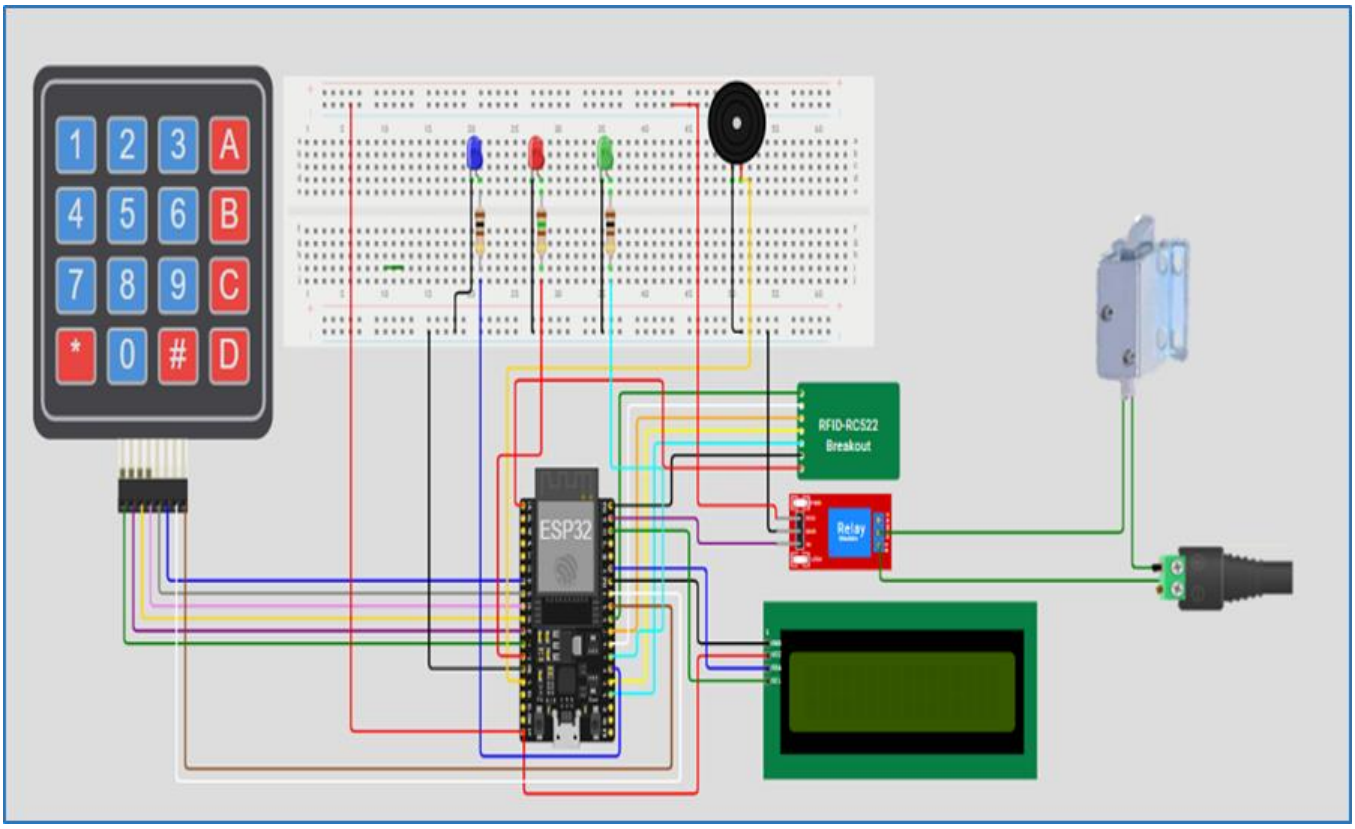
Simulation du système



Programme de la cart esp32

```
#define BLYNK_TEMPLATE_ID "TMPL5TRtHte5O"
#define BLYNK_TEMPLATE_NAME "ON in OFF LED"
#define BLYNK_AUTH_TOKEN "KuMkEkWSUO1tg7IX1eo1KNh7cs4yfFBL"

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <SPI.h>
#include <MFRC522.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <HTTPClient.h>
#include <time.h>
#include <FS.h>
#include <SPIFFS.h>

#define RELAY_PIN 23
#define LED_GREEN 4
#define LED_RED 12
#define LED_BLUE 0
#define BUZZER_PIN 13
#define SS_PIN 5
#define RST_PIN 15

// Google Sheets script URL
const char* googleScriptUrl =
"https://script.google.com/macros/s/AKfycbw2ut88qPf8a_DFcOIMsDr-
49l2hkJdeCOaRQKB6OgKxS6MfdWt25fjTKMyvV6fJWKj/exec";

MFRC522 rfid(SS_PIN, RST_PIN);

String rfids[] = {"F32B729A", "53CFF5DA", "12121212", "34343434"};
String rfidUsers[] = {"NOUREDDINE_RFID", "HIND_RFID", "MOHAMMED_RFID",
"OMAIMA_RFID"};

String pinCodes[] = {"1111", "5555", "8888", "0000","2222"};
String pinUsers[] = {"Sabir", "Mohamad", "Hind", "Omaima","PIN_USR"};

LiquidCrystal_I2C lcd(0x27, 16, 2);

const byte ROWS = 4;
```

```
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {14, 27, 26, 25};
byte colPins[COLS] = {33, 32, 19, 18};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

String enteredCode = "";

int failedAttempts = 0;
bool locked = false;
unsigned long lockTime = 0;

unsigned long ledBuzzerStartTime = 0;
unsigned long relayStartTime = 0;
bool ledBuzzerOn = false;
bool relayOn = false;

String pendingLogName = "";
bool shouldLogAccess = false;

const char* ssid_ap = "ESP32-LOCK";
const char* password_ap = "12345678";
int previousClientCount = 0;

const char* ssid = "SABIR";
const char* password = "12345678";

unsigned long lastWifiConnectionAttempt = 0;
const unsigned long wifiReconnectInterval = 10000;
bool wifiConnected = false;

void openDoor(String name);
void checkCode(String code);
void checkKeypad();
void checkRFID();
void lockSystem();
void unlockSystem();
void handleRelayTiming();
void resetDisplay();
void centerPrint(String text, int line);
void manageWiFiConnection();
void logAccess(String name);
void storeOfflineAccess(String name);
```

```
void sendStoredAccesses();
String getFormattedTime();

BLYNK_WRITE(V0) {
  if (locked) return;
  int pinValue = param.asInt();
  if (pinValue == 1) {
    openDoor("Blynk_User");
  }
}

void setup() {
  Serial.begin(115200);
  if(!SPIFFS.begin(true)){
    Serial.println("An Error has occurred while mounting SPIFFS");
  }
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_BLUE, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(0, OUTPUT);
  digitalWrite(RELAY_PIN, HIGH);
  digitalWrite(LED_GREEN, LOW);
  digitalWrite(LED_RED, LOW);
  digitalWrite(LED_BLUE, LOW);
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(0, LOW);
  lcd.init();
  lcd.backlight();
  resetDisplay();
  SPI.begin(16, 2, 17, 5);
  rfid.PCD_Init();
  configTime(0, 0, "pool.ntp.org", "time.nist.gov");
  WiFi.mode(WIFI_AP_STA);
  WiFi.softAP(ssid_ap, password_ap);
  Serial.print("AP IP address: ");
  Serial.println(WiFi.softAPIP());
  WiFi.begin(ssid, password);
  lastWifiConnectionAttempt = millis();
}

void loop() {
  if (locked) {
    if (millis() - lockTime >= 30000) {
      unlockSystem();
    } else {
      manageWiFiConnection();
      if (wifiConnected) {
```

```
        Blynk.run();
      }
      return;
    }
  }
  checkKeypad();
  checkRFID();
  int currentClientCount = WiFi.softAPgetStationNum();
  if (currentClientCount > previousClientCount) {
    Serial.println("WiFi client connected - open door ");
    openDoor("WiFi_User");
  }
  previousClientCount = currentClientCount;
  handleRelayTiming();
  manageWiFiConnection();
  if (wifiConnected) {
    Blynk.run();
  }
}

void manageWiFiConnection() {
  if (WiFi.status() != WL_CONNECTED) {
    wifiConnected = false;
    if (millis() - lastWifiConnectionAttempt >= wifiReconnectInterval) {
      Serial.println("Attempting to reconnect WiFi...");
      WiFi.disconnect();
      WiFi.begin(ssid, password);
      lastWifiConnectionAttempt = millis();
    }
  } else {
    if (!wifiConnected) {
      wifiConnected = true;
      Serial.println("WiFi connected");
      Blynk.config(BLYNK_AUTH_TOKEN);
      Blynk.connect();
      sendStoredAccesses();
    }
  }
}

void centerPrint(String text, int line) {
  lcd.setCursor(0, line);
  lcd.print("                ");
  lcd.setCursor((16 - text.length()) / 2, line);
  lcd.print(text);
}

void resetDisplay() {
  lcd.clear();
```

```
  lcd.setCursor(0, 0);
  lcd.print("Entrez le code");
  lcd.setCursor(0, 1);
  lcd.print("Glissez la cart");
}

void checkKeypad() {
  if (locked) return;
  char key = keypad.getKey();
  if (!key) return;
  if (key >= '0' && key <= '9') {
    if (enteredCode.length() < 4) {
      enteredCode += key;
      lcd.setCursor(0, 1);
      lcd.print(enteredCode + "    ");
    }
  } else if (key == '#') {
    if (enteredCode.length() == 4) checkCode(enteredCode);
    enteredCode = "";
  } else if (key == '*') {
    if (enteredCode.length() > 0) {
      enteredCode.remove(enteredCode.length() - 1);
      lcd.setCursor(0, 1);
      lcd.print(enteredCode + "    ");
    }
  }
}

void checkRFID() {
  if (locked) return;
  if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial()) return;
  String uid = "";
  for (byte i = 0; i < rfid.uid.size; i++) {
    char buffer[3];
    sprintf(buffer, "%02X", rfid.uid.uidByte[i]);
    uid += buffer;
  }
  Serial.print("RFID UID: ");
  Serial.println(uid);
  for (int i = 0; i < sizeof(rfids) / sizeof(rfids[0]); i++) {
    if (uid == rfids[i]) {
      openDoor(rfidUsers[i]);
      rfid.PICC_HaltA();
      rfid.PCD_StopCrypto1();
      return;
    }
  }
  lcd.clear();
  centerPrint("RFID inconnu", 0);
```

```
  digitalWrite(LED_RED, HIGH);
  digitalWrite(BUZZER_PIN, HIGH);
  delay(1000);
  digitalWrite(LED_RED, LOW);
  digitalWrite(BUZZER_PIN, LOW);
  resetDisplay();
  failedAttempts++;
  if (failedAttempts >= 5) {
    lockSystem();
  }
  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();
}

void checkCode(String code) {
  for (int i = 0; i < sizeof(pinCodes)/sizeof(pinCodes[0]); i++) {
    if (code == pinCodes[i]) {
      openDoor(pinUsers[i]);
      failedAttempts = 0;
      return;
    }
  }
  failedAttempts++;
  digitalWrite(LED_RED, HIGH);
  digitalWrite(BUZZER_PIN, HIGH);
  lcd.clear();
  centerPrint("Code", 0);
  centerPrint("incorrect", 1);
  delay(1000);
  digitalWrite(LED_RED, LOW);
  digitalWrite(BUZZER_PIN, LOW);
  resetDisplay();
  if (failedAttempts >= 5) {
    lockSystem();
  }
}

void openDoor(String name) {
  lcd.clear();
  centerPrint("Bienvenue", 0);
  centerPrint(name, 1);
  digitalWrite(LED_GREEN, HIGH);
  digitalWrite(BUZZER_PIN, HIGH);
  ledBuzzerStartTime = millis();
  ledBuzzerOn = true;
  pendingLogName = name;
  shouldLogAccess = true;
  Blynk.virtualWrite(V0, 1);
}
```

```
void lockSystem() {
  lcd.clear();
  centerPrint("Systeme bloque", 0);
  centerPrint("30 secondes", 1);
  digitalWrite(LED_BLUE, HIGH);
  digitalWrite(0, HIGH);
  lockTime = millis();
  locked = true;
  failedAttempts = 0;
  if (wifiConnected) {
    Blynk.virtualWrite(V0, 0);
  }
}

void unlockSystem() {
  locked = false;
  digitalWrite(LED_BLUE, LOW);
  digitalWrite(0, LOW);
  resetDisplay();
}

void handleRelayTiming() {
  if (ledBuzzerOn && millis() - ledBuzzerStartTime >= 1000) {
    digitalWrite(LED_GREEN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
    ledBuzzerOn = false;
    digitalWrite(RELAY_PIN, LOW);
    relayStartTime = millis();
    relayOn = true;
    if (shouldLogAccess) {
      if (wifiConnected) {
        logAccess(pendingLogName);
      } else {
        storeOfflineAccess(pendingLogName);
      }
      shouldLogAccess = false;
    }
  }
  if (relayOn && millis() - relayStartTime >= 3000) {
    digitalWrite(RELAY_PIN, HIGH);
    relayOn = false;
    resetDisplay();
    if (wifiConnected) {
      Blynk.virtualWrite(V0, 0);
    }
  }
}
```

```
void logAccess(String name) {
  String timeStr = getFormattedTime();
  if (timeStr == "") timeStr = "No Time";
  String jsonData = "{\"name\":\"" + name + "\", \"time\":\"" + timeStr +
"\"}";
  HTTPClient http;
  http.begin(googleScriptUrl);
  http.addHeader("Content-Type", "application/json");
  int httpResponseCode = http.POST(jsonData);
  if (httpResponseCode <= 0) {
    Serial.print("Error sending to Google Sheets: ");
    Serial.println(httpResponseCode);
    storeOfflineAccess(name);
  }
  http.end();
}

void storeOfflineAccess(String name) {
  File file = SPIFFS.open("/access_log.txt", FILE_APPEND);
  if (file) {
    file.println(name + "," + String(millis()));
    file.close();
  }
}

void sendStoredAccesses() {
  if (!SPIFFS.exists("/access_log.txt")) return;
  File file = SPIFFS.open("/access_log.txt", FILE_READ);
  while (file.available()) {
    String line = file.readStringUntil('\n');
    int commaIndex = line.indexOf(',');
    if (commaIndex == -1) continue;
    String name = line.substring(0, commaIndex);
    String timeStr = "Offline-" + line.substring(commaIndex + 1);
    String jsonData = "{\"name\":\"" + name + "\", \"time\":\"" + timeStr +
"\"}";
    HTTPClient http;
    http.begin(googleScriptUrl);
    http.addHeader("Content-Type", "application/json");
    if (http.POST(jsonData) > 0) {
      delay(100);
    } else {
      break;
    }
    http.end();
  }
  file.close();
  SPIFFS.remove("/access_log.txt");
}
```

```
String getFormattedTime() {
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    return "";
  }
  char timeString[20];
  strftime(timeString, sizeof(timeString), "%Y-%m-%d %H:%M:%S", &timeinfo);
  return String(timeString);
}
```