1. Introducere

1.1. Scopul proiectului

Scopul proiectului este reprezentat de realizarea unui sistem de casă inteligentă care oferă funcții de monitorizare, securitate și control la distanță, utilizând microcontrolerele Arduino Mega 2560 și ESP32. Proiectul își propune să colecteze date din mediul înconjurător și despre activitatea utilizatorilor printr-o serie de senzori, să le afișeze local pe un ecran LCD și să le transmită către o interfață web găzduită de ESP32.

1.2. Descrierea generală a sistemului

Sistemul este format din mai mulți senzori și module interconectate, fiecare având un rol specific în gestionarea securității și confortului locuinței. Arduino Mega colectează datele de la senzori precum: temperatură și umiditate (DHT11), distanță (HC-SR04), detecție de gaz (MQ-2), detecție flacără (senzor IR), și input de la utilizator (tastatură și RFID). De asemenea, Arduino controlează buzzerul și ecranul LCD pentru feedback local. Datele colectate de Arduino sunt transmise serial către modulul ESP32. ESP32, la rândul său, găzduiește o pagină web unde afișează aceste date în timp real și permite vizualizarea fluxului video de la camera asociată, gestionând totodată și procesul de autentificare prin browser.

2. Realizarea proiectului

2.1. Componentele hardware utilizate

- **Arduino Mega 2560:** placă de dezvoltare cu numeroși pini I/O, ideală pentru proiecte complexe. Controlează logica și interacțiunea cu toți senzorii și perifericele conectate direct
- **ESP32:** microcontroler cu Wi-Fi integrat. Acționează ca un server web, primind date de la Arduino Mega prin comunicare serială, afișându-le pe o interfață web pentru monitorizare de la distanță și gestionând fluxul video de la camera OV2640
- **senzor de flacără IR:** detectează radiațiile infraroșii emise de foc
- **HC-SR04:** măsoară distanța față de obstacole cu ultrasunete
- **DHT11:** măsoară temperatura și umiditatea aerului
- **senzor MQ-2:** detectează monoxidul de carbon și alte gaze nocive, oferă protecție împotriva scurgerilor de gaz
- **modul cameră OV2640:** cameră digitală pentru captură de imagini. Permite supravegherea vizuală a spațiului monitorizat, fiind integrată cu modulul ESP32 pentru streaming web
- **modul RFID RC522:** citește cartele și taguri RFID pentru autentificare
- **tastatură matricială 4x4:** permite introducerea de coduri PIN
- **buzzer:** emite semnale sonore pentru alerte sau notificări, folosit ca feedback auditiv pentru acțiuni precum apăsări de taste sau autentificări
- **LCD 16x2 cu I2C:** afișaj text pentru date și mesaje, economisește pini prin comunicație I2C
- **breadboard și fire de conexiune:** permite testarea circuitelor fără lipire.

2.2. Schema electrică

Conform schemei Fritzing atașate și detaliilor din cod:

- **tastatura matriceală 4x4** este conectată la pinii analogici ai Arduino-ului utilizați ca digitali: rândurile la A0, A1, A2, A3 și coloanele la A4, A5, A6, A7
- **senzorul DHT11** este conectat la pinul digital 11 al Arduino-ului, pentru a citi date de temperatură și umiditate
- **senzorul HC-SR04** este conectat la pinii digitali 9 (TRIG) și 10 (ECHO) ai Arduino-ului, pentru a măsura distanța
- **senzorul de flacără** este conectat la pinul analogic A9 al Arduino-ului
- **buzzerul** este conectat la pinul digital 7 al Arduino-ului
- **modulul RFID RC522** comunică prin interfața SPI, utilizând pinii SPI impliciti ai Arduino Mega (MOSI, MISO, SCK) și pinii digitali 49 (RST) și 53 (SS) pentru controlul specific al modulului
- **senzorul MQ-2** este conectat la pinul analogic A8 al Arduino-ului pentru semnalul de detecție gaz

- **LCD-ul (16x2)** folosește interfața I2C, fiind conectat la pinii I2C ai Arduino-ului (SDA și SCL). Conexiunile sunt etichetate general ca galben = SDA și mov = SCL

- **ESP32** (incluzând modulul camerei OV2640) este conectat la alimentare separat. Comunică cu Arduino Mega prin **Serial1**, unde pinii RXD1 (pinul 4) și TXD1 (pinul 5) ai ESP32 sunt conectați la pinii corespunzători TX1 (pinul 18) și RX1 (pinul 19) ai Arduino Mega, pentru transmiterea datelor senzorilor și a stărilor de autentificare.

Toate punctele de **GND (masă)** sunt comune în întregul circuit pentru a asigura o bună funcționare și referință de tensiune.

Conexiunile sunt etichetate clar pe schema Fritzing cu:

- roșu = alimentare 5V/ 3.3V
- negru = GND
- galben = SDA
- mov = SCL.

2.3. Alimentare

Alimentarea se realizează prin două cabluri USB:

- un cablu conectat la **Arduino Mega 2560**, care alimentează majoritatea senzorilor și perifericelor conectate direct la acesta

- un al doilea cablu USB conectat la **ESP32**, care alimentează placa și modulul camerei. Cele două plăci (Arduino Mega și ESP32) împart același **GND (masă comună)** pentru a permite comunicarea corectă între ele.

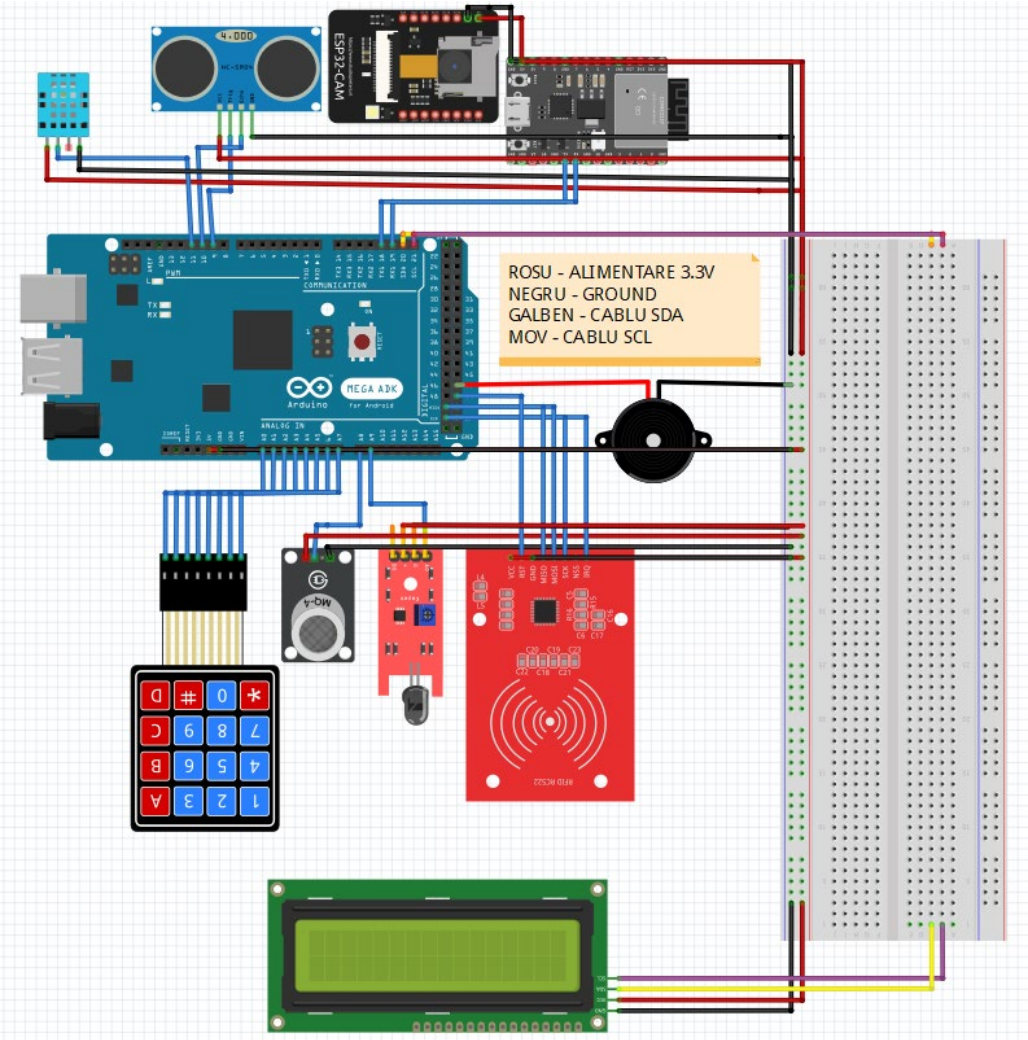

Figura 1. Schema Fritzing a montajului

3. Codul sursă

3.1. Cod ESP32

```
#include <WiFi.h>
#include <WebServer.h>

const char* ssid     = "Oaierless";
const char* password = "11nustiucesapun12";

#define BUZZER_PIN 7
#define RXD1 4
#define TXD1 5

WebServer server(80);

String lastRFID      = "— niciun tag —";
String lastSensorRaw = "— încă nu am date —";
String loginStatus   = "";
bool keypadAccess    = false;

void beep() { tone(BUZZER_PIN, 1000, 100); }

int getDistance() {
  int idx = lastSensorRaw.indexOf("D=");
  if (idx < 0) return 9999;
  return lastSensorRaw.substring(idx+2).toInt();
}

void handleRoot() {
  int dist = getDistance();
  String html = R"rawliteral(
<!DOCTYPE html><html lang='ro'>
<head><meta charset='UTF-8'><meta name='viewport' content='width=device-width,initial-scale=1.0'>
<title>Smart Home</title>
<meta http-equiv='refresh' content='5'>
<style>body{font-family:Arial;text-
align:center;background:#eef2f3;padding:20px;}h1{color:#333;}p{font-size:1.2em;}
.alert{color:#c00;margin-top:10px;}button{padding:15px30px;font-
size:1.2em;margin:10px;background:#4a90e2;color:#fff;
border:none;border-
radius:8px;cursor:pointer;transition:0.3s;}button:hover{background:#357ABD;transform:scale(1.05);}</
style></head>
<body><h1>♡ Detectare Acces</h1><p>Distanță: )rawliteral" + String(dist) + R"rawliteral(
cm</p>)rawliteral";

  if (dist < 150) html += "<form action='/login'><button>Intră în Login</button></form>";
  else html += "<p class='alert'>❌ Te afli prea departe! (<150 cm)</p>";


  html += "</body></html>";
  server.send(200, "text/html", html);
```

```
}

void handleLogin() {
  String html = R"rawliteral(
<!DOCTYPE html><html lang='ro'>
<head><meta charset='UTF-8'><meta name='viewport' content='width=device-width,initial-scale=1.0'>
<title>Login</title>
<style>body{font-family:Arial;text-align:center;background:#f9f9f9;padding:20px;}h1{color:#444;}
button{padding:15px 25px;font-
size:1em;margin:10px;background:#4CAF50;color:#fff;border:none;border-
radius:8px;cursor:pointer;transition:0.3s;}
button:hover{background:#45a049;transform:scale(1.05);}</style></head>
<body><h1>🔒 Login Smart Home</h1><p>Asteptam rfid, cod acces sau cod browser</p>
<form action='/login/keypad'><button>🖐️ Keypad</button></form>
<form action='/login/browser'><button>🌐 Browser PIN</button></form>
<form action='/login/rfid'><button>💳 RFID</button></form>
<br><a href='/'>Înapoi</a></body></html>)rawliteral";
  server.send(200, "text/html", html);
  beep();
}

void handleKeypadPage() {
  if (keypadAccess) {
    String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'>"
      "<title>Acces PERMIS</title>"
      "<script>setTimeout(function(){ window.location.href='/dashboard'; }, 3000);</script>"
      "<style>body{font-family:Arial;text-align:center;padding:20px;}</style></head><body>"
      "<h1>🖐️ Keypad Login</h1>"
      "<p>✅ PIN tastatura: ACCES PERMIS!</p>"
      "<p><small>(Redirecționare automată în 3 secunde...)</small></p>"
      "<br><a href='/login'>Înapoi</a></body></html>";
    server.send(200, "text/html", html);
    beep();

    keypadAccess = false;  // Reset acces după afișare
  } else {
    String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'>"
      "<title>Keypad Login</title><meta http-equiv='refresh' content='5'>"
      "<style>body{font-family:Arial;text-align:center;padding:20px;}</style></head><body>"
      "<h1>🖐️ Introdu codul pe tastatura fizică</h1>"
      "<p>Așteptăm confirmarea de la Arduino Mega...</p>"
      "<p><small>(Pagina se reîncarcă la fiecare 5 secunde)</small></p>"
      "<br><a href='/login'>Înapoi</a></body></html>";
    server.send(200, "text/html", html);
    beep();
  }
}

void handleLoginBrowser() {
  String html = R"rawliteral(
<!DOCTYPE html><html><head><meta charset='UTF-8'><meta name='viewport'
content='width=device-width,initial-scale=1.0'>
```

```
<title>Browser PIN</title>

<style>body{font-family:Arial;text-align:center;padding:20px;}input,button{padding:10px;font-
size:1em;margin:10px;} </style></head><body>
<h1>🌐 Introdu PIN</h1>
<form action='/login/browser/check' method='GET'>
<input name='code' maxlength='4' pattern='[0-9]{4}' placeholder='0000' required>
<button type='submit'>OK</button></form>
<br><a href='/login'>Înapoi</a></body></html>)rawliteral";
  server.send(200, "text/html", html);
}

void handleLoginBrowserCheck() {
  String code = server.arg("code");
  if (code == "0000") {
    loginStatus = "✅ Acces PERMIS!";
    beep();
    server.sendHeader("Location", "/dashboard", true);
    server.send(302, "text/plain", "");
  } else {
    loginStatus = "❌ PIN greșit!";
    beep();
    server.sendHeader("Location", "/login/browser", true);
    server.send(302, "text/plain", "");
  }
}

void handleLoginRFID() {
  if (loginStatus.indexOf("PERMIS") >= 0) {
    String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'>"
      "<title>Acces PERMIS</title>"
      "<script>setTimeout(function(){ window.location.href='/dashboard'; }, 3000);</script>"
      "<style>body{font-family:Arial;text-align:center;padding:20px;}</style></head><body>"
      "<h1>💳 RFID Login</h1>"
      "<p>Ultimul RFID UID: " + lastRFID + "</p>"
      "<p>✅ Acces: " + loginStatus + "</p>"
      "<p><small>(Redirecționare automată în 3 secunde...)</small></p>"
      "<br><a href='/login'>Înapoi</a></body></html>";
    server.send(200, "text/html", html);
    beep();

    loginStatus = ""; // Reset
  } else {
    String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'>"
      "<title>Login RFID</title><meta http-equiv='refresh' content='5'>"
      "<style>body{font-family:Arial;text-align:center;padding:20px;}</style></head><body>"
      "<h1>💳 RFID Login</h1>"
      "<p>Asteptam rfid...</p>"
      "<p>Ultimul RFID UID: " + lastRFID + "</p>"
      "<p>Acces: " + loginStatus + "</p>"
      "<br><a href='/login'>Înapoi</a></body></html>";
    server.send(200, "text/html", html);
```

```
    beep();
  }

}

void handleDashboard() {
  String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'>"
    "<title>Dashboard</title><meta http-equiv='refresh' content='5'>"
    "<style>body{font-family:Arial;text-align:center;background:#eef2f3;padding:20px;}"
    "h1{color:#333;}button{padding:10px 20px;margin:10px;}</style></head><body>"
    "<h1>📊 Dashboard Smart Home</h1>"
    "<button onclick=\"location.href='/senzor/temp'\">🌡 Temperatura</button>"
    "<button onclick=\"location.href='/senzor/umid'\">💧 Umiditate</button>"
    "<button onclick=\"location.href='/senzor/gaz'\">🔥 Gaz</button>"
    "<button onclick=\"location.href='/senzor/flacara'\">🔥 Flacără</button>"
    "<button onclick=\"location.href='/senzor/distanta'\">📏 Distanță</button>"
    "<button onclick=\"location.href='/senzor/camera'\">📹 Cameră</button>"
    "<br><a href='/'>Înapoi</a></body></html>";
  server.send(200, "text/html", html);
}
void handleCamera() {
  String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'><title>📹
Cameră</title></head><body>"
                "<h1>📹 Flux video</h1>"
                "<iframe src='http://192.168.51.138:80/stream' width='100%' height='480' frameborder='0'
allowfullscreen></iframe>"
                "<br><a href='/dashboard'>Înapoi la Dashboard</a></body></html>";
  server.send(200, "text/html", html);
}

void handleSenzorTemp() {
  int idxT = lastSensorRaw.indexOf("T=");
  String temp = (idxT >= 0) ? lastSensorRaw.substring(idxT + 2, lastSensorRaw.indexOf(",", idxT)) :
"necunoscut";

  String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'><title>🌡
Temperatura</title>"
    "<style>body{font-family:Arial;text-align:center;background:#eef2f3;padding:20px;}"
    "h1{color:#333;}p{font-size:1.3em;}</style></head><body>"
    "<h1>🌡 Temperatura</h1>"
    "<p><strong>" + temp + " °C</strong></p>"
    "<br><a href='/dashboard'>Înapoi la Dashboard</a></body></html>";

  server.send(200, "text/html", html);
}

void handleSenzorUmid() {
  int idxH = lastSensorRaw.indexOf("H=");
  String hum = (idxH >= 0) ? lastSensorRaw.substring(idxH + 2, lastSensorRaw.indexOf(",", idxH)) :
"necunoscut";
```

```
  String html = "<!DOCTYPE html><html><head><meta charset='UTF-8'><title>💧 Umiditate</title>"
    "<style>body{font-family:Arial;text-align:center;background:#eef2f3;padding:20px;}"
    "h1{color:#333;}p{font-size:1.3em;}</style></head><body>"

    "<h1>💧 Umiditate</h1>"
    "<p><strong>" + hum + " %</strong></p>"
    "<br><a href='/dashboard'>Înapoi la Dashboard</a></body></html>";

  server.send(200, "text/html", html);
}

void handleSenzorGaz() {
  int idxG = lastSensorRaw.indexOf("G=");
  String gaz = (idxG >= 0) ? lastSensorRaw.substring(idxG + 2, lastSensorRaw.indexOf(",", idxG)) :
"necunoscut";

  String msg;
  if (gaz != "necunoscut") {
    int gazVal = gaz.toInt();
    if (gazVal > 300) {
      msg = "🔥 Gaz detectat!";
    } else {
      msg = "✅ Nicio scurgere de gaz.";
    }
  } else {
    msg = "❓ Status necunoscut.";
  }

  String html = "<!DOCTYPE html><html><head><meta http-equiv='refresh' content='1' charset='UTF-
8'><title>🔥 Gaz</title>"
    "<style>body{font-family:Arial;text-align:center;background:#eef2f3;padding:20px;}"
    "h1{color:#333;}p{font-size:1.3em;}</style></head><body>"
    "<h1>🔥 Gaz</h1>"
    "<p><strong>" + msg + "</strong></p>"
    "<br><a href='/dashboard'>Înapoi la Dashboard</a></body></html>";

  server.send(200, "text/html", html);
}


void handleSenzorFlacara() {
  int idxF = lastSensorRaw.indexOf("F=");
  String flacara = "necunoscut";
  if (idxF >= 0) {
    int idxVirgula = lastSensorRaw.indexOf(",", idxF);
    if (idxVirgula > idxF) {
      flacara = lastSensorRaw.substring(idxF + 2, idxVirgula);
    } else {
      flacara = lastSensorRaw.substring(idxF + 2); // Ultimul camp
    }
  }
```

```
  String msg;
  if (flacara == "DA") {
    msg = "🔥 Flacără detectată!";
  } else if (flacara == "NU") {

    msg = "✅ Nicio flacără detectată.";
  } else {
    msg = "❓ Status necunoscut.";
  }

  String html = "<!DOCTYPE html><html><head><meta http-equiv='refresh' content='1' charset='UTF-
8'><title>🔥 Flacără</title>"
    "<style>body{font-family:Arial;text-align:center;background:#eef2f3;padding:20px;}"
    "h1{color:#333;}p{font-size:1.3em;}</style></head><body>"
    "<h1>🔥 Flacără</h1>"
    "<p><strong>" + msg + "</strong></p>"
    "<br><a href='/dashboard'>Înapoi la Dashboard</a></body></html>";

  server.send(200, "text/html", html);
}

void handleSenzorDist() {
  int idxD = lastSensorRaw.indexOf("D=");
  String dist = (idxD >= 0) ? lastSensorRaw.substring(idxD + 2) : "necunoscut";

  String html = "<!DOCTYPE html><html><head><meta http-equiv='refresh' content='3' charset='UTF-
8'><title>📏 Distanță</title>"
    "<style>body{font-family:Arial;text-align:center;background:#eef2f3;padding:20px;}"
    "h1{color:#333;}p{font-size:1.3em;}</style></head><body>"
    "<h1>📏 Distanță</h1>"
    "<p><strong>" + dist + " cm</strong></p>"
    "<br><a href='/dashboard'>Înapoi la Dashboard</a></body></html>";

  server.send(200, "text/html", html);
}

void setup() {
  pinMode(BUZZER_PIN, OUTPUT);
  Serial.begin(115200);
  Serial1.begin(9600, SERIAL_8N1, RXD1, TXD1);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) { delay(300); Serial.print("."); }
  Serial.println("\n✅ Conectat! IP=" + WiFi.localIP().toString());

  server.on("/", handleRoot);
  server.on("/login", handleLogin);
  server.on("/login/keypad", handleKeypadPage);
  server.on("/login/browser", handleLoginBrowser);
  server.on("/login/browser/check", handleLoginBrowserCheck);
  server.on("/login/rfid", handleLoginRFID);
```

```
  server.on("/dashboard", handleDashboard);
  server.on("/senzor/temp", handleSenzorTemp);
  server.on("/senzor/umid", handleSenzorUmid);
  server.on("/senzor/gaz", handleSenzorGaz);
  server.on("/senzor/flacara", handleSenzorFlacara);
  server.on("/senzor/distanta", handleSenzorDist);

  server.on("/senzor/camera", handleCamera);
  server.begin();
}

void loop() {
  server.handleClient();

  while (Serial1.available()) {
    String line = Serial1.readStringUntil('\n');
    line.trim();

    if (line.startsWith("RFID:")) {
      lastRFID = line.substring(5);
      Serial.println("UID detectat complet: " + lastRFID);
      if (line.indexOf("PERMIS") >= 0) {
        loginStatus = "✅ RFID: ACCES PERMIS!";
        beep();
      } else {
        loginStatus = "❌ RFID: ACCES RESPINS!";
        beep();
      }
    } else if (line.startsWith("PIN:PERMIS")) {
      loginStatus = "✅ PIN tastatura: ACCES PERMIS!";
      keypadAccess = true;
      beep();
    } else if (line.startsWith("SENZORI:")) {
      lastSensorRaw = line.substring(8);
    }
  }
}
```

3.2. Descrierea codului pentru ESP32

Programul ESP32 funcționează ca un server web centralizat pentru un sistem de casă inteligentă. Acesta se conectează la rețeaua Wi-Fi specificată (ssid, password) și găzduiește o interfață web pe portul 80. Interfața web include o pagină principală (/) care afișează o distanță (probabil de la un senzor de proximitate) și oferă un buton de "Login" dacă utilizatorul este la o distanță mai mică de 150 cm. Pagina de login (/login) prezintă trei opțiuni de autentificare: prin tastatura fizică (/login/keypad) conectată la Arduino, prin introducerea unui PIN direct în browser (/login/browser, cod fix: "0000") sau prin scanarea unui tag RFID (/login/rfid). Odată autentificat, utilizatorul este redirecționat către "Dashboard" (/dashboard), de unde poate accesa pagini dedicate pentru monitorizarea diverșilor senzori (temperatură, umiditate, gaz, flacără, distanță). Fiecare pagină de senzor (/senzor/temp, /senzor/umid, etc.) interpretează și afișează valorile specifice primite de la Arduino. De asemenea, există o funcționalitate de vizualizare a fluxului video (/senzor/camera) de la o sursă externă (specificată ca http://192.168.51.138:80/stream), indicând integrarea unei camere IP (probabil o ESP32-CAM separată). Comunicarea cu Arduino Mega se realizează prin portul serial Serial1 (pinii RXD1 4, TXD1 5), prin care ESP32 primește statusuri RFID,

statusuri de PIN și citiri brute de la senzori. Programul utilizează un buzzer (conectat la pinul 7) pentru a oferi feedback sonor la evenimente cheie, cum ar fi tentativele de autentificare reușite sau eșuate.

3.3. Cod Arduino

```
#include <SPI.h>
#include <MFRC522.h>
#include <Keypad.h>
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define BUZZER_PIN 7

#define DHTPIN   11
#define DHTTYPE  DHT11
DHT dht(DHTPIN, DHTTYPE);

#define RST_PIN  49
#define SS_PIN   53
MFRC522 rfid(SS_PIN, RST_PIN);
const byte AUTHORIZED_UID[4] = { 0x93, 0x1A, 0xD7, 0x99 };

const byte ROWS = 4, COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = { A0, A1, A2, A3 };
byte colPins[COLS] = { A4, A5, A6, A7 };
Keypad keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

#define MQ2_PIN    A8
#define FLAME_PIN  A9
#define TRIG_PIN   9
#define ECHO_PIN   10

const String correctCode = "0000";
String enteredCode = "";

LiquidCrystal_I2C lcd(0x27, 16, 2); // Adresa I2C a LCD-ului

String lcdState = "login"; // login, rfid, dashboard

void beepOK()  { tone(BUZZER_PIN, 1000, 200); }
void beepNOK() { tone(BUZZER_PIN, 500, 150); delay(200); tone(BUZZER_PIN, 500, 150); }
void beepKey() { tone(BUZZER_PIN, 1500, 80); }

void setup() {
  pinMode(BUZZER_PIN, OUTPUT);
  Serial.begin(9600);
```

```
  Serial1.begin(9600);

  dht.begin();

  SPI.begin();
  rfid.PCD_Init();

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Sistem SmartHome");

  Serial.println("✅ Mega gata!");
}

void loop() {
  // La fiecare 5 secunde – mesaj general pe LCD
  static unsigned long lastMsg = 0;
  if (millis() - lastMsg >= 5000) {
    lastMsg = millis();
    lcd.clear();
    if (lcdState == "login") {
      lcd.setCursor(0, 0);
      lcd.print("Alege modalitate");
      lcd.setCursor(0, 1);
      lcd.print("de login");
    } else if (lcdState == "rfid") {
      lcd.setCursor(0, 0);
      lcd.print("Scaneaza RFID...");
    } else if (lcdState == "dashboard") {
      lcd.setCursor(0, 0);
      lcd.print("Alege senzor din");
      lcd.setCursor(0, 1);
      lcd.print("meniu");
    }
  }

  // RFID
  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
    lcdState = "rfid";
    Serial.print("UID tag: ");
    String uidString = "";
    for (byte i = 0; i < rfid.uid.size; i++) {
      Serial.print(rfid.uid.uidByte[i], HEX);
      Serial.print(" ");
      uidString += String(rfid.uid.uidByte[i], HEX) + " ";
    }
    Serial.println();
```

```
 bool ok = true;
  for (byte i = 0; i < 4; i++) {
    if (rfid.uid.uidByte[i] != AUTHORIZED_UID[i]) { ok = false; break; }

  }

  if (ok) {
    Serial1.println("RFID:PERMIS");
    Serial.println("RFID:PERMIS");
    beepOK();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("RFID: ACCES OK");
    delay(3000);
    lcdState = "dashboard";
  } else {
    Serial1.println("RFID:RESPINS");
    Serial.println("RFID:RESPINS");
    beepNOK();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("RFID: ACCES NO");
    delay(3000);
    lcdState = "login";
  }

  rfid.PICC_HaltA();
  delay(300);
}

// Tastatura
char key = keypad.getKey();
if (key) {
  beepKey();
  if (key == '#') {
    if (enteredCode == correctCode) {
      Serial1.println("PIN:PERMIS");
      Serial.println("PIN:PERMIS");
      beepOK();
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("PIN: ACCES OK");
      delay(3000);
      lcdState = "dashboard";
    } else {
      Serial1.println("PIN:RESPINS");
      Serial.println("PIN:RESPINS");
      beepNOK();
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("PIN: ACCES NO");
      delay(3000);
      lcdState = "login";
```

```
      }
      enteredCode = "";
    }
    else if (key == '*') {

      enteredCode = "";
    }
    else {
      if (enteredCode.length() < 4) {
        enteredCode += key;
      }
    }
    delay(150);
  }

  // Senzori la 5 secunde
  static unsigned long lastSensor = 0;
  if (millis() - lastSensor >= 5000) {
    lastSensor = millis();
    float T = dht.readTemperature(), H = dht.readHumidity();
    if (isnan(T)||isnan(H)) { T = H = -1; }
    int G = analogRead(MQ2_PIN);
    int F = analogRead(FLAME_PIN);

    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    long dur = pulseIn(ECHO_PIN, HIGH);
    int dist = (dur/2) / 29.1;

    String s = "SENZORI:T=" + String(T)
             + ",H=" + String(H)
             + ",G=" + String(G)
             + ",F=" + String(F)
             + ",D=" + String(dist);
    Serial1.println(s);
    Serial.println(s);
  }
}
```

3.4. Descrierea codului pentru Arduino Mega

Programul Arduino Mega este inima sistemului, gestionând direct senzorii și autentificarea locală. Acesta citește date de la senzori precum DHT11 (temperatură/umiditate), MQ-2 (gaz), senzorul de flacără, și HC-SR04 (distanță). De asemenea, gestionează autentificarea prin tastatură matricială (PIN "0000") și prin RFID (comparând UID-uri cu unul predefinit). Statusurile de acces și valorile senzorilor sunt afișate pe un ecran LCD I2C și transmise continuu către modulul ESP32 prin comunicație serială (Serial1). Un buzzer oferă feedback sonor pentru diverse evenimente, inclusiv succesul sau eșecul autentificărilor.

4. Biblioteci necesare

Arduino Mega (RFID, tastatură, senzori, LCD)
- **SPI.h**: permite comunicarea cu module SPI, cum ar fi modulul RFID MFRC522
- **MFRC522.h**: gestionează citirea tag-urilor RFID de la modulul MFRC522 pentru autentificare
- **Keypad.h**: facilitează citirea intrărilor de pe tastatura matricială, utilizată pentru codul PIN
- **DHT.h**: citește valorile de temperatură și umiditate de la senzorul DHT11
- **Wire.h**: biblioteca I2C, esențială pentru comunicarea cu ecranul LCD
- **LiquidCrystal_I2C.h**: controlează afișajul text pe ecranul LCD prin interfața I.

ESP32 (Server web, Serial1, WiFi, tonuri)
- **WiFi.h**: permite conectarea ESP32 la rețeaua Wi-Fi și accesul la internet
- **WebServer.h**: transformă ESP32 într-un server web, gestionând paginile HTML și interfața utilizatorului.

5. Modul de funcționare

1. Pagina de access
Aplicația poate fi accesată doar dacă senzorul de distanță detectează pe cineva la mai puțin de 150cm.

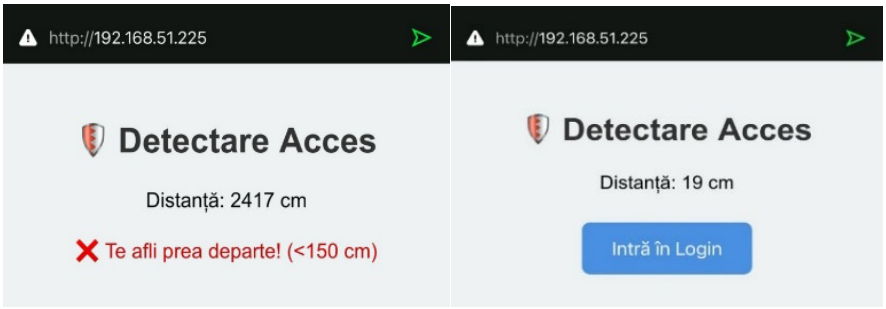

Figura 2. Detectarea distanței față de persoana din fața ușii

Figura 3. Prezentarea generală a casei

2. Pagina de login
Pagina de login oferă utilizatorului 3 opțiuni prin care acesta poate primi acces:
- tastatură (keypad)
- direct din browser
- RFID

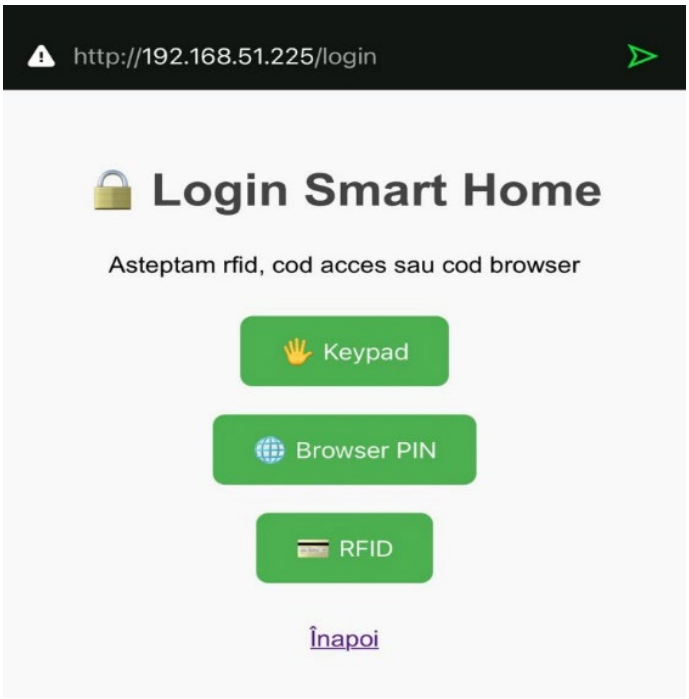

Figura 4. Pagina de login

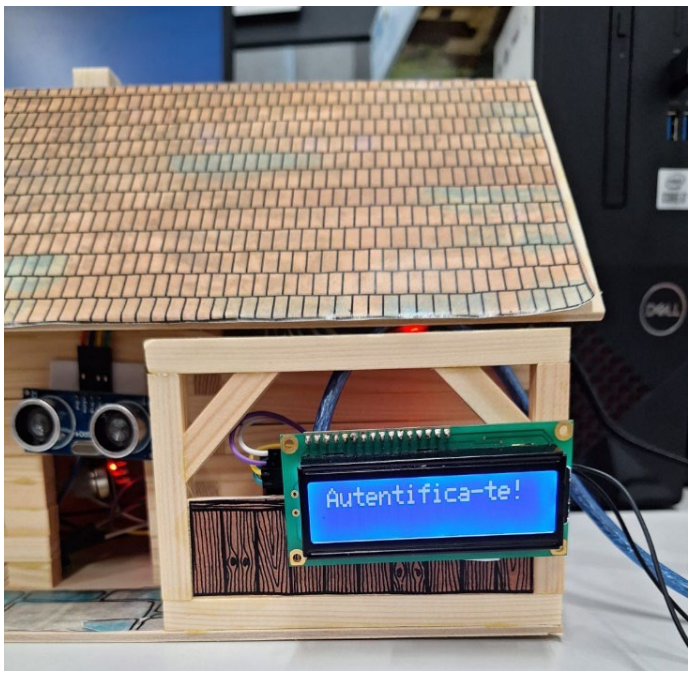

Figura 5. Afișarea mesajului de întâmpinare

Figura 6. Alegerea modalității de acces

Dacă utilizatorul alege modalitatea de acces prin keypad, acesta trebuie să introducă codul corect de la tastatură. Odată ce a făcut acest lucru, pe ecranul LCD este afișat mesajul că accesul a fost permis. De asemenea, în pagina de web va fi afișat un mesaj de "Acces permis", apoi după o perioadă de 3 secunde pagina se va reîncărca.

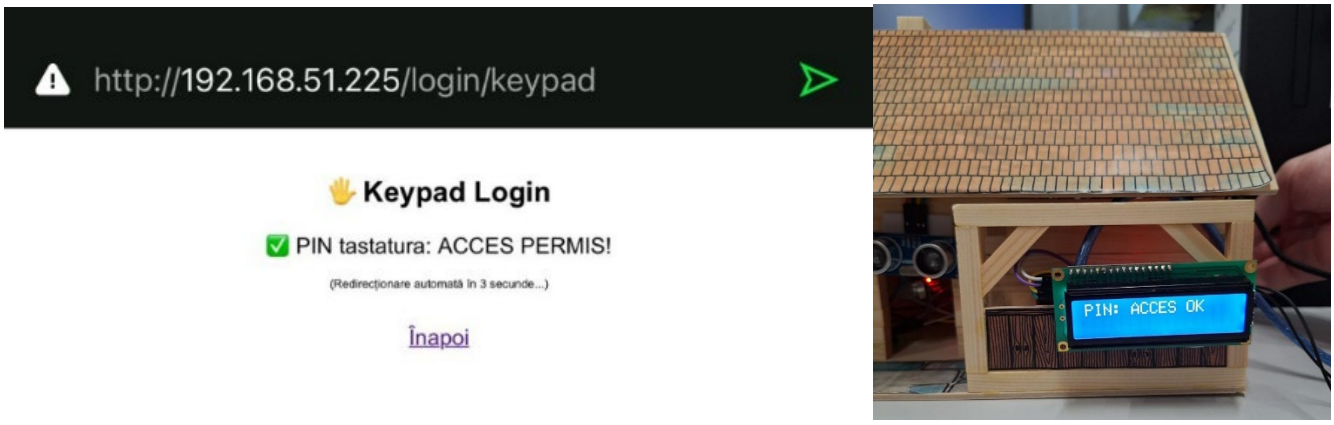

Figura 7. Modalitatea de login prin keypad

În cazul în care utilizatorul alege modalitatea de acces prin RFID, odată ce acesta a scanat tag-ul, pe ecranul LCD va fi afișat mesajul de "Acces permis". Dacă acesta va scana o cartelă de acces ce nu are codul inclus în script, acesta nu va primi accesul.

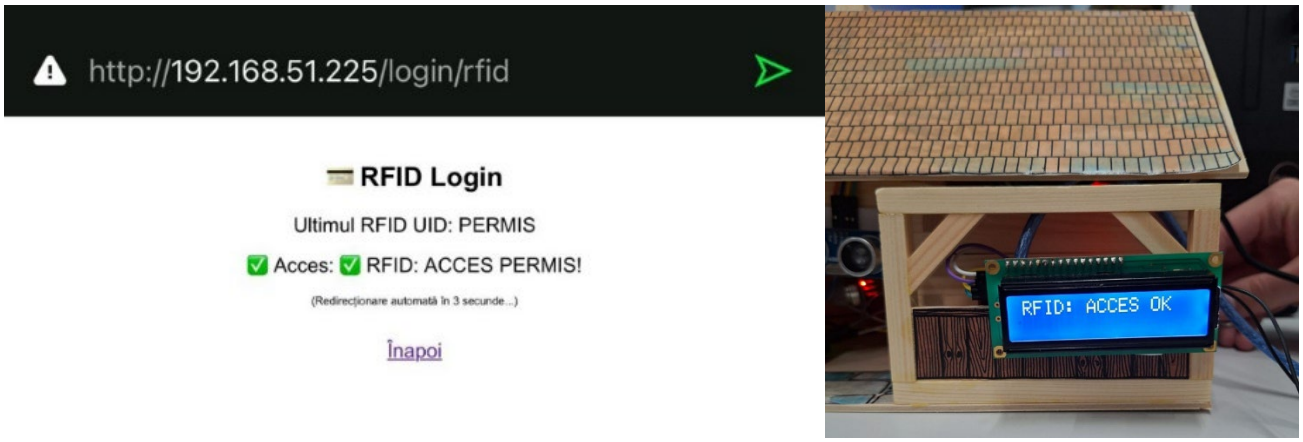

Figura 8. Modalitatea de login prin RFID

3. Dashboard

După ce utilizatorul a primit accesul, un dashboard cu senzori va fi afișat, de unde acesta poate selecta butoanele pentru a vizualiza datele fiecărui senzor. În tot acest timp, pe ecranul LCD vor fi afișate valorile citite de senzori din casă.

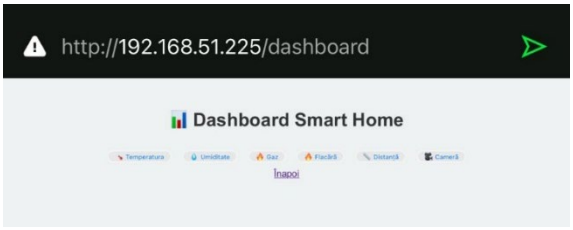

Figura 9. Panoul de control al casei

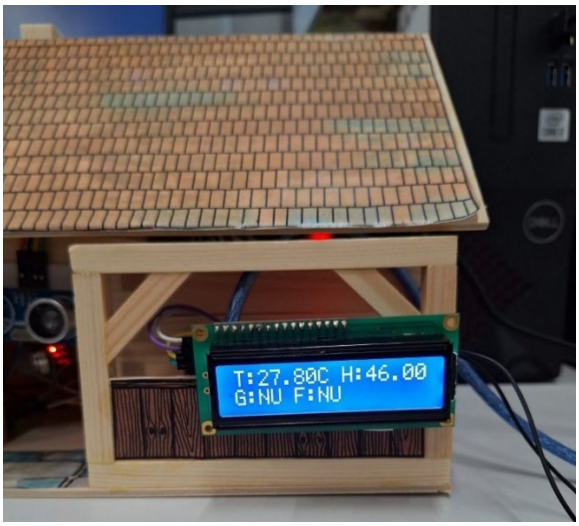

Figura 10. Afișarea valorilor citite de senzori

4. Temperatură

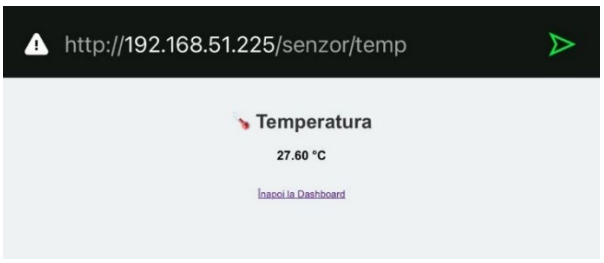

Figura 11. Afișarea valorii temperaturii

5. Gaz

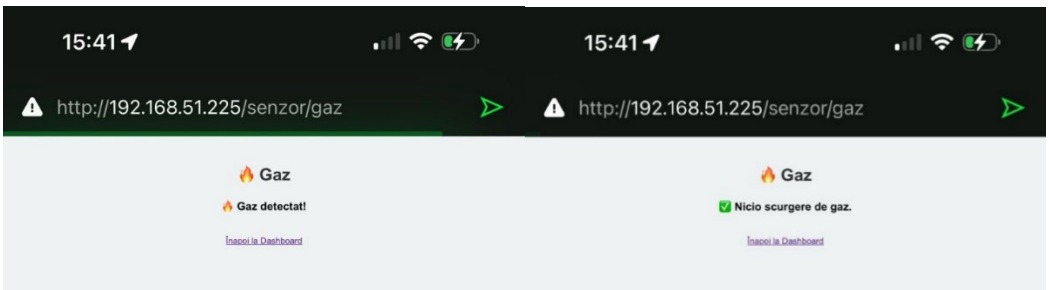

Figura 12. Stările senzorului de gaz

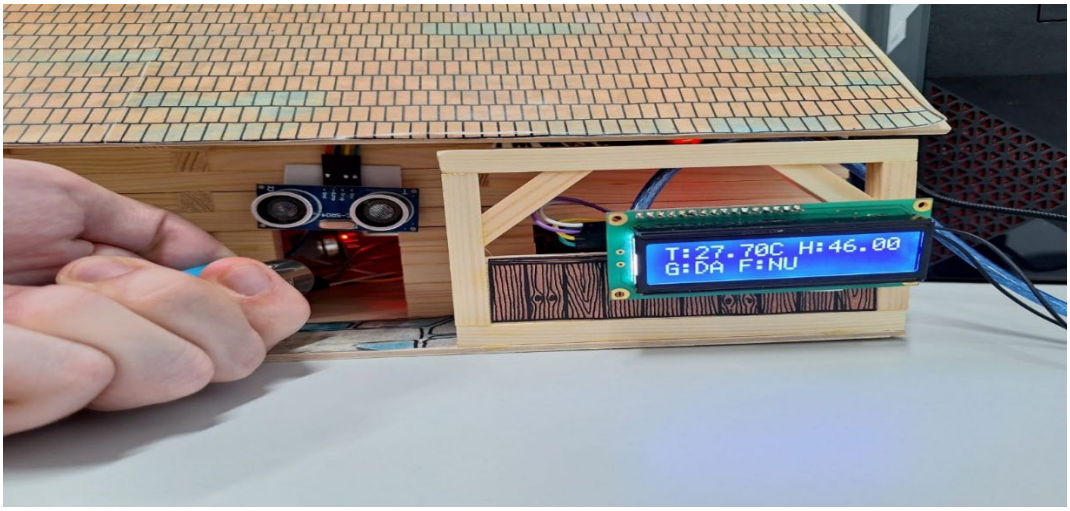

Figura 13. Exemplificarea funcționării senzorului de gaz, G: DA

6. Umiditate

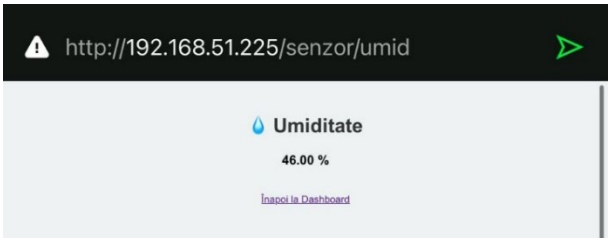

Figura 14. Afișarea valorii citite de senzorul de umiditate

7. Flacără

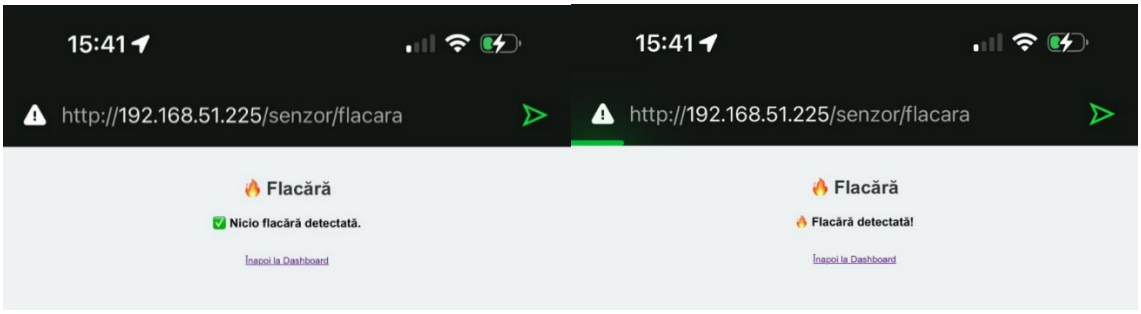

Figura 15. Stările de funcționare ale senzorului de flacără

Figura 16. Exemplificarea funcționării senzorului de flacără, F: DA

8. Camera video

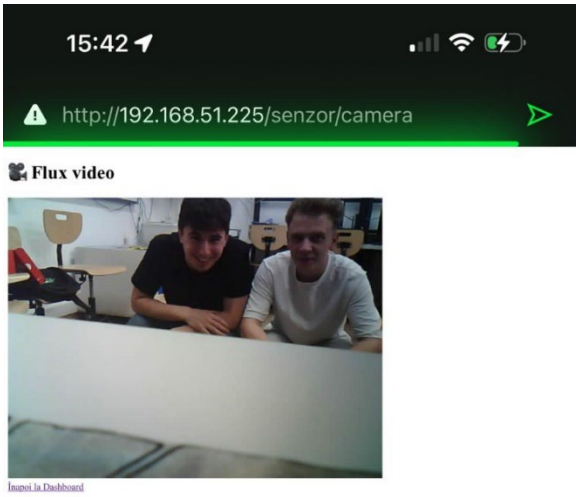

Figura 17. Exemplificarea modului de funcționare a camerei video

6. Concluzie

Finalizarea proiectului a dus la dezvoltarea unui sistem de casă inteligentă complex și funcțional, integrând eficient Arduino Mega și ESP32 pentru monitorizare, securitate și control la distanță. Prin combinarea colectării datelor și autentificării locale (Arduino) cu funcționalitatea de server web și conectivitatea Wi-Fi (ESP32), sistemul oferă o soluție robustă pentru automatizarea unei locuințe. Interfața web intuitivă permite interacțiunea de la distanță, vizualizarea datelor de la senzori și streaming video. Metodele multiple de autentificare și monitorizarea detaliată a factorilor de mediu subliniază caracterul complet al soluției. Astfel, proiectul atinge scopul de a crea un sistem ce îmbunătățește confortul și securitatea prin utilizarea inteligentă a componentelor și interconectării microcontrolerelor.

Bibliografie

1. https://www.arduino.cc/reference/en/
2. https://docs.espressif.com/
3. https://github.com/adafruit/DHT-sensor-library
4. https://github.com/miguelbalboa/rfid
5. https://github.com/Chris–A/Keypad
6. https://github.com/johnrickman/LiquidCrystal_I2C
7. https://www.youtube.com/@RandomNerdTutorials
8. https://www.youtube.com/@DroneBotWorkshop
9. https://www.youtube.com/@GreatScottLab