Mailbox connect

Par : NGANKOUE THIRRY

I. Project Summary

Develop a connected mailbox capable of sending a notification each time a new mail is deposited in the mailbox via a push notification system such as PushOver or Telegram and also send an email to a Gmail mailbox in order to 'to be further processed in the google cloud via google script.

The notification and email must include a photo of the inside of the mailbox.

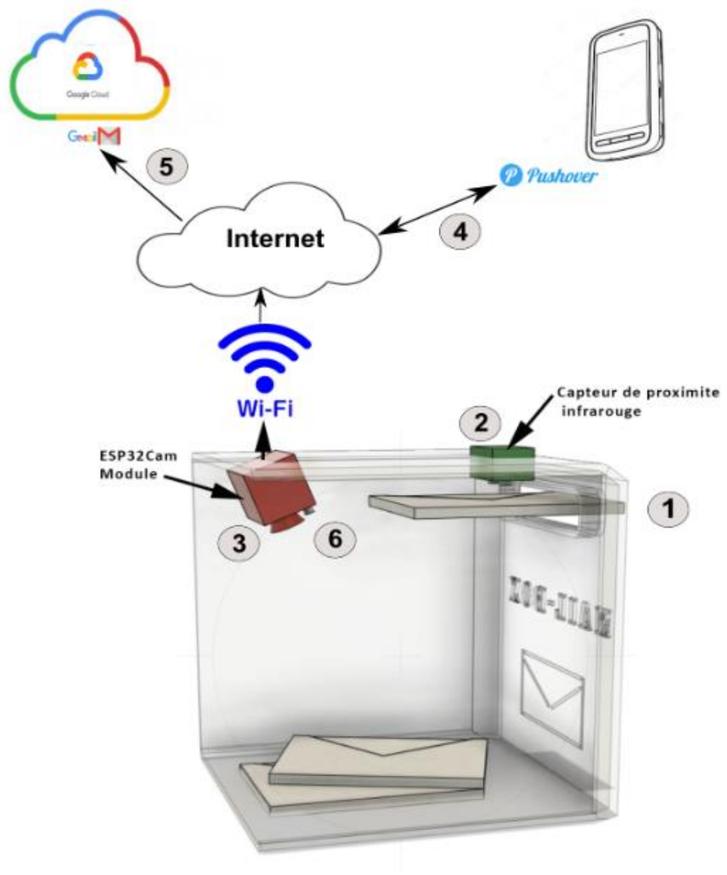
II. Problem Definition

- Useless back and forth to an empty mailbox
- Not being aware when the postman has passed
- The phases of impatience and anxiety when we are waiting for a letter
- Do not know the type of mail posted

III. Challenges & Motivation

Having ourselves been the victim of unnecessary back and forth to an empty mailbox and phases of impatience and anxiety when we wait for a mail, we have chosen the subject of the connected mailbox in order to provide a solution to its various problems in our everyday life and that of many people around the world.

IV. Technical Description



The system will consist of an infrared proximity detector as well as an esp32cam module.

The esp32cam module includes a camera, a led that can be used as a flash, a wifi interface that can connect to the internet as well as an SD card module that can possibly be used for local saving of images.

1. A new envelope is introduced into the box

2. The infrared proximity sensor detects the envelope and wakes up the esp32cam module

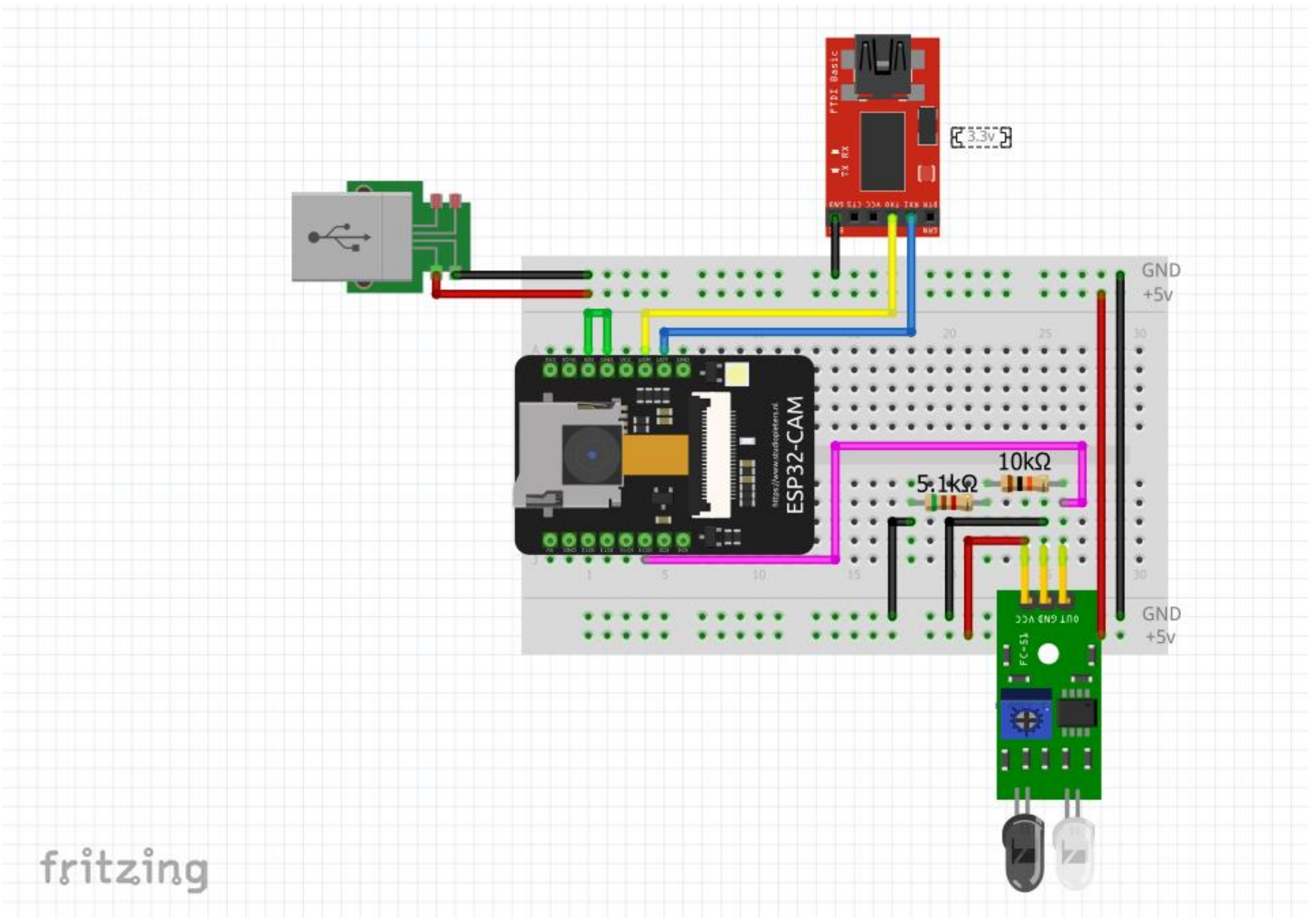
3. Once awakened, the esp32cam module turns on the flash and takes a photo.

4. The module then connects to the internet and sends a notification with the photo to a mobile device via the Pushover service

5. The module then sends an email to a gmail address with the photo in order to store the image on the google cloud.

6. The esp32cam module disconnects from the internet and wifi then returns to standby mode

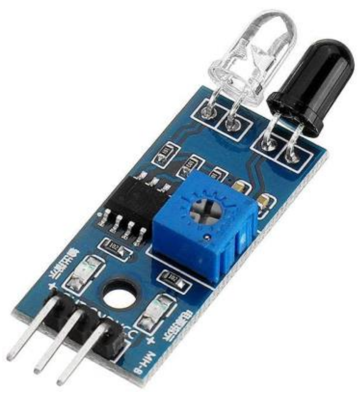
V. Schematic



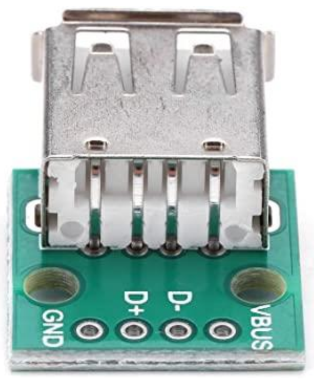
Circuit diagrams of connected mailbox

VI. Matrials

Resistance 1/4W 3.3 k Ohm



Infrared avoidance sensor



USB Breakout board

Mailboxes

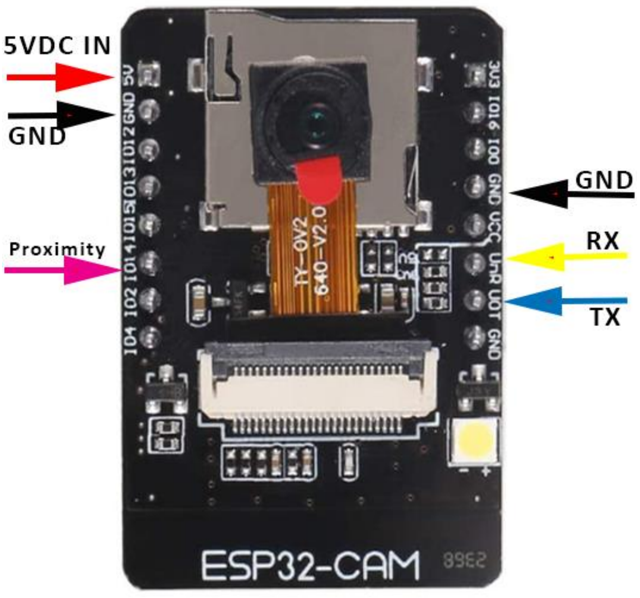
5 V DC USBPower Bank

FTDI232RL

ESP32 Cam Module

VII. Connection:

Even if the esp32Cam module operates on 3.3V it is recommended to power the esp32cam module by its 5Vdc input pin.

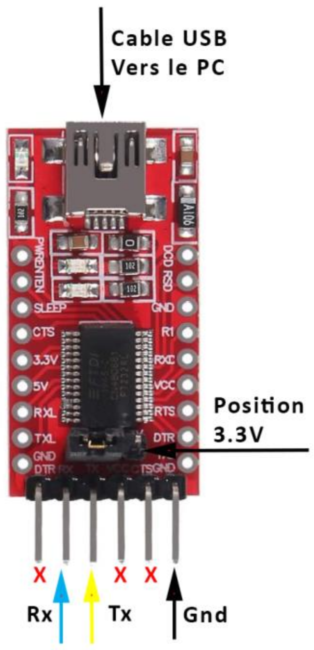

5VDC IN
GND
Proximity
GND
RX
TX
5V GND IO12 IO13 IO15 IO14 IO2 IO4
3V3 IO16 IO0 GND VCC U0R U0T GND
TY-OV2 640-V2.0
ESP32-CAM

The FTDI 232L module must be positioned on the 3.3V position.

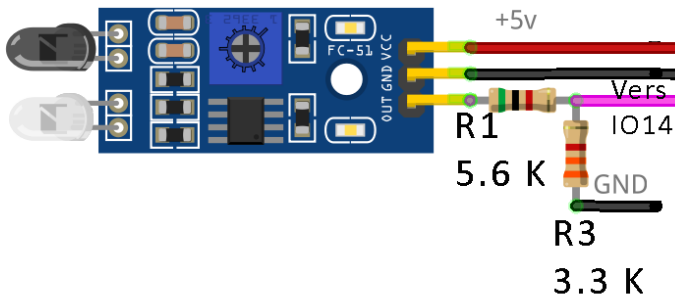
The Gnd pin must be connected to one of the Gnd pins of the Esp32Cam module

The TX pin of the FTDI module must be connected to the VOR Pin of the Esp32Cam module

The RX pin of the FTDI module must be connected to the VOT pin of the Esp32Cam module

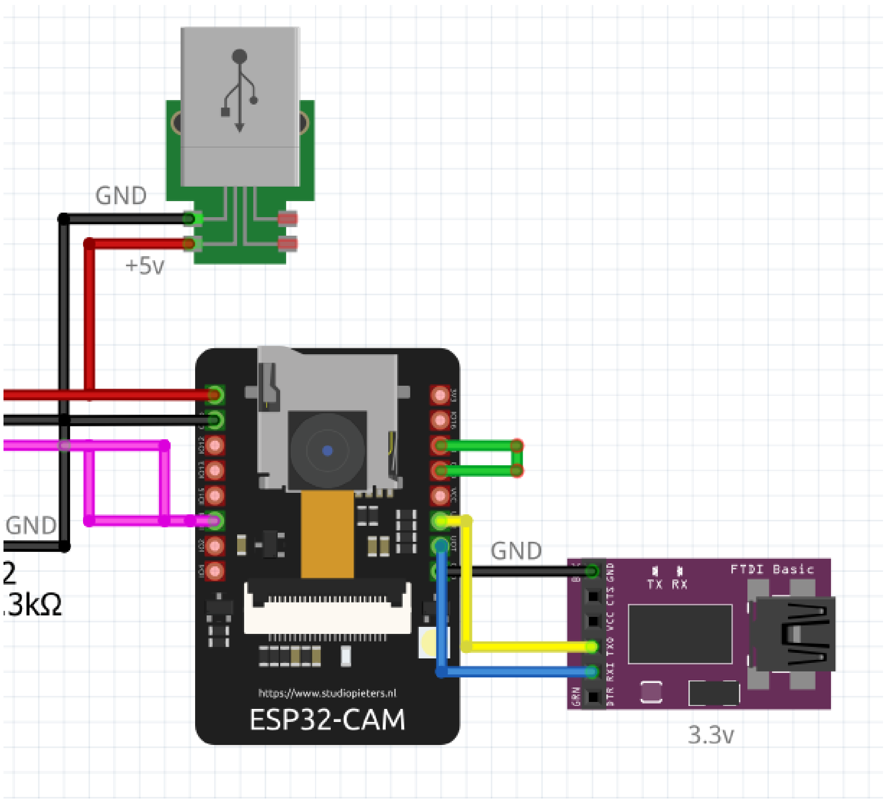


The Infrared avoidance sensor module must be powered by the 5Vdc power supply in order to stay on when the Esp32Cam module goes into standby mode in order to preserve the battery. The output of the module should be lowered to 3.3v in order to protect the input of the Esp32cam module. To lower the output voltage of the Infrared Avoidance Sensor module, we use a voltage divider resistor bridge.
We have chosen a value of 5.6 K Ohm for R1 and 3.3K Ohm for R2.
Other values can be used as long as the 5Vdc is transformed into 3.3Vdc.



VIII. Programming the esp32 module

The FTDI323L module is only needed when programming the Esp32cam module.
It is not necessary to leave it connected all the time during normal use.
The IO0 pin of the Esp32Cam module must be connected to the Gnd pin of the Esp32Cam module in order to start the module in programming mode.
Disconnect the IO0 pin from the Gnd pin of the esp32Cam module and press the reset button to start the module in normal mode.



IX. Alimentation

Pour alimenter le système nous avons deux solutions:

La premiere la plus simple est d'alimenter le système par une alimentation murale USB 5v. Avec cela le système pourra être allumé en permanence et ne nécessitera pas de recharge.

La deuxième solution avec un power bank usb 5v elle nécessite une gestion de la batterie par le microcontrôleur afin de déclencher une notification quand la batterie atteint un niveau bas.

X. Codes

```
#include "esp_camera.h"

#include "Arduino.h"

#include "FS.h"

#include "SD_MMC.h"

#include "soc/soc.h"

#include "soc/rtc_cntl_reg.h"

#include "driver/rtc_io.h"

#include <EEPROM.h>

#include "SD.h"

#include <WiFi.h>

#include "ESP32_MailClient.h"

#include "SPIFFS.h"

#include "esp_timer.h"

#include "img_converters.h"

#include "driver/rtc_io.h"


char ssid[] = "TP-Link_F4D8";

char password[] = "93655576";


// Envoyer un email via Gmail

#define emailSenderAccount    "ngankouechristian@gmail.com"

#define emailSenderPassword   "thierry94"

#define emailRecipient        "ngankouechristian@gmail.com"

#define emailRecipient2       "5v53xu8xqf@pomail.net"

#define smtpServer            "smtp.gmail.com"

#define smtpServerPort        465 //587 //465

#define emailSubject          "Thierry tu a un nouveau courrier notification"
```

```
SMTPData smtpData;


// Pushover settings

char pushoversite[] = "api.pushover.net";

char apitoken[] = "aszcfk9s7mi4ij89rvfscbmdggezc4";

char userkey [] = "um7vbt89udh9s1zidrz1gcoax1c9iq";

int length;

WiFiClient client;


// Pin definition for 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

#define TIME_TO_SLEEP  180            //time ESP32 will go to sleep (in seconds)
```

```
#define uS_TO_S_FACTOR 1000000ULL   //conversion factor for micro seconds to seconds */

int pictureNumber = 0;



void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  pinMode(4, OUTPUT);
  pinMode(33, OUTPUT);
  pinMode(14, INPUT);
  digitalWrite(4, LOW);
  digitalWrite(33, HIGH);
  Serial.begin(115200);
  //Serial.setDebugOutput(true);
  Serial.println();



  delay(500);
  Serial.println("Bonjour");
  if (!SPIFFS.begin(true)) {
    Serial.println("Une erreur est survenue lors du montage de SPIFFS");
    ESP.restart();
  }
  else {
    Serial.println("SPIFFS monte avec succe");
  }
  SPIFFS.format();
  EEPROM.begin(400);
  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_QVGA; // FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

  // Initialisation de la camera
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Erreur d'initialisation de la camera avec le code 0x%x", err);
    return;
  }

  //************************************** Connection
********************************************************************
  digitalWrite(33, LOW);
  Serial.print("Connectiion");

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(200);
  }

  Serial.println();
  Serial.println("WiFi connecte.");
  Serial.println();

}
void loop() {
  if (digitalRead(14)==LOW) {
    takePicture();
    sendNotification();
  }
}

// Callback fonction pour le statut d'envoi du courriel
void sendCallback(SendStatus msg) {
  // affiche le statut actuel
  Serial.println(msg.info());
  if (msg.success()) {
    Serial.println("----------------");
  }
}

byte pushover(char *pushovermessage)
{
  String message = pushovermessage;

  length = 81 + message.length();

  if (client.connect(pushoversite, 80))
  {
    client.println("POST /1/messages.json HTTP/1.1");
    client.println("Host: api.pushover.net");
    client.println("Connection: close\r\nContent-Type: application/x-www-form-urlencoded");
    client.print("Content-Length: ");
    client.print(length);
    client.println("\r\n");;
    client.print("token=");
    client.print(apitoken);
    client.print("&user=");
    client.print(userkey);
    client.print("&message=");
    client.print(message);
    while (client.connected())
    {
      while (client.available())
      {
        char ch = client.read();
        Serial.write(ch);
      }
    }
    client.stop();
  }
}
void takePicture() {

  //****** flash allumé  ***************************************
  digitalWrite(4, HIGH);
  camera_fb_t * fb = NULL;

  // Prise de la photo
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Echec de la prise de vue par la camera");
    return;
  }
  //****** flash eteint ****************************************
  digitalWrite(4, LOW);
  String path = "/Courrier.jpg";
  File file = SPIFFS.open(path, FILE_WRITE);
  // Chemin de sauvegarde de l'image



  if (!file) {
    Serial.println("Impossible d'ouvrir le fichier en mode ecriture");
  }
  else {
    file.write(fb->buf, fb->len); // payload (image), payload length
    Serial.printf("Fichier sauve avec le chemin suivant: %s\n", path);

  }
  file.close();
  esp_camera_fb_return(fb);
}
void sendNotification() {
  Serial.println("Preparation de l'envois du couriel");
  Serial.println();
  pushover("Vous avez un nouveau courrier!!!");

  //  Email SMTP Serveur , port, compte et mot de passe
  smtpData.setLogin(smtpServer, smtpServerPort, emailSenderAccount, emailSenderPassword);
  // Nome de l'expediteur et du compte
  smtpData.setSender("Nouveau courrier", emailSenderAccount);


  // Priorite du message High, Normal, Low ou 1 a 5 (1 est la plus forte)
  smtpData.setPriority("High");


  // Sujet du message
  smtpData.setSubject(emailSubject);


  // Corp du message en HTML
  smtpData.setMessage("<div style=\"color:#2f4468;\"><h1>Vous avez un nouveau courrier !</h1><p>-
Envoye depuis ESP32cam</p></div>", true);



  // Ajout du destinataire
  smtpData.addRecipient(emailRecipient);
  smtpData.addRecipient(emailRecipient2);
  smtpData.setFileStorageType(MailClientStorageType::SPIFFS);
  smtpData.addAttachFile("/Courrier.jpg");


  smtpData.setSendCallback(sendCallback);


  //Envois du courriel
  if (!MailClient.sendMail(smtpData))
    Serial.println("Erreur lors de l'envoi du couriel, " + MailClient.smtpErrorReason());


  //Efface toutes les donnees de l'objet Email pour liberer la memoire
  digitalWrite(33, HIGH);
  smtpData.empty();
```

```
}
```