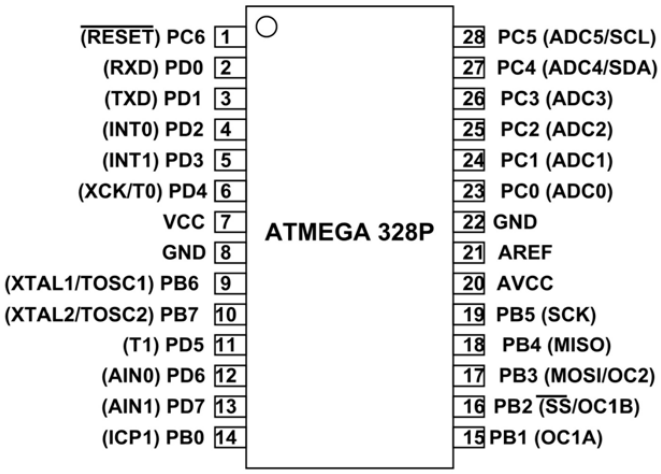
If you've decided to use the FTDI adapter, here's how you connect it to the ESP32-CAM module.



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

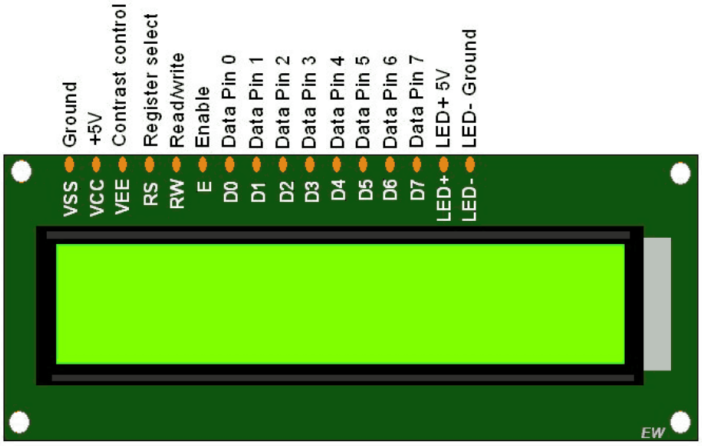
A servo motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. Servo motors are rated in kg/cm (kilogram 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

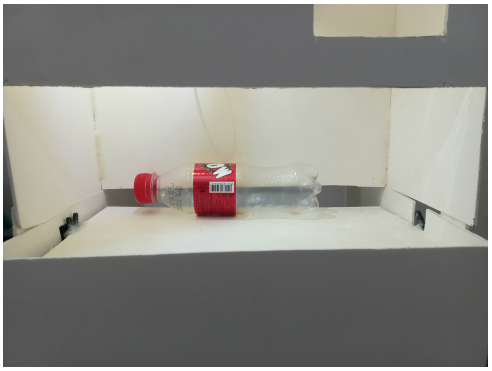
LCDs (Liquid Crystal Displays) are used in embedded system applications for displaying various parameters and status of the system. LCD 16x2 is a 16-pin device that has 2 rows that can accommodate 16 characters each. LCD 16x2 can be used in 4-bit mode or 8-bit mode. It is also possible to create custom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

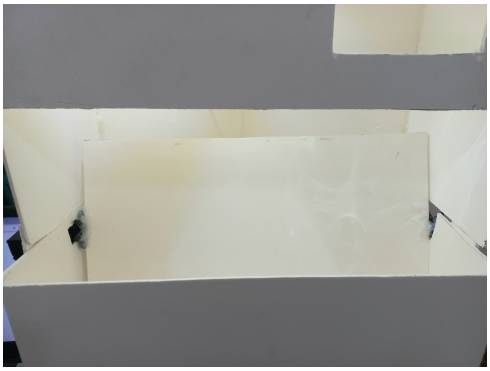


Working Process

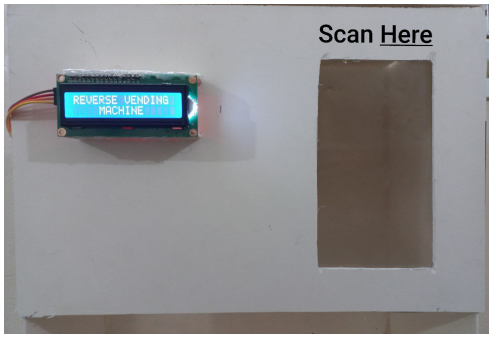
1. After placing the bottle into the compartment ESP-32 cam uses machine learning model[1] to detect the bottle.

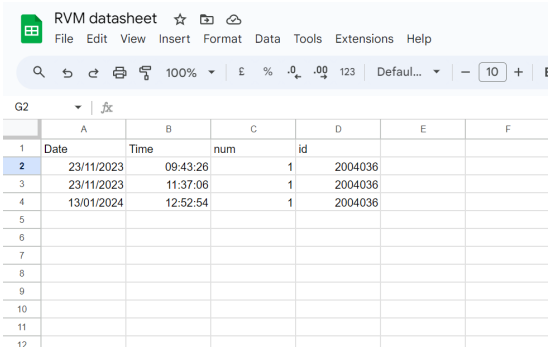
When a bottle is detected, the servo motor moves 90 degree to drop the bottle into the bin.



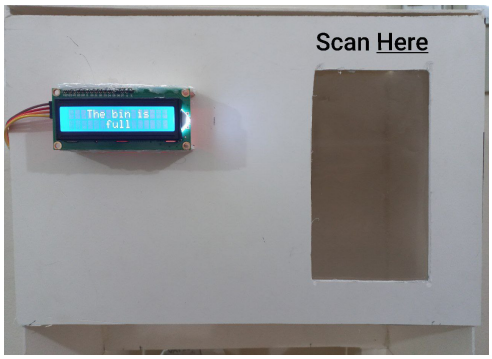


2. Upon accepting the bottle a QR code containing users account ID is scanned via 2nd ESP-32 cam. When the QR code is scanned the buzzer alerts the user that the scan is done. Then the credit gets added to their account.





3. 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)

```
const char* ssid     = "monir zx online";
const char* password = "01718146157";   // http://192.168.125.77

const char* apssid = "ESP32-CAM";
const char* appassword = "12345678";


#include <WiFi.h>
#include <Wire.h>
#include <WiFiClientSecure.h>
#include "esp_camera.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include <ESP32Servo.h>
#include <HTTPClient.h>


String ID = "AKfycbyC39YkVjLq7lJavRwuKNP70LZgdO-n3eT2j0HcwPdfTN2J-peTtAtYt5RmVYBOD9I"; //--> spreadsheet script ID
const char* host = "script.google.com";

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

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
       HTTPS!
62
63   client.setInsecure(); // this is the magical line that
      makes everything work
64
65   if (!client.connect(host, httpPort)) { //works!
66     // Serial.println("connection failed");
67     return;
68   }
69
70   //-----------------------------------------Processing
      data and sending data
71   String url = "/macros/s/" + ID + "/exec?num=";
72
73   url += 1;
74   // Serial.print("Requesting URL: ");
75   // Serial.println(url);
76
77   // This will send the request to the server
78   client.print(String("GET ") + url + " HTTP/1.1\r\n" +
79                "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
       position in variable 'pos'
91    Serial.println("motor moved 1");
92    delay(2000);
93
94    myservo.write(0);
95    Serial.println("motor moved 2");
96    update_google_sheet();
97
98    delay(50000);
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+" ,
      P3= "+P3+" ,P4= "+P4+" ,P5= "+P5+" ,P6= "+P6+" ,P7= "+P
      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       if (P1=="QQVGA")
169         s->set_framesize(s, FRAMESIZE_QQVGA);
170       else if (P1=="HQVGA")
171         s->set_framesize(s, FRAMESIZE_HQVGA);
172       else if (P1=="QVGA")
173         s->set_framesize(s, FRAMESIZE_QVGA);
174       else if (P1=="CIF")
175         s->set_framesize(s, FRAMESIZE_CIF);
176       else if (P1=="VGA")
177         s->set_framesize(s, FRAMESIZE_VGA);
178       else if (P1=="SVGA")
179         s->set_framesize(s, FRAMESIZE_SVGA);
180       else if (P1=="XGA")
181         s->set_framesize(s, FRAMESIZE_XGA);
182       else if (P1=="SXGA")
183         s->set_framesize(s, FRAMESIZE_SXGA);
184       else if (P1=="UXGA")
185         s->set_framesize(s, FRAMESIZE_UXGA);
186       else
187         s->set_framesize(s, FRAMESIZE_QVGA);
188     }
189     else if (cmd=="quality") {
190       sensor_t * s = esp_camera_sensor_get();
191       int val = P1.toInt();
192       s->set_quality(s, val);
193     }
194     else if (cmd=="contrast") {
195       sensor_t * s = esp_camera_sensor_get();
196       int val = P1.toInt();
```

```
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
      .indexOf("HTTPS")==0))
217         Feedback=tcp_https(domain,request,port,wait);
218       else
219         Feedback=tcp_http(domain,request,port,wait);
220     }
221     else if (cmd=="linenotify") {    //message=xxx&
      stickerPackageId=xxx&stickerId=xxx
222       String token = P1;
223       String request = P2;
224       Feedback=LineNotify(token,request,1);
```

```
225       if (Feedback.indexOf("status")!=-1) {
226         int s=Feedback.indexOf("{");
227         Feedback=Feedback.substring(s);
228         int e=Feedback.indexOf("}");
229         Feedback=Feedback.substring(0,e);
230         Feedback.replace("\"","");
231         Feedback.replace("{","");
232         Feedback.replace("}","");
233       }
234     }
235     else if (cmd=="sendCapturedImageToLineNotify") {
236       Feedback=sendCapturedImageToLineNotify(P1);
237       if (Feedback=="") Feedback="The image failed to send.
      <br>The framesize may be too large.";
238     }
239     else {
240       Feedback="Command is not defined.";
241     }
242     if (Feedback=="") Feedback=Command;
243   }
244
245   void setup() {
246     WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
247
248     Serial.begin(115200);
249     Serial.setDebugOutput(true);
250     // Serial.println();
251
252     pinMode (LED_BUILTIN, OUTPUT);//Specify that LED pin is
      output
```

```
253
254   myservo.attach(12);
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
      higher quality
284       config.fb_count = 1;
285     }
286     // camera init
287     esp_err_t err = esp_camera_init(&config);
288     if (err != ESP_OK) {
289       Serial.printf("Camera init failed with error 0x%x",
      err);
290       delay(1000);
291       ESP.restart();
292     }
293
294     //drop down frame size for higher initial frame rate
295     sensor_t * s = esp_camera_sensor_get();
296     s->set_framesize(s, FRAMESIZE_QVGA);  //UXGA|SXGA|XGA|
      SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
297
298     ledcAttachPin(4, 4);
299     ledcSetup(4, 5000, 8);
300
301     WiFi.mode(WIFI_AP_STA);
302
303     //WiFi.config(IPAddress(192, 168, 201, 100), IPAddress(1
      92, 168, 201, 2), IPAddress(255, 255, 255, 0));
304
305     WiFi.begin(ssid, password);
306
307     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)
      apssid).c_str(), appassword);
321     Serial.println("");
322     Serial.println("STAIP address: ");
323     Serial.println(WiFi.localIP());
324
325     for (int i=0;i<5;i++) {
326       ledcWrite(4,10);
327       delay(200);
328       ledcWrite(4,0);
329       delay(200);
330     }
331   }
332   else {
333     WiFi.softAP((WiFi.softAPIP().toString()+"_"+(String)
      apssid).c_str(), appassword);
334
335     for (int i=0;i<2;i++) {
```

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

```
 models/coco-ssd@2.1.0"> </script>
</head><body>
<img id="ShowImage" src="" style="display:none">
<canvas id="canvas" width="0" height="0"></canvas>
<table>
<tr>
  <td><input type="button" id="restart" value="Restart"
 ></td>
  <td colspan="2"><input type="button" id="getStill"
 value="Start Detect" style="display:none"></td>
</tr>
<tr>
  <td>Object</td>
  <td colspan="2">
      <select id="object" onchange="count.innerHTML='';"
 >

        <option value="bottle">bottle</option>

      </select>
  </td>
  <td><span id="count" style="color:red"><span></td>
</tr>
<tr>
  <td>ScoreLimit</td>
  <td colspan="2">
    <select id="score">
      <option value="1.0">1</option>
      <option value="0.9">0.9</option>
      <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
    x240)</option>
419         <option value="HQVGA">HQVGA(240x176)</option>
420         <option value="QQVGA">QQVGA(160x120)</option>
421       </select>
422     </td>
423   </tr>
424   <tr>
425     <td>Flash</td>
426     <td colspan="2"><input type="range" id="flash" min="0"
     max="255" value="0"></td>
427   </tr>
428   <tr>
429     <td>Quality</td>
430     <td colspan="2"><input type="range" id="quality" min="
    10" max="63" value="10"></td>
431   </tr>
432   <tr>
433     <td>Brightness</td>
434     <td colspan="2"><input type="range" id="brightness"
    min="-2" max="2" value="0"></td>
435   </tr>
436   <tr>
437     <td>Contrast</td>
438     <td colspan="2"><input type="range" id="contrast" min=
    "-2" max="2" value="0"></td>
439   </tr>
440   <tr><td>Rotate</td>
441     <td align="left" colspan="2">
442       <select onchange="document.getElementById('canvas').
```

```
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     var count = document.getElementById('count');
464     var result = document.getElementById('result');
465     var flash = document.getElementById('flash');
466     var lastValue="";
467     var myTimer;
468     var restartCount=0;
469     var Model;
470     getStill.onclick = function (event) {
```

```
471         clearInterval(myTimer);
472         myTimer = setInterval(function(){error_handle()
    ;},5000);
473         ShowImage.src=location.origin+'/?getstill='+Math.
    random();
474       }
475       function error_handle() {
476         restartCount++;
477         clearInterval(myTimer);
478         if (restartCount<=2) {
479           result.innerHTML = "Get still error. <br>Restart
    ESP32-CAM "+restartCount+" times.";
480           myTimer = setInterval(function(){getStill.click()
    ;},10000);
481         }
482         else
483           result.innerHTML = "Get still error. <br>Please
    close the page and check ESP32-CAM.";
484       }
485       ShowImage.onload = function (event) {
486         clearInterval(myTimer);
487         restartCount=0;
488         canvas.setAttribute("width", ShowImage.width);
489         canvas.setAttribute("height", ShowImage.height);
490         if (mirrorimage.value==1) {
491           context.translate((canvas.width + ShowImage.width)
     / 2, 0);
492           context.scale(-1, 1);
493           context.drawImage(ShowImage, 0, 0, ShowImage.width
    , ShowImage.height);
```

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

```
519       fetch(document.location.origin+'/?contrast='+this.
    value+';stop');
520     }
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:
    canvas.height;
535         var objectCount=0;
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             context.rect(x, y, width, height);
549             context.stroke();
550             context.lineWidth = "2";
551             context.fillStyle = "red";
552             context.font = Math.round(s/30) + "px Arial";
553             context.fillText(Predictions[i].class, x, y);
554             //context.fillText(i, x, y);
555             result.innerHTML+= "[ "+i+" ] "+Predictions[i
    ].class+", "+Math.round(Predictions[i].score*100)+"%, "
    +Math.round(x)+", "+Math.round(y)+", "+Math.round(width
    )+", "+Math.round(height)+"<br>";
556
557             if (Predictions[i].class==object.value&&
    Predictions[i].score>=score.value) {
558               objectCount++;
559             }
560           }
561           count.innerHTML = objectCount;
562         }
563         else {
564           result.innerHTML = "Unrecognizable";
565           count.innerHTML = "0";
566         }
567
568       //if (count.innerHTML != lastValue) {
569           lastValue = count.innerHTML;
570           if (objectCount>0) {
571             //$.ajax({url: document.location.origin+'/?
    serial='+object.value+';stop', async: false});
```

```
572             $.ajax({url: document.location.origin+'/?
    detectCount='+object.value+';'+String(objectCount)+';
    stop', async: false});
573           }
574       //}
575         try {
576           document.createEvent("TouchEvent");
577           setTimeout(function(){getStill.click();},250);
578         }
579         catch(e) {
580           setTimeout(function(){getStill.click();},150);
581         }
582       });
583     }
584     function getFeedback(target) {
585       var data = $.ajax({
586       type: "get",
587       dataType: "text",
588       url: target,
589       success: function(response)
590         {
591           result.innerHTML = response;
592         },
593         error: function(exception)
594         {
595           result.innerHTML = 'fail';
596         }
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
      ="";P6="";P7="";P8="";P9="";
607   ReceiveState=0,cmdState=1,strState=1,questionstate=0,
      equalstate=0,semicolonstate=0;
608
609    WiFiClient client = server.available();
610
611   digitalWrite(LED_BUILTIN, HIGH);
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                 if(!fb) {
630                   // Serial.println("Camera capture failed")
    ;
631                   delay(1000);
632                   ESP.restart();
633                 }
634 
635                 client.println("HTTP/1.1 200 OK");
636                 client.println("Access-Control-Allow-Origin:
     *");
637                 client.println("Access-Control-Allow-Headers
    : Origin, X-Requested-With, Content-Type, Accept");
638                 client.println("Access-Control-Allow-Methods
    : GET,POST,PUT,DELETE,OPTIONS");
639                 client.println("Content-Type: image/jpeg");
640                 client.println("Content-Disposition: form-
    data; name=\"imageFile\"; filename=\"picture.jpg\"");
641                 client.println("Content-Length: " + String(
    fb->len));
642                 client.println("Connection: close");
643                 client.println();
644 
645                 uint8_t *fbBuf = fb->buf;
646                 size_t fbLen = fb->len;
647                 for (size_t n=0;n<fbLen;n=n+1024) {
648                   if (n+1024<fbLen) {
649                     client.write(fbBuf, 1024);
650                     fbBuf += 1024;
651                   }
```

```
652                   else if (fbLen%1024>0) {
653                     size_t remainder = fbLen%1024;
654                     client.write(fbBuf, remainder);
655                   }
656                 }
657 
658                 esp_camera_fb_return(fb);
659 
660                 pinMode(4, OUTPUT);
661                 digitalWrite(4, LOW);
662               }
663               else {
664 
665                 client.println("HTTP/1.1 200 OK");
666                 client.println("Access-Control-Allow-Headers
    : Origin, X-Requested-With, Content-Type, Accept");
667                 client.println("Access-Control-Allow-Methods
    : GET,POST,PUT,DELETE,OPTIONS");
668                 client.println("Content-Type: text/html;
    charset=utf-8");
669                 client.println("Access-Control-Allow-Origin:
     *");
670                 client.println("Connection: close");
671                 client.println();
672 
673                 String Data="";
674                 if (cmd!="")
675                   Data = Feedback;
676                 else {
677                   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
      wait)
716 {
717     WiFiClient client_tcp;
718
719     if (client_tcp.connect(domain.c_str(), port))
720     {
721       Serial.println("GET " + request);
722       client_tcp.println("GET " + request + " HTTP/1.1");
723       client_tcp.println("Host: " + domain);
724       client_tcp.println("Connection: close");
725       client_tcp.println();
726
727       String getResponse="",Feedback="";
728       boolean state = false;
729       int waitTime = 3000;   // timeout 3 seconds
730       long startTime = millis();
731       while ((startTime + waitTime) > millis())
732       {
```

```
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))
    break;
749          }
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   if(!fb) {
764     Serial.println("Camera capture failed");
765     delay(1000);
766     ESP.restart();
767     return "";
768   }
769
770   WiFiClientSecure client_tcp;
771   client_tcp.setInsecure();   //run version 1.0.5 or above
772
773   Serial.println("Connect to notify-api.line.me");
774
775   if (client_tcp.connect("notify-api.line.me", 443)) {
776     Serial.println("Connection successful");
777
778     String message = "Welcome to Taiwan.";
779     String head = "--Taiwan\r\nContent-Disposition: form-
      data; name=\"message\"; \r\n\r\n" + message + "\r\n--
      Taiwan\r\nContent-Disposition: form-data; name=\"
      imageFile\"; filename=\"esp32-cam.jpg\"\r\nContent-Type
      : image/jpeg\r\n\r\n";
780     String tail = "\r\n--Taiwan--\r\n";
781
782     uint16_t imageLen = fb->len;
783     uint16_t extraLen = head.length() + tail.length();
784     uint16_t totalLen = imageLen + extraLen;
785
786     client_tcp.println("POST /api/notify HTTP/1.1");
787     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(
          totalLen));
791       client_tcp.println("Content-Type: multipart/form-data;
           boundary=Taiwan");
792       client_tcp.println();
793       client_tcp.print(head);
794
795       uint8_t *fbBuf = fb->buf;
796       size_t fbLen = fb->len;
797       for (size_t n=0;n<fbLen;n=n+1024) {
798         if (n+1024<fbLen) {
799           client_tcp.write(fbBuf, 1024);
800           fbBuf += 1024;
801         }
802         else if (fbLen%1024>0) {
803           size_t remainder = fbLen%1024;
804           client_tcp.write(fbBuf, remainder);
805         }
806       }
807
808       client_tcp.print(tail);
809       esp_camera_fb_return(fb);
810
811       int waitTime = 10000;   // timeout 10 seconds
812       long startTime = millis();
813       boolean state = false;
814       while ((startTime + waitTime) > millis())
815       {
```

```
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)

```
    const char* ssid = "monir zx online";
const char* password = "01718146157";


const char* apssid = "esp32-cam1";
const char* appassword = "12345678";

String Feedback="";
String Command="",cmd="",P1="",P2="",P3="",P4="",P5="",P6=
   "",P7="",P8="",P9="";

byte ReceiveState=0,cmdState=1,strState=1,questionstate=0,
   equalstate=0,semicolonstate=0;

#include <WiFi.h>
#include "esp_camera.h"
#include <HTTPClient.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "quirc.h"
#include <HardwareSerial.h>
#include "esp_system.h"
String ID = "
   AKfycbxOATD7b1rALzoFIdvE810D53u7oJHPkEKzCCcWP9N5ZNUB5_PsVKWlEk_Om
   -SNOkuasQ"; //--> spreadsheet script ID
const char* host = "script.google.com";
TaskHandle_t Task;

```

```
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#define BUZZER_PIN        15  // ESP32-cam pin GPIO15 connected to Buzzer's pin

void update_google_sheet(String id)
{
  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClientSecure client;
  const int httpPort = 443; // 80 is for HTTP / 443 is for HTTPS!

  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
     data and sending data
60  String url = "/macros/s/" + ID + "/exec?num=";
61
62  url += 1;
63  url += "&id=";
64  url += String(id);
65
66  Serial.print("Requesting URL: ");
67  Serial.println(url);
68
69  // This will send the request to the server
70  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
71               "Host: " + host + "\r\n" +
72               "Connection: close\r\n\r\n");
73
74  Serial.println();
75  Serial.println("closing connection");
76  digitalWrite(BUZZER_PIN, HIGH); // turn on
77  delay(1500);
78  digitalWrite(BUZZER_PIN, LOW); // turn off
79  Serial.print('A');
80  delay(50000);
```

```
}

struct QRCodeData
{
  bool valid;
  int dataType;
  uint8_t payload[1024];
  int payloadLen;
};

struct quirc *q = NULL;
uint8_t *image = NULL;
camera_fb_t * fb = NULL;
struct quirc_code code;
struct quirc_data data;
quirc_decode_error_t err;
struct QRCodeData qrCodeData;
String QRCodeResult = "";

WiFiServer server(80);
WiFiClient client;

camera_config_t config;

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  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",
      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
      created task */
214               0);        /* pin task to core 0 */
215
216   Serial.print("listenConnection running on core ");
217   Serial.println(xPortGetCoreID());
218 }
219
220 void loop() {
221   listenConnection();
222 }
223
224 void QRCodeReader( void * pvParameters ){
225   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
      ->width, fb->height, fb->len);
246       memcpy(image, fb->buf, fb->len);
247       quirc_end(q);
248       //Serial.printf("quirc_end\r\n");
249
250       int count = quirc_count(q);
251       if (count > 0) {
252         Serial.println(count);
253         quirc_extract(q, 0, &code);
254         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     Serial.printf("Mask: %d\n", data->mask);
278     Serial.printf("Length: %d\n", data->payload_len);
279     Serial.printf("Payload: %s\n", data->payload);
280
281     QRCodeResult = (const char *)data->payload;
282     Serial.println(QRCodeResult);
283     update_google_sheet(QRCodeResult);
```

```
}


static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<!DOCTYPE html>
<head>
  <title></title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
   initial-scale=1">
</head>
<body>
<canvas id="canvas" width="320" height="240"></canvas><br>
Flash<input type="range" id="flash" min="0" max="255"
   value="0">
<input type="button" value="Get Still" onclick="getStill()
   ;"><br>
<div id="result" style="color:red"></div>
</body>
</html>

<script>
  var canvas = document.getElementById('canvas');
  var context = canvas.getContext('2d');
  var flash = document.getElementById('flash');
  var result = document.getElementById('result');

  flash.onchange = function() {
    var query = document.location.origin+"/?flash="+flash.
   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.
      responseText
319         if (arrayBuffer) {
320           var byteArray = new Uint8Array(arrayBuffer);
321           var imgData=context.getImageData(0,0,canvas.width,
      canvas.height);
322           var val = 0;
323           for (var i=0;i<imgData.data.length;i+=4) {
324             val = parseInt(byteArray[i/4], 10);
325                 imgData.data[i]=val;
326                 imgData.data[i+1]=val;
327                 imgData.data[i+2]=val;
328                 imgData.data[i+3]=255;
329             }
330             context.putImageData(imgData,0,0);
331             setTimeout(function(){getStill();}, 100);
332           }
333       };
334
335       xhr.send(null);
336     }
337     getStill();
```

```
</script>
)rawliteral";

void listenConnection() {
  Feedback="";Command="";cmd="";P1="";P2="";P3="";P4="";P5
   ="";P6="";P7="";P8="";P9="";
  ReceiveState=0,cmdState=1,strState=1,questionstate=0,
   equalstate=0,semicolonstate=0;

  client = server.available();

  if (client) {
    String currentLine = "";

    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        getCommand(c);
        if (c == '\n') {
          if (currentLine.length() == 0) {
            if (cmd=="getstill") {
              getStill();
            } else {
              mainPage();
            }
            Feedback="";
            break;
          } else {
            currentLine = "";
          }
```

```
366           } else if (c != '\r') {
367             currentLine += c;
368           }
369
370           if ((currentLine.indexOf("?")!=-1)&&(currentLine.
    indexOf(" HTTP")!=-1)) {
371             if (Command.indexOf("stop")!=-1) {
372               client.println();
373               client.println();
374               client.stop();
375             }
376             currentLine="";
377             Feedback="";
378             ExecuteCommand();
379           }
380         }
381       }
382       delay(1);
383       client.stop();
384     }
385 }
386
387 void mainPage() {
388       client.println("HTTP/1.1 200 OK");
389       client.println("Access-Control-Allow-Headers: Origin,
    X-Requested-With, Content-Type, Accept");
390       client.println("Access-Control-Allow-Methods: GET,POST
    ,PUT,DELETE,OPTIONS");
391       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,
    PUT,DELETE,OPTIONS");
422   client.println("Content-Type: application/octet-stream")
    ;
423   client.println();
424
425   for (size_t n=0;n<fbLen;n=n+1024) {
426     if (n+1024<fbLen) {
427       client.write(fbBuf, 1024);
428       fbBuf += 1024;
429     }
430     else if (fbLen%1024>0) {
431       size_t remainder = fbLen%1024;
432       client.write(fbBuf, remainder);
433     }
434   }
435   esp_camera_fb_return(fb);
436
437   pinMode(4, OUTPUT);
438   digitalWrite(4, LOW);
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
      to 0x27 for a 16 chars and 2 line display
16
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
      Output
37    pinMode(echoPin, INPUT); // Sets the echoPin as an Input
38    lcd.init();                      // initialize the lcd
39    // Print a message to the LCD.
40    lcd.backlight();
41    lcd.setCursor(0,0);
42    lcd.print("REVERSE VENDING");
43    lcd.setCursor(0,1);
44    lcd.print("    MACHINE    ");
45  }
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';
```

```
}
else {
var sheet_id = '1ClNE3e6AVFAl5YZ_-
   vDhdz8hNxUryIkL4sdLqJnZkEQ'; // Spreadsheet ID
var sheet = SpreadsheetApp.openById(sheet_id).
   getActiveSheet();
var newRow = sheet.getLastRow();
var value1 = sheet.getRange(newRow, 3).getValue();
var value2 = sheet.getRange(newRow, 4).getValue();
var rowData = [];
if (value1 == 1 && value2 == ''){
  var Curr_Date = new Date();
  rowData[0] = Curr_Date; // Date in column A
  var Curr_Time = Utilities.formatDate(Curr_Date, "Asia/
   Bangkok", 'HH:mm:ss');
  rowData[1] = Curr_Time; // Time in column B
  for (var param in e.parameter) {
    Logger.log('In for loop, param=' + param);
    var value = stripQuotes(e.parameter[param]);
    Logger.log(param + ':' + e.parameter[param]);
    switch (param) {

      case 'num':
      rowData[2] = value;
      result = 'OK';
      break;

      case 'id':
      rowData[3] = value;
      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 = '1ClNE3e6AVFAl5YZ_-
       vDhdz8hNxUryIkL4sdLqJnZkEQ'; // Spreadsheet ID
```

```
var sheet = SpreadsheetApp.openById(sheet_id).
   getActiveSheet();
var newRow = sheet.getLastRow();
var rowData = [];
var Curr_Date = new Date();
rowData[0] = Curr_Date; // Date in column A
var Curr_Time = Utilities.formatDate(Curr_Date, "Asia/
   Bangkok", 'HH:mm:ss');
rowData[1] = Curr_Time; // Time in column B
for (var param in e.parameter) {
Logger.log('In for loop, param=' + param);
var value = stripQuotes(e.parameter[param]);
Logger.log(param + ':' + e.parameter[param]);
switch (param) {

      case 'num':
      rowData[2] = value;
      result = 'OK';
      break;

      default:
      result = "unsupported parameter";
    }

  }

Logger.log(JSON.stringify(rowData));
var newRange = sheet.getRange(newRow, 1, 1, rowData.length
   );
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

[2] https://github.com/alvarowolfx/ESP32QRCodeReader

[3] https://github.com/NishantSahay7/Nodemcu-to-Google-Sheets/blob/master/Script.txt