# FaceId Security System

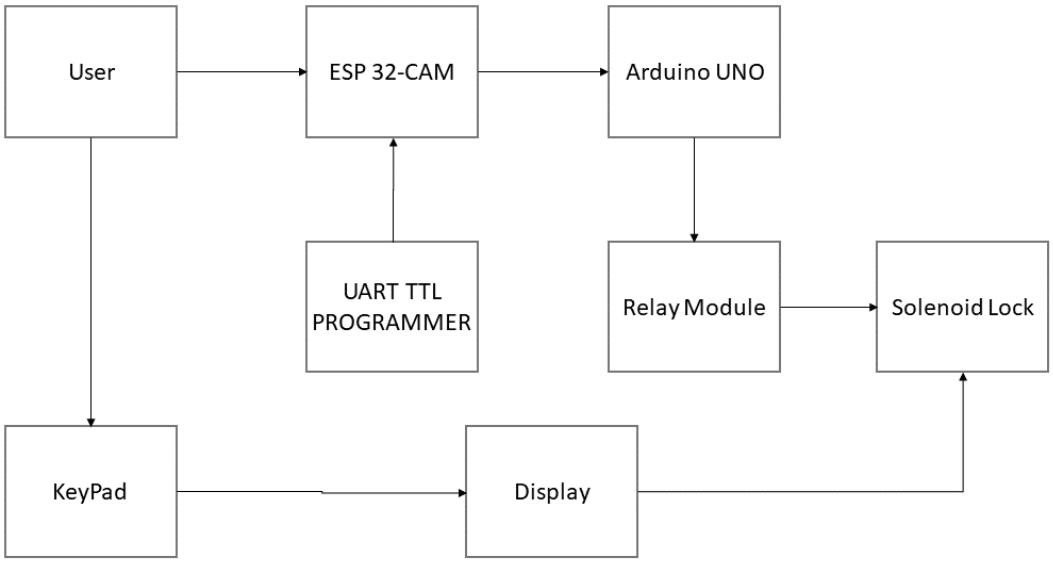
by Uleia Răzvan-Alexandru

## Introduction

The **FaceId Security System** is based on recognizing the user's face and unlocking the lock if the face matches. If the system fails to recognize the face 3 times, then the user will be directed to introduce the pin manually to be able to unlock the lock.

## General Description

The user will put his face within the camera reach. Firstly, the user will be asked to introduce a PIN to secure the lock. The ESP32-CAM will detect the user's face, and the user is able to add his face to the camera's feed. After this the lock will be activated. If the user wants to unlock the specific lock, he just has to put his face in the camera such that it will detect and recognize it. If somehow there is a software error and the face cannot be recognized, the user will be asked to introduce the PIN he just created to be able to unlock it.

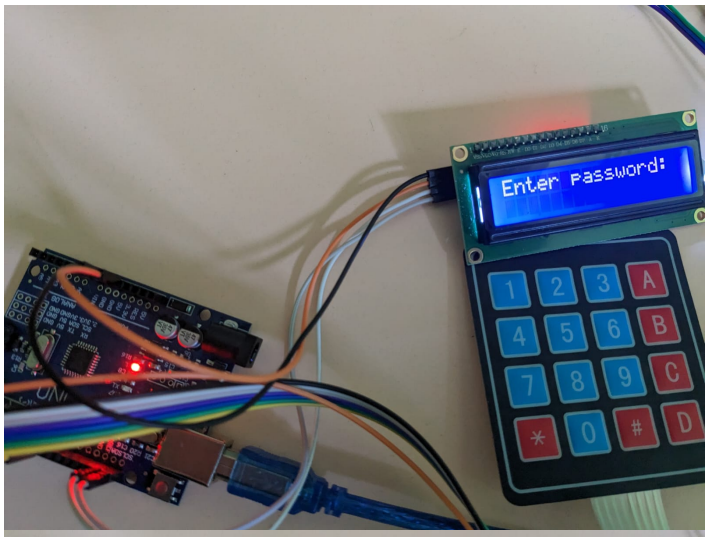


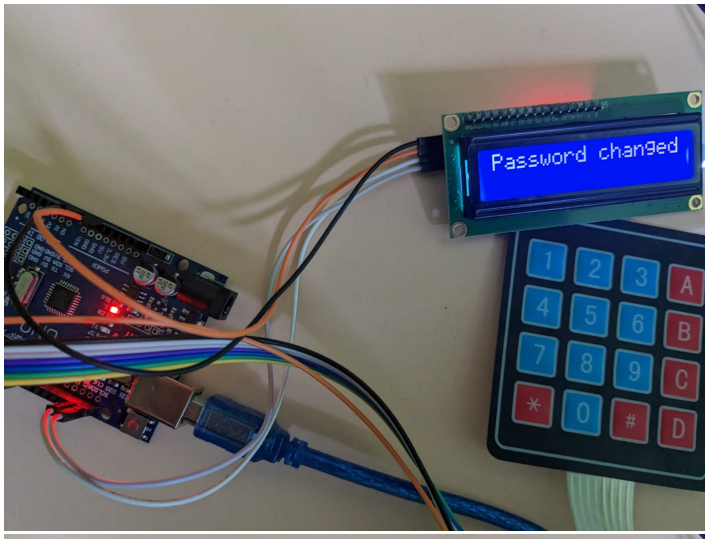
# Hardware Design

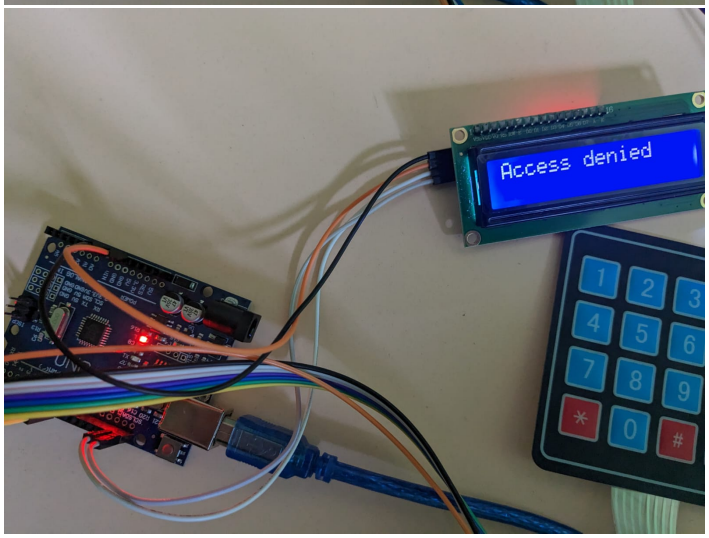
The components used in the project:

- Arduino UNO
- ESP 32-Cam
- Buzzer
- Relay Module
- UART TTL Programmer
- 12V Solenoid Lock
- 12V Battery
- 7805 Regulator
- Display LCD 2×16
- 4×4 Keypad

*this is not the final version, I will upload more irl pictures as soon as I get my wires delivered*




## Software Design

```
#include "esp_camera.h"
#include <WiFi.h>
#include <ESPAsyncWebSrv.h>
//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is
selected
//            Partial images will be transmitted if image exceeds buffer
size
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
```

```
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM

#include "camera_pins.h"

#define output 2

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
AsyncWebServer server(8080);
void startCameraServer();


bool activateRelay = false;
bool isFaceDetected = false;
unsigned long prevMillis=0;
unsigned long currentMillis;
int interval = 5000;

void setup() {

  pinMode(output, OUTPUT);
  prevMillis = 0;

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();


  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 PSRAM IC present, init with UXGA resolution and higher JPEG quality
  //                      for larger pre-allocated frame buffer.
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // 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;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE) ||
defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif


  WiFi.softAP(ssid, password);
  IPAddress apIP = WiFi.softAPIP();
  Serial.print("Access Point IP address: ");
  Serial.println(apIP);
  server.begin();

  startCameraServer();
}
```

```
void loop() {

 // Serial.println(isFaceDetected);
  delay(2000);

  //Serial.println(isFaceDetected);
  delay(2000);
  currentMillis = millis();
  if(isFaceDetected==true)
  {
    activateRelay=true;
    if ((currentMillis - prevMillis >= interval) && activateRelay == true){
    digitalWrite(output,LOW);
    delay(2000);
    prevMillis = currentMillis;
    activateRelay=false;
     isFaceDetected=false;
     delay(2000);
}
    digitalWrite(output, HIGH);



 /* if (isFaceDetected == true) {
    // Activate the relay by setting the relay pin to HIGH
    delay(2000);
    digitalWrite(output, LOW);
    delay(2000);
  } else {
    // Deactivate the relay by setting the relay pin to LOW
    dleay(2000);
    digitalWrite(output, HIGH);
    delay(2000);
  }*/
}
}
```

This is the code for the camera to work. I have used the “CameraWebServer” example as a guiding point and I have done the necessary modifications.

In it, I have modified the “httpds.cpp” file so that when the ESP32-CAM detects my face, it will send a message in the Serial. The message from the Serial is read by Arduino Uno through connecting it to the RXD and TXD pins of the ESP32-CAM. The message sends a voltage through the Relay Module and the Relay Module Opens the 12V Solenoid Lock making it open.

```
const int relayPin = 10;

#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```
#include <EEPROM.h>


const byte ROWS = 4;
const byte COLS = 4;

char keys[COLS][ROWS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {2,3,4,5};
byte colPins[COLS] = {6,7,8,9};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
char password[5]; // Default password
char enteredPassword[5];   // Entered password
bool changeMode = false;
bool passwordChanged = false;

const int passwordAddress = 0;

void setup() {
  // Initialize the serial communication
  Serial.begin(115200);
  lcd.begin(16, 2);   // Initialize the LCD
  lcd.clear();        // Clear the LCD screen

  lcd.print("Hello User");  // Display initial message on LCD
  delay(2000);
  lcd.clear();
  lcd.print("Enter password:");

  // Set the relay pin as an output
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, HIGH);
}

void loop() {
  // Check if there are any incoming serial messages
  if (Serial.available()) {
    // Read the incoming message
    char message = Serial.read();
    delay(2000);
    Serial.println(message);
    // Check if the message indicates a face detection
    if (message == 'q' ) {
```

```
      Serial.println("Triggering Relay");
      // Trigger the relay module by setting the relay pin high
      digitalWrite(relayPin, LOW);
      delay(5000);  // Keep the relay activated for 1 second
      digitalWrite(relayPin, HIGH);  // Turn off the relay
    }
  }

  char key = keypad.getKey(); // Get the pressed key

  if (key != NO_KEY) { // If a key is pressed
    if (key == 'A') { // Enter password change mode
      lcd.clear();
      lcd.print("Enter new pass:");
      changeMode = true;
    } else if (key == 'D') { // Verify entered password
      if (changeMode) {
        lcd.clear();
        lcd.print("Password changed");
        delay(2000);
        lcd.clear();
        lcd.print("Verify it:");
        changeMode = false;
        EEPROM.put(passwordAddress, password); // Store the new password in
EEPROM
      } else {
        EEPROM.get(passwordAddress, password); // Retrieve the password from
EEPROM
        if (strcmp(enteredPassword, password) == 0) {
          lcd.clear();
          lcd.print("Access granted");
          digitalWrite(relayPin, LOW);
          delay(5000);  // Keep the relay activated for 1 second
          digitalWrite(relayPin, HIGH);  // Turn off the relay
        } else {
          lcd.clear();
          lcd.print("Access denied");
        }
        delay(2000);
        lcd.clear();
        lcd.print("Enter password:");
        memset(enteredPassword, 0, sizeof(enteredPassword)); // Clear
entered password
      }
    } else { // Append entered digit to password
      if (changeMode) {
        lcd.setCursor(strlen(enteredPassword), 1); // Set cursor position to
the end of entered password
        lcd.print(''); // Display "" instead of entered digits
        strcat(enteredPassword, &key);
```

```
        strcpy(password, enteredPassword); // Update the password array when
a new password is entered
      } else {
        lcd.setCursor(strlen(enteredPassword), 1); // Set cursor position to
the end of entered password
        lcd.print(''); // Display "" instead of entered digits
        strcat(enteredPassword, &key);
      }
    }
  }
}
```

This is the code that allows the Relay Module to be actioned by the KeyPad aswell

# Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

# Concluzii

# Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

# Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

# Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Link to project page  Export to PDF