Activity 13. Smart House

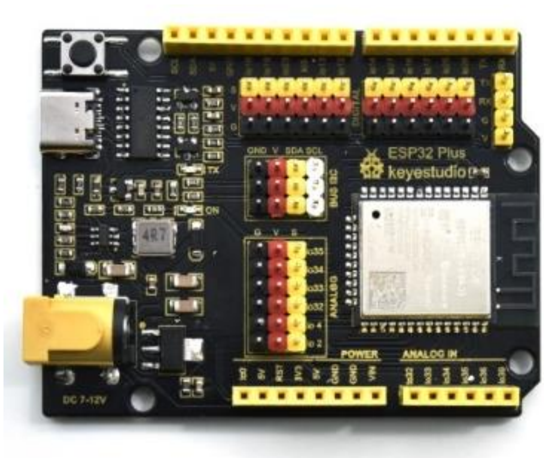
Project carried out and documented by: Dumitrescu Dragoș-George, grade XI-a D, mathematics-computer science specialization, "Constantin Cantacuzino" National Pedagogical College Târgoviște.

1. Introduction

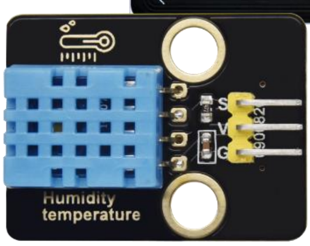
- This project aims to show the need for automation in people's daily lives.
- Smart House has as its main objective the automation of our home and as secondary objectives, the better understanding of the variables that affect our lives and how we can change the way we live in a simple and fast way.

2. Required components

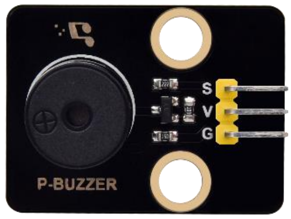
- ESP32 Plus Development Board WiFi + Bluethoot
- Electronic components used: RGB led module, 2 button modules, RFID module, piezo buzzer module, DC motor, yellow led, 2 servo motors, I2C LCD module
- Sensors: XHT11 temperature and humidity sensor, PIR motion sensor, steam sensor, gas sensor.
- The ESP32 board created specifically for this project helps us with many IO ports both digital and analog, but also through WiFi and Bluetooth connectivity through which we can control various functions of the house by phone or computer.



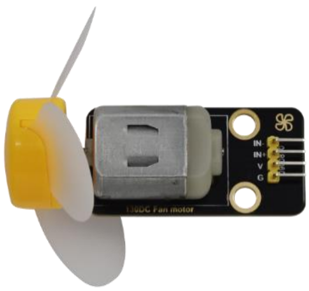
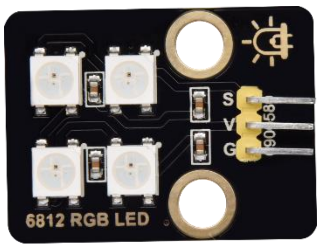
- The RFID module helps us by using RFID cards or tags to control who has access or not in the house.
- The PIR motion sensor is used to let us know if someone is at our door.



- The LCD module is used to display various information stand alone, without the need to connect to a PC.
- The combined temperature and humidity sensor can turn on the heater or open the window if one of the variables exceeds the critical limits.
- The steam sensor is used to close the window if it starts raining.

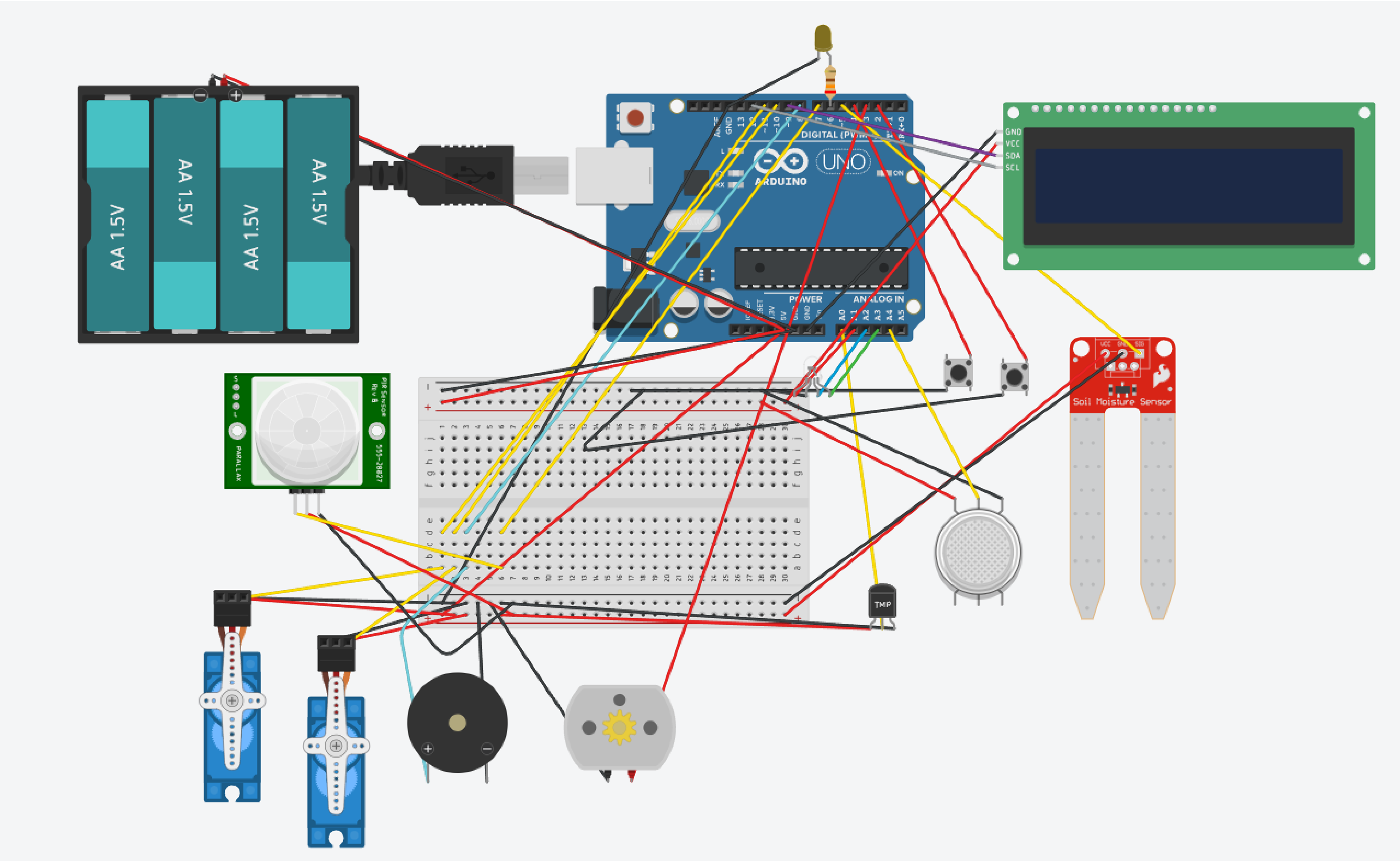


- The gas sensor is used to open the window if there are gas leaks.
- The piezo buzzer module notifies us if there are gas leaks through a constant repetitive sound, being connected to the gas sensor, until the house is ventilated.
- The DC motor drives the fan controlled by the temperature and humidity sensor.
- The RGB led module is used for the good condition of the inhabitants of the house

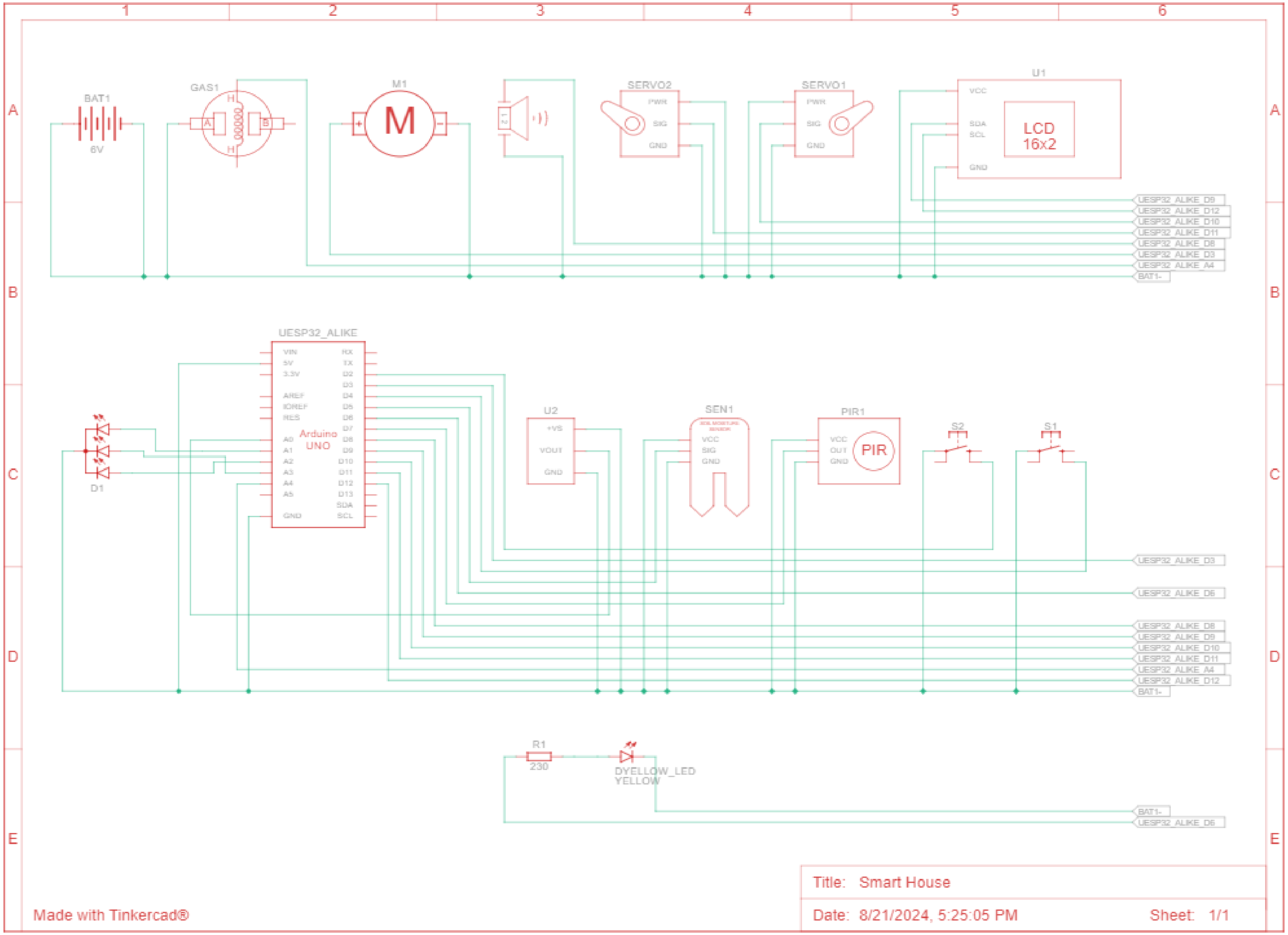



and to announce a problem or calamity.
- The 2 push-button modules are used to operate the LCD screen and adjust various functions of the smart home.
- The yellow LED is used to show the operation of all the smart systems of the house, the caste no longer flashes at a constant interval if there are problems.

3. Circuit diagram



4. Wiring diagram



5. Working mode

Being a specific patented design for this application, the ESP32 module is modified to have the necessary ports for the sensors used, not needing a breadboard. The assembly is done by following the tutorial with images offered by the developer, the assembly steps include both the construction of the house itself, but also the connection of the sensors and motors.

URL: https://docs.keyestudio.com/projects/KS5009/en/latest/docs/

6. Programming

- Software used: Arduino IDE
- Libraries used: DHT.h, Arduino.h, Adafruit_Sensor.h, ESPmDNS.h, WiFi.h, Wire.h, LiquidCrystal_I2C.h, MFRC522_I2C.h, ESP32Servo.h, Adafruit_NeoPixel.h.

Code

```
#include <DHT.h> for temperature sensor
#include <DHT_U.h>
#define DHTPIN 17
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

#include <Arduino.h> for specific sensors
#include <Adafruit_Sensor.h>

#include <ESPmDNS.h> pt wifi connection
#include <WiFi.h>
#include <WiFiClient.h>
Item string  = "0";
const char *ssid = "XID";
const char *password = "mateinfo11D";
WiFiServer server(80);

#include <Wire.h> pt i2c bus

#include <LiquidCrystal_I2C.h> for LCD
LiquidCrystal_I2C LCD(0x27, 16, 2);

#include "MFRC522_I2C.h" for RFID reader
MFRC522 MFRC522(0x28);
String passwordCard = "";

#include <ESP32Servo.h> pt servo
Servo servoWindow, servoDoor;

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
```

```
#include <avr/power.h>  Required for 16 MHz Adafruit Trinket
#endif
#define PIXEL_PIN 26 // Digital IO pin connected to the NeoPixels.
#define PIXEL_COUNT 4 // Number of NeoPixels
Adafruit_NeoPixel strip(PIXEL_COUNT, PIXEL_PIN, NEO_GRB + NEO_KHZ800);
boolean oldState = HIGH;
int mode = 0;
boolean newState;

#define yellowLed 12
#define steamSens 34
#define fanInMin 18
#define fanInPlus 19
#define motionSense 14
#define leftBtn 16
#define rightBtn 27
#define gasSens 23
#define buzz 25
unsigned long previousMillis = 0;
unsigned long previousMillisGas = 0;
const long interval = 1000;
const long intervalDoor = 5000;
these above used to get rid of the delay function for considerable long times
(>100 ms);
bool yLedState = 0;
int leftBtnMode = 0;
int posDoor = 0;
int servoDoorPin = 13;
int posWindow = 50;
int servoWindowPin = 5;
int directionMotor = 150;
int directionMotor2 = 0;
int aux = 0;
String doorPos = "closed";
String windowPos = "closed";
String temp = "";
Wet string  = "";
bool testMessageUsa = false;
int lastLeftBtnMode = 0;
int motor1, motor2;
bool fanOverWrite = false, windowOverWrite = false, doorOverWrite = false;

void initPins()
{
  pinMode(yellowLed, OUTPUT);
  pinMode(leftBtn, INPUT);
  pinMode(motionSens, INPUT);
  pinMode(fanInMin, OUTPUT);
  pinMode(fanInPlus, OUTPUT);
  pinMode(rightBtn, INPUT);
  pinMode(gasSense, INPUT);
```

```
  pinMode(steamSens, INPUT);
  pinMode(buzz, OUTPUT);
}

void initServo()
{
  servoDoor.attach(servoDoorPin);
  servoWindow.attach(servoWindowPin);
  servoDoor.write(posDoor);
  servoWindow.write(posWindow);
}
void initLcd()
{
  LCD.init();
  LCD.backlight();
  LCD.clear();
}
void wifiSetup()
{
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  Serial.println("TCP server started");
  MDNS.addService("http", "tcp", 80);
}
void initRFID()
{
  MFRC522.PCD_Init(); // initialize MFRC522
  ShowReaderDetails(); // PCD display - MFRC522 read carder
  Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
}
void initRGB()
{
  strip.begin(); // Initialize NeoPixel strip object (REQUIRED)
  strip.show(); // Initialize all pixels to 'off'
}

// SETUP FUNCTON START-----------------
void setup()
{
  Serial.begin(9600);
  initPins();
```

```
  initServo();
  initLcd();
  Wire.begin();
  initRFID();
  wifiSetup();
  Serial.println(F("DHTxx test!"));
  dht.begin();
  initRGB();
}
// SETUP FUNCTON END----------------
float h = dht.readHumidity();
float t = dht.readTemperature();
void gasDetect() // 0 and gas, 1 nu and gas
{
  unsigned long currentMillisGas = millis();
  bool gasval = digitalRead(gasSens);
  if (!windowOverWrite)
    if (!gasval)
{
      openWindow();
      digitalWrite(buzz, HIGH);
      delay(5);
      digitalWrite(buzz, LOW);
      delay(5);
}
    else
      digitalWrite(buzz, LOW);
}
void rainDetect() // 0 is not rain, 1 is rain
{
  if (!windowOverWrite)
    if (digitalRead(steamSens))
{
      closeWindow();
}
    else
{
      openWindow();
}
}

void lcdSetting()
{
  if (digitalRead(leftBtn) == 0)
{
    LCD.clear();
    delay(200);
    leftBtnMode = (leftBtnMode + 1) % 4;
}
  switch (leftBtnMode)
{
```

```
  Case 0:
{
    LCD.setSlider(0, 0);
    LCD.print("Welcome to OUR");
    LCD.setCursor(0, 1);
    LCD.print("SMART HAUS");
    break;
}
  Case 1:
{
    LCD.setSlider(0, 0);
    LCD.print("Temp: ");
    LCD.print(temp);
    LCD.print(" C");
    LCD.setCursor(0, 1);
    LCD.print("Humidity:");
    LCD.print(wet);
    LCD.print("%");
    break;
}
  Case 2:
{
    LCD.setSlider(0, 0);
    LCD.print("Door is ");
    LCD.print(doorPos);
    LCD.setCursor(0, 1);
    LCD.print("Window is ");
    LCD.print(windowPos);
    break;
}
  Case 3:
{
    LCD.setSlider(0, 0);
    LCD.print("Connected on:");
    LCD.setCursor(0, 1);
    LCD.print(WiFi.localIP());
    break;
}
  Case 4:
{
    LCD.setSlider(0, 0);
    LCD.print("Someone is at ");
    LCD.setCursor(0, 1);
    LCD.print("the front door");
    break;
}
  Case 5:
{
    LCD.setSlider(0, 0);
    LCD.print("Card recognised");
    LCD.setCursor(0, 1);
```

```
    LCD.print("Welcome back!    ");
    break;
}
  Case 6:
{
    LCD.setSlider(0, 0);
    LCD.print("Card error");
    LCD.setCursor(0, 1);
    LCD.print("Access denied");
    break;
}
  Case 7:
{
    LCD.setSlider(0, 0);
    LCD.print("Hidden menu");
    LCD.setCursor(0, 1);
    LCD.print("ON Slots");
    break;
}
}
}
void doorMotion()
{
  if (!doorOverWrite)
    if (!digitalRead(motionSens))
{
      closeDoor();
      if (testMessageUsa)
{
        LCD.clear();
        leftBtnMode = lastLeftBtnMode;
        testMessageUses = false;
}
}
}
void openDoor()
{
  servoDoor.write(45);
  doorPos = "opened";
}
void closeDoor()
{
  servoDoor.write(0);
  doorPos = "closed";
}
void openWindow()
{
  servoWindow.write(100);
  windowPos = "opened";
}
void closeWindow()
```

```
{
  servoWindow.write(0);
  windowPos = "closed";
}
void fanOn()
{
  motor2 = directionMotor2;
  motor1 = directionMotor;
  analogWrite(fanInMin, motor2);
  analogWrite(fanInPlus, motor1);
}
void fanOff()
{
  motor2 = 0;
  motor1 = 0;
  analogWrite(fanInMin, motor2);
  analogWrite(fanInPlus, motor1);
}
void wifiSettings()
{
 WiFiClient client = server.available();
  if (!client)
{
    return;
}
  while (client.connected() & !customer.available())
{
    delay(1);
}
 String req = client.readStringUntil('\r');
  int addr_start = req.indexOf(' ');
  int addr_end = req.indexOf(' ', addr_start + 1);
  if (addr_start == -1 || addr_end == -1)
{
    Serial.print("Invalid request: ");
    Serial.println(req);
    return;
}
  req = req.substring(addr_start + 1, addr_end);
  item = req;
  Serial.println(item);
 String s;
  if (req == "/") // Browser accesses address can read the information sent by
the client.println(s);
{
 IPAddress ip = WiFi.localIP();
 String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +
String(ip[3]);
    s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<! DOCTYPE
HTML>\r\n<html>Hello from ESP32 at ";
    s += ipStr;
```

```
    s += "</html>\r\n\r\n";
    Serial.println("Sending 200");
    customer.println(s); // Send the string S, you can read the information when
visiting the address of E smart home using a browser.
client.print(s);
}
  if (req == "/led/on")
{
    customer.println("turn on the LED");
    newState = LOW;
    mode = 4;
}
  if (req == "/led/off")
{
    customer.println("turn off the LED");
    newState = LOW;
    mode = 0;
}
  if (req == "/red/on")
{
    customer.println("RED LED tower");
    newState = LOW;
    mode = 1;
}
  if (req == "/green/on")
{
    customer.println("GREEN LED tower");
    newState = LOW;
    mode = 2;
}
  if (req == "/blue/on")
{
    customer.println("BLUE LED tower");
    newState = LOW;
    mode = 3;
}
  if (req == "/fan/on")
{
    customer.println("Turning the fan ON");
    fanOn();
    fanOverWrite = true;
}
  if (req == "/fan/off")
{
    customer.println("Turning the fan OFF");
    fanOff();
    fanOverWrite = false;
}
  if (req == "/window/off")
{
    customer.println("Opening the Window");
```

```
    openWindow();
    windowOverWrite = false;
}
  if (req == "/window/on")
{
    windowOverWrite = true;
    customer.println("Closing the Window");
    closeWindow();
}
  if (req == "/door/on")
{
    customer.println("Opening the Door");
    openDoor();
    doorOverWrite = true;
}
  if (req == "/door/off")
{
    customer.println("Closing the Door");
    closeDoor();
    doorOverWrite = false;
}
  customer.stop();
}
void rfidRead()
{
  if (!MFRC522.PICC_IsNewCardPresent() || !MFRC522.PICC_ReadCardSerial())
{
    delay(50);
    passwordCard = "";

    return;
}
select one of door cards. UID and SAK are mfrc522.uid.

save UID
  Serial.println(F("Card UID:"));
  for (byte i = 0; and < MFRC522.uid.size; i++)
{
    Serial.Print(MFRC522.uid.uidByte[i] < 0x10? " 0" : " ");
// Serial.print(mfrc522.uid.uidByte[i], HEX);
    Serial.Print(MFRC522.uid.uidByte[i]);
    passwordCard = passwordCard + String(mfrc522.uid.uidByte[i]);
}
  if (passwordCard == "83179172230") // The card number is correct, open the door
{
    passwordCard = "";
    if (!doorOverWrite)
      openDoor();
    LCD.clear();
    if (lastLeftBtnMode = !5)
      lastLeftBtnMode = leftBtnMode;
```

```
    leftBtnMode = 5;
    testMessageUses = true;
}
  else // The card number is wrong,LCD displays error
{
    passwordCard = "";
    if (!doorOverWrite)
      closeDoor();
    LCD.clear();
    if (lastLeftBtnMode = !6)
      lastLeftBtnMode = leftBtnMode;
    leftBtnMode = 6;
    testMessageUses = true;
}
Serial.println(password);
}
void dhtRead()
{
  delay(50);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  temp = String(t, 1);
  wet = String(h, 1);

  if (isnan(h) || isnan(t))
{
    Serial.println(F("Failed to read from dht sens"));
    return;
}
  if (!fanOverWrite)
    if ((t > 31) || (h > 70))
      fanOn();
    else
      fanOff();
}

void hiddenMenu()
{
  if (digitalRead(steamSens) & & !digitalRead(rightBtn))
{
    LCD.clear();
    leftBtnMode = 7;
}
}
void rgbLed()
{
Get current button state.
  newState = digitalRead(rightBtn);
Check if state changed from high to low (button press).

  if ((newState == LOW) & & (oldState == HIGH))
```

```
{
Short delay to debounce button.
    delay(20);
Check if button is still low after debounce.
    newState = digitalRead(rightBtn);
    if (newState == LOW)
 { // Yes, still low
      if (++mode > 4)
        mode = 0; // Advance to next mode, wrap around after #8
      switch (mode)
 { // Start the new animation...
      Case 0:
        colorWipe(strip.Color(0, 0, 0), 50); // Black/off
        break;
      Case 1:
        colorWipe(strip.Color(255, 0, 0), 50); //Red
        break;
      Case 2:
        colorWipe(strip.Color(0, 255, 0), 50); //Green
        break;
      Case 3:
        colorWipe(strip.Color(0, 0, 255), 50); //Blue
        break;
      Case 4:
        colorWipe(strip.Color (127, 127, 127), 50); // whitish half
}
}
}
Set the last-read button state to the old state.
  oldState = newState;
}
// LOOP FUNCTION START-----------------
void loop()
{
  hiddenMenu();
  dhtRead();
  wifiSettings();
  unsigned long currentMillis = millis();
  gasDetect();
  if (currentMillis - previousMillis > interval)
{
    yLedState = !yLedState;
    digitalWrite(yellowLed, yLedState);
    doorMotion();
    previousMillis = currentMillis;
}

  rfidRead();
  rainDetect();
  lcdSetting();
  rgbLed();
```

```
}
// LOOP FUNCTION END-----------------
void colorWipe(uint32_t color, int wait)
{
  for (int i = 0; and < strip.numPixels(); i++)
 { // For each pixel in strip...
    strip.setPixelColor(i, color); // Set pixel's color (in RAM)
    strip.show(); // Update strip to match
    delay(wait); // Pause for a moment
}
}
void ShowReaderDetails()
{
attain the MFRC522 software
 Byte V = MFRC522.PCD_ReadRegister(MFRC522.VersionReg);
  Serial.print(F("MFRC522 Software Version: 0x"));
  Serial.print(v, HEX);
  if (v == 0x91)
    Serial.print(F(" = v1.0"));
  else if (v == 0x92)
    Serial.print(F(" = v2.0"));
  else
    Serial.print(F(" (unknown)"));
  Serial.println("");
when returning to 0x00 or 0xFF, may fail to transmit communication signals
  if ((v == 0x00) || (v == 0xFF))
{
    Serial.println(F("WARNING: Communication failure, is the MFRC522 properly
connected?"));
}
}
```

7. Conclusions

The project had its difficulties, found both in the construction part, but also in the programming part, which make us learn new things and how to think differently about a problem. The Smart Home helps us to help ourselves by acclimatizing humans with technology that can make our lives easier and better. A single sensor will have no power, but a multitude of sensors and human interfaces, closely linked and perfected to work together, can bring about radical changes in everyone's life.

8. Bibliographic references

- Keyestudio IoT Smart Home Kit for ESP32,
  URL: https://docs.keyestudio.com/projects/KS5009/en/latest/index.html
- 250+ ESP32 Projects, Tutorials and Guides with Arduino IDE,
  URL: https://randomnerdtutorials.com/projects-esp32/
- Database with the libraries used and the code written. URL:

https://drive.google.com/drive/folders/1i-62LxW3sEDUMpLTA_b0HygQe_WZIGaf?usp=sharing