Universitatea "POLITEHNICA" Timişoara

Departamentul de ID/IFR și educație digitală (DeL)

Informatică

Proiect sincretic

Sistem inteligent de blocare a ușilor

Student: Bănățean Alexandru-Ioan

Anul 2 Info ID

Timișoara, 2026

Cuprins

1. Descrierea proiectului

Proiectul propune implementarea unui sistem inteligent de blocare a ușilor, controlat de la distanță și integrat într-o arhitectură IoT bazată pe protocolul MQTT.
Sistemul permite:

- Controlul închiderii și deschiderii unei încuietori electrice
- Autentificare locală prin tastatură, RFID sau cod PIN
- Monitorizarea stării ușii (închisă/deschisă)
- Notificări în timp real către utilizator și/sau aplicație
- Arhivarea evenimentelor într-o bază de date.

Acest tip de sistem este util în clădiri rezidențiale, birouri, depozite sau spații industriale unde accesul trebuie controlat și monitorizat.

2. Analiza SWOT

Strenghts (puncte forte)

- Control de la distanță în timp real prin MQTT
- Monitorizare continuă a accesului.
- Cost redus de implementare (ESP32, servo/solenoid)
- Scalabilitate ridicată — pot fi adăugate mai multe uși/senzori

Weaknesses (puncte slabe)

- Necesită conexiune WiFi stabilă
- Dependență de securitatea implementării MQTT
- Consum de energie crescut dacă se folosește un electromagnet continuu

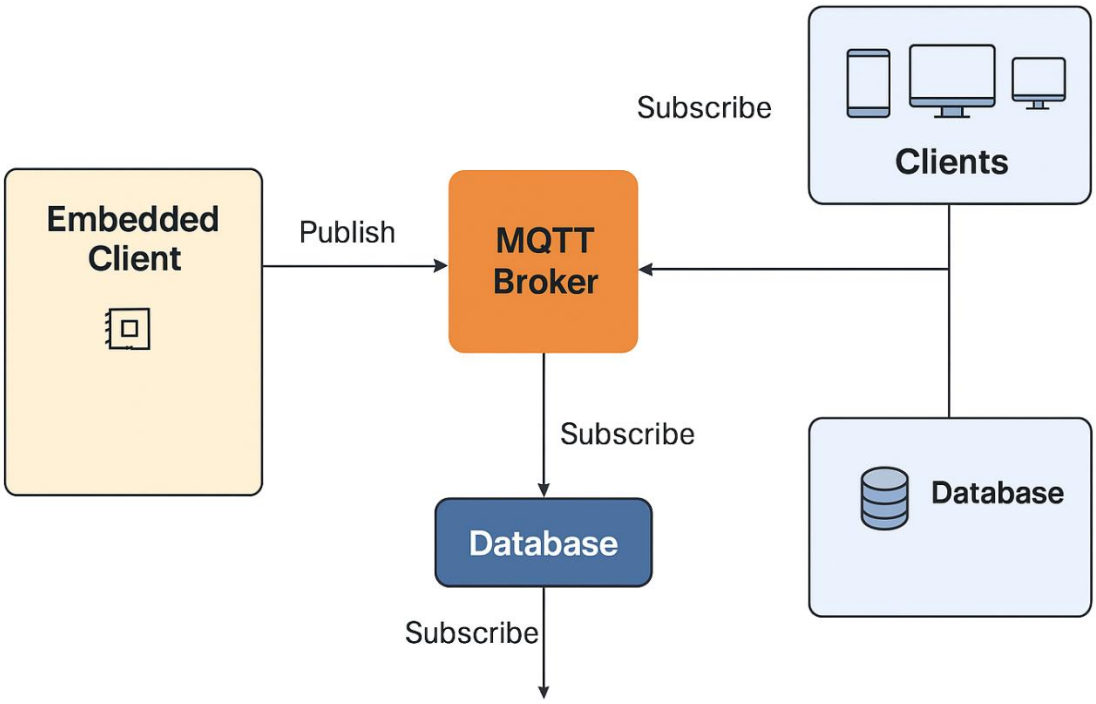
Opportunities (oportunități)

- Integrare cu aplicații mobile, sisteme de alarmă sau camere video
- Extindere către sisteme smart home complete
- Implementare autentificare multi-factor

Threats (amenințări)

- Atacuri cibernetice dacă MQTT nu este securizat (SSL/TLS)
- Defecțiuni hardware (servo, solenoid)
- Probleme de alimentare electrică

3. Diagrama arhitecturală/descrierea elementelor



🔹 **Client MQTT embedded**

Dispozitiv IoT montat pe ușă:

- Microcontroler: **ESP32** / ESP8266
- Senzori:
    - Senzor de poziție ușă (reed switch)
    - RFID (opțional)
    - Tastatură numerică (opțional)
- Actuator:
    - Servo motor/electromagnet (solenoid)

- Funcții:
    - Trimite starea ușii (door/status)
    - Primește comenzi de la broker (door/command)
    - Publică evenimente de acces (door/event)

🔹 **Broker MQTT**

Exemple:

- Mosquitto (local)
- HiveMQ Cloud

🔹 **Client(ți) receptori**

Poate fi:

- o aplicație mobilă
- un dashboard web
- un PC cu un client MQTT (MQTT Explorer etc.)

Funcții:

- Primesc notificări
- Trimit comenzi de blocare/deblocare
- Vizualizează istoricul accesărilor

🔹 **Baza de date**

Tip: MySQL / SQLite / Firebase
Tabele recomandate:

1. **users** – utilizatori autorizați
2. **events** – log de acces
3. **door_status** – starea ușii în timp real

4. Descrierea funcțiilor majore

Client embedded (ESP32)

- Citește senzorul de poziție a ușii
- Inițiază conexiunea WiFi + MQTT
- Primește comenzi pentru acționarea încuietorii
- Trimite periodic starea (online/offline, open/closed)
- Trimite evenimente: acces aprobat, acces respins, eroare actuator.

Broker MQTT

- Rutează mesaje între clientul embedded și clienții receptori
- Gestionează topicurile create conform structurii ierarhice
- Oferă QoS (Quality of Service) pentru livrarea sigură a mesajelor.

Clientul receptor

- Trimite comanda de „lock/unlock”
- Afișează starea ușii
- Stochează datele în DB
- Generează alerte/mesaje către utilizator.

5. Diagrama bazei de date + descriere

Tabele recomandate

1. users - Reține utilizatorii autorizați să deschidă ușa.

id	name	rfid_code	pin	role
1	Alexandru	23C21G	7468	admin

2. events - Evenimente de acces, blocare/deblocare, încercări eșuate.

id	user_id	event_type	timestamp
31	1	unlocked	2025-12-10 17:03

3. door_status - Menține ultima stare a ușii.

id	status	updated_at
1	closed	2025-12-10 18:30

6. Concluzii

Sistemul inteligent de blocare a ușilor oferă o soluție modernă, sigură și scalabilă pentru controlul accesului cu ajutorul tehnologiilor IoT.
Utilizarea MQTT asigură un timp de răspuns rapid, iar integrarea cu o bază de date permite monitorizarea accesărilor. Sistemul poate fi extins ușor cu funcții suplimentare precum: camere video, geofencing, autentificare biometrică sau aplicații mobile dedicate.

7. Bibliografie

- MQTT Topics, Wildcards, & Best Practices
- Wokwi for VS Code
- Building a Smart Door Lock System with ESP32
- RFID Security Door Lock Using ESP32

8. Cod sursă EN (github)

SmartDoorLock

9. Cod sursă RO

```
/*
 Sistem inteligent de blocare a ușii – ESP32 + MQTT

 Acest proiect implementează un sistem de tip Smart Door Lock folosind:
 - microcontroler ESP32
 - protocolul MQTT (publish / subscribe)
 - mesaje JSON (biblioteca ArduinoJson)

 Funcționalități principale:
 - conectare la rețeaua WiFi
 - conectare la un broker MQTT
 - primirea comenzilor de blocare / deblocare
 - controlul mecanismului de blocare (releu / solenoid / servo)
 - monitorizarea stării ușii (deschis / închis)
 - publicarea stării și a evenimentelor sub formă de mesaje JSON

 NOTĂ:
 - Aceasta este implementarea de bază (fără RFID, keypad etc.)
 - Codul este structurat pentru a putea fi extins ușor
*/

#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
```

```
// ====================================================
// CONFIGURARE REȚEA WIFI
// ====================================================
// AICI trebuie introduse datele rețelei WiFi
const char* WIFI_SSID     = "YOUR_WIFI_SSID";      // <-- numele rețelei WiFi
const char* WIFI_PASSWORD = "YOUR_WIFI_PASSWORD";  // <-- parola WiFi

// ====================================================
// CONFIGURARE MQTT
// ====================================================
// Broker MQTT public (pentru testare)
// Poate fi înlocuit cu Mosquitto local sau HiveMQ Cloud
const char* MQTT_BROKER = "test.mosquitto.org";
const int   MQTT_PORT   = 1883;

// ID unic pentru clientul MQTT
const char* MQTT_CLIENT_ID = "door1-esp32";

// Topic-uri MQTT utilizate în sistem
const char* TOPIC_CMD    = "home/door1/cmd";    // primește comenzi (lock / unlock)
const char* TOPIC_STATUS = "home/door1/status";  // publică starea ușii
const char* TOPIC_EVENT  = "home/door1/event";   // publică evenimente

// ====================================================
// CONFIGURARE PINI HARDWARE
```

```
// ================================================
const int PIN_LOCK = 23;   // actuator blocare (releu / solenoid)
                    // LOW  = ușa blocată
                    // HIGH = ușa deblocată

const int PIN_REED = 22;   // senzor poziție ușă (reed switch)
                    // HIGH = ușa închisă
                    // LOW  = ușa deschisă

// ================================================
// OBIECTE GLOBALE
// ================================================
WiFiClient espClient;
PubSubClient mqttClient(espClient);

// ================================================
// VARIABILE DE STARE
// ================================================
bool lastDoorClosed = false;              // starea anterioară a ușii
unsigned long lastStatusPublish = 0;      // momentul ultimei publicări
const unsigned long STATUS_INTERVAL = 5000;  // publicare periodică (5 secunde)

// ================================================
// FUNCȚIE DE CONECTARE LA WIFI
// ================================================
void connectWiFi() {
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  // Așteaptă până când ESP32 se conectează la WiFi
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}

// ====================================================
// FUNCȚIE GENERICĂ PENTRU PUBLICARE JSON
// ====================================================
// Publică un obiect JSON pe un topic MQTT
void publishJson(const char* topic, JsonDocument& doc, bool retained = false) {
  char buffer[128];  // buffer pentru mesajul JSON
  size_t length = serializeJson(doc, buffer);
  mqttClient.publish(topic, (uint8_t*)buffer, length, retained);
}

// ====================================================
// PUBLICARE STARE UȘĂ (JSON)
// ====================================================
void publishStatus(bool closed) {
  JsonDocument doc;
  doc["device"] = "door1";
  doc["door"] = closed ? "closed" : "open";
  doc["timestamp"] = millis();
```

```
  publishJson(TOPIC_STATUS, doc, true); // mesaj reținut (retained)
}

// ====================================================
// PUBLICARE EVENIMENT SISTEM (JSON)
// ====================================================
void publishEvent(const char* eventType) {
  JsonDocument doc;
  doc["device"] = "door1";
  doc["event"] = eventType;
  doc["timestamp"] = millis();

  publishJson(TOPIC_EVENT, doc);
}

// ====================================================
// CALLBACK MQTT – PROCESARE COMENZI
// ====================================================
// Această funcție este apelată la primirea unui mesaj MQTT
void mqttCallback(char* topic, byte* payload, unsigned int length) {
  String message;

  // Conversie payload (bytes) în String
  for (unsigned int i = 0; i < length; i++) {
    message += (char)payload[i];
```

```
  }
  message.trim();

  // Procesare comenzi
  if (message.equalsIgnoreCase("lock")) {
    digitalWrite(PIN_LOCK, LOW);   // blochează ușa
    publishEvent("lock_command_received");
  }
  else if (message.equalsIgnoreCase("unlock")) {
    digitalWrite(PIN_LOCK, HIGH);  // deblochează ușa
    publishEvent("unlock_command_received");
  }
}

// ======================================================
// FUNCȚIE DE CONECTARE LA BROKER MQTT
// ======================================================
void connectMQTT() {
  while (!mqttClient.connected()) {
    if (mqttClient.connect(MQTT_CLIENT_ID)) {
      mqttClient.subscribe(TOPIC_CMD);   // abonare la topic-ul de comenzi
      publishEvent("mqtt_connected");
    } else {
      delay(2000); // reîncearcă după 2 secunde
    }
  }
```

```
}

// ======================================================
// SETUP – RULEAZĂ O SINGURĂ DATĂ
// ======================================================
void setup() {
  // Configurare pini
  pinMode(PIN_LOCK, OUTPUT);
  pinMode(PIN_REED, INPUT);

  // Stare inițială: ușa blocată
  digitalWrite(PIN_LOCK, LOW);

  // Conectare la WiFi
  connectWiFi();

  // Configurare MQTT
  mqttClient.setServer(MQTT_BROKER, MQTT_PORT);
  mqttClient.setCallback(mqttCallback);
}

// ======================================================
// LOOP – RULEAZĂ CONTINUU
// ======================================================
void loop() {
  // Asigură conexiunea MQTT
```

```
 if (!mqttClient.connected()) {
  connectMQTT();
 }
 mqttClient.loop();

 // Citește starea ușii
 bool doorClosed = (digitalRead(PIN_REED) == HIGH);

 // Publică starea dacă s-a schimbat
 if (doorClosed != lastDoorClosed) {
  publishStatus(doorClosed);
  lastDoorClosed = doorClosed;
 }

 // Publică starea periodic (heartbeat)
 if (millis() - lastStatusPublish > STATUS_INTERVAL) {
  publishStatus(doorClosed);
  lastStatusPublish = millis();
 }
}
```