**Projektová dokumentácia**

**ESP32-CAM video zvonček**

*Cieľom tohto projektu bolo vytvoriť systém využívajúci arudino ide a model ESP 32 CAM, ktorý zachytí obraz pomocou fotoaparátu a odošle upozornenie na Whats up. Na základe tohto upozornenia si vieme v aplikácií Home asistent prezrieť video ktoré zachytáva aktuálny obraz.*

***Schéma zapojenia :***

Obrázok, na ktorom je diagram, schematický

Automaticky generovaný popis

1. Po spracovaní predbežných návrhov (skíc) na grafický dizajn sme následne detailne vypracovali 3D návrh zvončeka. A to konkrétne pomocou nástrojov 3D modelovania v programe Fusion 360.

Takto sme precízne premenili náš 2D náčrt na 3D model. Vytvorili sme tak prvky ako tlačidlo a kryt mechanizmu zvonenia. Vybrali sme vhodné materiály pre 3D tlač, berúc do úvahy faktory ako je pevnosť, odolnosť a estetika. Použitím textúry farby a iných vzhľadov aby bol dizajn zvončeka skutočne vizuálne príťažlivý.

1. Ďalším krokom bolo naprogramovanie Esp 32 cam - pomocou arduino ide. To zahŕňalo inštaláciu príslušných knižníc ako je knižnica Esp 32 cam a knižnica Whats up api. Jednotlivé kódy v arduino ide je nutné zlúčiť, predovšetkým však treba prihliadať na príslušné funkcie kódov!

Inšpiráciu sme čerpali z nasledujúcich vzorových stránok:

**<https://randomnerdtutorials.com/esp32-cam-video-streaming-web-server-camera-home-assistant/>**

[**https://randomnerdtutorials.com/esp32-send-messages-whatsapp/**](https://randomnerdtutorials.com/esp32-send-messages-whatsapp/)

***Kombinácia a následné zlúčenie kódov:***

#include "esp\_camera.h"

#include <WiFi.h>

#include "esp\_timer.h"

#include "img\_converters.h"

#include "Arduino.h"

#include "fb\_gfx.h"

#include "soc/soc.h" //disable brownout problems

#include "soc/rtc\_cntl\_reg.h"  //disable brownout problems

#include "esp\_http\_server.h"

#include <HTTPClient.h>

//Replace with your network credentials

const char\* ssid = "Názov Wifi";

const char\* password = "Heslo";

String apiKey = "1668684";

String phone\_number = "+421 telefónne číslo";

String url;

const int buttonPin = 2; // Set the pin number for the button

volatile bool buttonstate = false; // Use a bool instead of an int to store the button state

void IRAM\_ATTR buttonISR() {

  static unsigned long last\_interrupt\_time = 0; // Keep track of the last interrupt time

  unsigned long interrupt\_time = millis();

  if (interrupt\_time - last\_interrupt\_time > 100) { // Only trigger the button state change if at least 200 ms have passed since the last interrupt

    buttonstate = true;

  }

  last\_interrupt\_time = interrupt\_time;

}

#define PART\_BOUNDARY "123456789000000000000987654321"

#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

static const char\* \_STREAM\_CONTENT\_TYPE = "multipart/x-mixed-replace;boundary=" PART\_BOUNDARY;

static const char\* \_STREAM\_BOUNDARY = "\r\n--" PART\_BOUNDARY "\r\n";

static const char\* \_STREAM\_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";

httpd\_handle\_t stream\_httpd = NULL;

static esp\_err\_t stream\_handler(httpd\_req\_t \*req){

  camera\_fb\_t \* fb = NULL;

  esp\_err\_t res = ESP\_OK;

  size\_t \_jpg\_buf\_len = 0;

  uint8\_t \* \_jpg\_buf = NULL;

  char \* part\_buf[64];

  res = httpd\_resp\_set\_type(req, \_STREAM\_CONTENT\_TYPE);

  if(res != ESP\_OK){

    return res;

  }

  while(true){

    fb = esp\_camera\_fb\_get();

    if (!fb) {

      Serial.println("Camera capture failed");

      res = ESP\_FAIL;

    } else {

      if(fb->width > 400){

        if(fb->format != PIXFORMAT\_JPEG){

          bool jpeg\_converted = frame2jpg(fb, 80, &\_jpg\_buf, &\_jpg\_buf\_len);

          esp\_camera\_fb\_return(fb);

          fb = NULL;

          if(!jpeg\_converted){

            Serial.println("JPEG compression failed");

            res = ESP\_FAIL;

          }

        } else {

          \_jpg\_buf\_len = fb->len;

          \_jpg\_buf = fb->buf;

        }

      }

    }

    if(res == ESP\_OK){

      size\_t hlen = snprintf((char \*)part\_buf, 64, \_STREAM\_PART, \_jpg\_buf\_len);

      res = httpd\_resp\_send\_chunk(req, (const char \*)part\_buf, hlen);

    }

    if(res == ESP\_OK){

      res = httpd\_resp\_send\_chunk(req, (const char \*)\_jpg\_buf, \_jpg\_buf\_len);

    }

    if(res == ESP\_OK){

      res = httpd\_resp\_send\_chunk(req, \_STREAM\_BOUNDARY, strlen(\_STREAM\_BOUNDARY));

    }

    if(fb){

      esp\_camera\_fb\_return(fb);

      fb = NULL;

      \_jpg\_buf = NULL;

    } else if(\_jpg\_buf){

      free(\_jpg\_buf);

      \_jpg\_buf = NULL;

    }

    if(res != ESP\_OK){

      break;

    }

    //Serial.printf("MJPG: %uB\n",(uint32\_t)(\_jpg\_buf\_len));

  }

  return res;

}

void startCameraServer(){

  httpd\_config\_t config = HTTPD\_DEFAULT\_CONFIG();

  config.server\_port = 80;

  httpd\_uri\_t index\_uri = {

    .uri       = "/",

    .method    = HTTP\_GET,

    .handler   = stream\_handler,

    .user\_ctx  = NULL

  };

  //Serial.printf("Starting web server on port: '%d'\n", config.server\_port);

  if (httpd\_start(&stream\_httpd, &config) == ESP\_OK) {

    httpd\_register\_uri\_handler(stream\_httpd, &index\_uri);

  }

}

void setup() {

  WRITE\_PERI\_REG(RTC\_CNTL\_BROWN\_OUT\_REG, 0); //disable brownout detector

  Serial.begin(115200);

  Serial.setDebugOutput(false);

  pinMode(buttonPin, INPUT\_PULLUP); // Set the button pin as input and enable the internal pull-up resistor

  attachInterrupt(digitalPinToInterrupt(buttonPin), buttonISR, FALLING); // Attach the interrupt to the button pin for detecting falling edge (button press)

  WiFi.begin(ssid, password);

  Serial.println("Connecting to WiFi");

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println();

  Serial.println("Connected to the WiFi network");

  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\_SXGA;

    config.jpeg\_quality = 10;

    config.fb\_count = 2;

  } else {

    config.frame\_size = FRAMESIZE\_SVGA;

    config.jpeg\_quality = 12;

    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);

    return;

  }

  // Wi-Fi connection

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");

  Serial.print(WiFi.localIP());

  // Start streaming web server

  startCameraServer();

}

void loop() {

String localIP = WiFi.localIP().toString();

  if (buttonstate) { // Check if the button is pressed

    message\_to\_whatsapp("Niekto zvoní: http://" + String(localIP)); // Send the message

    buttonstate = false; // Reset the button state to false

    Serial.print(String(localIP));

  }

  delay(300);

}

void message\_to\_whatsapp(String message) {

  url = "https://api.callmebot.com/whatsapp.php?phone=" + phone\_number + "&apikey=" + apiKey + "&text=" + urlencode(message);

  postData();

}

void postData() {

  int httpCode;

  HTTPClient http;

  http.begin(url);

  httpCode = http.POST(url);

  if (httpCode == 200) {

    Serial.println("Sent ok.");

  }

  else {

    Serial.println("Error.");

  }

  http.end();

}

String urlencode(String str) {

    String encodedString="";

    char c;

    char code0;

    char code1;

    char code2;

    for (int i =0; i < str.length(); i++){

      c=str.charAt(i);

      if (c == ' '){

        encodedString+= '+';

      } else if (isalnum(c)){

        encodedString+=c;

      } else{

        code1=(c & 0xf)+'0';

        if ((c & 0xf) >9){

            code1=(c & 0xf) - 10 + 'A';

        }

        c=(c>>4)&0xf;

        code0=c+'0';

        if (c > 9){

            code0=c - 10 + 'A';

        }

        code2='\0';

        encodedString+='%';

        encodedString+=code0;

        encodedString+=code1;

      }

      yield();

    }

    return encodedString;

}

1. Postup pokračoval vytvorením a inštaláciou chat bot-u priamo do telefónu. Po získaní príslušných údajov je potrebné vloženie špecifických chat ID údajov priamo do arduino kódu. Takisto je potrebné údaje vložiť aj do ESP cam.
2. Modul ESP 32 sme pripojili k zdroju napájania a uistili sme sa že poskytuje vhodné napätie a prúd podľa špecifikácií modulu.
3. Na zachytenie snímky sme použili model fotoaparátu, experimentovali sme s rôznymi rozlíšeniami obrazu nastaveniami kvality a spúšťačmi snímania obrazu(napr. stačenie tlačidla, časovač) aby vyhovovali našim špecifickým potrebám.
4. Funkčnosť projektu preverujeme aj pomocou knižnice Whats up api táto odošle správu s upozornením na skupinu.
5. Nainštalovali sme si Home asistent na raspberry pi.
6. Nakonfigurovali sme modul ESP 32 cam ako komponent kamery v aplikácií Home asistent. Na pripojenie kamery sme použili spomínanú aplikáciu. Home Assistant je open-source platforma domácej automatizácie, ktorá nám umožnila ovládať a automatizovať rôzne inteligentné domáce zariadenia z jedného jednotného rozhrania. Zaujalo nás, že je navrhnutý tak, aby fungoval na rôznych zariadeniach, ako je napríklad Raspberry Pi.
7. Pomocou ESP home sme nastavili Esp 32 Dev kit. Použili sme ho z dôvodu užívateľsky prívetivého webového konfiguračného rozhrania, ktoré nám umožňovalo definovať ESP a jeho funkcie pomocou jazyka založeného na YAML.

Po definovaní konfigurácie na ESP home automaticky vygeneruje firmware ktorý je možné nahrať na dosku ESP 32 pomocou arduina ide.

1. Esp home sme spojili s platformou Home asistent.

**KÓD:**

esphome:

  name: kontrola\_branicky

  friendly\_name: Rele

esp32:

  board: nodemcu-32s

  framework:

    type: arduino

# Enable logging

logger:

# Enable Home Assistant API

api:

  encryption:

    key: "BG0Y5vRmixCA6tr11ChTWQeMzrYWM+aA7eHIba9uq18="

ota:

  password: "feb20aa5d5543f40eada9d121d49ada6"

switch:

  - platform: gpio

    name: "Kontorla-bráničky"

    pin: 21

wifi:

  ssid: "Názov wifi"

  password: "Heslo"

  # Enable fallback hotspot (captive portal) in case wifi connection fails

  ap:

    ssid: "Rele Fallback Hotspot"

    password: "GUNHRgivelcb"

captive\_portal:

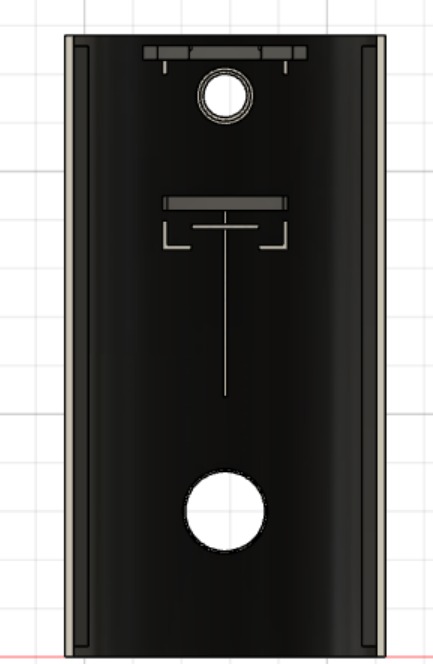
1. Zostavili sme hardvérové komponenty do funkčného systému zvončeka vrátane modulu ESP 32, druhého modulu ESP cam a relé (1 Channel DC 12VLatching Relay Module with Touch Bistable Switch MCU Control One Channel Relay) Spodné otvory zvončeka slúžia na napájanie - 12V napája relátko a 5V napája ESP 32 a ESP cam.

*Použitím ESP 32 a ESP home sa nám podarilo vytvoriť vlastný zvonček a integrovať ho s open source platformou - Home asistent.*

*Takto môže plnohodnotne fungovať ako súčasťou inteligentnej domácnosti.*

*Fotodokumentácia modelu:*

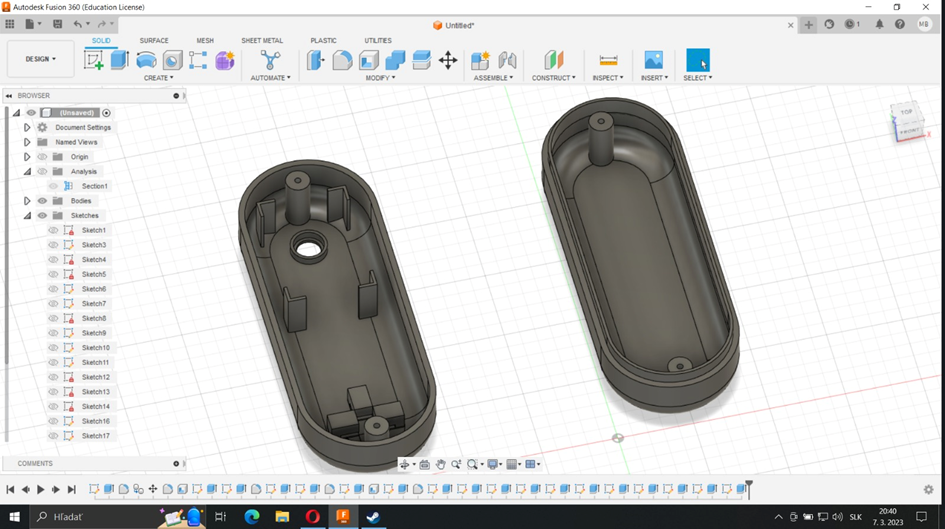
Obrázok, na ktorom je osoba, vnútri, modrá, ruka

Automaticky generovaný popis 

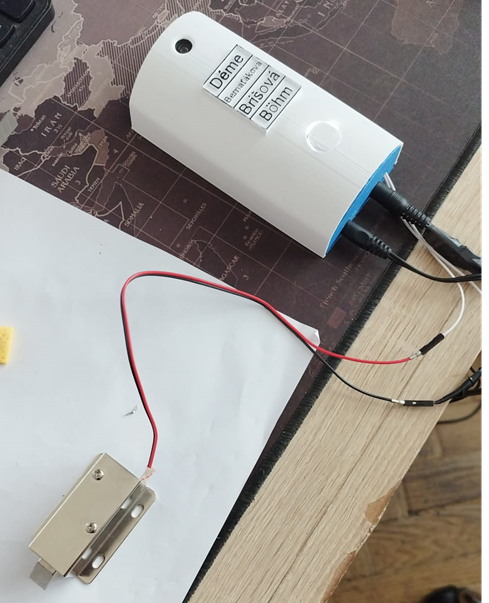
Obrázok, na ktorom je diagram

Automaticky generovaný popisObrázok, na ktorom je text, čierny, biely

Automaticky generovaný popis



Obrázok, na ktorom je kábel, elektronika, elektrinžinierstvo, elektroinštalácia

Automaticky generovaný popis

Obrázok, na ktorom je text, multimédiá, snímka obrazovky, gadget

Automaticky generovaný popis

Relé – kontrola bráničky

Obrázok, na ktorom je text, kábel, elektronika, dizajn

Automaticky generovaný popis

Výsledný produkt : **ESP32-CAM video zvonček**

Zdroje:

[**https://randomnerdtutorials.com/esp32-cam-video-streaming-web-server-camera-home-assistant/**](https://randomnerdtutorials.com/esp32-cam-video-streaming-web-server-camera-home-assistant/)

[**https://randomnerdtutorials.com/esp32-send-messages-whatsapp/**](https://randomnerdtutorials.com/esp32-send-messages-whatsapp/)