SMART BIRD HOUSE

By Pablo Castillo and Fátima Mendoza

1. Introduction

As many people know birds are as important to the environment as to the humans, for example they help with cross-pollination which help with the production of healthy seeds that humans can use in agriculture, which at the same time help to the seed propagation that the environment needs to grow more nature. Therefor for this and more reason they should have appropriate places to nest.

Because of deforestation many birds don’t have safe places to nest so as the causatives persons the least we can do is give them a safe place to live and eat. Bird houses provide secure spaces where birds can nestle meanwhile people can have a new hobby that benefits the wildlife. There are many benefits of having a bird house, they improve your mental health, you get to have a new wild pet as you see them in your garden, and you can see the eggs hatch. And with the computer vision you get to learn about them and have more data.

For the smart bird house with computer vision people can watch when a bird come and know the conditions of the house at that specific moment. When a bird come the camera will take a picture and upload it to Google Drive, the name of the file with be the date and time in which have been taken. It will also send a email with the temperature and humidity at the time the picture was taken.

2. Methods and Materials

2.1. How to build a bird house

It’s important to build the bird house according to the specific species you are looking for. Steps for building a bird house:

- Choose your bird.
- Investigate the need for your specific bird so it fulfils its needs.
- Select the proper materials.
- Build the house so it won’t need much maintenance, it needs to stay dry and warm.
- Provide ventilation.
- Make it safe from predators and do not add perches.
- Locate your bird house.

A Guide to Building and Placing Birdhouses

9 Best Wood for Birdhouse

The best location for a bird house depends on the needs of the bird species. Although there are some recommendations such as been away from trees and predators by being mounted on a pole that is 5 to 30 feet off the ground. Also is suggested to be camouflaged so predators don’t find it easy and attacked it.

For setting a bird house there are some recommendations to consider, for example:

- Accessible for people to watch it. It’s true that it needs to be camouflaged for predators but is also important to bird watchers to observe it and maintain, to keep it clean when is emptied and of course to see the birds.
- Nesting materials. It is recommended to have the bird house furnished with some nesting materials so birds can be comfortable, for example with pine needles, dead grass, string and thread to mention some of them.
- Bird houses should be spaced apart. Some birds are territorial so the houses should be spaced by at least 25 feet apart, so the birds are comfortable in their houses.
- Shading from the sun is not required. Keep in mind the color of the house and the material so it won’t overheat in the summer. It’s not necessary that the house is place in a shadow place, but it can provide some protection from the afternoon sun.

The Best Location For A Bird House

2.2. Hardware

2.2.1. ESP32-CAM

DEBO CAM ESP32

Product Specifications

The ESP32-CAM has a very competitive small-size camera module that can operate independently as a minimum system with a footprint of only 27*40.5*4.5mm and a deep sleep current of up to 6mA.

ESP-32CAM can be widely used in various loT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals and other loT applications. It is an ideal solution for loT applications.

ESP-32CAM adopts DIP package and can be directly inserted into the backplane to realize rapid production of products, providing customers with high-reliability connection mode, which is convenient for application in various loT hardware terminals.

2.2.2. Mini PIR Motion Sensor

Details mini PIR motion sensor

The Grove – mini PIR motion sensor is a compact, low power consumption, and cost-effective PIR sensor which is suitable for applications with relatively less detection distance requirements.

Features

- Adjustable sensitivity: a reserved pin out on board to solder a slide rheostat to adjust the sensitivity
- Easy to use: Grove compatible interface and supports both Arduino and Raspberry Pi platforms
- Mini size:20mm x20mm x12mm

2.2.3. Temperature and Humidity Sensor DHT22

DHT22 - Digital Temperature and Humidity Sensor

DHT22 Datasheet (PDF) - List of Unclassifed Manufacturers

The DHT22 is a basic, low-cost digital temperature and humidity sensor. Due to its low cost and ease of use, you'll find project examples all over the web for these simple sensors.

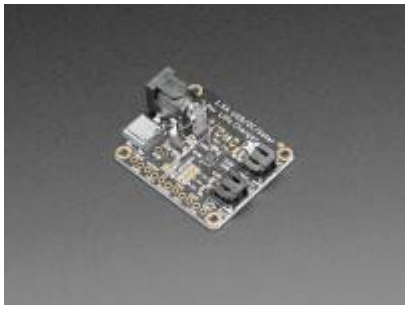
The DHT22 uses a capacitive humidity sensor and thermistor to measure the surrounding air, then provides that data via a digital output signal on the data pin. They're easy to use, however, they require some careful timing to grab data. Body size 27mm x 59mm x 13.5mm

2.2.4. UartSBee V5

Details UartSBee V5

UartSBee v5' is FTDI cable compatible USB to Serial adapter equipped with BEE socket(20pin 2.0mm). The integrated FT232RL can be used for programming or communicating with MCUs. On the other hand, you might connect your PC to various wireless applications via a Bee compatible module. UartSBee provides breakouts for the bit-bang mode pins of FT232RL as well.This Bit-bang mode pins (8 I/O pins) can be used as a replacement for applications involving PC parallel port which is scares now a day.

2.2.5. Universal USB/DC/Solar Lithium Ion/Polymer charger

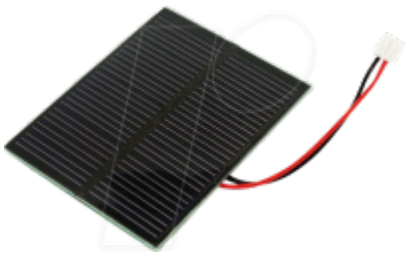


Description Universal USB/DC/Solar Lithium Ion/Polymer charger

Adafruit Universal USB / DC / Solar Lithium Ion/Polymer charger

This charger is the only one you need to keep all your Lithium Polymer (LiPoly) or Lithium Ion (LiIon) rechargeable batteries topped up. No matter the power source at your disposal! The Adafruit Universal USB / DC / Solar Lithium Ion/Polymer Charger can use USB, DC or Solar power, with a wide 5-10V input voltage range! The charger chip is super smart, and will reduce the current draw if the input voltage starts to dip under 4.5V, making it a perfect near-MPPT solar charger that you can use with a wide range of panels.

2.2.6. Solar panel

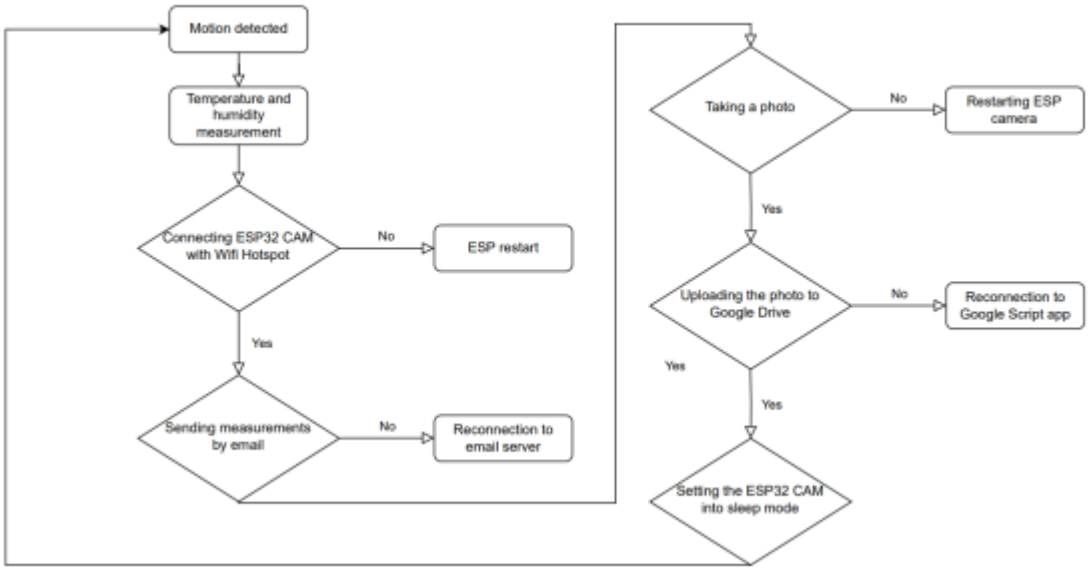


Solar panel technology

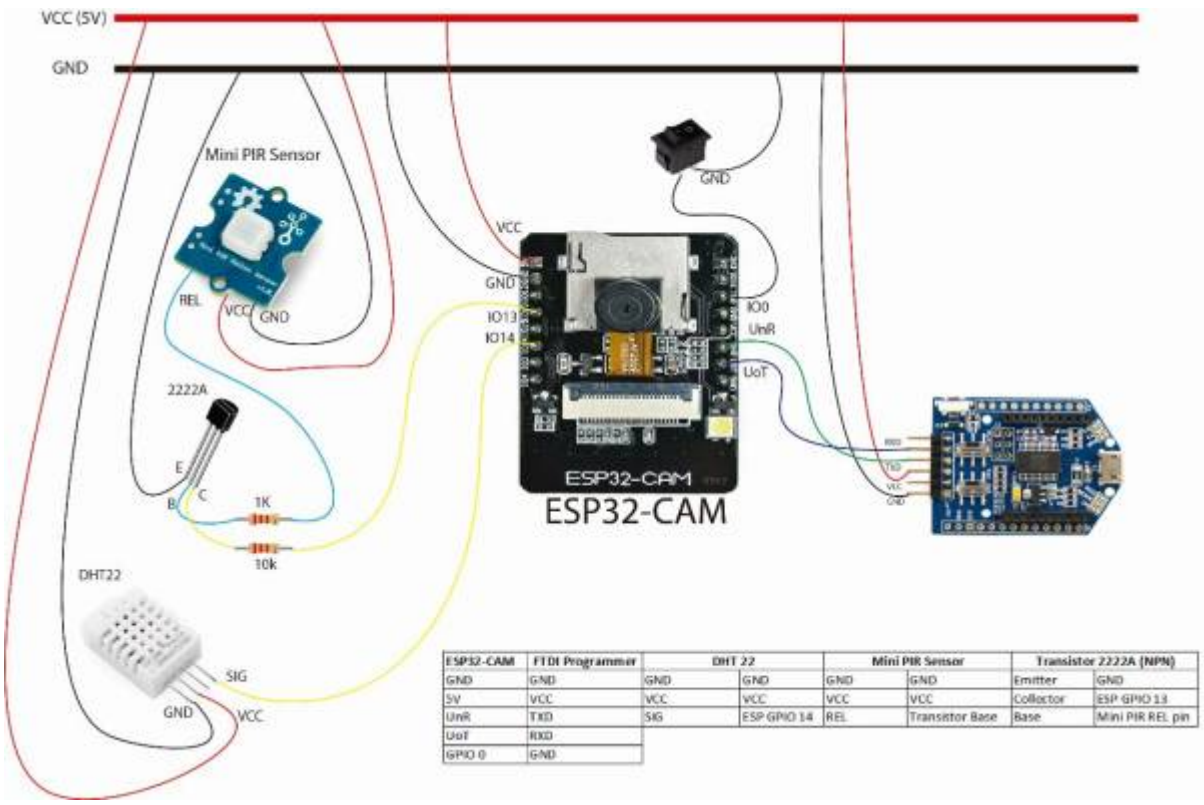
A solar pannel is a component of a photovoltaic system that is made out of a series of photovoltaic cells arranged to generate electricity using sunlight. Size 13 cm x 15 cm.

2.3. Software

2.3.1.Flow chart of Smart Bird House

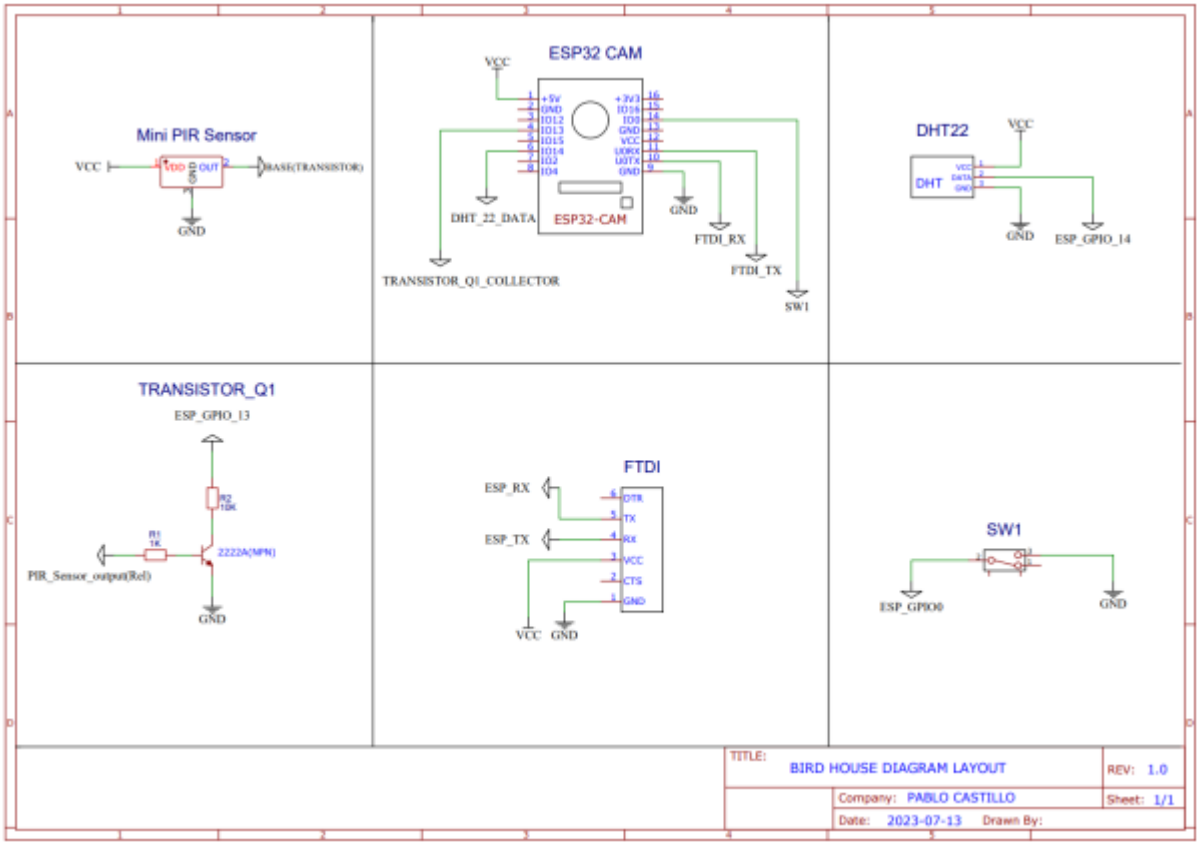


2.3.2.Hardware Set Up



ESP32-CAM	FTDI Programmer	DHT 22		Mini PIR Sensor		Transistor 2222A (NPN)	
GND	GND	GND	GND	GND	GND	Emitter	GND
5V	VCC	VCC	VCC	VCC	VCC	Collector	ESP GPIO 13
UnR	TXD	SIG	ESP GPIO 14	REL	Transistor Base	Base	Mini PIR REL pin
UoT	RXD						
GPIO 0	GND						

2.3.3.Electric Diagram of Smart Bird House



2.3.4.Raw Code

========== *Including the libraries. #include <WiFi.h> #include <WiFiClientSecure.h> #include “soc/soc.h” #include “soc/rtc_cntl_reg.h” #include “Base64.h” #include “esp_camera.h” #include <string.h> #include “Arduino.h” #include “FS.h”* SD Card ESP32 #include “SD_MMC.h” *SD Card ESP32 #include “driver/rtc_io.h” #include “ESP_Mail_Client.h” #include “DHT.h”*

int IRpin = 13; int state = 0; ========== *Defining Temp. and Hum. Sensor #define DHTTYPE DHT22* DHT 22 (AM2302), AM2321 *Initialize DHT sensor pin. #define DHTPIN 14 DHT dht(DHTPIN, DHTTYPE);*
========== CAMERA_MODEL_AI_THINKER GPIO. #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
========== LED Flash PIN (GPIO 4) #define FLASH_LED_PIN 4

========== *Enter your WiFi ssid and password. const char* ssid = “FRITZ!Box Gastzugang”; const char* password = “VillaKunterbunt”;*

========== *Replace with your “Deployment ID” and Folder Name. String myDeploymentID = “AKfycbwxhr3qbdDC105rDPUUwsJXcITatlL1rqdQh1dKv385-uil9Z8OXrNOQ5wXUR3f5F8R”; String myMainFolderName = “ESP32-CAM”;*

========== *Mail setup / **The smtp host name e.g. smtp.gmail.com for GMail*/ #define SMTP_HOST “smtp.gmail.com” #define SMTP_PORT 465 /* The sign in credentials */ #define AUTHOR_EMAIL “pablocastillopaypalpersonal@gmail.com” #define AUTHOR_PASSWORD “wfbqnnacjufaiafo” /* Recipient's email*/ #define RECIPIENT_EMAIL “pablocastillo417@gmail.com” /* Declare the global used SMTPSession object for SMTP transport */ SMTPSession smtp; /* Callback function to get the Email sending status */ void smtpCallback(SMTP_Status status);***
========== *Variables for Timer/Millis. unsigned long previousMillis = 0; const int Interval = 20000; -> Capture and Send a photo every 20 seconds. ========== Variable to set capture photo with LED Flash. Set to “false”, then the Flash LED will not light up when capturing a photo. Set to “true”, then the Flash LED lights up when capturing a photo. bool LED_Flash_ON = false; Initialize WiFiClientSecure. WiFiClientSecure client; Test_Con() This subroutine is to test the connection to “script.google.com”. void Test_Con() { const char* host = “script.google.com”; while (1) { Serial.println(“———-”); Serial.println(“Connection Test…”); Serial.println(“Connect to ” + String(host)); client.setInsecure(); if (client.connect(host, 443)) { Serial.println(“Connection successful.”); Serial.println(“———-”); client.stop(); break; } else { Serial.println(“Connected to ” + String(host) + “ failed.”); Serial.println(“Wait a moment*

for reconnecting."); Serial.println("———–"); client.stop(); } delay(1000); } } SendCapturedPhotos() Subroutine for capturing and sending photos to Google Drive. void SendCapturedPhotos() { const char host = "script.google.com"; Serial.println(); Serial.println("———–"); Serial.println("Connect to " + String(host)); client.setInsecure(); ————————————- The process of connecting, capturing and sending photos to Google Drive. if (client.connect(host, 443)) { Serial.println("Connection successful."); if (LED_Flash_ON == true) { digitalWrite(FLASH_LED_PIN, HIGH); delay(100); } else { digitalWrite(FLASH_LED_PIN, LOW); } …… Taking a photo. Serial.println(); Serial.println("Taking a photo…"); for (int i = 0; i <= 3; i++) { camera_fb_t* fb = NULL; fb = esp_camera_fb_get(); if (!fb) { Serial.println("Camera capture failed"); Serial.println("Restarting the ESP32 CAM."); delay(1000); ESP.restart(); return; } esp_camera_fb_return(fb); delay(200); } camera_fb_t* fb = NULL; fb = esp_camera_fb_get(); if (!fb) { Serial.println("Camera capture failed"); Serial.println("Restarting the ESP32 CAM."); delay(1000); ESP.restart(); return; } if (LED_Flash_ON == true) digitalWrite(FLASH_LED_PIN, LOW); Serial.println("Taking a photo was successful."); …… …… Sending image to Google Drive. Serial.println(); Serial.println("Sending image to Google Drive."); Serial.println("Size: " + String(fb→len) + "byte"); String url = "/macros/s/" + myDeploymentID + "/exec?folder=" + myMainFolderName; client.println("POST " + url + " HTTP/1.1"); client.println("Host: " + String(host)); client.println("Transfer-Encoding: chunked"); client.println(); int fbLen = fb→len; char* input = (char*)fb→buf; int chunkSize = 3 * 1000; –> must be multiple of 3. int chunkBase64Size = base64_enc_len(chunkSize); char output[chunkBase64Size + 1]; Serial.println(); int chunk = 0; for (int i = 0; i < fbLen; i += chunkSize) { int l = base64_encode(output, input, min(fbLen - i, chunkSize)); client.print(l, HEX); client.print("\r\n"); client.print(output); client.print("\r\n"); delay(100); input += chunkSize; Serial.print("."); chunk++; if (chunk % 50 == 0) { Serial.println(); } } client.print("0\r\n"); client.print("\r\n"); esp_camera_fb_return(fb); …… …… Waiting for response. Serial.println("Waiting for response."); long int StartTime = millis(); while (!client.available()) { Serial.print("."); delay(100); if* [1] *{ Serial.println(); Serial.println("No response."); break; } } Serial.println(); while (client.available()) { Serial.print(char(client.read())); } …… } else { Serial.println("Connected to " + String(host) + " failed."); } ————————————- Serial.println("———–"); client.stop(); }*
========== Callback function to get the Email sending status void smtpCallback(SMTP_Status status) { / Print the current status */ Serial.println(status.info()); /* Print the sending result */ if (status.success()) { Serial.println("———-"); ESP_MAIL_PRINTF("Message sent success: %d\n", status.completedCount()); ESP_MAIL_PRINTF("Message sent failed: %d\n", status.failedCount()); Serial.println("———-\n"); for (size_t i = 0; i < smtp.sendingResult.size(); i++) { /* Get the result item */ SMTP_Result result = smtp.sendingResult.getItem(i); ESP_MAIL_PRINTF("Message No: %d\n", i + 1); ESP_MAIL_PRINTF("Status: %s\n", result.completed ? "success" : "failed"); ESP_MAIL_PRINTF("Date/Time: %s\n", MailClient.Time.getDateTimeString(result.timestamp, "%B %d, %Y %H:%M:%S").c_str()); ESP_MAIL_PRINTF("Recipient: %s\n", result.recipients.c_str()); ESP_MAIL_PRINTF("Subject: %s\n", result.subject.c_str()); } Serial.println("———-\n"); You need to clear sending result as the memory usage will grow up. smtp.sendingResult.clear(); } } VOID SETUP() void setup() { put your setup code here, to run once: Disable brownout detector. WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); initiating the DHT Sensor dht.begin(); Getting temp. and hum. values float h = dht.readHumidity(); Read temperature as Celsius*

*(the default) float t = dht.readTemperature(); Compute heat index in Celsius (isFahreheit = false) float hic = dht.computeHeatIndex(t, h, false); Converting to string type String hum = String(h); String temp = String(t); String heat_index_celsius = String(hic); initializing Serial monitor Serial.begin(115200); Serial.println(); delay(1000); pinMode(FLASH_LED_PIN, OUTPUT); Setting the ESP32 WiFi to station mode. Serial.println(); Serial.println("Setting the ESP32 WiFi to station mode."); WiFi.mode(WIFI_STA); ————————————————- The process of connecting ESP32 CAM with WiFi Hotspot / WiFi Router. Serial.println(); Serial.print("Connecting to : "); Serial.println(ssid); WiFi.begin(ssid, password); The process timeout of connecting ESP32 CAM with WiFi Hotspot / WiFi Router is 20 seconds. If within 20 seconds the ESP32 CAM has not been successfully connected to WiFi, the ESP32 CAM will restart. I made this condition because on my ESP32-CAM, there are times when it seems like it can't connect to WiFi, so it needs to be restarted to be able to connect to WiFi. int connecting_process_timed_out = 20; -> 20 = 20 seconds. connecting_process_timed_out = connecting_process_timed_out * 2; while (WiFi.status() != WL_CONNECTED) { Serial.print("."); delay(500); if (connecting_process_timed_out > 0) connecting_process_timed_out–; if (connecting_process_timed_out == 0) { Serial.println(); Serial.print("Failed to connect to "); Serial.println(ssid); Serial.println("Restarting the ESP32 CAM."); delay(1000); ESP.restart(); } } Serial.println(); Serial.print("Successfully connected to "); Serial.println(ssid); /* Set the network reconnection option */ MailClient.networkReconnect(true); /** Enable the debug via Serial port * 0 for no debugging * 1 for basic level debugging */ smtp.debug(1); /* Set the callback function to get the sending results */ smtp.callback(smtpCallback); /* Declare the Session_Config for user defined session credentials */ Session_Config config_mail; /* Set the session config */ config_mail.server.host_name = SMTP_HOST; config_mail.server.port = SMTP_PORT; config_mail.login.email = AUTHOR_EMAIL; config_mail.login.password = AUTHOR_PASSWORD; config_mail.login.user_domain = ""; /* Set the NTP config time */ config_mail.time.ntp_server = F("pool.ntp.org,time.nist.gov"); config_mail.time.gmt_offset = 2; config_mail.time.day_light_offset = 0; /* Declare the message class */ SMTP_Message message; /* Set the message headers */ message.sender.name = F("ESP"); message.sender.email = AUTHOR_EMAIL; message.subject = F("Smart Bird House Report"); message.addRecipient(F("Pablo"), RECIPIENT_EMAIL); Send raw text message String textMsg = String("The temperature is: " + temp + " \nHumidity: " + hum + " \nheat index: " + heat_index_celsius + ". \nThe picture taken has been uploaded to your folder in Google Drive. - Sent from Smart Bird House"); message.text.content = textMsg.c_str(); message.text.charSet = "us-ascii"; message.text.transfer_encoding = Content_Transfer_Encoding::enc_7bit; message.priority = esp_mail_smtp_priority::esp_mail_smtp_priority_low; message.response.notify = esp_mail_smtp_notify_success | esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay; /* Connect to the server */ if (!smtp.connect(&config_mail)) { ESP_MAIL_PRINTF("Connection error, Status Code: %d, Error Code: %d, Reason: %s", smtp.statusCode(), smtp.errorCode(), smtp.errorReason().c_str()); ESP.restart(); return; } if (!smtp.isLoggedIn()) { Serial.println("\nNot yet logged in."); } else { if (smtp.isAuthenticated()) Serial.println("\nSuccessfully logged in."); else Serial.println("\nConnected with no Auth."); } /* Start sending Email and close the session */ if (!MailClient.sendMail(&smtp, &message)) ESP_MAIL_PRINTF("Error, Status Code: %d, Error Code: %d, Reason: %s", smtp.statusCode(), smtp.errorCode(), smtp.errorReason().c_str()); ————————————————- Set the camera ESP32 CAM. Serial.println(); Serial.println("Set the camera ESP32 CAM…"); 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; init with high specs to pre-allocate larger buffers if (psramFound()) { config.frame_size = FRAMESIZE_UXGA; config.jpeg_quality = 10; 0-63 lower number means higher quality config.fb_count = 2; } else { config.frame_size = FRAMESIZE_SVGA; config.jpeg_quality = 8; 0-63 lower number means higher quality config.fb_count = 1; } camera init esp_err_t err = esp_camera_init(&config); if (err != ESP_OK) { Serial.printf("Camera init failed with error 0x%x", err); Serial.println(); Serial.println("Restarting the ESP32 CAM."); delay(1000); ESP.restart(); } sensor_t* s = esp_camera_sensor_get(); Selectable camera resolution details : -UXGA = 1600 x 1200 pixels -SXGA = 1280 x 1024 pixels -XGA = 1024 x 768 pixels -SVGA = 800 x 600 pixels -VGA = 640 x 480 pixels -CIF = 352 x 288 pixels -QVGA = 320 x 240 pixels -HQVGA = 240 x 160 pixels -QQVGA = 160 x 120 pixels s→set_framesize(s, FRAMESIZE_XGA); –> UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA Serial.println("Setting the camera successfully."); Serial.println(); delay(1000); Test_Con(); SendCapturedPhotos(); Serial.println(); delay(15000); rtc_gpio_hold_en(GPIO_NUM_4); esp_sleep_enable_ext0_wakeup(GPIO_NUM_13, 0); Serial.println("Going to sleep now"); delay(1000); esp_deep_sleep_start(); } VOID LOOP() void loop() { } ===== 3. Results ===== ===== 4. Discussion and Conclusion ===== === 4.1. Improvements === As always there are some improvements that can be made. It would be better to use an infrared camera so even when it's dark the pictures are visible. Also for the current design of the house there can be some improvements by providing more holes so it has more ventilation and it wont accumulate water. ===== 5. Bibliography ===== This links are helpful to build, design and select the proper materials for your bird house: Materials Used for Building Birdhouses Tips for Birdhouse Design and Building

1)

StartTime + 10 * 1000) < millis(