Faculty of Engineering at Shoubra
Electrical Engineering Department

كلية الهندسة بشبرا

FACULTY OF ENGINEERING AT SHOUBRA

Names :

1) Mohamed ahmed ibrahem mansour.
2) Mohamed osama mohamed bakr .
3) Mohamed osama aowad aowad.
4) Mohamed ahmed saleh.
5) Mohamed ahmed rabea.
6) Mohamed saad abdelnaeem.
7) Maher shehta abdo badran.

Reoprt about : QR Scanner by ESP-32 CAM

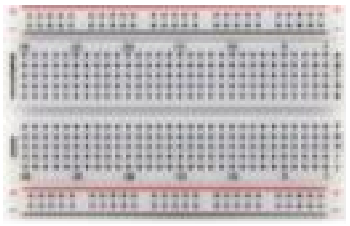
Submitted to Dr \ Tarek Elewa , **Subject :** electronics

Outlines :-

Introduction About the project :

A device to scan an QR code and do one of the two applications :
1) Turning on/off red and green LEDs .
2) Entering a website .

Appendix & Components :



breadboard

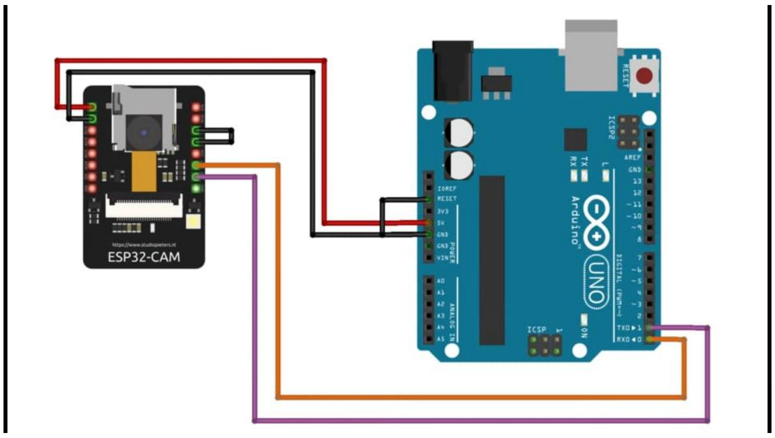
Arduino

Encoder

ESP-32

Uploading the code on esp32 CAM :

1) Esp-32 CAM connection :



2) How to upload the code on esp-32 CAM :

** We connected the connection as it is in the picture, then we connected the Arduino to the laptop and uploaded the code, then we opened the serial monitor to get the IP that we will enter from the page.

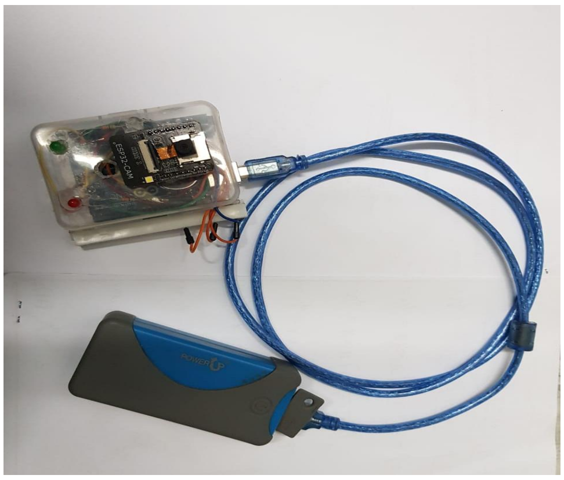
The code :-

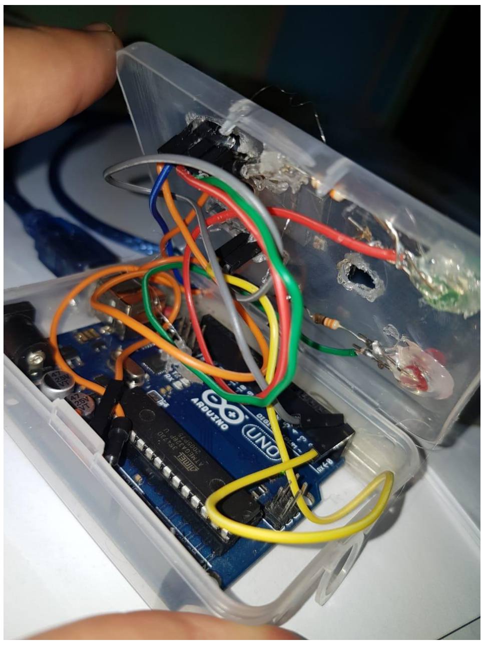
```
const char* ssid = "OPPO A92";
const char* password = "Mohamed6644";
// http://192.168.4.1
const char* apssid = "esp32-cam";
const char* appassword = "12345678";

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,semic
olonstate=0;

#include <WiFi.h>
#include "esp_camera.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "quirc.h"

TaskHandle_t Task;

//ESP32-CAM
#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 LED_Green   12
#define LED_Red     13

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();
  pinMode(LED_Red, OUTPUT);
  pinMode(LED_Green, OUTPUT);

//  https://github.com/espressif/esp32-
camera/blob/master/driver/include/esp_camera.h
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 10000000;
  config.pixel_format = PIXFORMAT_GRAYSCALE;
  config.frame_size = FRAMESIZE_QVGA;
  config.jpeg_quality = 15;
  config.fb_count = 1;


  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    ESP.restart();
  }

  sensor_t * s = esp_camera_sensor_get();
  s->set_framesize(s, FRAMESIZE_QVGA);

  //s->set_vflip(s, 1);
  //s->set_hmirror(s, 1);

  //(GPIO4)
  ledcAttachPin(4, 4);
  ledcSetup(4, 5000, 8);

  WiFi.mode(WIFI_AP_STA);  // WiFi.mode(WIFI_AP); WiFi.mode(WIFI_STA);

  //Client IP
  //WiFi.config(IPAddress(192, 168, 201, 100), IPAddress(192, 168, 201,
2), IPAddress(255, 255, 255, 0));

  for (int i=0;i<2;i++) {
    WiFi.begin(ssid, password);

    delay(1000);
    Serial.println("");
    Serial.print("Connecting to ");
    Serial.println(ssid);

    long int StartTime=millis();
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        if ((StartTime+5000) < millis()) break;
    }

    if (WiFi.status() == WL_CONNECTED) {
      WiFi.softAP((WiFi.localIP().toString()+"_"+(String)apssid).c_str()
, appassword);   //設定 SSID 顯示客戶端 IP
      Serial.println("");
      Serial.println("STAIP address: ");
      Serial.println(WiFi.localIP());
      Serial.println("");
      for (int i=0;i<5;i++) {
        ledcWrite(4,10);
        delay(200);
        ledcWrite(4,0);
        delay(200);
      }
      break;
    }
  }

  if (WiFi.status() != WL_CONNECTED) {
    WiFi.softAP((WiFi.softAPIP().toString()+"_"+(String)apssid).c_str(),
appassword);

    for (int i=0;i<2;i++) {
      ledcWrite(4,10);
      delay(1000);
      ledcWrite(4,0);
      delay(1000);
    }
  }


  //WiFi.softAPConfig(IPAddress(192, 168, 4, 1), IPAddress(192, 168, 4,
1), IPAddress(255, 255, 255, 0));
  Serial.println("");
  Serial.println("APIP address: ");
  Serial.println(WiFi.softAPIP());
  Serial.println("");


  pinMode(4, OUTPUT);
  digitalWrite(4, LOW);

  server.begin();

  xTaskCreatePinnedToCore(
             QRCodeReader, /* Task function. */
             "Task",   /* name of task. */
             10000,     /* Stack size of task */
             NULL,      /* parameter of the task */
             1,         /* priority of the task */
             &Task,    /* Task handle to keep track of created task */
             0);        /* pin task to core 0 */

  Serial.print("listenConnection running on core ");
  Serial.println(xPortGetCoreID());
}
```

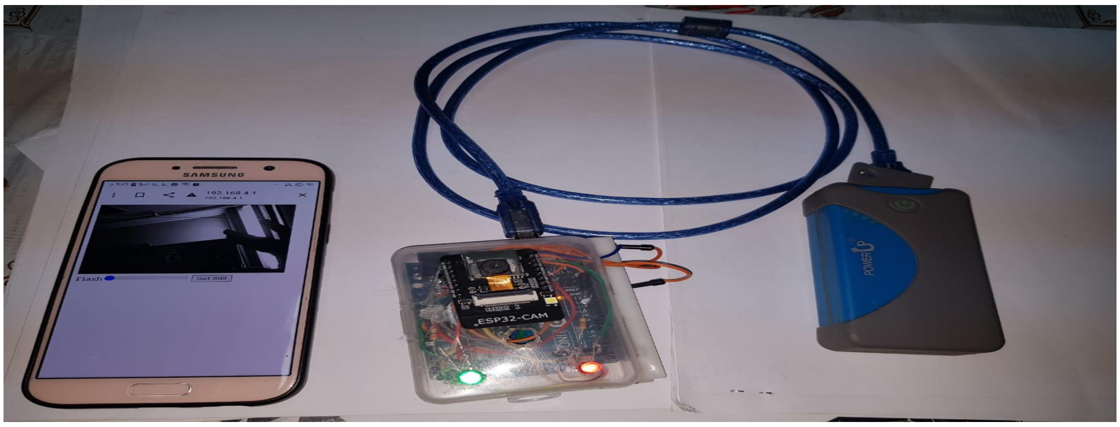
```
void loop() {
  listenConnection();
}

void QRCodeReader( void * pvParameters ){
  Serial.print("QRCodeReader running on core ");
  Serial.println(xPortGetCoreID());

  while(1){
      q = quirc_new();
      if (q == NULL){
        Serial.print("can't create quirc object\r\n");
        continue;
      }

      fb = esp_camera_fb_get();
      if (!fb)
      {
        Serial.println("Camera capture failed");
        continue;
      }

      //Serial.printf("quirc_begin\r\n");
      quirc_resize(q, fb->width, fb->height);
      image = quirc_begin(q, NULL, NULL);
      //Serial.printf("Frame w h len: %d, %d, %d \r\n", fb->width, fb-
>height, fb->len);
      memcpy(image, fb->buf, fb->len);
      quirc_end(q);
      //Serial.printf("quirc_end\r\n");

      int count = quirc_count(q);
      if (count > 0) {
        Serial.println(count);
        quirc_extract(q, 0, &code);
        err = quirc_decode(&code, &data);

        if (err){
          Serial.println("Decoding FAILED");
          QRCodeResult = "Decoding FAILED";
        } else {
          Serial.printf("Decoding successful:\n");
          dumpData(&data);
          /*
          qrCodeData.dataType = data.data_type;
          for (int j = 0; j < data.payload_len; j++)
          {
            qrCodeData.payload[j] = data.payload[j];
          }
          qrCodeData.valid = true;
          qrCodeData.payload[data.payload_len] = '\0';
          qrCodeData.payloadLen = data.payload_len;
          */
          //vTaskDelay(3000/portTICK_RATE_MS);
        }
        Serial.println();
      }

      esp_camera_fb_return(fb);
      fb = NULL;
      image = NULL;
      quirc_destroy(q);
  }
}

void dumpData(const struct quirc_data *data)
{
  Serial.printf("Version: %d\n", data->version);
  Serial.printf("ECC level: %c\n", "MLHQ"[data->ecc_level]);
  Serial.printf("Mask: %d\n", data->mask);
  Serial.printf("Length: %d\n", data->payload_len);
  Serial.printf("Payload: %s\n", data->payload);

  QRCodeResult = (const char *)data->payload;
}


void ExecuteCommand() {
  //Serial.println("");
  //Serial.println("Command: "+Command);





  if (cmd!="getstill") {
    Serial.println("cmd= "+cmd+" ,P1= "+P1+" ,P2= "+P2+" ,P3=
"+P3+" ,P4= "+P4+" ,P5= "+P5+" ,P6= "+P6+" ,P7= "+P7+" ,P8= "+P8+" ,P9=
"+P9);
    Serial.println("");
  }

  //  http://192.168.xxx.xxx?cmd=P1;P2;P3;P4;P5;P6;P7;P8;P9
  if (cmd=="your cmd") {
    // You can do anything
    // Feedback="<font color=\_Red\">Hello World</font>";
  }
  else if (cmd=="ip") {  //APIP, STAIP
```

```
    Feedback="AP IP: "+WiFi.softAPIP().toString();
    Feedback+="<br>";
    Feedback+="STA IP: "+WiFi.localIP().toString();
  }
  else if (cmd=="mac") {
    Feedback="STA MAC: "+WiFi.macAddress();
  }
  else if (cmd=="restart") {  //WIFI
    ESP.restart();
  }
  else if (cmd=="digitalwrite") {
    ledcDetachPin(P1.toInt());
    pinMode(P1.toInt(), OUTPUT);
    digitalWrite(P1.toInt(), P2.toInt());
  }
  else if (cmd=="digitalread") {
    Feedback=String(digitalRead(P1.toInt()));
  }
  else if (cmd=="analogwrite") {
    if (P1=="4") {
      ledcAttachPin(4, 4);
      ledcSetup(4, 5000, 8);
      ledcWrite(4,P2.toInt());
    }
    else {
      ledcAttachPin(P1.toInt(), 9);
      ledcSetup(9, 5000, 8);
      ledcWrite(9,P2.toInt());
    }
  }
  else if (cmd=="analogread") {
    Feedback=String(analogRead(P1.toInt()));
  }
  else if (cmd=="touchread") {
    Feedback=String(touchRead(P1.toInt()));
  }
  else if (cmd=="framesize") {
    sensor_t * s = esp_camera_sensor_get();
    int val = P1.toInt();
    s->set_framesize(s, (framesize_t)val);
  }
  else if (cmd=="quality") {
    sensor_t * s = esp_camera_sensor_get();
    int val = P1.toInt();
    s->set_quality(s, val);
  }
  else if (cmd=="contrast") {
    sensor_t * s = esp_camera_sensor_get();
    int val = P1.toInt();
    s->set_contrast(s, val);
  }
  else if (cmd=="brightness") {
    sensor_t * s = esp_camera_sensor_get();
    int val = P1.toInt();
    s->set_brightness(s, val);
  }
  else if (cmd=="hmirror") {
    sensor_t * s = esp_camera_sensor_get();
    int val = P1.toInt();
    s->set_hmirror(s, val);
  }
  else if (cmd=="vflip") {
    sensor_t * s = esp_camera_sensor_get();
    int val = P1.toInt();
    s->set_vflip(s, val);
  }
  else if (cmd=="serial") {
    Serial.print(P1);
  }
  else if (cmd=="restart") {
    ESP.restart();
  }
  else if (cmd=="flash") {
    ledcAttachPin(4, 4);
    ledcSetup(4, 5000, 8);
    int val = P1.toInt();
    ledcWrite(4,val);
  }
  else if(cmd=="servo") {  // (0-180)
    ledcAttachPin(P1.toInt(), 3);
    ledcSetup(3, 50, 16);

    int val = 7864-P2.toInt()*34.59;
    if (val > 7864)
       val = 7864;
    else if (val < 1638)
      val = 1638;
    ledcWrite(3, val);
  }
  else if (cmd=="relay") {
    pinMode(P1.toInt(), OUTPUT);
    digitalWrite(13, P2.toInt());
  }
  else {
    Feedback="Command is not defined.";
  }
  if (Feedback=="") Feedback=Command;
}

// web code
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;
    fetch(query);
  }

  function getStill() {
    var query = document.location.origin+"/?getstill";
    fetch(query).then(function(response) {
      return response.text();
    }).then(function(text) {
      result.innerHTML = text.split(",")[1];
      text = text.split(",")[0];
      context.clearRect(0, 0, canvas.width, canvas.height);
      var imgData=context.getImageData(0,0,canvas.width,canvas.height);
      var n = 0;
      for (var i=0;i<imgData.data.length;i+=4) {
        var val = parseInt(text.substr(2*n,2), 16);
        imgData.data[i]=val;
        imgData.data[i+1]=val;
        imgData.data[i+2]=val;
        imgData.data[i+3]=255;
        n++;
      }
      context.putImageData(imgData,0,0);
      setTimeout(function(){getStill();}, 100);
      let link = result.innerHTML;
      if(link != "" && link != "Decoding FAILED" && link != "LED RED
ON" && link != "LED RED OFF" && link != "LED GREEN ON" && link != "LED
GREEN OFF"){
        window.open(link);
      }
    })
  }
</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,sem
icolonstate=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 = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }

        if ((currentLine.indexOf("?")!=-1)&&(currentLine.indexOf("
HTTP")!=-1)) {
          if (Command.indexOf("stop")!=-1) {  // ->
http://192.168.xxx.xxx?cmd=aaa;bbb;ccc;stop
            client.println();
            client.println();
            client.stop();
          }
          currentLine="";
          Feedback="";
          ExecuteCommand();
        }
      }
    }
    delay(1);
    client.stop();
  }
}

void mainPage() {
    //Feedback
    client.println("HTTP/1.1 200 OK");
    client.println("Access-Control-Allow-Headers: Origin, X-Requested-
With, Content-Type, Accept");
    client.println("Access-Control-Allow-Methods:
GET,POST,PUT,DELETE,OPTIONS");
    client.println("Content-Type: text/html; charset=utf-8");
    client.println("Access-Control-Allow-Origin: *");
    client.println("Connection: close");
    client.println();

    String Data="";
    if (cmd!="")
      Data = Feedback;
    else {
      Data = String((const char *)INDEX_HTML);
    }
    int Index;
    for (Index = 0; Index < Data.length(); Index = Index+1024) {
      client.print(Data.substring(Index, Index+1024));
    }
}

void getStill() {
  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    return;
  }

  client.println("HTTP/1.1 200 OK");
  client.println("Access-Control-Allow-Origin: *");
  client.println("Access-Control-Allow-Headers: Origin, X-Requested-
With, Content-Type, Accept");
  client.println("Access-Control-Allow-Methods:
GET,POST,PUT,DELETE,OPTIONS");
  client.println("Content-Type: text/plain");
  client.println("Connection: close");
  client.println();

  uint8_t *fbBuf = fb->buf;
  size_t fbLen = fb->len;
  String val = "";
  String v = "";
  if (QRCodeResult == "LED RED ON") digitalWrite(LED_Red, HIGH);
  if (QRCodeResult == "LED RED OFF") digitalWrite(LED_Red, LOW);
  if (QRCodeResult == "LED GREEN ON") digitalWrite(LED_Green, HIGH);
  if (QRCodeResult == "LED GREEN OFF") digitalWrite(LED_Green, LOW);

   if (QRCodeResult == "Decoding FAILED" ){
    digitalWrite(LED_Red, HIGH);
    delay(500);
    digitalWrite(LED_Red, LOW);
  }
  if (QRCodeResult != "Decoding FAILED" && QRCodeResult != "LED RED ON"
&& QRCodeResult != "LED RED OFF" && QRCodeResult != "LED GREEN ON" &&
QRCodeResult != "LED GREEN OFF" && QRCodeResult != ""){
    digitalWrite(LED_Green, HIGH);
    delay(500);
    digitalWrite(LED_Green, LOW);
  }

  for (int n=0;n<fbLen;n++) {
    v = String(fbBuf[n], HEX);
    if (v.length()==1)
      val += "0"+v;
    else
      val += v;
    if ((n+1)%1024==0) {
      client.print(val);
      val = "";
    }
  }
  if (val!="")
    client.print(val);

  val = ","+QRCodeResult;
  client.print(val);
  QRCodeResult = "";

  esp_camera_fb_return(fb);
  pinMode(4, OUTPUT);
  digitalWrite(4, LOW);
}

void getCommand(char c) {
  if (c=='?') ReceiveState=1;
  if ((c==' ')||(c=='\r')||(c=='\n')) ReceiveState=0;

  if (ReceiveState==1)
  {
    Command=Command+String(c);

    if (c=='=') cmdState=0;
    if (c==';') strState++;

    if ((cmdState==1)&&((c!='?')||(questionstate==1)))
cmd=cmd+String(c);
    if ((cmdState==0)&&(strState==1)&&((c!='=')||(equalstate==1)))
P1=P1+String(c);
    if ((cmdState==0)&&(strState==2)&&(c!=';')) P2=P2+String(c);
    if ((cmdState==0)&&(strState==3)&&(c!=';')) P3=P3+String(c);
    if ((cmdState==0)&&(strState==4)&&(c!=';')) P4=P4+String(c);
    if ((cmdState==0)&&(strState==5)&&(c!=';')) P5=P5+String(c);
    if ((cmdState==0)&&(strState==6)&&(c!=';')) P6=P6+String(c);
    if ((cmdState==0)&&(strState==7)&&(c!=';')) P7=P7+String(c);
    if ((cmdState==0)&&(strState==8)&&(c!=';')) P8=P8+String(c);
    if ((cmdState==0)&&(strState>=9)&&((c!=';')||(semicolonstate==1)))
P9=P9+String(c);

    if (c=='?') questionstate=1;
    if (c=='=') equalstate=1;
    if ((strState>=9)&&(c==';')) semicolonstate=1;
  }
}
```

Some of our project images :-







QR codes :

Red on red off

Green on green off

Wikipedia QR

Summary:-

Our project is a QR code scanner. We used an esp-32 cam and programmed it so that when a QR scan starts, it opens the page on the mobile that is connected to it via wifi. It can also perform a specific function via the QR, such as turning lights on and off. Many applications, such as the supermarket, the absence and presence of students, employees, and we can also use it in smart homes. Here we started to upload the code to the esp-32 via the Arduino, and here we also used the Arduino because if we wanted to display the QR on a serial monitor