Project Report **Image Capturing Device**

# Table of Contents

[**Table of Contents**](#_4lqp25cx7kth) **1**

[**Introduction**](#_qlyr26gflke6) **2**

[**Working Principle:**](#_i4izftxw1cj7) **2**

[**Circuit Components:**](#_eael7qg8ayvl) **4**

[**Internal Integration:**](#_27xga6ycb9fz) **6**

[**Further Modifications:**](https://docs.google.com/document/d/1y1UFa0kcwyB-IAsKW0_lgTslaTb3gVJEAlgQX5mX2tU/edit#heading=h.bwcga9uz1lhg)[**7**](https://docs.google.com/document/d/1y1UFa0kcwyB-IAsKW0_lgTslaTb3gVJEAlgQX5mX2tU/edit#heading=h.bwcga9uz1lhg)

[**Limitations and Precautions:**](#_al4656mu6lu8) **7**

[**Applications:**](#_nb6gpvrjngsr) **8**

[**Testings and Results:**](#_5e06a7h6m8i6) **8**

[**Reference:**](#_ier9f0rkx59d) **8**

[**Appendices:**](#_qbkd08j03aae) **8**

[Appendix 1: Main Code](#_xj5y124o3bh1) 8

[Appendix 2: Library header file](#_l5mf8f878p52) 19

[Appendix 3: Library C file](#_vxewsguh6yeu) 22

# 

# Introduction

This image capturing device includes a remote optical module configured to capture an image of a particular object or place, that will be sent to the user’s email ID at regular intervals selected by the user.

It is a 12-volt-operated device that is connected to a 12-volt external adapter with a built-in battery backup feature which is used as an alternate power source. It uses a push button as a power cut-off switch to manually turn on the device and an additional LED is integrated for indication of Wifi connectivity, whether the device is connected to a network or not, searching for network and capturing images sent to particular email status.

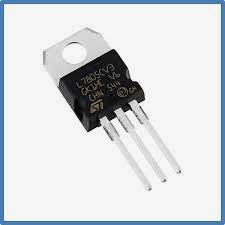
We are using ESP-32 as the main controller for the device which has a built-in Camera and Wifi module.

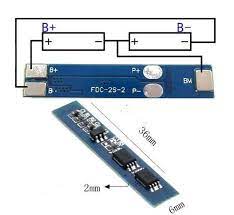
# Working Principle:

The LED will blink when the device is on by switch and the LED will turn Off when the device is connected to the WIFI. If the device is not connected to the WIFI, after 1 minute of network searching, the device automatically goes to deep sleep mode, after 30 minutes of Deep Sleep time interval, the device will restart again and search for the WIFI connection and the whole process will repeat itself.

* Switching On: When the button is pushed, the device starts searching for the WIFI, until the WIFI is not connected the LED continues to blink for a time interval of 1 minute after which the device goes into deep sleep mode if the WIFI network is not found.
* Memory Storage Check: After successful connection with the WIFI the device then checks for the availability of the storage space (which in our device is a SPIFFS Filesystem). If the device does not find any storage or storage device in it, then the device will restart again and again until it does not find any storage.
* Camera Checking: After getting a storage space, the device will then check for its camera. If the device does not find any camera, it will be restarted. This will go on until the camera is not found
* Image Capturing: After the confirmation of the availability of all of the above three things, the device will start capturing the image from wherever the face of the camera will be. Until a fully clear image is not captured, the device will keep clicking the images and delete the unclear image from the device.
* Storing Image: After a clear image is captured, the image is stored in the memory card of the device.
* Transferring Image: From the device, an automatic email is generated to the user with the help of an online SMTP server (which transfers the image from the controller to the user). Firstly, the server checks the credentials and generates an email with the subject ESP32-CAM-D1 (here D1 is for device-1 ) the connection to the server link is found in reference.
* Indication: When the image is sent to the user with a mail, the LED is blinked twice to indicate that the image is sent successfully and when the image is not sent due to any kind of error so there will be no indication from the LED. After all this, the image is deleted from the device memory so that it can store further images.
* Deep Sleep Mode: When all these things are done the device goes to deep sleep mode for 30 minutes and again performs all the above tasks.

# Circuit Components:

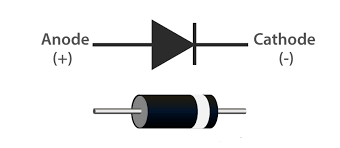
1. **Regulator IC:** Here for regulation, we use LM7805 for 5-volt constant power to the controller
2. **Heat Sink:** A heat sink is used in this device to prevent regulating IC from any heating issue.
3. **Lithium Cells**: 2 Lithium cells of 2000mAh are used for backup power supply.
4. **2S-BMS:** Abattery management system is used for 2 series lithium cells for short circuit protection and overcharges protection.
5. **DC-DC Buck converter**: Here LM2596 is used to convert the 12V power supply to 8V.
6. **ESP32-CAM:** ESP-32 is used as the main controller with the integrated camera feature.
7. **Diode:** It is used to verify the one-way flow of current.
8. LM7805 IC (2) Heat Sink 

(3) Lithium Cells (4) 2S BMS

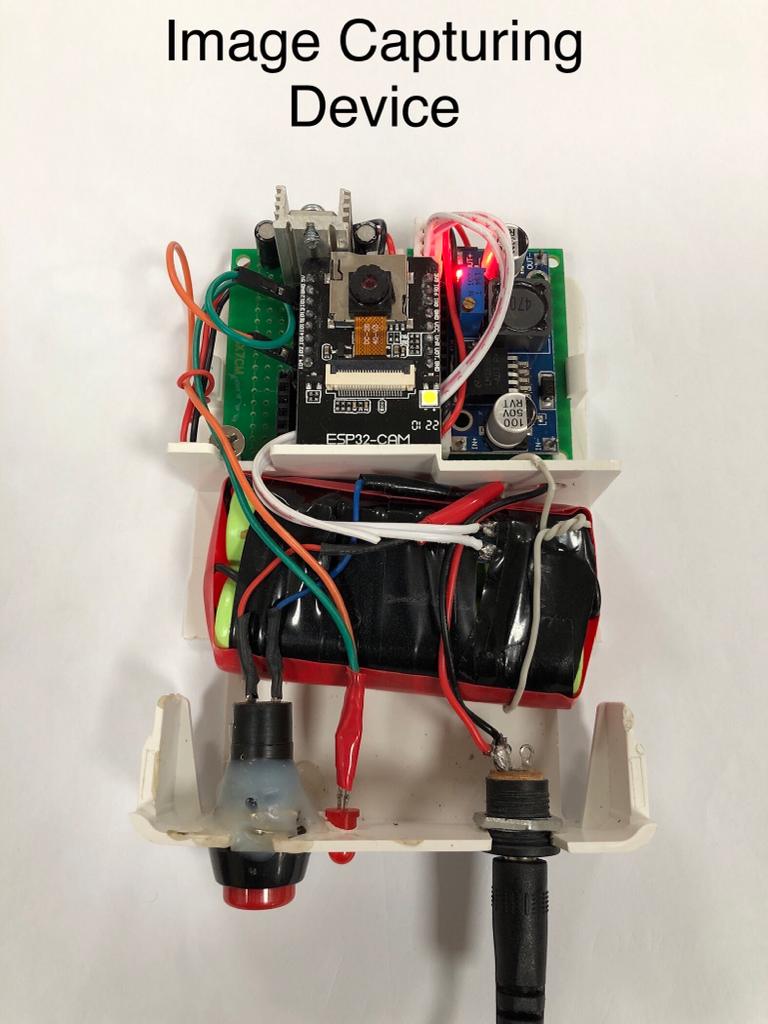


(5) DC-DC Buck Converter (6)ESP32-CAM

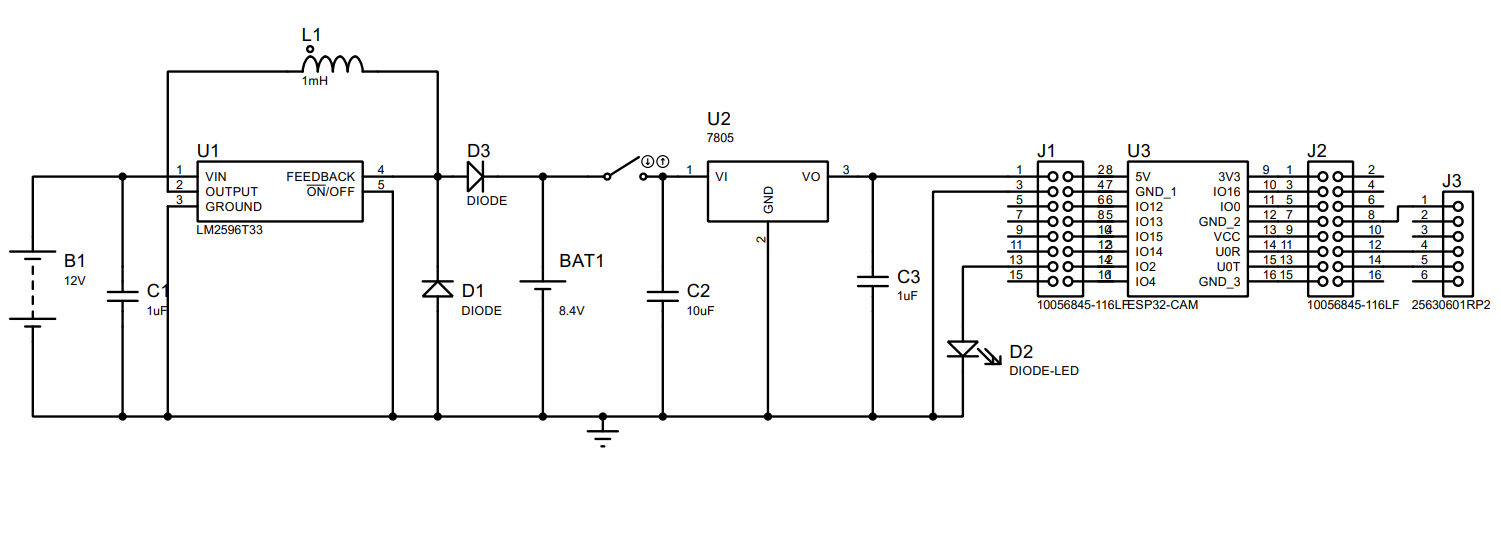
(7) Diode



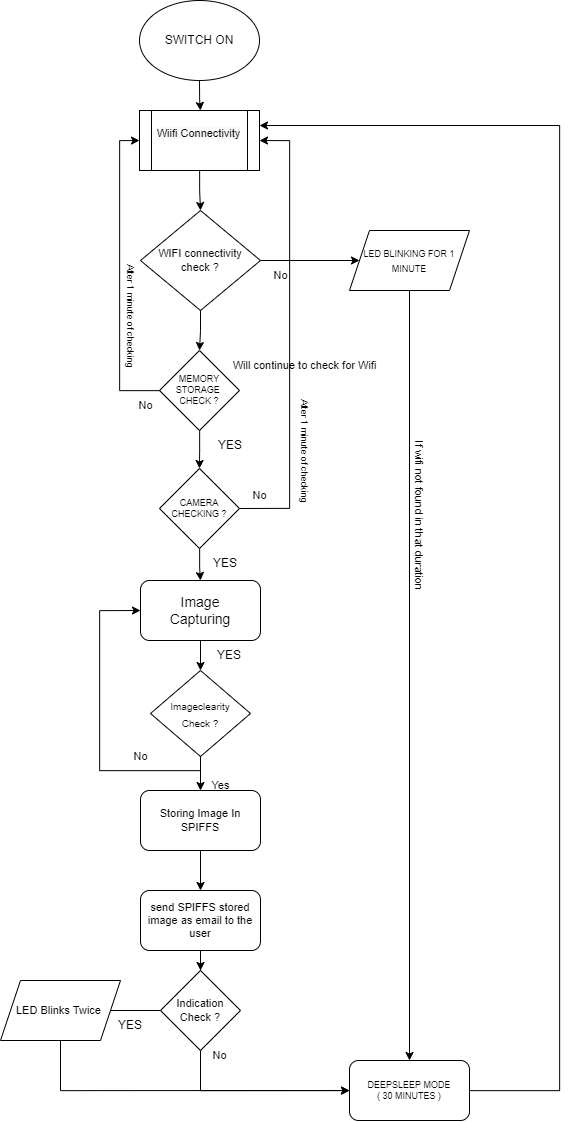
# Internal Integration:



# Circuit Diagram:



# Block Diagram:



# Further Modifications:

* Remote Configuration: configure the sent email address and time interval of image capturing.
* EEPROM: We can use EEPROM which is a built-in controller that works as a non-volatile memory we can use it for default and current configuration purposes
* Charging LED indicators: We can show the charging of the batteries with the help of different LEDs or LCDs, which will inform us about the percentage of the charging of the batteries.
* Flash Light: We can also add Flash Light for the night which can be controlled by any I/O pin.
* With a little modification, this can be used as face recognizer
* Can integrate gsm module to send data through wifi or gsm

# Limitations and Precautions:

* Requires Wifi with stable internet connectivity.
* Requires memory card and RAM.
* Requires camera.
* Require 12 volts power supply.
* Should be prevented from water.
* Can not work efficiently on higher temperatures (greater than 55°C)
* Should be placed in a specific position with correct angle

# Applications:

* Agriculture Purpose
* Meter Reading
* Remote Attendance

# Testings and Results:

* Power Cut Off Switch
* LED indicating
* Charging of battery through BMS (Battery Management system)
* Regulation of Regulator IC
* Power Delivery of Bug Convertor
* At complete battery drainage voltage cut-off
* Working under hot temperature
* Battery backup up to one day ( when one cell is weak )
* Charging and Operation of device at a particular instant
* Normal operations of ESP-32 controller takes 0.2 mA

# Reference:

* <https://randomnerdtutorials.com/program-upload-code-esp32-cam/>
* <https://randomnerdtutorials.com/esp32-cam-send-photos-email/>
* <https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/>
* <https://app.diagrams.net/#G1eBnD3pUr8RUZQPrGkUhp_P8xqFHlZiyt>

# Appendices:

## Appendix 1: Main Code

| /\*  \* main.ino  \*  \* Created on: 06/09/2022  \* Author: Muhammad Danish  \*/  #include "esp\_camera.h" #include "SPI.h" #include "driver/rtc\_io.h" #include "ESP32\_MailClient.h" #include <FS.h> #include <SPIFFS.h> #include <WiFi.h>   //------------------WIFI--------------- // REPLACE WITH YOUR NETWORK CREDENTIALS //const char\* ssid = "RCAI"; //const char\* password = "RCAIned@123"; const char\* ssid = "Extensity"; const char\* password = "password1"; #define NWT\_TIMEOUT 1\*60\*1000 //trying for 1 min   //------------------------------------     //--------------------------ESP-CAM----------- //sending picture to gmail at particular time interval //unsigned long tm\_now = -5 \* 60 \* 1000; //unsigned long Alert\_tm = 5 \* 60 \* 1000; //1 min \* 60 second \* 1000 ms = 1 min // ledPin refers to ESP32-CAM GPIO 4 (flashlight) #define Network\_led 2 #define FLASH\_GPIO\_NUM 4  // To send Emails using Gmail on port 465 (SSL), you need to create an app password: https://support.google.com/accounts/answer/185833 #define emailSenderAccount "engrmuhammaddanish001@gmail.com" #define emailSenderPassword "ntcxyfmngcwwoxay" #define smtpServer "smtp.gmail.com" #define smtpServerPort 465 #define emailSubject "ESP32-CAM Photo Captured" #define emailRecipient "smartdanish96@gmail.com" //#define emailRecipient "engrmuhammaddanish001@gmail.com" #define CAMERA\_MODEL\_AI\_THINKER  #if defined(CAMERA\_MODEL\_AI\_THINKER) #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 #else #error "Camera model not selected" #endif  bool run\_mode = false; //use for capture image or not  // The Email Sending data object contains config and data to send SMTPData smtpData;  // Photo File Name to save in SPIFFS #define FILE\_PHOTO "/photo.jpg" //---------------------------------------------------------   //---------------------------------DEEP-SLEEP------------------ //deep sleep variable #define uS\_TO\_S\_FACTOR 1000000 /\* Conversion factor for micro seconds to seconds \*/ #define TIME\_TO\_SLEEP 30\*60 /\* set to 30 min: Time ESP32 will go to sleep (in seconds) \*/  RTC\_DATA\_ATTR int bootCount = 0;  /\*  Method to print the reason by which ESP32  has been awaken from sleep \*/ void print\_wakeup\_reason() {  esp\_sleep\_wakeup\_cause\_t wakeup\_reason;   wakeup\_reason = esp\_sleep\_get\_wakeup\_cause();   switch (wakeup\_reason)  {  case ESP\_SLEEP\_WAKEUP\_EXT0 : Serial.println("Wakeup caused by external signal using RTC\_IO"); break;  case ESP\_SLEEP\_WAKEUP\_EXT1 : Serial.println("Wakeup caused by external signal using RTC\_CNTL"); break;  case ESP\_SLEEP\_WAKEUP\_TIMER : Serial.println("Wakeup caused by timer"); break;  case ESP\_SLEEP\_WAKEUP\_TOUCHPAD : Serial.println("Wakeup caused by touchpad"); break;  case ESP\_SLEEP\_WAKEUP\_ULP : Serial.println("Wakeup caused by ULP program"); break;  default : Serial.printf("Wakeup was not caused by deep sleep: %d\n", wakeup\_reason); break;  } }  //----------------------------------------------------------------------  void setup() {  WRITE\_PERI\_REG(RTC\_CNTL\_BROWN\_OUT\_REG, 0); //disable brownout detector   // initialize digital pin ledPin as an output  // pinMode(FLASH\_GPIO\_NUM, PULL\_UP);  pinMode(Network\_led,OUTPUT);    Serial.begin(115200);  Serial.println();   Serial.print(millis());  Serial.println("ms: start Time");  //------------DEEP SLEEP MODE----------   //Increment boot number and print it every reboot  ++bootCount;  Serial.println("Boot number: " + String(bootCount));   //Print the wakeup reason for ESP32  print\_wakeup\_reason();   /\*  First we configure the wake up source  We set our ESP32 to wake up every 5 seconds  \*/  esp\_sleep\_enable\_timer\_wakeup(TIME\_TO\_SLEEP \* uS\_TO\_S\_FACTOR);  Serial.println("Setup ESP32 to sleep for every " + String(TIME\_TO\_SLEEP) +  " Seconds");   //------------DEEP SLEEP MODE END------------------   //--------------------WIFI-CONNECTIVITY-----------  //Connect to Wi-Fi  WiFi.begin(ssid, password);  Serial.print("Connecting to WiFi...");  while (WiFi.status() != WL\_CONNECTED && millis() <= NWT\_TIMEOUT) { //if wifi not found then trying for 1 min  digitalWrite(Network\_led,!digitalRead(Network\_led));  Serial.print(".");  delay(500);  }    if (WiFi.status() != WL\_CONNECTED) {  Serial.println();  Serial.println("WIFI NOT FOUND");  digitalWrite(Network\_led,HIGH);  run\_mode = LOW;  } else {   // Print ESP32 Local IP Address  Serial.print("IP Address: http://");  Serial.println(WiFi.localIP());  digitalWrite(Network\_led,LOW);  run\_mode = HIGH;  }  Serial.println();   //--------------------WIFI-CONNECTIVITY-END-----------   if(run\_mode){ //IF RUN MODE IS ACTIVATE THEN CAPTURE THE IMAGE   //-------------------ESP-CAM-MODE------------  if (!SPIFFS.begin(true)) {  Serial.println("An Error has occurred while mounting SPIFFS");  ESP.restart();  }  else {  delay(500);  Serial.println("SPIFFS mounted successfully");  }     camera\_config\_t config;  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 = 20000000;  config.pixel\_format = PIXFORMAT\_JPEG;   if (psramFound()) {  // config.frame\_size = FRAMESIZE\_UXGA;  // config.jpeg\_quality = 10;  // config.fb\_count = 2;  config.frame\_size = FRAMESIZE\_SVGA;  config.jpeg\_quality = 10;  config.fb\_count = 1;  } else {  config.frame\_size = FRAMESIZE\_SVGA;  config.jpeg\_quality = 12;  config.fb\_count = 1;  }   // Initialize camera  esp\_err\_t err = esp\_camera\_init(&config);  if (err != ESP\_OK) {  Serial.printf("Camera init failed with error 0x%x", err);  return;  }  //-------------------ESP-CAM-MODE-END------------    //------------CAPTURED IMAGE SEND TO SERVER----------  delay(200);   // digitalWrite(FLASH\_GPIO\_NUM, HIGH);  capturePhotoSaveSpiffs();  // digitalWrite(FLASH\_GPIO\_NUM, LOW);  sendPhoto();   //------------CAPTURED IMAGE SERVER END-----------  }    //------------DEEP SLEEP MODE----------    /\*  Now that we have setup a wake cause and if needed setup the  peripherals state in deep sleep, we can now start going to  deep sleep.  In the case that no wake up sources were provided but deep  sleep was started, it will sleep forever unless hardware  reset occurs.  \*/  Serial.println("Going to sleep now");  digitalWrite(Network\_led,LOW);  delay(200);  Serial.flush();  // digitalWrite(FLASH\_GPIO\_NUM, LOW);  esp\_deep\_sleep\_start();  Serial.println("This will never be printed");   //------------DEEP SLEEP MODE END------------------    }  void loop() {  }  // Check if photo capture was successful bool checkPhoto( fs::FS &fs ) {  File f\_pic = fs.open( FILE\_PHOTO );  unsigned int pic\_sz = f\_pic.size();  return ( pic\_sz > 100 ); }  // Capture Photo and Save it to SPIFFS void capturePhotoSaveSpiffs( void ) {  camera\_fb\_t \* fb = NULL; // pointer  bool ok = 0; // Boolean indicating if the picture has been taken correctly   do {  // Take a photo with the camera  Serial.println("Taking a photo...");   fb = esp\_camera\_fb\_get();  if (!fb) {  Serial.println("Camera capture failed");  return;  }   // Photo file name  Serial.printf("Picture file name: %s\n", FILE\_PHOTO);  File file = SPIFFS.open(FILE\_PHOTO, FILE\_WRITE);   // Insert the data in the photo file  if (!file) {  Serial.println("Failed to open file in writing mode");  }  else {  file.write(fb->buf, fb->len); // payload (image), payload length  Serial.print("The picture has been saved in ");  Serial.print(FILE\_PHOTO);  Serial.print(" - Size: ");  Serial.print(file.size());  Serial.println(" bytes");  }  // Close the file  file.close();  esp\_camera\_fb\_return(fb);   // check if file has been correctly saved in SPIFFS  ok = checkPhoto(SPIFFS);  } while ( !ok ); }  void sendPhoto( void ) {  // Preparing email  Serial.println("Sending email...");  // Set the SMTP Server Email host, port, account and password  smtpData.setLogin(smtpServer, smtpServerPort, emailSenderAccount, emailSenderPassword);   // Set the sender name and Email  smtpData.setSender("ESP32-CAM-D1", emailSenderAccount);   // Set Email priority or importance High, Normal, Low or 1 to 5 (1 is highest)  smtpData.setPriority("High");   // Set the subject  smtpData.setSubject(emailSubject);   // Set the email message in HTML format  smtpData.setMessage("<h2>Photo captured with ESP32-CAM-DEVICE-1 and attached in this email.</h2>", true);  // Set the email message in text format  //smtpData.setMessage("Photo captured with ESP32-CAM and attached in this email.", false);   // Add recipients, can add more than one recipient  smtpData.addRecipient(emailRecipient);  //smtpData.addRecipient(emailRecipient2);   // Add attach files from SPIFFS  smtpData.addAttachFile(FILE\_PHOTO, "image/jpg");  // Set the storage type to attach files in your email (SPIFFS)  smtpData.setFileStorageType(MailClientStorageType::SPIFFS);   smtpData.setSendCallback(sendCallback);   // Start sending Email, can be set callback function to track the status  if (!MailClient.sendMail(smtpData)){  Serial.println("Error sending Email, " + MailClient.smtpErrorReason());  }else{  for(int i=0; i<2 ; i++){  digitalWrite(Network\_led,HIGH);  delay(250);  digitalWrite(Network\_led,LOW);  delay(250);  }  }  // Clear all data from Email object to free memory  smtpData.empty();   //delete created file  if(SPIFFS.remove(FILE\_PHOTO)){  Serial.println("- file deleted");  } else {  Serial.println("- delete failed");  }  }  // Callback function to get the Email sending status void sendCallback(SendStatus msg) {  //Print the current status  Serial.println(msg.info()); } |
| --- |

## 

## 

## 

## Appendix 2: Library header file

| /\*  \* Copyright (c) 2013 Adam Rudd.  \* See LICENSE for more information  \* https://github.com/adamvr/arduino-base64   \*/ #ifndef \_BASE64\_H #define \_BASE64\_H  /\* b64\_alphabet:  \* Description: Base64 alphabet table, a mapping between integers  \* and base64 digits  \* Notes: This is an extern here but is defined in Base64.c  \*/ extern const char b64\_alphabet[];  /\* base64\_encode:  \* Description:  \* Encode a string of characters as base64  \* Parameters:  \* output: the output buffer for the encoding, stores the encoded string  \* input: the input buffer for the encoding, stores the binary to be encoded  \* inputLen: the length of the input buffer, in bytes  \* Return value:  \* Returns the length of the encoded string  \* Requirements:  \* 1. output must not be null or empty  \* 2. input must not be null  \* 3. inputLen must be greater than or equal to 0  \*/ int base64\_encode(char \*output, char \*input, int inputLen);  /\* base64\_decode:  \* Description:  \* Decode a base64 encoded string into bytes  \* Parameters:  \* output: the output buffer for the decoding,  \* stores the decoded binary  \* input: the input buffer for the decoding,  \* stores the base64 string to be decoded  \* inputLen: the length of the input buffer, in bytes  \* Return value:  \* Returns the length of the decoded string  \* Requirements:  \* 1. output must not be null or empty  \* 2. input must not be null  \* 3. inputLen must be greater than or equal to 0  \*/ int base64\_decode(char \*output, char \*input, int inputLen);  /\* base64\_enc\_len:  \* Description:  \* Returns the length of a base64 encoded string whose decoded  \* form is inputLen bytes long  \* Parameters:  \* inputLen: the length of the decoded string  \* Return value:  \* The length of a base64 encoded string whose decoded form  \* is inputLen bytes long  \* Requirements:  \* None  \*/ int base64\_enc\_len(int inputLen);  /\* base64\_dec\_len:  \* Description:  \* Returns the length of the decoded form of a  \* base64 encoded string  \* Parameters:  \* input: the base64 encoded string to be measured  \* inputLen: the length of the base64 encoded string  \* Return value:  \* Returns the length of the decoded form of a  \* base64 encoded string  \* Requirements:  \* 1. input must not be null  \* 2. input must be greater than or equal to zero  \*/ int base64\_dec\_len(char \*input, int inputLen);  #endif // \_BASE64\_H |
| --- |

## Appendix 3: Library C file

| /\*  \* Copyright (c) 2013 Adam Rudd.  \* See LICENSE for more information  \* https://github.com/adamvr/arduino-base64   \*/ #if (defined(\_\_AVR\_\_)) #include <avr\pgmspace.h> #else #include <pgmspace.h> #endif  const char PROGMEM b64\_alphabet[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  "abcdefghijklmnopqrstuvwxyz"  "0123456789+/";  /\* 'Private' declarations \*/ inline void a3\_to\_a4(unsigned char \* a4, unsigned char \* a3); inline void a4\_to\_a3(unsigned char \* a3, unsigned char \* a4); inline unsigned char b64\_lookup(char c);  int base64\_encode(char \*output, char \*input, int inputLen) {  int i = 0, j = 0;  int encLen = 0;  unsigned char a3[3];  unsigned char a4[4];   while(inputLen--) {  a3[i++] = \*(input++);  if(i == 3) {  a3\_to\_a4(a4, a3);   for(i = 0; i < 4; i++) {  output[encLen++] = pgm\_read\_byte(&b64\_alphabet[a4[i]]);  }   i = 0;  }  }   if(i) {  for(j = i; j < 3; j++) {  a3[j] = '\0';  }   a3\_to\_a4(a4, a3);   for(j = 0; j < i + 1; j++) {  output[encLen++] = pgm\_read\_byte(&b64\_alphabet[a4[j]]);  }   while((i++ < 3)) {  output[encLen++] = '=';  }  }  output[encLen] = '\0';  return encLen; }  int base64\_decode(char \* output, char \* input, int inputLen) {  int i = 0, j = 0;  int decLen = 0;  unsigned char a3[3];  unsigned char a4[4];    while (inputLen--) {  if(\*input == '=') {  break;  }   a4[i++] = \*(input++);  if (i == 4) {  for (i = 0; i <4; i++) {  a4[i] = b64\_lookup(a4[i]);  }   a4\_to\_a3(a3,a4);   for (i = 0; i < 3; i++) {  output[decLen++] = a3[i];  }  i = 0;  }  }   if (i) {  for (j = i; j < 4; j++) {  a4[j] = '\0';  }   for (j = 0; j <4; j++) {  a4[j] = b64\_lookup(a4[j]);  }   a4\_to\_a3(a3,a4);   for (j = 0; j < i - 1; j++) {  output[decLen++] = a3[j];  }  }  output[decLen] = '\0';  return decLen; }  int base64\_enc\_len(int plainLen) {  int n = plainLen;  return (n + 2 - ((n + 2) % 3)) / 3 \* 4; }  int base64\_dec\_len(char \* input, int inputLen) {  int i = 0;  int numEq = 0;  for(i = inputLen - 1; input[i] == '='; i--) {  numEq++;  }   return ((6 \* inputLen) / 8) - numEq; }  inline void a3\_to\_a4(unsigned char \* a4, unsigned char \* a3) {  a4[0] = (a3[0] & 0xfc) >> 2;  a4[1] = ((a3[0] & 0x03) << 4) + ((a3[1] & 0xf0) >> 4);  a4[2] = ((a3[1] & 0x0f) << 2) + ((a3[2] & 0xc0) >> 6);  a4[3] = (a3[2] & 0x3f); }  inline void a4\_to\_a3(unsigned char \* a3, unsigned char \* a4) {  a3[0] = (a4[0] << 2) + ((a4[1] & 0x30) >> 4);  a3[1] = ((a4[1] & 0xf) << 4) + ((a4[2] & 0x3c) >> 2);  a3[2] = ((a4[2] & 0x3) << 6) + a4[3]; }  inline unsigned char b64\_lookup(char c) {  if(c >='A' && c <='Z') return c - 'A';  if(c >='a' && c <='z') return c - 71;  if(c >='0' && c <='9') return c + 4;  if(c == '+') return 62;  if(c == '/') return 63;  return -1; } |
| --- |