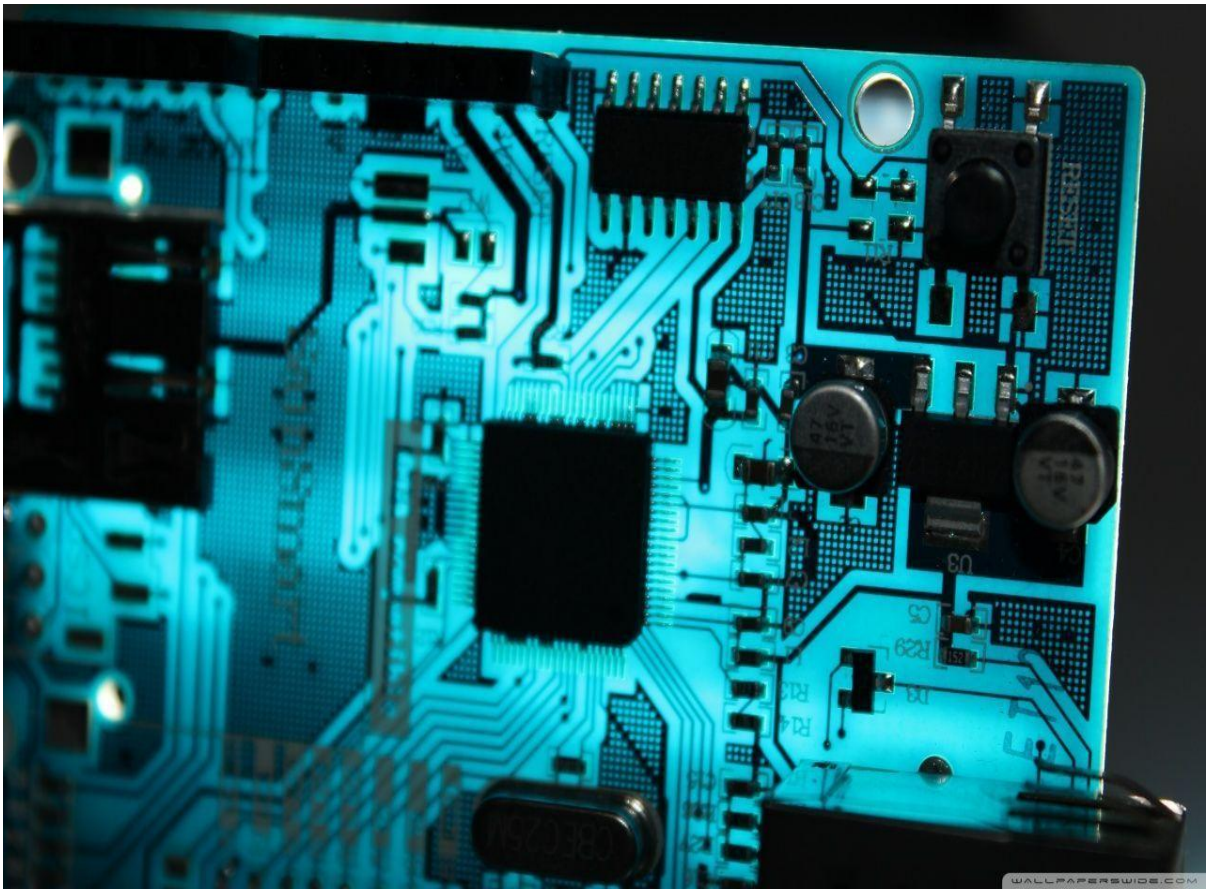
Gáz szivárgás riasztó



Projektvezető: Dr. Losonczi Lajos

Készítette: Szász Arnold-Levente, Számítástechnika

Sapientia EMTE, 2021

Tartalomjegyzék

Bevezetés

A modern világnak egyre nagyobb érdeklődése van az automatizált rendszerek iránt, mindent szeretnénk, ha automatikusan egy előre definiált módon egy gép, egy okos eszköz elvégezne helyettünk, ezzel is megkönnyítve a mindennapi életet. Éppen ezért olyan elterjedtek a modern háztartásokban az IoT rendszerek, amelyek lényegében olyan különböző elektronikai eszközöket jelent, amelyek képesek felismerni valamilyen lényegi információt, és azt egy internet alapú hálózaton egy másik eszközzel kommunikálni. (Wikipedia, 2020) Ez egy igen gyors léptékben fejlődő iparág, mivel nagy az igény a kreatív, intelligens megoldásokra. Az emberek szeretnék, ha házaikban automatikusan szabályozódna a hőmérséklet, az ablakok maguktól nyílnának, ha éppen friss levegőre lenne szükség, vagy akár az életre veszélyes jelenségek felügyelve lennének egy intelligens eszköz által, ilyen lehet például a füst vagy a gázszivárgás érzékelő.

Ezeket a rendszereket általában egy vagy több mikrovezérlővel és érzékelők, szenzorok hadával oldják meg. Az érzékelők, szenzorok adatot generálnak, a mikrovezérlő, mint feldolgozó egység értelmezi azt, és előre definiált műveletet hajt végre. Az adatokat megosztja egy általában internet segítségével elérhető szerverrel, ahonnan a felhasználó bármikor nyomon követheti éppen mi zajlik otthonában, illetve felülírva a az automatikát saját maga kezdeményezhet műveleteket, például felhúzhatja a fűtést távolról.

A projekt célja

Célunk egy olyan rendszer létrehozása, amely képes éghető gáz koncentrációját mérni egy adott helyiségben, a gázszint értékét képes megjeleníteni egy erre létrehozott weboldalon. Ha a gáz koncentráció érték meghalad egy bizonyos küszöb értéket, akkor a rendszer automatikusan közbeavatkozik, azonban lehetőség van manuális beavatkozásra is. A webszerver, illetve a feldolgozó egység szerepét egy ESP32 mikrovezérlő látja el. Feladata a webszerver elérhetőségét biztosítani, és az MQ-2-es gázkoncentrációt érzékelő szenzor adatait leolvasni, illetve egy előre definiált módon viselkedni.

Megvalósítandó feladatok

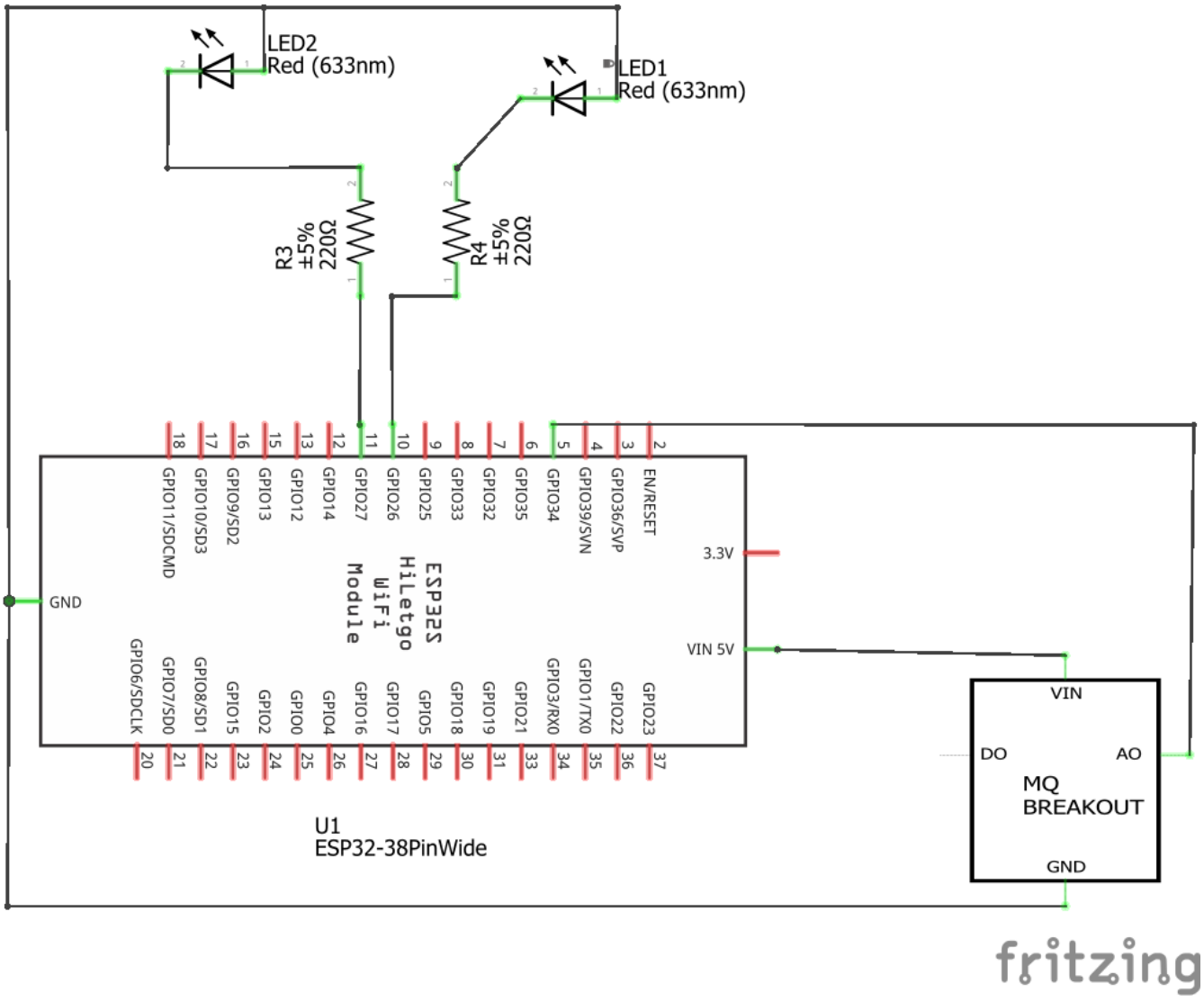
- ESP32 mikrovezérlő, MQ-2 gázkoncentráció szenzor, ledek, huzalok beszerzése
- Ehhez kapcsolódó dokumentáció átnézése és megértése
- Konkrét rendszer megtervezése, amely megfelel a projekt kiírásnak
    - Használható, barátságos internetes felület létrehozása
    - Szenzor adatok beolvasás
    - Beavatkozó egység manuális. illetve automatizált módon történő elindítása
- Megfelelő projekt dokumentáció létrehozása

Tervezési folyamat

A tervezés során figyelembe kellett venni a projekt kiírás részletei, azaz használni kellett egy 32 bittes mikróvezérlőt, egy gáz koncentrációt érzékelő szenzort, illetve ledeket, amelyek a megszakító rendszert kívánják szemléltetni.

Első lépésben a projektvezető tanár által megadott dokumentációt kellett át olvasni és megérteni, hogy miként működnek az ilyen rendszerek, és miként lehet őket programozni adott célra és feladatra.

Az alkatrészek beszerzése után következett a konkrét terv elkészítése és az áramköri rajz létrehozása.



Áramköri rajz

Az áramköri rajzból kivehető, miként vannak egymáshoz kapcsolva a valóságban is az alkatrészek, milyen lábakon kapcsolódnak a ledek az ESP32 laphoz, illetve milyen bekötés szükséges a szenzor szakszerű üzemeltetéséhez. Az áramköri rajzból kiderül pontosan milyen alkatrészre van

szükségünk, mennyi összekötő kábelre, és segítségével jobban el tudjuk képzelni, hogy a valóságban miként is fog kinézni és működni az általunk tervezett rendszer.

Alkatrészek

A projekt különböző alkatrészekből épül fel, amelyek az áramköri rajzon is jól láthatóak. Az általam választott mikróvezérlő egy ESP32-DEVKITC-32D ESPRESSIF, a gázkoncentrációt érzékelő szenzor az a WAVESHARE MQ-2, illetve található még 2 db piros led, 2 db 270Ω ellenállás, össze kötő huzalok, USB adatkábel és egy külső akkumulátor tápforrás gyanánt.

ESP32 mikrovezérlő

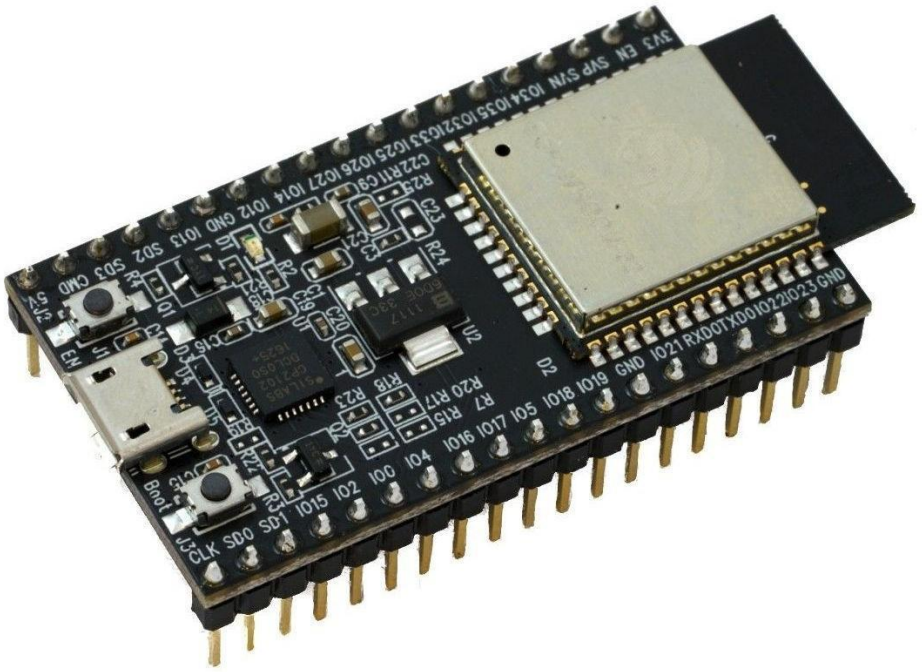
Az ESP32 egy SoC (System on Chip) mikrovezérlő amely olcsó, alacsony energiafogyasztású rendszer, beépített WI-FI modullal és kettős módú Bluetooth-szal rendelkezik. Gyártják egy, illetve kétmagos Tensilica Xtensa LX6 mikroprocesszor változatban is, mindemellett beépített antenna kapcsolókat, RF balun-t, erősítőt, alacsony zajszintű vevőerősítőt, szűrőket és energiagazdálkodási modulok tartalmaz. Összesen 38 pin áll rendelkezésre, beleértve a 3,3V-os tápegységet, az I2C, UART és GPIO interfészeket. Az ESP32-t az Espressif System fejlesztette és készítette. (Espressif, 2016)

ESP32-DEVKITC-32D ESPRESSIF tartalmaz:

- CPU: Xtensa kétmagos (vagy egymagos) 32-bit LX6 mikroprocesszort, amely 160-240 MHz működik
- Memória: 520 KiB SRAM
- Wi-Fi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR és BLE
- 12-bit SAR ADC egész 18 csatorna
- 2 × 8-bit DACs
- 10 × érintés érzékelő szenzor
- 4 × SPI
- 2 × I²S interfész
- 2 × I²C interfész
- 3 × UART
- SD/SDIO/CE-ATA/MMC/eMMC host vezérlő
- SDIO/SPI slave vezérlő
- Ethernet MAC interfész, dedikált DMA and IEEE 1588 Precision Time Protocol támogatás
- CAN bus 2.0
- Infravörös távirányító (TX / RX, akár 8 csatorna)
- Motor PWM
- LED PWM (akár 16 csatorna)
- Hall effektus érzékelő
- Rendkívül kis teljesítményű analóg előerősítő
- IEEE 802.11 standard védelmi funkciók támogatva, beleértve WFA, WPA/WPA2 és WAPI
- Biztonságos boot
- Flash titkosítás
- 1024-bit OTP, akár 768 bites az ügyfelek számára

- Titkosítási hardveres gyorsítás: AES, SHA-2, RSA, elliptikus görbe kriptográfia (ECC), véletlenszám-generátor (RNG)
- Belső alacsony esés szabályozó
- Egyéni energia tartomány az RTC számára
- 5 µA mély alvási áram
- Ébresztés GPIO megszakítás, időzítő, ADC mérések, kapacitív érintésérzékelő megszakítás után (Espressif, 2017)

Az ESP32 lapot az ESP8266 lap utódjaként fejlesztették, elsősorban két ütemezett mag megvalósításában különbözik. Az 8266-hoz képest megnövekedett a GPIO port és a funkcióinak száma is. Jelenleg az Espressif több változatban is gyártják ezt a mikrovezérlőt.



ESP32

Az esp eszközt lehetőség van felprogramozni Windows, Linux, MacOS operációs rendszerről is, mivel az Arduino IDE elérhető a felsorolt operációs rendszereken. Segítségével intelligens otthonokat létrehozhatunk, mert az IoT világban az ESP32 lapot rengeteg megvalósításban használják. Lehetőség van a számtalan előre definiált és megvalósított rendszer közül választani, vagy meg lehet tervezni a sajátunkat teljesen a nulláról, ehhez Az Arduino IDE megfelelő környezetet biztosít. (Badamasi, 2014)

MQ-2 gázkoncentráció szenzor

Az MQ-2 modul hasznos a gázszivárgás érzékeléséhez otthoni és ipari környezetben egyaránt. Alkalmas H2, LPG, CH4, CO, alkohol, füst vagy propán kimutatására. Nagy érzékenységének és gyors reagálási idejének köszönhetően a lehető leggyorsabb mérést teszi lehetővé. A szenzor analóg kimenettel rendelkezik. Közvetlenül csatlakozik az ESP32 mikrokontrollerhez.

Fontosabb adatok:

- Üzemi feszültség: 5V
- Terhelésállóság: 20 KΩ
- Fűtőellenállás: 33Ω ± 5%
- Fűtésfogyasztás: <800mw
- Ellenállás érzékelése: 10 KΩ – 60 KΩ
- Koncentrációs hatókör: 200 – 10000ppm
- Előmelegítés ideje: 24 óra felett (Lastminuteengineers, 2020)

MQ-2

Összesen 4 lába van, VCC – 5v táplálás, GND – földelés, D0 – digitális kimenet, A0 – analóg kimenet. Az MQ-2-es gázérzékelő szenzor úgy működik, hogy amikor az ón-dioxidot (félvezető részecskéket) magas hőmérsékleten melegítik a levegőben, oxigén adszorbeálódik a felszínen. A tiszta levegőben az ón-dioxidban lévő donor elektronok az oxigén felé vonzódnak, amely az érzékelő anyag

felületén adszorbeálódik. Ez megakadályozza az elektromos áram áramlását. Redukáló gázok jelenlétében az adszorbeált oxigén felületi sűrűsége csökken, amikor reagál a redukáló gázokkal. Az elektronokat ezután az ón-dioxidba engedik, így az áram szabadon áramolhat az érzékelőn. (K. Keshamoni, S. Hemanth, 2017)

Arduino IDE

Az arduino egy nyílt forrású - ingyenes platform melyet az Atmell AVR mikrokontrollereire alapozva fejlesztettek ki. A C programozási nyelvet felhasználó keretrendszer segítségével viszonylag egyszerűen írhatunk programokat AVR alapú eszközökre. Az elmúlt években rengeteg alaplapot és kiegészítő eszközt dobtak piacra, melyek megkönnyítik a munkát. Az alaplapok felszereltsége nagyon változatos. Tartalmazzák a bootloaderrel feltöltött mikrovezérlőn kívül a stabil tápegységet, a perifériák csatlakoztatásának lehetőségeit, és opcionálisan a program betöltéshez szükséges számítógépes csatlakozási lehetőséget. Ez általánosan egy USB-soros konverter. (Massimo Banzi, Michael Shiloh, 2015)

Felhasználási szempontból rendkívül rugalmas, tulajdonképpen csak a képzeletünk és az általunk birtokolt mikrovezérlő szab határokat. Lehet használni mérésre, szabályozásra, vezérlésre, játékra.

A konfiguráció után, amelyben hozzáadjuk a lapunkat, ha esetleg nem alapértelmezett, beállítjuk a megfelelő virtuális soros portot (COM3, COM4 stb.), mehet is a projekt létrehozása. A konkrét kódot úgynevezett sketch-be írjuk. A megírt kód gombnyomásra ellenőrzésre, fordításra majd betöltésre kerül a mikrovezérlőbe.

Beágyazott vezérlő programok

Sketchet használ az Arduino egy program megírásához. Ez a kódegység, amelyet feltöltenek és futtatnak egy Arduino lapon. Két speciális funkció van, amelyek mindenképpen szerepelnek egy ilyen sketchbe, setup() és loop(). A setup() egyszer kerül meghívásra amikor a sketch elindul. Itt történik meg a telepítési folyamatok elvégzése, mint a pin módok beállításai, könyvtár inicializáció. A loop() függvény újra és újra meghívódik, itt olyan kódrészletek vannak, amelyek folyamatosan, többször le kell fussanak. Az újra futást a delay() függvénnyel tudjuk szabályozni, megadva mennyit várakozzon a loop-ba lévő kódsorozat újra elvégzésre. Minden egyes sketch kötelezően tartalmazza ezeket, ha még nem is használja fel semmire. (Arduino, 2020)

Kezdetben szükséges megadni a különböző pineket, melyik mire használatos, megadjuk a Wi-Fi kapcsolási adatokat, illetve megadjuk melyik porton üzemel a szerver.

```
WebServer server(80);
#define MQ2 34
#define manualOut 26
#define automaticOut 27
//Enter your SSID and PASSWORD
const char* ssid = "ssid";
const char* password = "password";
```

A továbbiakban három általam megírt függvény következik. HandleRoot, kiolvassa a Header állományból a weboldal html kódját és elküldi a szervernek. HandleMQ2, analóg módon kiolvassa a

szenzor adatot a megfelelő pinről, majd elküldi a szervernek, ha arra érkezett kérelem. ManualInterrupt, a manuális beavatkozást végzi, ha erre kérelem jött akkor felgyújtja a ledet, vagy le oltja a LOW és HIGH értékek változtatásával.

```
void handleRoot() {
 String htmlCode = MAIN_page; //Read HTML contents
 server.send(200, "text/html", htmlCode); //Send web page
}
void handleMQ2() {
 int sensoreRead = analogRead(MQ2);
 String mqValue = String(sensoreRead);
 server.send(200, "text/plane", mqValue); //Send gas sensor value only to client ajax request
}
void manualInterrupt() {
 if (digitalRead(manualOut) == LOW) {
  Serial.println("int");
  digitalWrite(manualOut, HIGH);
 }
 else {
  Serial.println("not");
  digitalWrite(manualOut, LOW);
 }
}
```

A setup részben először a SerialMonitor beállítása történik majd a wifire való csatlakozás. While ciklusba fut a csatlakozási próbálkozás. Ha sikeresen csatlakozott az ESP lap a wifihez, akkor a SerialMonitorba kiíródik a lokális IP cím. Következik a pinMode-ok kiválasztása, a fentebb deklarált pinek beállításra kerülnek, attól függően, hogy kimenet vagy bemenet a pinMode nevű függvény segítségével. A szerverhez szükséges függvények következnek, maga a megjelenítést, szenzor adat frissítés végül a szerver konkrét elindítása. (Rui Santos, Sara Santos)

```
Serial.begin(115200);
 WiFi.mode(WIFI_STA); //Connect to your wifi
 WiFi.begin(ssid, password);
 while (WiFi.waitForConnectResult() != WL_CONNECTED) {
  Serial.print(".");
 }
 //If connection successful show IP address in serial monitor
 Serial.println(WiFi.localIP());  //IP address assigned to your ESP
 //------------------------------------------------------------
 pinMode(MQ2, INPUT);
 pinMode(manualOut, OUTPUT);
 pinMode(automaticOut, OUTPUT);
 server.on("/", handleRoot);      //This is display page
 server.on("/readGas", handleMQ2);//To get update of gas sensor Value only
 server.on("/manualInterrupt", manualInterrupt);
 server.begin();                  //Start server
 Serial.println("HTTP server started");
```

A loop részben az automatikus megszakítás történik, illetve a szerver fenntartása. Az automatikus megszakítás úgy működik, hogy ha a leolvasott szenzor értéke meghaladja a

küszöbértéket, akkor kigyúl egy led, ha visszaesik a normál értékre akkor kikapcsolja a ledet. (Rui Santos, Sara Santos)

```
server.handleClient();
int limit = 3000;
int sensorValue = analogRead(MQ2);
if (sensorValue < limit)
{
  digitalWrite(automaticOut, LOW);
}
else {
  digitalWrite(automaticOut, HIGH);
}
delay(1);
```

Annak érdekében, hogy a program dinamikusan tudja frissíteni a szenzor értékét a weboldalon AJAX (Asynchronous JavaScript and XML) használtam. Az AJAX lehetővé teszi a weboldalak aszinkron frissítését azáltal, hogy kis mennyiségű adatot cserél a szerverrel a kulisszák mögött. Ez azt jelenti, hogy a weboldal egyes részei frissíthetők az egész oldal újratöltése nélkül.

Felhasználói interfész programok

A felhasználói interfész egy weboldal, amelyet akár globálisan is el lehet érni. Maga a weboldal HTML, CSS, JavaScript kombinációjával készült. A HTML leírja s megjeleníti a weboldalon található komponenseket, CSS dizájnt ad, JavaScript felelős az AJAX requestek teljesítésért, beletartozik a szenzor értékének a dinamikus cseréje, illetve a kézi beavatkozásnál a gombnyomás érzékelése és a kérelem küldése a szerver felé.

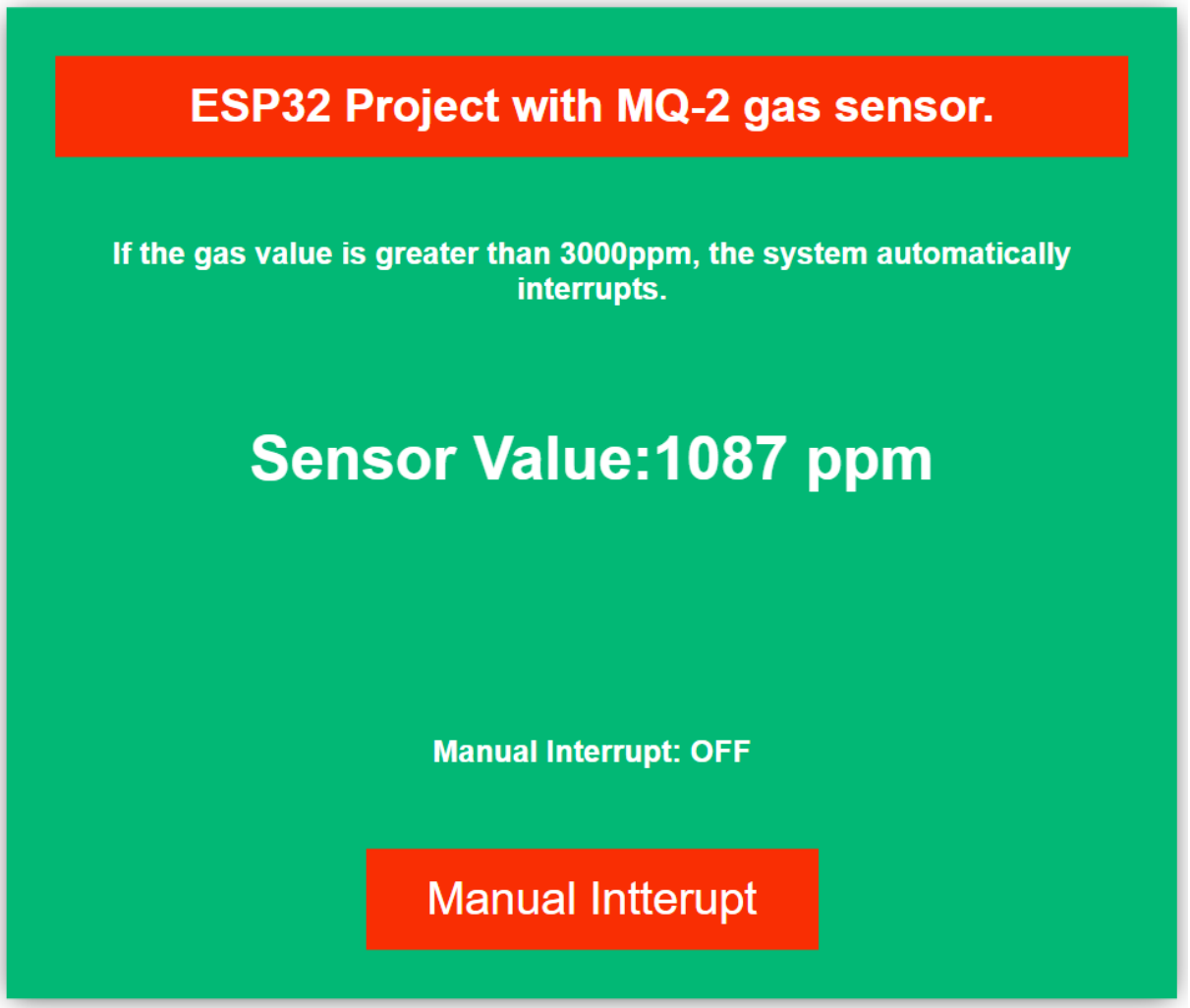
```
setInterval(function () {
        getData();
    }, 1000); //1000mSeconds update rate
    function getData() {
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
              if(this.responseText>="3000"){
               audio.play();
               document.getElementById("warning").style.visibility="visible";
                }
                else{
                  document.getElementById("warning").style.visibility="hidden";
                  }
                document.getElementById("GasValue").innerHTML =
                    this.responseText;
            }
        };
        xhttp.open("GET", "readGas", true);
        xhttp.send();
    }
```

Egy http request hívódik meg 1 másodpercenként a setInterval js függvény segítségével, amelyben figyeljük azt, hogy a válasz értéke, azaz a szenzor értéke nagyobb mint 3000, ha igen, akkor egy figyelmeztető hangot bocsát ki, a képernyőn megjelenik egy figyelmeztető kép, és a háttérbe elindul az automatikus megszakítás.

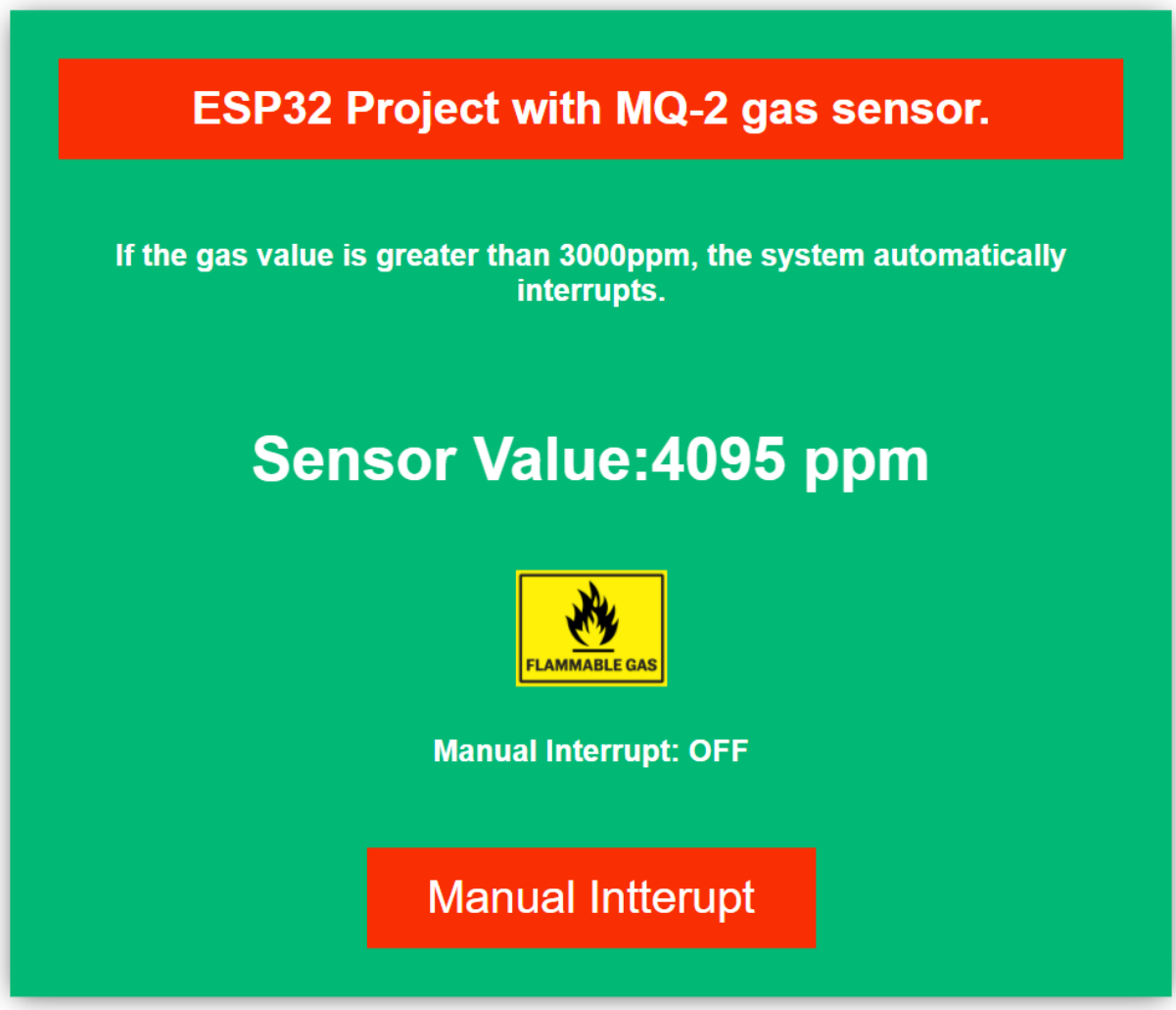
```
document.getElementById("btnManual").addEventListener("click", () => {
    var state = document.getElementById("spanManual").innerHTML;
    if (state == "ON") {
        document.getElementById("spanManual").innerHTML = "OFF";
    }
    else {
        document.getElementById("spanManual").innerHTML = "ON";
    }
    xhttp.open("GET", "manualInterrupt", true);
    xhttp.send();
})
```

A fenti függvény felelős a manuális megszakítás miatt. Ha az erre előre definiált gombot lenyomja a felhasználó akkor, a képernyőn megjelenik a manuális megszakítás jelenlegi állapota, illetve elküldi a szervernek, hogy éppen manuális beavatkozás történt és el kell végezze a szerver oldali műveleteket.

Ha a felhasználó sikeresen felcsatlakozott a weboldalra akkor a következőt fogja látni:



Alaphelyzetben a képernyőn egy rövid információ sáv található meg, mikor fog bekövetkezni az automatikus megszakítás. Láthatjuk a szenzor jelenlegi értékét és a manuális megszakítás jelenlegi állapotát, alatta a manuális megszakítást végző gombot.



Gázszivárgás esetén, ha a szenzor értéke meghaladja a küszöbértéket, akkor egy figyelmeztető hang, egy figyelmeztető kép jelenik meg pluszba a képernyőn, eközben a háttérben el végződik az automatikus megszakítás.

A felhasználói felületet globálissá lehet tenni, egy egyszerű kis program, az Ngrok segítségével. Az Ngrok egy cross-platformos alkalmazás, melynek segítségével a lokálisan futtatott webszerverek globálisan is elérhetők anélkül, hogy nyilvános IP címmel rendelkezne, mert az ngrok.com aldomainjén található. Tulajdonképpen egy hosszú élettartamú TCP tunnelt hoz létre közvetlenül a helyi gépre.

Összegzés

A projekt bemutatóra sikerült egy működő rendszert létrehozni, amely képes mérni a gázkoncentrációt és bizonyos küszöbérték esetén riasztást és automatikus megszakítást hajt végre. A szenzor adatok dinamikusan megtekinthetőek a weboldalon, ahol lehetőség van manuális beavatkozásra is. Úgy gondolom a projekt alatt hasznos tudásra tettem szert, mind a mikrovezérlők, mind az ezekkel való munka terén. Ez a tudás kezdésnek elég lesz új, általam hasznosnak vélt alkalmazási módok kipróbálásához.

Könyvészet, referenciák

Arduino. (2020). Retrieved from https://www.arduino.cc/en/tutorial/sketch

Badamasi, Y. A. (2014). The working principle of an Arduino. *1th International Conference on Electronics, Computer and Computation (ICECCO)*, (pp. 1-4). Abuja,.

Espressif. (2016). Retrieved from https://www.espressif.com/en/products/socs/esp32

Espressif. (2017). Retrieved from ESP32 Series: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

K. Keshamoni, S. Hemanth. (2017). Smart Gas Level Monitoring, Booking & Gas Leakage Detector over IoT. *2017 IEEE 7th International Advance Computing Conference (IACC)*, (pp. 330-332). Hyderabad.

Lastminuteengineers. (2020). Retrieved from https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/

Massimo Banzi, Michael Shiloh. (2015). *Getting Started with Arduino.* Maker Media.

Rui Santos, Sara Santos. (n.d.). *ESP32 Web Server with Arduino IDEA.*

Rui Santos, Sara Santos. (n.d.). Ultimate Guide for Arduino Sensors/Moduls.

Wikipedia. (2020, Június 28). Retrieved from Dolgok internetje: https://hu.wikipedia.org/wiki/Dolgok_internetje