ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK Hallgató neve: **Rédics Dániel**
2025 Hallgató törzskönyvi száma: **T/006908/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT
FELADATLAP

Hallgató neve: **Rédics Dániel**
Törzskönyvi száma: T/006908/FI1290/N
Neptun kódja: [redacted]

A dolgozat címe:
Okosotthon kialakítás és webalkalmazás-alapú irányítás Smart Home Design and Web Application-Based Control

Intézményi konzulens: Somlyai László
Külső konzulens:

Beadási határidő: 2024. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
IoT, beágyazott rendszerek és robotika

specializáció

A feladat

A feladat célja egy webalkalmazással irányított okos otthon rendszer megtervezése és megvalósítása, melyben egy mikrokontrolleres rendszer felelős az érzékelők adatainak gyűjtéséért és továbbításáért egy weboldalra. A weboldalon a felhasználó lehetőséget kap az adatok megtekintésére és a szenzorok távoli felügyeletére. A projekt során válassza ki a legmegfelelőbb eszközöket a feladat hatékony megvalósítása érdekében. A mikrokontrolleres rendszernek képesnek kell lennie különböző érzékelők (pl. hőmérséklet, páratartalom, mozgás stb.) adatainak gyűjtésére és ezek továbbítására egy webalkalmazás felé. Egy felhasználóbarát webalkalmazásnak kell rendelkezésre állnia, amely megjeleníti az érzékelőktől érkező adatokat és lehetőséget biztosít a szenzorok távoli felügyeletére. A webalkalmazásnak intuitív felhasználói felülettel kell rendelkeznie, amely lehetővé teszi az adatok könnyű navigálását és a szenzorok állapotának egyszerű felügyeletét.

A dolgozatnak tartalmaznia kell:

- a megoldandó feladat pontos leírását és részletes elemzését,
- a már működő rendszerek vizsgálatát,
- a felhasznált számítási rendszerek választásának okait,
- a rendszertervet és a döntési indoklásokat,
- a megvalósítás menetét,
- a tesztelési tervet, a tesztelés eredményeit, a fejlesztés során felmerült problémákat és azok megoldásait,
- az elkészült rendszer felhasználási-, és továbbfejlesztési lehetőségeit.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

Ph.

Fleiner R.

Dr. habil. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2026. december 15.**
(ÓE HKR 54.§ (10) bekezdés szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

Dr. Lovasi László
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2024. 05. 15.

.....

hallgató aláírása

Absztrakt

A szakdolgozatom egy webes alkalmazással vezérelt intelligens otthoni rendszer tervezését és megvalósítását mutatja be. A rendszer egy mikrokontroller-alapú egységet használ az érzékelőadatok gyűjtésére és továbbítására egy webszerverre a távoli felügyelet és vezérlés érdekében. A webes alkalmazás felhasználóbarát felületet biztosít az érzékelőadatok megjelenítéséhez és az érzékelő beállításainak kezeléséhez. Rendszertervezés: mikrokontroller-alapú egység, mely összegyűjti az érzékelők adatait (hőmérséklet, páratartalom, mozgás stb.), és továbbítja azokat a webszerverre. Webalkalmazás: megjeleníti az érzékelők adatait, lehetővé teszi az érzékelők távvezérlését, és intuitív felhasználói felülettel rendelkezik. Tesztelés és értékelés: a rendszer működésének és az adatok pontosságának átfogó tesztelése, a fejlesztés során felmerülő lehetséges problémák azonosítása és megoldása. Alkalmazás és jövőbeli irányok: lehetséges alkalmazások az otthoni automatizálásban, a biztonsági rendszerekben és a környezeti megfigyelésben. A szakdolgozatom során meg kellett ismerkednem a mikrokontrollerek fajtáival, ezek fejlesztői környezetével, a hozzá felhasznált szenzorokkal, a weboldal készítéssel és ezen eszközök kommunikációjával.

Abstract

My thesis presents the design and implementation of a web application driven smart home system. The system uses a microcontroller-based unit to collect and transmit sensor data to a web server for remote monitoring and control. The web application provides a user-friendly interface to display sensor data and manage sensor settings. System design: a microcontroller-based unit that collects sensor data (temperature, humidity, motion, etc.) and transmits it to a web server. Web application: displays sensor data, allows remote control of sensors and has an intuitive user interface. Testing and evaluation: comprehensive testing of system operation and data accuracy, identification and resolution of potential problems encountered during development. Applications and future directions: potential applications in home automation, security systems and environmental monitoring. During my thesis I had to learn about the types of microcontrollers, their development environment, the sensors used for them, the creation of web pages and the communication between these devices

Tartalomjegyzék

1. Bevezetés

A mai világban az IoT (Internet Of Things) és az okosotthon rendszerek egyre több figyelmet kapnak. Napról napra egyre kényelmesebb az életünk, és ebben a hatalmas szerepet játszanak ezek az eszközök, vegyük például a világítást, a fűtést és hűtést, biztonsági rendszereket és háztartási gépeket.

Az okosotthonok koncepciója nem újdonság. Az 1960-as évek óta léteznek elképzelések az intelligens otthonokról, de az 1970-es években fejlesztették ki az X10 nevű vezérlőrendszert. Az X10 egy 1975-ben kitalált integrált áramköri projekt, amely lehetővé tette, hogy az otthoni eszközök először kommunikáljanak egymással. A projektet a Pico Electronics találta fel, egy skót vállalat, amely akkoriban inkább az első egychipes számológép feltalálásával végzett úttörő munkájáról volt ismert. Az első X10 egy vezérlőpulton és távirányítón keresztül tette lehetővé az otthoni lámpák és készülékek összekapcsolását, vezérlését és automatizálását. Ezek az eszközök az otthon elektromos vezetékein keresztül tudtak kommunikálni. A felhasználóknak a telepítéshez mindössze annyit kellett tenniük, hogy a központot és a modulokat bedugják egy konnektorba, majd a modulok segítségével csatlakoztatják a vezérelni kívánt lámpát és készüléket. Ez azt jelentette, hogy a telepítés nagyon egyszerű volt, és nem volt szükség új kábelezésre. A vezérlőpult 16 különböző üzemmódot tartalmazott az eszközök számára, amelyek közül néhányat a távirányítóról is lehetett vezérelni. A jövőbeli elképzelések azt mutatják, hogy a jövő intelligens otthonában talán csak sétálgatnunk kell majd, mert az otthon minden mást elintéz helyettünk. Bár a teljes automatizálás egy nagyon érdekes és megosztó téma, az olyan technológiák, mint az X10, megnyitják az utat, hogy az ilyen képzeletbeli élethez hasonló életet élhessünk. Lassan, de biztosan az X10 után újabb eszközök jöttek létre, amelyek hasonló célt szolgáltak: a fizikai fáradalmakat egy gombnyomással elvégezhetővé tenni.[1]

Napjainkban a legnépszerűbb funkciók között van a világításvezérlés, hőmérséklet-szabályozás, biztonsági rendszerek vezérlése, ajtó- és zárvezérlés, háztartási gépek irányítása és a hangvezérlés, de ezeken kívül számos hasznos funkcionalitást helyeznek a gyártók a piacra.

A szakdolgozat célja egy webalkalmazással irányított okos otthon rendszer megtervezése és megvalósítása, melyben egy mikrokontrolleres rendszer felelős az érzékelők (hőmérséklet, páratartalom, gáz, mozgás) adatainak gyűjtéséért és továbbításáért egy weboldalra. A weboldalon a felhasználó lehetőséget kap az adatok megtekintésére és a szenzorok távoli felügyeletére. Egy felhasználóbarát webalkalmazás áll rendelkezésre, amely megjeleníti az érzékelőktől érkező adatokat és lehetőséget biztosít a szenzorok távoli felügyeletére. A webalkalmazás intuitív felhasználói felülettel rendelkezik, amely lehetővé teszi az adatok könnyű navigálását és a szenzorok állapotának egyszerű felügyeletét. A szakdolgozatomban vizsgálni fogom ezen rendszerek működését, összetételét és a hozzájuk tartozó szenzorokat. A szakdolgozat tartalmazza a már működő rendszerek elemzését, melyekből a saját modellemet hozom létre, kitérek a mikrokontrollerek és szenzorok választási lehetőségeire, majd a megvalósítás menetéről.

2. Referencia projektek áttekintése

Ebben a pontban a szakdolgozat elkészítéséhez szükséges, irodalomkutatás céljából megvizsgált kész projektek kerülnek bemutatásra, elemzem őket és a szakdolgozatomhoz kellő információkat kigyűjtöm. A vizsgálat során a hasonló témájú vagy releváns területeken megvalósított projektek kerülnek megfigyelés alá, különös figyelmet fordítva azok célkitűzéseire, módszertanára, alkalmazott technológiáira, eredményeire, valamint az esetleges kihívásokra és megoldásokra. Az elemzések során kiemelem azokat a szempontokat, amelyek közvetlenül vagy közvetetten hozzájárulhatnak a szakdolgozatomban megfogalmazott kutatási kérdések megválaszolásához és a célkitűzések eléréséhez. Az irodalomkutatásból származó információkat strukturált módon rendszerezem, és összegzem azokat az ismereteket, amelyek beépíthetők a szakdolgozatomba. Az elemzések révén nemcsak a meglévő tudásanyag bővítése a cél, hanem új ötletek generálása és az egyéni megközelítés kialakítása is.

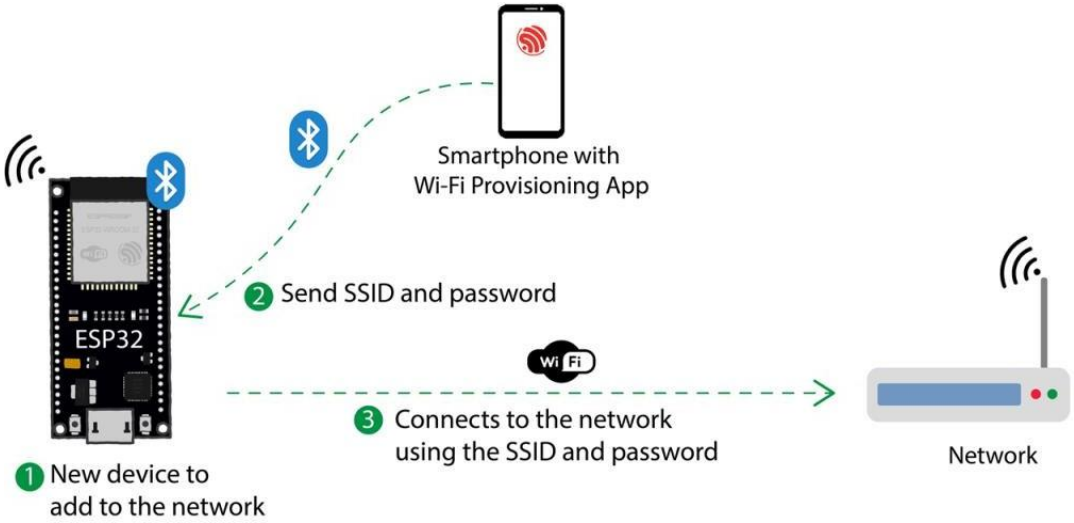
2.1 Wi-Fi Provisioning Applikáció Bluetooth Low Energy használatával

Ez egy lehetőség az IoT-projektek számára, amelyek Wi-Fi hitelesítő adatokat igényelnek az internethez való csatlakozáshoz, anélkül, hogy a fejlesztés során keményen be kellene kódolni azokat az Arduino vázlatba. A Wi-Fi üzembe helyezés egy új Wi-Fi eszköz (állomás) Wi-Fi hálózathoz (hozzáférési ponthoz) való csatlakoztatásának folyamata. Ebben az esetben egy ESP32-t szeretnénk csatlakoztatni egy Wi-Fi hálózathoz. A provisioning folyamat során az ESP32-t betöltjük annak a hálózatnak a nevével (SSID) és jelszavával, amelyhez csatlakozni szeretnénk. Az ESP32 Bluetooth Low Energy-n keresztüli üzembe helyezéséhez egy másik BLE-kompatibilis eszközzel, általában egy okostelefonnal kell csatlakoznunk az ESP32-hez BLE-n keresztül, és el kell küldenünk a Wi-Fi hitelesítő adatokat. Használhatunk egy Android vagy iOS alkalmazást, vagy egy webes Bluetooth alkalmazást.[2]

Rövid összefoglaló: Van egy új eszköz, amelyet egy ismert hálózathoz szeretne csatlakoztatni. Esetünkben egy ESP32-t. BLE-n keresztül csatlakozik az ESP32-hez. A Wi-Fi provisioning alkalmazásban megadja és elküldi annak a hálózatnak az SSID-jét és jelszavát, amelyhez az ESP32-t csatlakoztatni szeretnénk. Végül, az ESP32 a megadott hitelesítő adatokkal csatlakozik a hálózathoz. Ettől a pillanattól kezdve bármilyen más Wi-Fi-vel kapcsolatos feladatot elvégezhet. [2]

Ez a projekt segítséget nyújt számomra a Bluetooth Low Energy használatához és a Wi-Fi kapcsolat kialakításában.

Az alábbi ábrán (2.1. ábra) látható a modellezése.



2.1. ábra: ESP32 Wi-Fi Provisioning
(https://randomnerdtutorials.com/esp32-wi-fi-provisioning-ble-arduino/)

2.2 ESP32 Webszerver készítése, LED irányító csúszkával

Ebben a projektben egy ESP32 mikrokontrolleren alapuló webszervert hozunk létre egy LED-es fény vezérlésére. A projekt bemutatja a csúszka hozzáadásának módját a webszerverhez a felhasználói bevitel gyűjtéséhez, a LED beállításának lekérését a csúszkáról, és az érték tárolását egy változóban, amelyet az ESP32 később felhasználhat a LED vezérlésére. Az ESP32 kontrollert programozzuk egy webszerver futtatására, amely egy csúszkával ellátott weboldalt jelenít meg. Amikor a felhasználó a csúszkát mozgatja, HTTP-kérést küld az ESP32-nek az új csúszkaértékkel. A HTTP-kérelem a következő formátumban érkezik: GET/slider?value=SLIDERVALUE, amelyben a SLIDERVALUE egy 0 és 255 közötti szám. A HTTP-kérésből az ESP32 megkapja a csúszka aktuális értékét, az ESP32 a csúszka értékének megfelelően állítja be a PWM ciklusidejét.[3]

A bemutatott rendszer segítségével egy egyszerű webszerver létrehozásához elegendő információt kapok.

2.3 Webszerver készítése, DHT11/DHT22 szenzor használatával

Ez a projekt elkészít egy aszinkron ESP32 webszervert, melyhez hozzáköt egy DHT11 vagy DHT22 szenzort (hőmérséklet és páratartalom érzékelő). A segítségével ki tudunk olvasni értéket páratartalom és hőmérséklet szenzorból, majd ezeket le tudjuk olvasni a weblap frissítése nélkül.[4]

A szakdolgozatom elkészítéséhez fontos szenzort mutat be az előbbi projekt, mely bemutatja a webes kezelését a DHT szenzornak.

2.3 Egyszerű webszerver készítése Arduino IDE környezetben

A projekt célja: egyszerű webszerver létrehozása Arduino IDE környezetben. Tartalmazza a webszerver definícióját, az ESP32 mikrokontroller üzemmódjainak bemutatását

A projektben bemutatásra kerül egy webszerver létrehozása, az ESP32 mikrokontroller üzemmódjainak beállítása és ezek után kettő darab LED és ellenállás bekötése következik. Amikor az ESP32 megkapja ezt a kérést, felismeri, hogy a felhasználó be akarja kapcsolni a LED-et. Ennek eredményeképpen bekapcsolja a LED-et, és egy dinamikus weboldalt küld a böngészőnek, amely a LED állapotát „bekapcsoltként” jeleníti meg. A feladat ezek után részletesen bemutatja a kód működését és megvalósítását mindkettő üzemmódban. [5]

2.4 ESP32 BH1750 környezeti fényérzékelővel

A BH1750 egy 16 bites környezeti fényérzékelő, amely I2C protokollon keresztül kommunikál. A fényerősségméréseket luxban (a megvilágítás SI-egységében) adja ki. Minimum 1 lux és maximum 65535 lux mérésére képes. A BH1750 egy környezeti fényérzékelő, így számos projektben használható. Például:

- A fényérzékelővel például a nappali vagy az éjszakai világítás érzékelésére
- a LED fényerejének a környezeti fényviszonyoknak megfelelő beállítása vagy bekapcsolása
- LCD-k és képernyők fényerejének beállítására
- annak érzékelésére, hogy egy LED világít-e

Ezek után egy egyszerű projektet készít, amely leolvassa a környezeti fény mértékét, és megjeleníti azt az Arduino IDE soros monitorán.[6]

2.6 ESP32 alapú IoT eszközök tervezése és implementációja

Ez a tanulmány olyan oktatási IoT-eszközöket és -technológiákat mutat be, amelyek egyszerűsítik az IoT-alkalmazások tervezését, megvalósítását és tesztelését. A cikk bemutatja a bevezető IoT-tanfolyamot, amelyet a diákok kezdetben végeznek, majd a későbbiekben bemutatja néhány olyan projektet, amelyet önállóan fejlesztenek és valósítanak meg. [7]

Eszközök és módszerek:

Hardver: ESP32, Analog Discovery 2, egyedi interfészlapok (Grove Creator Kit, Szenzorbővítő lapok, szenzorok).

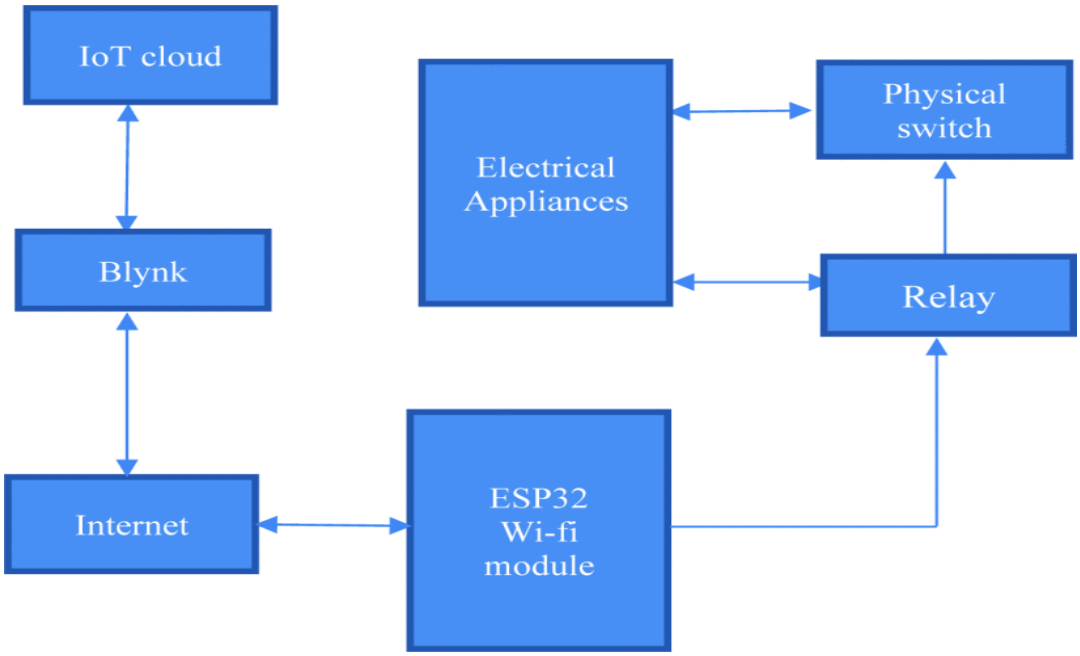
Szoftver: Arduino IDE és ESP32 könyvtárak

2.7 Hatékony otthoni automatizálás ESP32 NodeMCU használatával

Egy költséghatékony otthoni automatizálási rendszer, amely lehetővé teszi a felhasználók számára a háztartási készülékek távoli vezérlését egy Android-alkalmazás és MQTT-alapú kommunikáció segítségével. A szerzők relék és egy NodeMCU (ESP8266) mikrokontroller használatát javasolták, hogy lehetővé tegyék az elektromos kapcsolók távvezérlését egy Node-kiszolgálón keresztül. Ezek a megoldások arra összpontosítanak, hogy alacsony költségű és hatékony módot biztosítsanak a készülékek távoli irányítására. A szerzők rávilágítottak az intelligens eszközök távoli aktiválásának és kezelésének fontosságára az energiahatékonyság érdekében. Kiemelték az ESP8266 lap előnyeit, amely rendkívül alacsony energiafogyasztást kínál alacsony költség mellett, így ideális választás az IoT-alapú otthoni automatizáláshoz. Ez a technológia nemcsak az energiagazdálkodást javítja, hanem a házat intelligens és hatékony élettérré alakítja át. Agarwal és munkatársai IoT-alapú megközelítést javasoltak a vezeték nélkül programozható intelligens otthoni automatizáláshoz. Rendszerük egy dedikált, interneten vagy LAN-on keresztül elérhető weboldalt használ a háztartási készülékek okostelefonokon vagy asztali számítógépeken keresztül történő irányítására. Ez a megközelítés demonstrálja az IoT-alapú megoldások rugalmasságát és skálázhatóságát az otthoni automatizálás teljesítményének fokozásában.[8]

A bemutatott rendszer feltárja a lehetőségeit az MQTT-alapú kommunikácónak, mely lehetőséget biztosít akár több mikrokontroller használatára.

A rendszer blokk diagramja az alábbi ábrán (2.7. ábra) látható.[8]



2.7. ábra: Block Diagram

Arduino IDE

Az Arduino integrált fejlesztőkörnyezet (IDE) egy nyílt forráskódú szoftver, amelyet arra terveztek, hogy kódot fordítson és töltsön fel az Arduino kártyákra, mint például az Arduino Uno, Mega és Micro. Java nyelven íródott, kompatibilis a macOS, Windows és Linux rendszerekkel, és felhasználóbarát felületet kínál beépített eszközökkel a kód hibakereséséhez és szerkesztéséhez. [8]

Blynk IoT Cloud

A Blynk egy sokoldalú IoT platform, amelyet iOS és Android eszközökre terveztek, és lehetővé teszi a felhasználók számára, hogy távolról kezeljék az olyan hardvereket, mint az Arduino, a Raspberry Pi és a NodeMCU. Intuitív grafikus felhasználói felületén (GUI) keresztül a felhasználók könnyen konfigurálhatnak widgeteket az eszközök interneten keresztüli felügyeletéhez és vezérléséhez. A hardver és a Blynk Cloud közötti hídként működő platform feldolgozza a parancsokat, megkönnyíti az adatok megjelenítését, és támogatja a valós idejű hardvervezérlést. [8]

Blynk alkalmazás működése

A Blynk alkalmazás egyszerű módot biztosít egyéni felületek létrehozására az IoT-projektek vezérléséhez. A felhasználók olyan widgeteket adhatnak hozzá, mint a kapcsolók és csúszkák, hogy az interneten keresztül kezelhessék az eszközök állapotát. Az alkalmazás közvetítőként működik, és a Blynk felhőn keresztül felhasználói parancsokat küld a csatlakoztatott hardvereknek. [8]

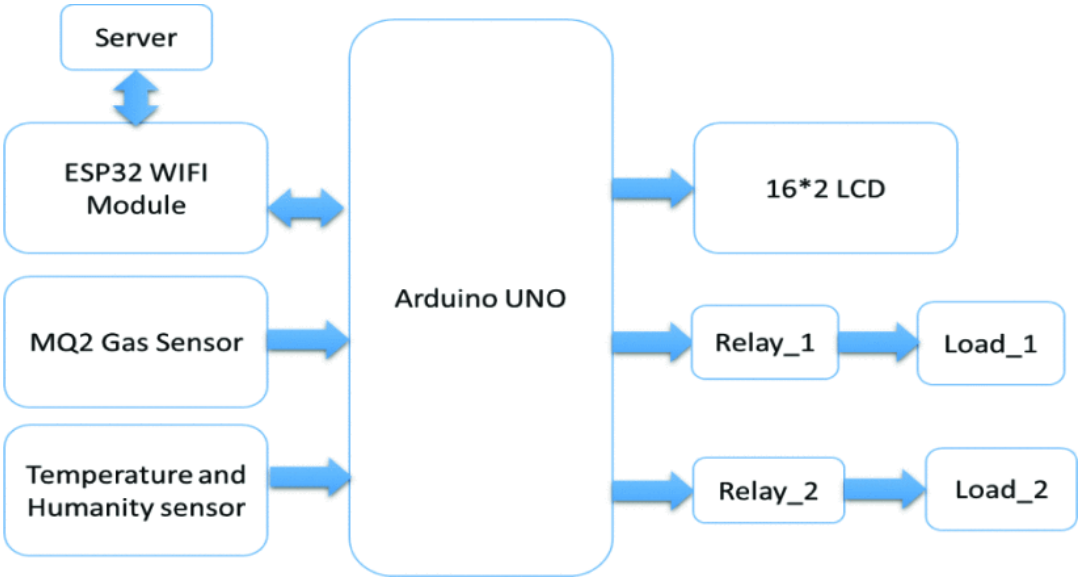
Felhasznált eszközök

NodeMCU ESP32, Relé

2.8 Egy IoT alapú intelligens otthoni automatizálási rendszer

A rendszer lehetővé teszi a regisztrált felhasználók számára, hogy egy ESP32 Wi-Fi modullal kommunikáló privát szerveren keresztül távolról, bármikor és bárhol irányítsák és felügyeljék az otthoni készülékeket. Ez a szerver, amelyet az OSI-modell alapján terveztek meg, biztonságos működést biztosít azáltal, hogy privát és illetéktelen felhasználók számára elérhetetlen. A rendszer egy Arduino UNO mikrokontrollerre és egy ESP32 Wi-Fi modulra épül. Az Arduino mikrokontroller kezeli a fő vezérlési műveleteket, beleértve az adatok lekérdezését, dekódolását és végrehajtását egy strukturált végrehajtási ciklusban. A kommunikációt és a frissítéseket az ESP32 modul kezeli, amely közvetlenül egy privát szerverhez csatlakozik. Az ESP32 frissíti az Arduinót, amikor a regisztrált felhasználó változtatásokat vagy frissítéseket végez. Jelenleg a prototípus két terhelést támogat, de a rendszer a jövőben bővíthető, hogy a felhasználói igények alapján több terhelést vagy háztartási készüléket is befogadjon. Előnyei a projektnek az alacsony költségű intelligens IoT otthoni automatizálás, energiatakarékosság, mivel ez egy automatizált rendszer, könnyen karbantartható házi terhek a szerveren keresztül, figyelemmel kísérheti az energiafogyasztást, mivel a terhelések futási ideje. Ezeken kívül a gázszivárgás érzékelése és bárki bárhonnan vezérelheti. [9]

Az alábbi ábrán (2.8. ábra) az architektúra látható.[9]



2.8.ábra: Architecture Model

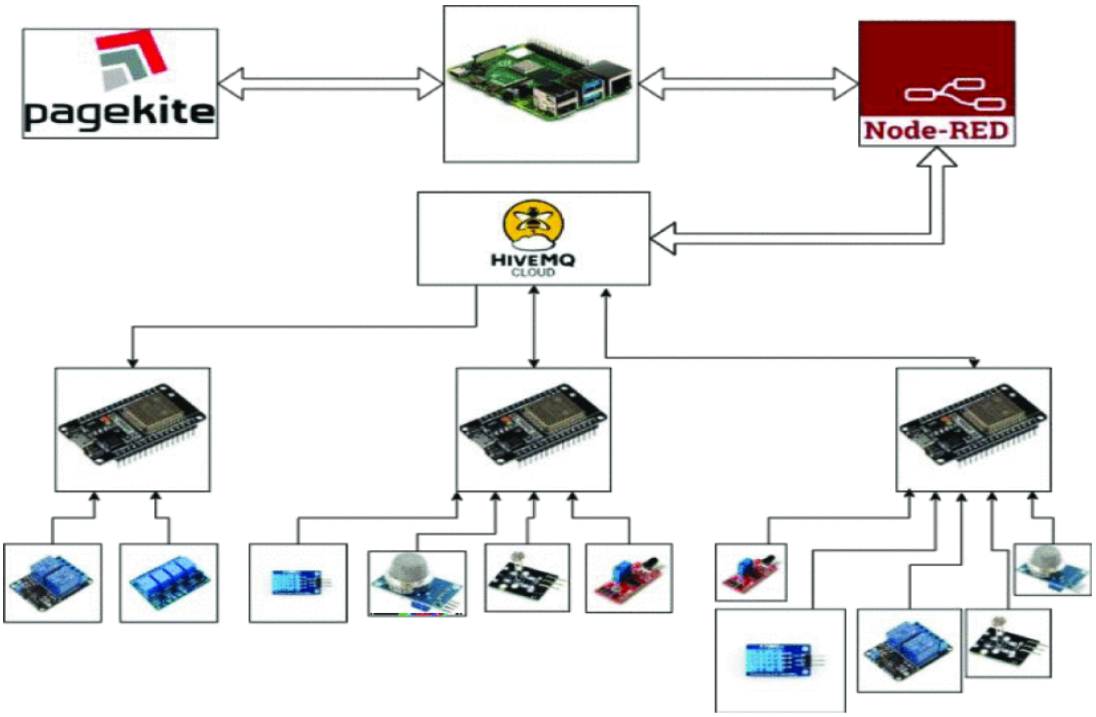
Felhasznált eszközök

Arduino UNO, ESP32 modul, relé, gáz szenzor, hőmérséklet és páratartalom mérő, LCD kijelző, égő

2.9 *Távoli elérésű otthoni automatizálás MQTT-vel: ESP32, Raspberry Pi, NodeRed*

Ez a projekt az MQTT protokollt használó újszerű megközelítést mutat be az otthoni automatizáláshoz. A rendszer három ESP32-t és egy Node-RED kiszolgálót használ, amelyek egy Raspberry Pi eszközön vannak elhelyezve, lehetővé téve a felhasználók számára, hogy a Raspberry Pi és a kliensterminálok segítségével egyaránt vezéreljék a háztartási készülékeket. A rendszer megkönnyíti az otthoni készülékek távvezérlését és felügyeletét bárhonnan, kihasználva a könnyű MQTT protokollt a csomópontok és a szerver közötti valós idejű kommunikációhoz. Ez biztosítja az érzékeny vezérlést és a hatékony adatmonitoringot. Az ESP32 csomópontok rugalmasságot és skálázhatóságot biztosítanak, míg a Node-RED intuitív felületet biztosít a rendszer kezeléséhez. A rendszer az MQTT protokollt használja a valós idejű adatátvitelhez, így a felhasználók egy felhasználóbarát felületen keresztül távolról felügyelhetik és irányíthatják otthonukat. A Raspberry Pi 4 központi átjáróként működik, integrálja a három ESP32 csomópontból származó adatokat, és MQTT közvetítőként szolgál a zökkenőmentes kommunikációhoz. Az 1. csomópont hat relé segítségével kezeli a közös teremben lévő készülékeket és a világítást, míg a 2. csomópont érzékelőkkel figyeli a külső körülményeket (pl. hőmérséklet, páratartalom és fényerősség), és automatikusan beállítja a világítást. A 3. csomópont hasonló funkciókat lát el a beltérben, átfogó környezeti felügyeletet biztosítva. A Node-RED műszerfala valós idejű adatmegjelenítést és vezérlést kínál, a Pagekite segítségével pedig távoli hozzáférést tesz lehetővé. [10]

Az alábbi ábrán (2.9. ábra) az architektúra látható.[10]



2.9.ábra: Model Architecture

Felhasznált eszközök

Raspberry Pi, ESP32, relé, DHT11 szenzor, MQ135 szenzor, KY018 szenzor, lángérzékelő

Fejlesztés

HiveMQ, Node-RED, Pagekite

2.10 Otthoni automatizálás Wi-Fi használatával: ESP32-alapú rendszer távvezérlésre és környezeti felügyeletre

A projekt célja egy otthoni automatizálási rendszer kifejlesztése Wi-Fi és ESP32 mikrokontrollerek használatával, relémodulok és DHT11 érzékelők integrálása a távvezérléshez és a környezeti felügyelethez, a rendszer funkcionalitásának, megbízhatóságának és felhasználóbarátságának tesztelése illetve a teljesítménymérők értékelése és felhasználói visszajelzések gyűjtése. [11]

Rendszertervezés

Az architektúra Wi-Fi-kapcsolatot és ESP32 mikrokontrollereket tartalmaz, mint központi egységeket, relé modulokkal a készülék vezérléséhez és DHT11 érzékelőkkel a hőmérséklet és a páratartalom ellenőrzéséhez. [11]

Hardveres beállítás

Az ESP32-t, a relémodulokat és a DHT11 érzékelőket a megfelelő csatlakoztathatóság biztosítása érdekében a kapcsolási rajzok és áramköri elrendezések alapján szereltük össze. [11]

Szoftverfejlesztés

A Wi-Fi, a relék és az érzékelők integrálásához Arduino IDE segítségével fejlesztettük ki a firmware-t, valamint egy felhasználóbarát okostelefon-alkalmazást a rendszer vezérléséhez. [11]

Rendszerintegráció

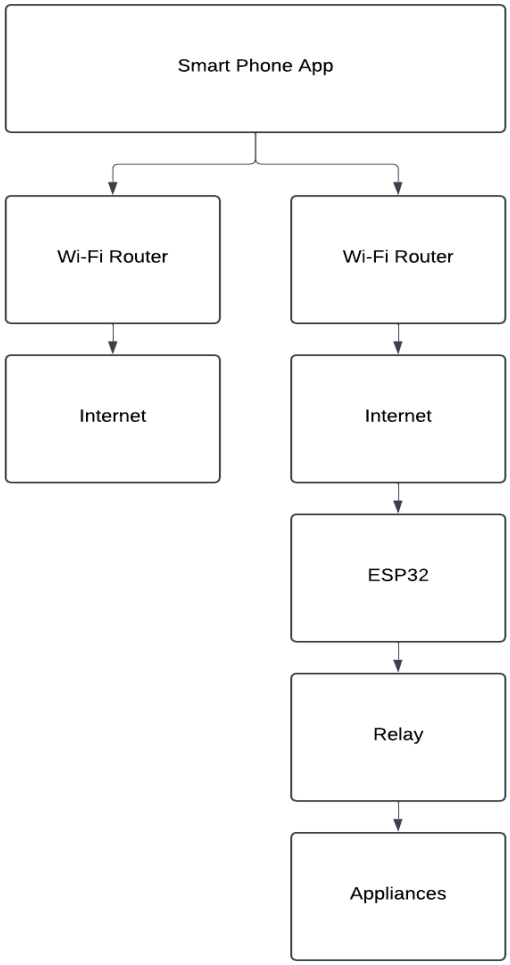
A hardvert és a szoftvert kombinálták, a Wi-Fi-t a zökkenőmentes kommunikációhoz konfigurálták, a kalibrálás pedig biztosította a pontos érzékelői leolvasásokat és a relék működését. [11]

Tesztelés és értékelés

Teszteket végeztek a távvezérlés, az érzékelők pontosságának, a válaszidőknek és a rendszer stabilitásának értékelésére, biztosítva az optimális teljesítményt. [11]

Adatelemzés

A tesztelési adatokat és a felhasználói visszajelzéseket elemezték a teljesítmény értékelése, a problémák azonosítása és a fejlesztésekhez szükséges információk összegyűjtése érdekében. Az alábbi ábrán (2.10. ábra) látható a blokk diagram.[11]

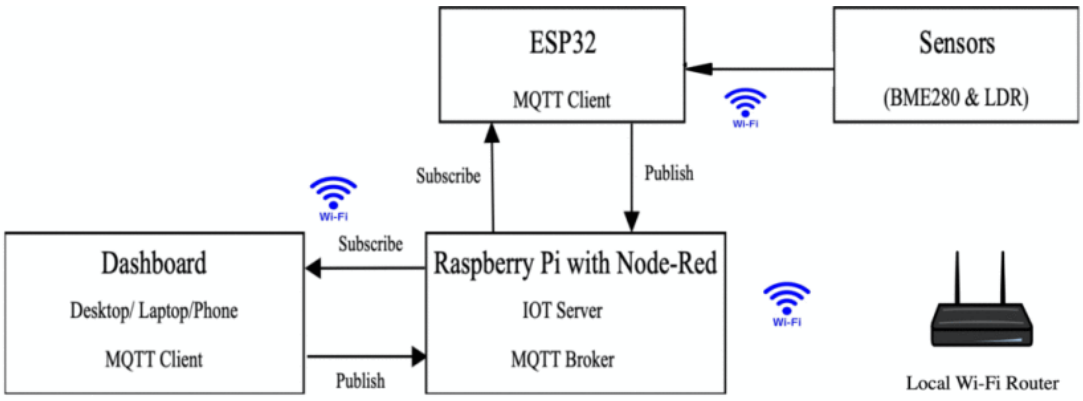


2.10.*ábra: Block Diagram*

A blokkdiagram egy átfogó otthoni automatizálási beállítást mutat be, ahol az okostelefon szolgál központi vezérlőfelületként. Az okostelefon egy Wi-Fi routerhez van csatlakoztatva, amely lehetővé teszi a kommunikációt a szélesebb internetes hálózattal. Az okostelefon és a csatlakoztatott készülékek között az ESP32 modul helyezkedik el, amely az adatfeldolgozás és -átvitel központjaként működik. Ez a modul közvetlenül kapcsolódik egy relémodulhoz, amely viszont a különböző háztartási készülékeket vezérli. Ez a konfiguráció lehetővé teszi a készülékek zökkenőmentes távvezérlését és felügyeletét az okostelefon-alkalmazáson keresztül, kihasználva a Wi-Fi és az internet által biztosított kapcsolódási lehetőségeket.[11]

2.11 ESP32, Raspberry Pi, Node-Red és MQTT protokoll alapú SCADA rendszer

Ebben a projektben egy alacsony költségű és nyílt forráskódú SCADA rendszert fejlesztenek ki, amely Wi-Fi-t, az MQTT protokollt és a Node-Redet használja. A rendszer részletes hőmérséklet-, nyomás-, páratartalom- és fénysűrűség-méréseket tud biztosítani. Emellett az otthoni eszközök állapotát is nyomon követi. Ezért a felhasználók megérthetik az otthoni eszközeik, például a világítás, a ventilátorok, a légkondicionálók vagy a fűtési/hűtési rendszerek állapotát, majd távolról vezérelhetik azokat. A SCADA-rendszer blokkdiagramja az ábrán látható. A megvalósításhoz ESP32-t használnak érzékelő átjáróként és RTU-ként, valamint Raspberry Pi2-t helyi szerverként. A SparkFun Atmospheric Breakout Sensor-BME280 érzékelő és egy LDR csatlakozik az ESP32-hez a kívánt adatok megszerzéséhez, valamint a műszerfalhoz a megszerzett érzékelőadatok webes fogadásához, megjelenítéséhez és kezeléséhez, miközben a Node-Red és az MQTT protokollt használja az adatcseréhez és az MQTT ügyfelek, például a végfelhasználók és az ESP32-hez csatlakoztatott érzékelők közötti kommunikációhoz, miközben a megszerzett adatokat SQLite-ban, mint webszerverben tárolják. [12] Az alábbi ábrán (2.11. ábra) látható a folyamat.[12]



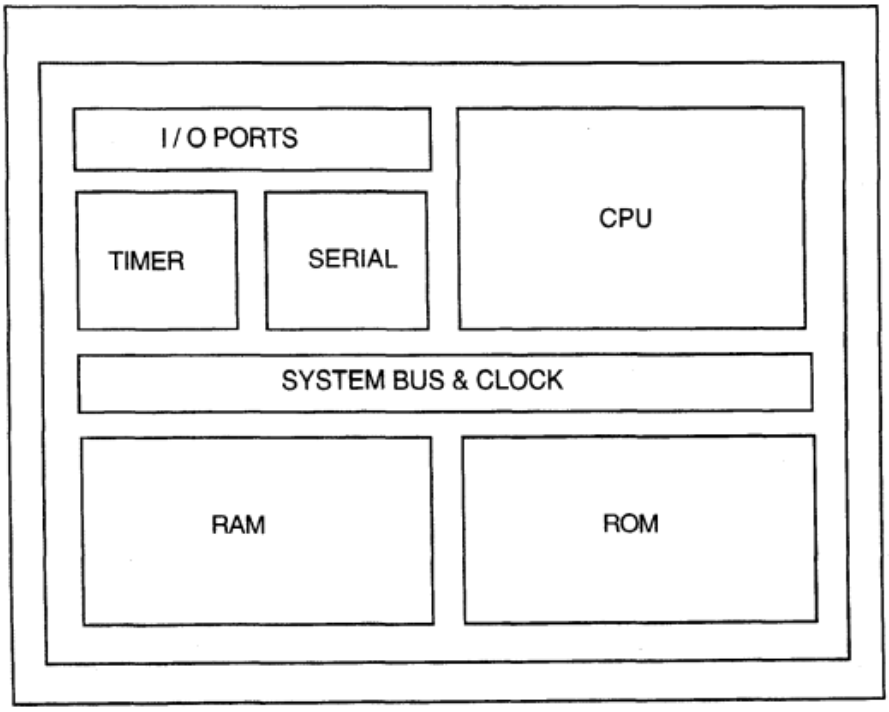
2.11.ábra: Folyamatábra

Az MQTT egy publish/subscribe protokoll. Ez egy könnyű és egyszerű üzenetküldő eszköz, amelyet megbízhatatlan hálózatokhoz, korlátozott eszközökhöz és alacsony sávszélességhez terveztek. Megfelelő megoldás ehhez a kialakításhoz, mivel egyszerű kommunikációt biztosít a kiszolgáló (MQTT bróker) és az ügyfelek (ESP32 mikrokontroller és számítógépek és mobileszközök) között.Az ügyfelek feliratkozhatnak a témákra, vagy közzétehetik az adatokat bármilyen típusú adat témáira. A bróker ezután szétosztja az adatokat minden olyan ügyfélnek, aki feliratkozott az adott témára. Az Eclipse Mosquitto szoftver brókerként fut a helyi szerveren (Raspberry Pi2).[12]

3. Mikrokontrollerek

A mikrokontroller, más néven mikrovezérlő egység (MCU), egy beágyazott rendszerkomponens, amely egyetlen chipen egyesíti a mikroprocesszort, a memóriát és a programozható bemeneti/kimeneti (I/O) perifériákat. Ezek a perifériák tipikusan tartalmaznak analóg-digitális átalakítókat (ADC), digitál-analóg átalakítókat (DAC), timereket, számlálókat, kommunikációs interfészeket (mint például SPI, I2C, UART) és impulzusszélesség-modulációs (PWM) egységeket. A mikrokontrollerek általában vezérlési feladatokra optimalizáltak, azaz arra, hogy érzékelőktől vegyenek jeleket, és azok alapján vezéreljenek különböző eszközöket. Például egy okos termosztátban a mikrokontroller érzékeli a szoba hőmérsékletét, és ennek alapján ki- vagy bekapcsolja a fűtést. Az IoT továbbfejlesztése és alkalmazási területeinek bővítése érdekében nagy teljesítményű, olcsó és alacsony fogyasztású megoldásokra van szükség a tárgyak internetének eszközeihez. Az IoT-eszközökkel szemben támasztott másik követelmény a kis méret; minél kisebb az eszköz mérete és súlya, annál szélesebb az alkalmazási területe.[13]

A megfelelő mikrokontroller kiválasztása kulcsfontosságú egy webvezérelt intelligens otthoni rendszer kiépítéséhez. A választás a projekt specifikus igényeitől függ. Fontos figyelembe venni a perifériák számát, a programozási nyelvet, a költségvetést és a kívánt funkcionalitást. Az alábbi ábrán (3. ábra) megtekinthető egy egyszerű mikrokontroller.[13]



3.ábra: Mikrokontroller

3.1 Arduino Uno

Az Arduino Uno egy gyakori választás a hasonló felépítésű projektek között mivel könnyen programozható a C++ nyelv egyszerűsített változatával, az Arduino IDE-vel. Számos online oktatóanyag és példa áll rendelkezésre. Kompatibilis számos elektronikai alkatrésszel, amelyek lehetővé teszik a felhasználók számára, hogy kiterjesszék a funkcionalitását, aktív közössége van, ahol a felhasználók segítséget és tanácsot kaphatnak egymástól. Ezek mellett pedig a megfizethető árkategóriába esik. Hátránya a többi vizsgált mikrokontrollerhez képest, hogy memóriakapacitása korlátozott, és nem tartalmaz beépített Wi-Fi vagy Bluetooth kapcsolatot, ami azt jelenti, hogy az eltervezett funkciók megvalósításához további hardverokra van szükség.[14]

3.2 STM32F407xx

Az STMicroelectronics STM32F407xx egy ARM Cortex M4 processzor alapú 32 bites mikrokontroller (MC), amely a nagy hatékonyságú digitális feldolgozáshoz készült. Képes 210 millió utasítás másodpercenkénti feldolgozására, és képes egyciklusú szorzásra és felhalmozásra. Ennek a mikrokontrollernek néhány fontos jellemzője a következő: 1 MB Flash memória, 192 KB RAM, amely 168 MHz-en (max.) működik, és 210 MIP csúcsátviteli teljesítményt biztosít, ST-LINK/V2 hibakereső a hardver szintű hibakereséshez. 3×12 bites A/D átalakítók: akár 24, 2×12 bites D/A átalakítók. Általános célú közvetlen memóriaelérés (DMA): 16-folyamos DMA vezérlő FIFO-kkal és burst támogatással. Akár 17 időzítő: akár tizenkét 16 bites és két 32 bites időzítő 168 MHz-ig, kiegészítő PWM-csatorna programozható holtidő-behelyezéssel. 15 kommunikációs interfész, mint például UART, I2C és SPI. Legfeljebb 140 általános célú be- és kimenet (GPIO). [15]

Hátránya a projekt szempontjából a beépített Wi-Fi és Bluetooth modul hiánya.

3.3 NodeMCU ESP8266

A NodeMCU egy nyílt forráskódú, Lua-alapú firmware és fejlesztői kártya, amely kifejezetten IoT-alapú alkalmazásokhoz készült. Tartalmazza az Espressif Systems ESP8266 Wi-Fi SoC-jén futó firmware-t, valamint az ESP-12 modulon alapuló hardvert. A NodeMCU ESP8266 fejlesztői kártya az ESP-12E modult tartalmazza, amely a Tensilica Xtensa 32 bites LX106 mikroprocesszorral rendelkező ESP8266 chipet tartalmazza. Ez a mikroprocesszor támogatja az RTOS-t, és 80 MHz és 160 MHz közötti állítható órajelen működik. A NodeMCU 128 KB RAM-mal és 4 MB Flash memóriával rendelkezik az adatok és programok tárolására. A nagy feldolgozási teljesítménye a beépített Wi-Fi / Bluetooth és Deep Sleep Operating funkciókkal ideális az IoT projektekhez. Költséghatékony, kis mérettel rendelkezik, alacsony az áramfelvétele, tartalmaz beépített Wi-Fi modult, programozható Arduino IDE környezetben, mely fontos lesz a későbbiekben a fejlesztői környezetek összehasonlításánál és választásánál, ezek mellett rengeteg dokumentáció található róla a nagy közösségével együtt.[16] Az alábbi ábrán (3.3. ábra) látható a NodeMCU ESP8266.

3.3.ábra: NodeMCU ESP8266
(https://www.flyrobo.in/blog/what-is-nodemcu-esp8266)

3.4 ESP32

Az ESP32 nagymértékben integrált, beépített antenna kapcsolókkal, RF balun, teljesítményerősítővel, alacsony zajszintű vételi erősítővel, szűrőkkel és energiagazdálkodási modulokkal. Az ESP32 magas funkcionalitást és sokoldalúságot biztosít az alkalmazások számára, minimális nyomtatott áramköri lap (PCB) igény mellett. [17]

Rendszer és memória

Az ESP32 egy kétmagos rendszer két Harvard architektúrájú Xtensa LX6 CPU-val. Az összes beágyazott memória, külső memória és periféria ezen CPU-k adatbuszán és/vagy utasításbuszán található. A mikrokontroller két maggal rendelkezik. Az adat- és utasításbusz címtartománya egyaránt 4 GB, a perifériák címtartománya pedig 512 KB. Ezenkívül a beágyazott memóriák 448KB ROM, 520KB SRAM és két 8KB RTC memória. A külső memória legfeljebb négyszer 16MB Flash-t támogat.[17]

Óra és időzítő

Az ESP32 használhatja vagy a 320 MHz-es belső fáziszárt hurkot (PLL) vagy egy külső kristályt. Lehetőség van egy oszcilláló áramkör használatára is óraforrásként 2-40MHz- en a CPU CLK főóra generálásához mindkét CPU mag számára. Ez az órajel lehet akár 160MHz is a nagy teljesítmény érdekében, vagy alacsonyabb az energiafogyasztás csökkentése érdekében. Minden más órajelet, például a perifériák APB_CLK-ját a master órajel vezérli.
Ezen kívül számos alacsony fogyasztású órajel is rendelkezésre áll, mint például a belső RTC_CLK, amelynek alapértelmezett frekvenciája 150 kHz, és lehetőség van a mély alvó (Deep Sleep) üzemmódokhoz való beállítására.
Négy 64 bites időzítő van általános célokra 16 bites előskálázókkal, 2 és 65536 közötti tartományban. Mindegyik időzítő az APB órajelet használja, általában 80 MHz-es frekvencián.
Ezek az időzítők felfelé vagy lefelé számolhatnak, befagyaszthatók és eseményeket indíthatnak. A 4 általános időzítőn kívül vannak időzítők is a PWM-vezérlő meghajtására.
Van 8 nagysebességű és 8 alacsony sebességű PWM-csatorna, amelyek mindegyikét négy időzítő vezérli.[17]

Blokk diagram és működés

Az ESP32 mikrokontroller felépítését úgy tervezték, hogy a következő protokollok szerint működjön: TCP/IP, teljes 802.11 b/g/n/e/i WLAN MAC és Wi-Fi Direct specifikáció. A mikrokontroller képes az elosztott vezérlő funkció (DCF) protokoll szerinti alapvető szolgáltatáskészlet (BSS) STA és Soft-AP műveleteket biztosítani.

Támogatja a legújabb Wi-Fi P2P protokollnak megfelelő P2P csoportos működést is. Így állomásként működhet, és csatlakoztatható az internethez vagy a kiszolgálóhoz és a hozzáférési ponthoz, hogy felhasználói felületet biztosítson például egy mobilalkalmazást futtató okostelefon számára. Az alábbi képen a blokk diagram látható.[17]

A mikrokontroller támogatja a v4.2 BR/EDR és a BLE Bluetooth-t, amely megfelel a jelenlegi szabványnak, és akár 4 Mbps sebességgel is képes működni.
Az ESP32 különböző energiaellátási módokban működhet - aktív üzemmódban (a

chip rádiója működik) és modem-alvó üzemmódban (a CPU teljesen működőképes, de a Wi-Fi és a Bluetooth ki van kapcsolva), Továbbá vannak könnyű és mély alvó üzemmódok, ahol vagy mindkét, vagy csak az egyik CPU alacsonyabb teljesítményen működik.

A GPIO-k közé tartozik két 12 bites ADC, összesen 18 csatornával. Ezek 9 bites, 10 bites és 12 bites felbontásra konfigurálhatók, -0dB, -6dB vagy -11dB csillapítással a különböző bemeneti tartományokhoz.

Az egyik ADC-csatorna a mágneses mezők érzékelése érdekében azintegrált Hall-szenzorhoz, míg a másik a -40°C és 125°C közötti tartományban működő hőmérsékletszenzorhoz csatlakozik a chip hőmérsékletének megfigyelésére.

Az ADC-k mellett két 8 bites DAC is található a digitális jelek analóg feszültségjel-kimenetekké történő átalakítására.

A GPIO-k közül tíz képes a kapacitív változások érzékelésére, és érintésérzékelőkhöz használható. Az ESP32 továbbá számos interfészt biztosít: egy Ethernet MAC interfész, egy SD/SDIO/MMC Host Controllert, három UART interfész 5Mbps-ig, két I2C busz interfész standard és gyors üzemmóddal, két I2C interfész 10kHz- től 10MHz-ig, egy 8 csatornás infravörös távvezérlő és egy 8 csatornás impulzusszámláló. A PWM-vezérlő digitális motorok meghajtására vagy digitális hullámformák létrehozására használható. Három SPI használható „slave" vagy „master" üzemmódban, akár 80MHz-es órajelen.[17]

3.5 Összehasonlítás

Az előző pontban taglaltak megtekinthetőek összehasonlító táblázat (3.5. ábra) formájában.[17]

Adatok	**ESP32**	**NodeMCU ESP8266**
Processzor	Xtensa Dual-Core 32-bit LX6 600 DMIPS	Xtensa Single-core 32-bit L106
CPU sebesség	80MHz	160MHz
Wi-Fi	Van beépítve	Van beépítve
Bluetooth	Van (BLE)	Nincs
Digitális Inputok/Outputok	Több	Több
Költség	Magasabb	Alacsonyabb
Méret	Nagyobb	Kisebb
Fejlesztési környezet	Arduino IDE vagy ESP-IDF	Arduino IDE

3.5.ábra: összehasonlító táblázat

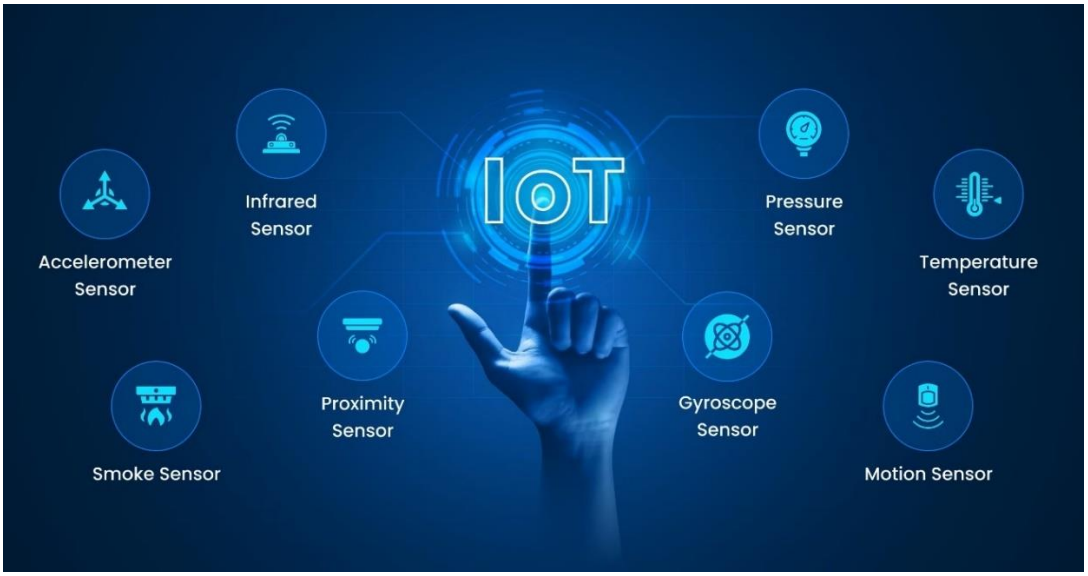
3.6 Összesítés

Az ESP32 gyorsabb, mint az ESP8266, több GPIO-val és több funkcióval rendelkezik, támogatja az analóg méréseket 18 csatornán, támogatja a Bluetooth-t, míg az ESP8266 nem, az ESP32 kétmagos, az ESP8266 pedig egymagos, azonban az ESP8266-nek kevesebb a költsége és szélesebb közösséggel rendelkezik források szempontjából. Mindkét kártya programozható az Arduino IDE vagy más támogatott IDE segítségével.

Mind az ESP32, mind az ESP8266 nagy teljesítményű mikrokontrollerek, amelyek jól alkalmazhatók webvezérelt intelligens otthoni rendszerekhez. A szakdolgozat elkészítése szempontjából a legfontosabb tényezők a Wi-Fi kapcsolat, mely mindkettőben megtalálható, a bővíthetőség, az ár-érték arány megtartása, megfelelő kommunikációs protokoll.

4. Szenzorok

Az érzékelők/szenzorok az IoT-rendszerek alapvető elemei, amelyek adatokat gyűjtenek a környezetből, és azokat feldolgozható és továbbítható formába alakítják.[18] A szakdolgozatomban fontos szerepe van a szenzoroknak, ezek az eszközök gyűjtik be a környezetből származó adatokat. Az alábbi ábra (4. ábra) illusztrál pár szenzor fajtát.



4.ábra: Szenzorok(https://www.intuz.com/guide-on-top-iot-sensor-types)

4.1 Közelségérzékelő

A közelségérzékelővel bármely közeli tárgy pozíciója könnyen, fizikai érintkezés nélkül is érzékelhető. Az elektromágneses sugárzás, például infravörös sugárzás kibocsátásával úgy találja meg egy tárgy jelenlétét, hogy egyszerűen a visszatérő jel bármilyen változását keresi.[19] Különböző típusú közelségérzékelők vannak, mint például

- induktív
- kapacitív
- ultrahangos
- fotoelektromos
- mágneses

Ezek különböző alkalmazásokat céloznak meg. Ezt a különleges érzékelőtípust leginkább a biztonságot és hatékonyságot igénylő alkalmazásokban használják. Az ilyen típusú érzékelők különböző alkalmazási területei a tárgyak érzékelése, a tárgyak számának számlálása, a forgás mértékének mérése, a tárgyak pozicionálása, az anyagérzékelés, a mozgásirány mérése, a parkolási érzékelők stb. A közelségérzékelőket számos iparágban használják a legjobban.[19]

4.2 Pozícióérzékelők

A helyzetérzékelő egy adott területen lévő ember vagy tárgyak jelenlétét érzékeli mozgásuk érzékelésével. Használható az otthoni biztonságban, hogy a tulajdonos bárhonnan nyomon követhesse a szobák és készülékek ajtóit és ablakait. Mindig tudatja velük a nyitott vagy zárt állapotot, és a betolakodókat a távollétükben is nyomon követheti. Használható az egészségügyben a betegek, ápolók és orvosok helyzetének nyomon követésére egy kórházban, a mezőgazdaságban a szarvasmarhák helyzetének észlelésére.[19]

4.3 Sebességérzékelők

Olyan érzékelő, amely állandó pozíciómérés és ismert időközönkénti pozícióértékek változásának sebességét számítja ki. A sebességérzékelő lehet lineáris vagy szögletes. A lineáris sebességérzékelő egy tárgy sebességét érzékeli egy egyenes vonal mentén, míg a szögsebességérzékelő azt érzékeli, hogy milyen gyorsan forog egy eszköz.[19]

4.4 Nyomásérzékelők

A nyomásérzékelők érzékelik az erő nagyságát és jelekké alakítják azt. Az ilyen típusú érzékelők az egészségmegőrzésben használhatók.[19]

4.5 Optikai érzékelők

Az optikai érzékelők az elektromágneses energiák, például a fény érzékelésére alkalmasak. Mivel passzívak az elektromos interfészek minden formájával szemben, ezeket széles körben használják a tárgyak interneteiben, például a digitális kamerákban. Az optikai érzékelők jól használhatóak az energiával, az egészségüggyel, a környezettel, az olajfinomítókkal, a vegyiparral, az iparral, a repülőgépiparral stb. kapcsolatos IoT-alkalmazásokban.[19]

4.6 Hőmérséklet és páratartalom szenzorok

A hőmérséklet- és páratartalom-érzékelő alacsony költségű, érzékeny elektronikus eszközök, amelyek érzékelik, mérik és jelentik a nedvességet és a levegő hőmérsékletét. Ez az egyik legfontosabb eszköz, amely széles körben elterjedt a fogyasztói, ipari, orvosbiológiai és környezetvédelmi stb. alkalmazásokban a hőmérséklet és a páratartalom mérésére és nyomon követésére egy adott helyen, különösen egy adatközpontban vagy egy szerverszobában. A piacon kapható hőmérséklet-páratartalom jeladók általában mérik a levegőben lévő hőmérséklet és relatív páratartalom mennyiségét, és bizonyos szabályok szerint elektromos jelekké vagy más jelformákká alakítják, és kimenetként a készüléket a műszerre vagy szoftverre adják ki, hogy megfeleljenek a felhasználók környezeti felügyeleti

igényeinek.[20]

Kettő típusú szenzort választottam ki bemutatásra, egyik a DHT11, a másik pedig a DHT22. Szempont volt a bemutatás céljából az optimalizálás, egy eszközön belül vizsgálható a környezet hőmérséklete és páratartalma, így nem szükséges két különböző szenzort beépíteni.

DHT11/DHT22

A DHT22 érzékelő a hőmérséklet és a páratartalom szabályozására szolgáló érzékelő, amely analóg feszültségkimenettel rendelkezik, amely mikrokontrollerrel tovább feldolgozható. A DHT22 modul egy recesszív elemként besorolt modul, mint például egy hőmérsékletmérő eszköz, amely előnyökkel rendelkezik az érzékelési adatok minőségének leolvasása szempontjából, amely érzékenyebb és gyorsabb a hőmérséklet és a páratartalom érzékelése szempontjából.[21]

A DHT11 érzékelő érzékeli a páratartalmat és a hőmérsékletet a környezetből. Ez az érzékelő 0 °C és 50 °C közötti hőmérsékletet és 20 % és 90 % közötti páratartalmat tud mérni ±1 °C és ±1 % pontossággal [21]. 3,5V és 5,5V közötti üzemi feszültségen és legfeljebb 0,3mA üzemi áramerősségen belül működik. A DHT11 érzékelőmodul a hőmérséklet mért értékét Celsius fokban adja meg.[22]

A DHT22 érzékelő jobb felbontással és szélesebb hőmérséklet- és páratartalom-mérési tartományban működik. Azonban egy kicsit drágább, és csak 2 másodperces időközönként kérhet leolvasást. A DHT11 szenzornak kisebb a mérési tartománya, éskevésbé pontos. Viszont másodpercenként kérheti az érzékelő leolvasásait. Ez egy kicsit olcsóbb is. Különbségeik ellenére hasonlóan működnek, és ugyanazt a kódot használhatja a hőmérséklet és a páratartalom leolvasására. Csak a kódban kell kiválasztani a használt érzékelőtípust.

4.7 Gáz/levegőminőség szenzorok

A környezeti gázok érzékelése és mérése nélkülözhetetlenné vált a különböző területeken, és a balesetek megelőzésétől, a berendezések meghibásodásának elkerülésétől a légszennyezettségre való figyelmeztetésig. és a megfelelő gázkeverék biztosítása a betegek számára a kórházakban. A gázszivárgás nagy méreteket ölthet, egész városnegyedeket vagy akár városokat is érinthetnek, hatalmas környezeti hatásokat okozva. [23]

A gáz és levegőminőség érzékelők közül az MQ szériát választottam bemutatásra, ezek pedig az MQ-2, MQ-3, MQ-4, MQ-135 szenzorok.

MQ2

Az MQ-2 gázérzékelőt a folyékony prémiumgázban (LPG) lévő gáz kimutatására használják. Az MQ-2 érzékelő ólom-dioxid (SnO2) anyagból készül, amelynek vezetőképessége tiszta levegőben nagyon alacsony. Ez a gázérzékelő érzékeny a butángázra és más típusú földgázokra, például a füstre (CO) és az alkoholra. Az MQ-2 érzékelő használható éghető gázok, például metán kimutatására is. [24]

MQ3

Az MQ-3 modul képes alkoholt, benzolt, CH4-et, hexánt, LPG-t és CO-t érzékelni. Az MQ-3 gázérzékelő alkalmas az alkohol kimutatására, ez a szenzor használható alkoholszondában. Nagy érzékenységgel rendelkezik az alkoholra és kis érzékenységű a benzolra. Az érzékeny MQ-3 gázérzékelő érzékeny anyaga az SnO2. A levegőben a szenzor vezetőképessége megnő az alkohollal együtt gázkoncentráció emelkedésével párhuzamosan egy egyszerű áramkör átalakítja a változást. vezetőképesség változását a gáz megfelelő kimeneti jelévé alakítja át.[25]

MQ4

Az MQ-4 modul képes a metán, CNG gáz érzékelésére. Az MQ3 alkoholérzékelő 5V DC feszültségen működik és körülbelül 750mW-ot fogyaszt. A metán, földgáz koncentrációját 300 ppm - 10000 ppm között képes érzékelni. [26]

MQ-135

Az MQ-135 egy olyan érzékelő, amely a levegő minőségét vagy szennyezettségét méri. Az MQ-135 gázérzékelő egy sokoldalú modul, az ammónia, az alkohol, a benzol, a füst, a CO2 és számtalan más gáz kimutatására.[27]

4.8 Mozgásérzékelő szenzorok

A mozgásérzékelők nélkülözhetetlenné váltak a modern biztonsági rendszerek számára, forradalmasítva otthonaink és vállalkozásaink védelmét. Ezeket a szerény eszközöket úgy tervezték, hogy érzékeljék a hatótávolságukon belüli mozgást, és azonnal reagáljanak, ha bármilyen gyanús tevékenységet észlelnek.

Infravörös mozgásérzékelő szenzor

Minden tárgy, beleértve az emberi testet is, az abszolút nulla (0 Kelvin / -273,15 °C) feletti hőmérsékleten infravörös sugárzás formájában hőenergiát bocsát ki. Minél melegebb egy tárgy, annál több sugárzást bocsát ki.

A PIR (Passive Infrared Receiver) egy infravörös alapú érzékelő. Az infravörös érzékelőkkel ellentétben azonban a legtöbb érzékelő egy IR LED-ből és egy fototranzisztorból áll. A PIR nem bocsát ki semmit, mint egy IR LED. Mivel a neve "passzív", ez az érzékelő csak a passzív infravörös sugarak energiájára reagál, amelyet minden általa érzékelt tárgy birtokol. Az ezzel az érzékelővel érzékelhető tárgy általában az emberi test. Ez a PIR-érzékelő úgy működik, hogy megragadja a passzív infravörös sugarakból származó hőenergiát, amellyel minden tárgy rendelkezik, ha a tárgy hőmérséklete abszolút nulla fölött van. Mint az emberi test, amelynek testhőmérséklete körülbelül 32 Celsius-fok, ami egy tipikusan meleg hőmérséklet, amely a környezetben található. [20] A PIR-érzékelők közül az egyik legnépszerűbb a HC-SR501.

4.9 LED

A világító diódák (LED-ek) a világítástechnika legnépszerűbb fényforrása. A LED-technológia az elmúlt évtizedekben virágzott. A nagy hatékonyság, a megbízhatóság, a robusztus felépítés, az alacsony energiafogyasztás és a tartósság a nagy fényerejű LED- eken alapuló szilárdtest-világítás gyors fejlődésének kulcstényezői közé tartoznak. A hagyományos fényforrások, például az izzószálas izzók és a fénycsövek vagy izzástól, vagy a gázok kisülésétől függenek. E két folyamatot nagy energiaveszteségek kísérik, amelyek a magas hőmérsékletnek és a nagy Stokes-eltolódási jellemzőknek tulajdoníthatók. Másrészt a félvezetők lehetővé teszik a fény előállításának hatékony módját. A félvezető anyagokból készült LED-ek képesek az elektromosságot közel egységnyi hatásfokkal fénnyé alakítani. [28]

Az érzékelőket minden IoT-alkalmazásban használják. A különböző típusú érzékelők és az IoT intelligens alkalmazásainak elemzése után ez ábra bemutatja, hogy milyen típusú érzékelőkre van szükség egy IoT-alkalmazásban az intelligens világ megteremtéséhez. Az alábbi táblázat (4.9. ábra) mutatja a javaslatokat.[19]

IoT Applications	Type of Sensors
Smart City	Velocity, Light, Accelerometer, Position, Temperature, Proximity, Humidity, Pressure, Infrared
Smart Environment	Light, Temperature, Humidity, Chemical, Gyroscope, Bio Sensors, Chemicals, Accelerometer, Optical
Smart Water	Temperature, Humidity, Occupancy, Water Quality
Smart Building	Light, Accelerometer, Chemical, Gyroscope, Magneto
Smart Health	Light, Gyroscope, Biosensors, Chemicals, Magneto, Accelerometer, Pressure
Smart Home	Light, Gyroscope, Biosensors, Chemicals, Magneto, Accelerometer, Temperature, Proximity, Position, Infrared
Smart Transport	Gyroscope, Pressure, chemicals, Magneto, Accelerometer, Temperature, Motion, Infrared
Smart Security	Light, Gyroscope, Chemical, Magneto, Accelerometer, Temperature, Infrared
Smart Agriculture	Temperature, Humidity, Water Quality, Chemical, Proximity, Position
Smart Retail	Light, Gyroscope, Chemical, Magneto, Accelerometer, Pressure, Position

4.9.ábra: Type of sensors to applications

5. Fejlesztői környezetek

5.1 ESP-IDF

Az ESP-IDF (Espressif IoT Development Framework) egy hivatalos fejlesztői keretrendszer az ESP32, ESP32-S, ESP32-C és ESP32-H SoC-sorozatokhoz. Önálló SDK-t biztosít bármilyen általános alkalmazásfejlesztéshez ezeken a platformokon, olyan programozási nyelvek használatával, mint a C és a C++. Az ESP-IDF jelenleg több millió eszközzel működik, és lehetővé teszi a hálózatra csatlakoztatott termékek széles skálájának létrehozását, az egyszerű villanykörtéktől és játékoktól kezdve a nagy készülékekig és ipari eszközökig. A legátfogóbb hozzáférést kínálja az ESP32 hardveres jellemzőihez és funkcióihoz. Alkalmas professzionális fejlesztésekhez és erőforrás- igényes projektekhez. Nyílt forráskódú. az ESP-IDF szabadon elérhető a GitHubon. Az ESP-IDF összetevőinek többsége forráskódban elérhető az Apache 2.0 licenc alatt. Az ESP-IDF számos szoftverkomponenst támogat, beleértve az RTOS-t, a perifériás illesztőprogramokat, a hálózati vermet, a különböző protokoll implementációkat és a gyakori alkalmazási esetekhez szükséges segédprogramokat. A nyílt forráskódú és szabadon elérhető fejlesztői eszközök, valamint a hivatalosan támogatott Eclipse és VSCode IDE-k biztosítják a fejlesztők számára az egyszerű használatot. [29]

5.2 Arduino IDE

Az Arduino IDE (Integrated Development Environment) egy nyílt forráskódú szoftver, amely kód írására és feltöltésére szolgál az Arduino kártyákra. Az IDE alkalmazás különböző operációs rendszereken, például Windows, Mac OS X és Linux operációs rendszereken használható. Támogatja a C és a C++ programozási nyelveket. Az Arduino IDE egy kezdőbarát, egyszerű felülettel és hatalmas online közösségi forrásokkal, támogatja az ESP32 lapokat előre telepített könyvtárakkal és példákkal. Könnyen használható gyors prototípusok készítéséhez és az alapvető programozási fogalmak elsajátításához. Problémát jelent a más keretrendszerekhez képest korlátozott funkcionalitás az összetett projektekhez, nem ideális memóriaszűkös alkalmazásokhoz. Kevésbé rugalmas a fejlesztés az ESP-IDF-hez képest. [30]

5.3 PlatformIO

A PlatformIO legfontosabb jellemzői:

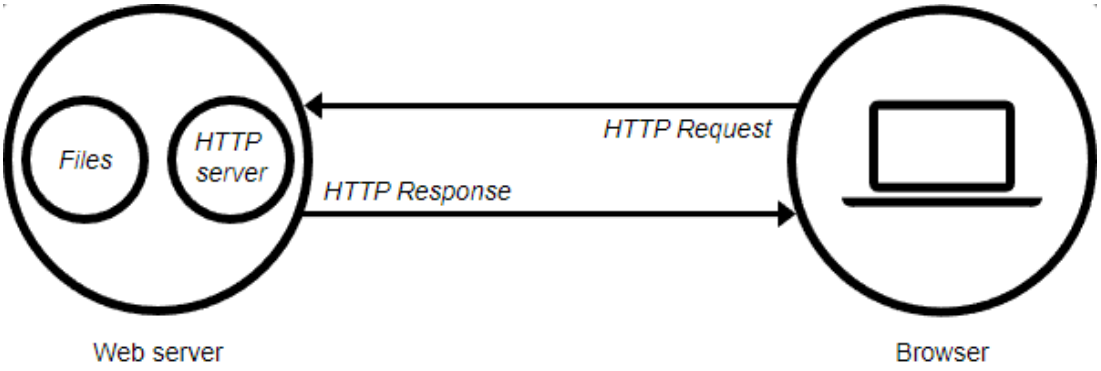
PlatformIO több mint 40 platformot, 20 keretrendszert és 1500 lapot támogat, lehetővé téve a fejlesztők számára, hogy több hardver- és szoftverkörnyezetben zökkenőmentesen dolgozzanak. Felhasználóbarát felületet biztosít olyan funkciókkal, mint az intelligens kódkiegészítés, a több projektre kiterjedő munkafolyamat és a beépített terminál. A PlatformIO átfogó hibakeresési környezetet kínál, amely kompatibilis a különböző hibakereső szondákkal és célrendszerekkel, megkönnyítve a hatékony hibaelhárítást és kódoptimalizálást.

Egységtesztelésre és kódelemzésre szolgáló eszközöket tartalmaz, amelyek lehetővé teszik a fejlesztők számára, hogy a fejlesztési ciklus korai szakaszában felismerjék a lehetséges hibákat, ezen kívül a PlatformIO lehetővé teszi a távoli fejlesztést. [30]

Összességében a PlatformIO egyesíti az Arduino IDE egyszerű használatát az ESP-IDF teljesítményével, támogat több kártyát és keretrendszert, beleértve az ESP32-t és az Arduino core-t, ezek mellett olyan fejlett funkciókat kínál, mint a kódkiegészítés, hibakeresés és verziókezelés, azonban bonyolultabb az Arduino IDE-hez képest.

6. Webszerver

A webszerver az a hely, ahol a weboldalakat tárolják, feldolgozzák és kiszolgálják a webes ügyfeleknek. A webes ügyfél nem más, mint egy webböngésző, amelyet számítógépünkön és telefonunkon használunk. A webes ügyfél és a webkiszolgáló egy speciális protokoll, a HTTP (Hypertext Transfer Protocol) segítségével kommunikál egymással. A legalapvetőbb szinten, amikor egy böngészőnek szüksége van egy webkiszolgálón tárolt fájlra, a böngésző HTTP-n keresztül kéri a fájlt. [31] Az alábbi ábra (6. ábra) bemutatja egy kérés-válasz működését.



6.ábra: Webserver (https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server)

Amikor a kérés eléri a megfelelő (hardveres) webkiszolgálót, a (szoftveres) HTTP-kiszolgáló elfogadja a kérést, megtalálja a kért dokumentumot, és visszaküldi azt a böngészőnek, szintén HTTP- n keresztül. Egy weboldal közzétételéhez statikus vagy dinamikus webkiszolgálóra van szükség. A statikus webkiszolgáló vagy stack egy számítógépből (hardver) és egy HTTP-kiszolgálóból (szoftver) áll. Azért nevezzük „statikusnak”, mert a kiszolgáló a tárolt fájlokat úgy küldi el a böngészőnek, ahogy vannak. A dinamikus webkiszolgáló egy statikus webkiszolgálóból és egy további szoftverből, leggyakrabban egy alkalmazáskiszolgálóból és egy adatbázisból áll. Azért nevezzük „dinamikusnak”, mert az alkalmazáskiszolgáló frissíti a tárolt fájlokat, mielőtt a HTTP-kiszolgálón keresztül elküldi a tartalmat a böngészőnek. Például a böngészőben látható végleges weboldalak létrehozásához az alkalmazáskiszolgáló egy HTML-sablont tölthet fel egy adatbázisból származó tartalommal. Az olyan webhelyek, mint az MDN vagy a Wikipedia több ezer weboldallal rendelkeznek. Az ilyen típusú webhelyek jellemzően csak néhány HTML-sablonból és egy hatalmas adatbázisból állnak, nem pedig több ezer statikus HTML-dokumentumból. Ez a felállás megkönnyíti a tartalom karbantartását és szállítását. [31]

A webszerver és a kommunikációs technológiák rendkívül fontos szerepet játszanak egy okosotthon ESP32 alapú megvalósításában. A webszerver szerepe: helyi vagy távoli hozzáférés biztosítása, eszközök irányítása, állapotok figyelése.

6.1 *A webszerver jellemzői*

Protokoll: kommunikációhoz használt protokollok, amelyek meghatározzák, hogyan történik az adatcsere a webszerver és a kliens között (pl. böngésző). [31]

HTTP (HyperText Transfer Protocol):

Az alapvető protokoll, amelyet a webszerverek használnak az adatok továbbítására. Nem titkosított, az adatforgalom szabadon olvasható, így érzékeny információk kiszivároghatnak.[31]

HTTPS (HTTP Secure):

A HTTPS az HTTP protokoll biztonságos változata. SSL/TLS tanúsítványt használ az adatforgalom titkosítására, így megvédi az adatokat az illetéktelen hozzáféréstől.[31]

Működési mód:

Szinkron szerverek:

Minden egyes klienskérés feldolgozása külön szálat vagy folyamatot indít.

Egyszerűbb implementáció, de nagy forgalomnál nem skálázható jól, mert sok rendszererőforrást fogyaszt.[32]

Aszinkron szerverek:

Egyetlen szálon vagy eseményvezérelt modell segítségével kezeli az összes kérést. Hatékonyabb, különösen nagy forgalmú alkalmazások esetén. Ideális a valós idejű alkalmazásokhoz, mint a chat, a streaming, vagy az IoT.[32]

Statikus és dinamikus tartalmak

Statikus tartalom:

HTML, CSS, JavaScript fájlok SPIFFS vagy LittleFS tárolóból**:** beépített fájlrendszerek, mint a SPIFFS vagy a LittleFS, lehetővé teszik statikus webes tartalmak tárolását és kiszolgálását. [33]

Dinamikus tartalom:

Weboldal frissítése valós időben (AJAX, WebSocket): Az AJAX és WebSocket technológiák használatával valós idejű adatfrissítés valósítható meg anélkül, hogy az egész oldalt újratöltenénk. REST API JSON formátumban: A RESTful API-k lehetővé teszik az eszközök közötti kommunikációt JSON formátumú adatokkal, egyszerűsítve az integrációt és az adatok feldolgozását. [33]

7. Kommunikáció

Egy webvezérelt intelligens otthoni rendszer kiépítésekor a mikrokontroller és a konfigurációs eszközök közötti megfelelő kommunikációs protokoll kiválasztása kulcsfontosságú.

7.1 Wi-Fi alapú kommunikáció és módok

A Wi-Fi modul lehetővé teszi az eszközök számára, hogy csatlakozzanak helyi hálózatokhoz vagy saját hozzáférési pontot hozzanak létre.

Helyi hálózatban való működés:

a mikrokontroller csatlakozhat egy meglévő Wi-Fi hálózathoz, így a hálózaton belüli eszközök közvetlenül kommunikálhatnak vele.[34]

Közvetlen hozzáférés böngészőn keresztül:

a mikrokontroller webszerverként működhet, lehetővé téve a felhasználók számára, hogy böngészőn keresztül érjék el és vezéreljék az eszközt.[34]

Távoli elérés port-forwarding vagy VPN segítségével:

A router beállításainak módosításával (port-továbbítás) vagy virtuális magánhálózat (VPN) használatával a mikrokontroller távolról is elérhetővé válik.[34]

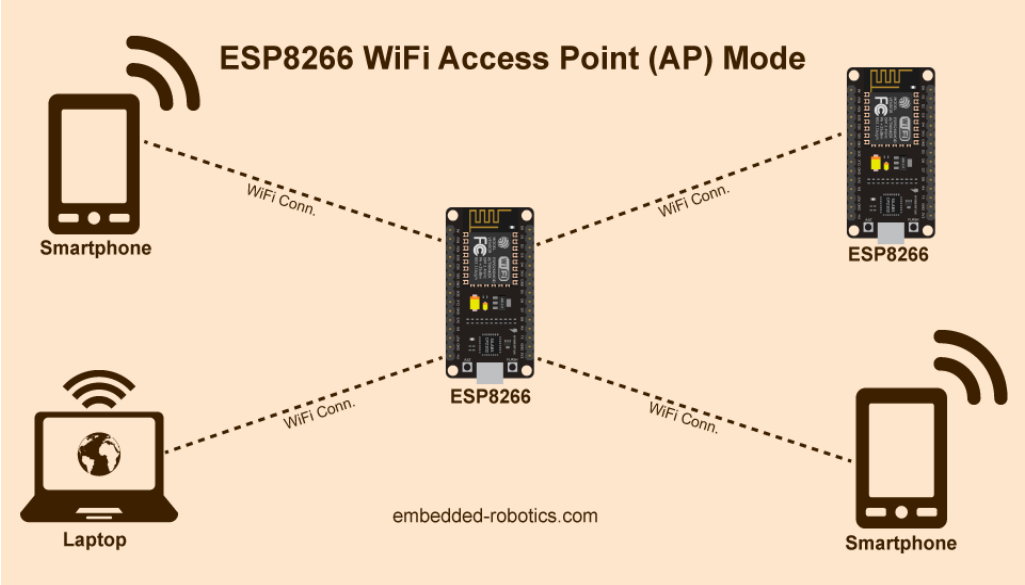
WiFi-modulos mikrokontrollereknél (például ESP8266, ESP32) általában három fő WiFi működési mód érhető el. Ezek az üzemmódok a WiFi-kommunikáció különböző típusait teszik lehetővé. Ezek az üzemmódok rugalmasan alkalmazhatók az adott alkalmazás igényeinek megfelelően.

AP mód

Az Access Point funkció lehetővé teszi, hogy a mikrokontroller vezeték nélküli hozzáférési pontként működjön, és más eszközök úgy csatlakozhassanak hozzá, mintha egy hagyományos Wi-Fi router lenne. Az AP segítségével létrehozhatunk saját helyi Wi-Fi hálózatot külső router vagy internetkapcsolat nélkül. Ez a képesség különböző forgatókönyvekben hasznos, például egy önálló IoT-hálózat létrehozásakor, helyi kommunikációs hálózat beállításakor az eszközök számára, vagy egy konfigurációs portál létrehozásakor a Wi-Fi hitelesítő adatok beállításához. Az AP DHCP-kiszolgálója alapértelmezés szerint engedélyezve van, és automatikusan hozzárendeli az IP-konfigurációkat a csatlakoztatott kliensekhez. Alternatívaként lehetőség van arra, hogy manuálisan adjuk meg az eszköz AP IP-konfigurációját, vagy használjuk az alapértelmezett beállítást.[35]

A konfiguráláshoz nincs szükség meglévő Wi-Fi hálózatra, illetve egyszerűbb beállítási folyamatot kínál kezdők számára, azonban a

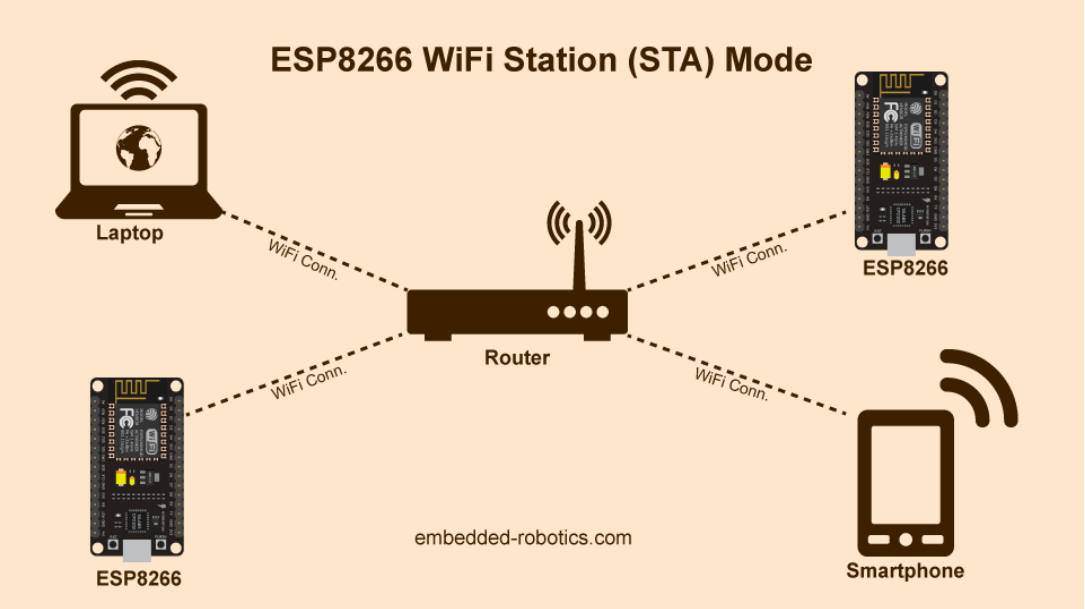
konfigurációs eszközök elveszítik az internetkapcsolatot, miközben az AP-hálózathoz csatlakoznak, az AP hálózat biztonsága gyenge lehet, ha nem megfelelően van konfigurálva. Nem alkalmas az intelligens otthoni eszközök és a webes felület közötti folyamatos adatátvitelre. A konfiguráció befejezése után a felhasználóknak le kell választaniuk az AP-ról.[35] Az alábbi ábrán (7.1. ábra) látható az AP mód működése.



7.1.ábra: AP mode (https://www.embedded-robotics.com/esp8266-wifi)

Station mód

Az STA mód a készüléket egy meglévő vezeték nélküli hálózat kliensévé alakítja. Ebben az üzemmódban az eszköz egy már meglévő vezeték nélküli hálózathoz csatlakozik, ahogyan egy okostelefon vagy laptop csatlakozik az otthoni Wi-Fi routerhez. Az STA üzemmód lehetővé teszi, hogy az eszköz csatlakozzon egy meglévő vezeték nélküli hálózathoz, és így hozzáférjen az internethez vagy kommunikáljon más eszközökkel. STA módban az eszköz csomóponttá válik a hálózaton, megkönnyítve az adatcserét és a kommunikációt más eszközökkel.[35] Az alábbi ábrán (7.2. ábra) látható a Station mód működése.



7.1.ábra: STA mode (https://www.embedded-robotics.com/esp8266-wifi)

STA+AP(kettős mód)

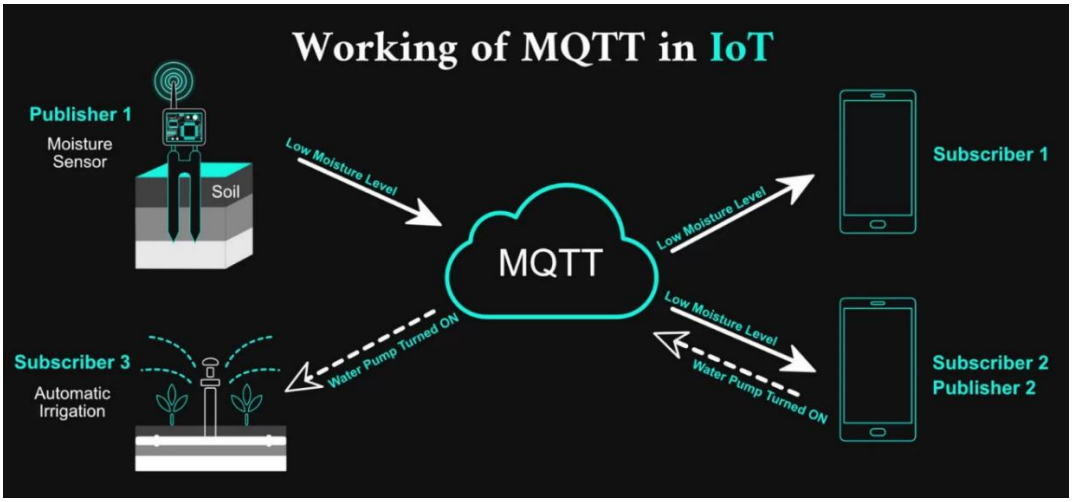
A mikrokontroller egyszerre működhet Station és Access Point módban, így egy meglévő Wi-Fi hálózathoz csatlakozik, miközben saját hálózatot is létrehoz. Ez ideális olyan alkalmazásokhoz, ahol a helyi hozzáférés és az internetkapcsolat egyaránt fontos. Például a felhasználó lokálisan vezérelheti az ESP32-t, miközben az adatok távolról is elérhetők. [35]

7.2 MQTT (Message Queuing Telemetry Transport)

Az MQTT az OASIS szabványos üzenetküldő protokollja a tárgyak internetéhez (IoT). Úgy tervezték, hogy egy rendkívül könnyű, publish/subscribe üzenetküldő protokoll legyen, amely ideális távoli eszközök összekapcsolására, kis kódlábnyom és minimális hálózati sávszélesség mellett. Az MQTT-t ma már számos iparágban használják, például az autóiparban, a gyártásban, a távközlésben, az olaj- és gáziparban stb. Az MQTT egy könnyű és gyors üzenetküldő protokoll, amelyet gyakran használnak IoT alkalmazásokban az eszközök közötti kommunikációhoz.[36]

Broker-alapú működés:

Az MQTT architektúrája egy központi broker köré épül, amely fogadja és továbbítja az üzeneteket a kliensek között. Az ESP32 kliensként működhet, amely adatokat publikál és fogad a broker segítségével.[36]

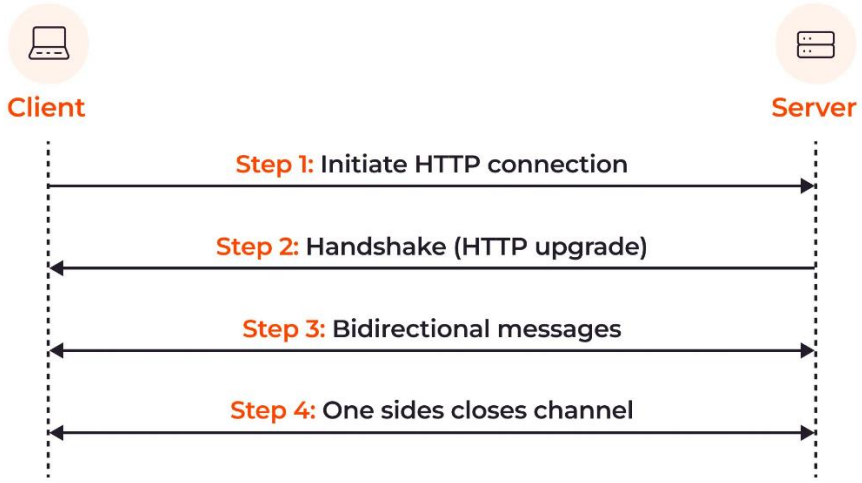


7.2.ábra: MQTT (https://psiborg.in/advantages-of-using-mqtt-for-iot-devices/)

7.3 Websocket

A WebSocket protokoll kétirányú kommunikációt tesz lehetővé az ügyfél egy ellenőrzött környezetben nem megbízható kódot futtató és egy távoli állomás között. [37]

amely engedélyezte az adott kód kommunikációját. A biztonság modell az eredet-alapú biztonsági modell, amelyet általában a következő módon használnakwebböngészők által is használt eredetmegjelölés. A protokoll egy nyitó kézfogásból áll amelyet alapvető üzenetkeretezés követ, a TCP-re rétegezve. A technológia célja, hogy mechanizmust biztosítson a böngészőalapú alkalmazásoknak, amelyeknek kétirányú kommunikációra van szükségük a kiszolgálókkal, amelynem támaszkodik több HTTP-kapcsolat megnyitására (például a XMLHttpRequest vagy <iframe>-ek és hosszú lekérdezés). Valós idejű, kétirányú kommunikáció: A WebSocket lehetővé teszi, hogy a szerver és a kliens valós időben küldjön és fogadjon üzeneteket anélkül, hogy minden egyes üzenethez új kapcsolatot kellene létrehozni.[37] Az alábbi ábrán (7.4. ábra) látható a websocket működése.



7.3.ábra.: Websocket (https://gcore.com/learning/what-is-websocket/)

7.4 Bluetooth

Amikor a Bluetooth először megjelent a piacon, nem volt Bluetooth Classic vagy Bluetooth Low Energy. Egyszerűen csak Bluetooth volt, egy olyan protokoll, amelyet a Bluetooth Low Energy bevezetése óta Bluetooth Classic néven emlegetünk. A Bluetooth Classic megkönnyíti a rövid hatótávolságú vezeték nélküli kommunikációt az eszközök között. A 2,4 GHz-es ipari, tudományos és orvosi (ISM) sávban működik, és lehetővé teszi az olyan eszközök, mint az okostelefonok, számítógépek, fejhallgatók és számtalan más elektronikai eszköz számára, hogy rövid távolságokon keresztül vezeték nélkül cseréljenek adatokat. Ez a technológia lehetővé teszi a magas szintű biztonságot nyújtó személyes területi hálózatok (PAN) létrehozását. Az eredetileg az 1990-es évek végén kifejlesztett Bluetooth Classic számos frissítésen esett át az adatátviteli képességek, a biztonság és az energiahatékonyság javítása érdekében. Különösen híres arról, hogy a legnépszerűbb szabványként szolgál a vezeték nélküli hang és a több gyártó eszközei közötti interoperabilitás terén. A Bluetooth Classic a mai napig számos Bluetooth-alkalmazásban használatos. Legyen szó vezeték nélküli fejhallgató és telefon összekapcsolásáról, kihangosított hívás lehetővé tételéről egy járműben, vagy adatok szinkronizálásáról orvosi megfigyelőeszközök között, a Bluetooth Classic az eredeti Bluetooth implementáció, amely megbízható és könnyen használható megoldást nyújt, és forradalmasította az eszközök közötti interakciót. A Bluetooth még mindig rendkívül népszerű vezeték nélküli szabvány a perifériák számára számos piacon és számos eszköztípusban, különösen a régebbi alkalmazásokban.[40]

7.5 Bluetooth Low Energy

A Bluetooth Low Energy (BLE) egy vezeték nélküli kommunikációs technológia, amelyet kifejezetten alacsony fogyasztású adatátvitelre terveztek. Széles körben használják különböző alkalmazásokban, beleértve a webvezérelt intelligens otthoni

rendszereket. A BLE a 2,4 GHz-es, engedély nélküli rádiófrekvencia-sávban működik, egyszerűbb modulációs sémát és rövidebb adatcsomagokat használ, ami elődjéhez képest (Bluetooth) jelentősen alacsonyabb energiafogyasztást eredményez. Ez a tulajdonság teszi a BLE-t ideális eszközzé az akkumulátorral működő eszközökhöz, ahol a hosszabb üzemidő létfontosságú. Az ultraalacsony energiafogyasztás mellett a Bluetooth LE több olyan egyedi tulajdonsággal is rendelkezik, amelyek megkülönböztetik a többi elérhető vezeték nélküli technológiától, többek között a klasszikus Bluetooth-eszközökhöz hasonlóan a BLE-eszközök is a Bluetooth Special Interest Group (SIG) által meghatározott szabványokat követik, és a különböző gyártók BLE-eszközei együttműködnek egymással. A BLE gyors frekvenciaugrással biztosítja a robusztus átvitelt más vezeték nélküli technológiák jelenlétében is.[41]

A BLE-t úgy fejlesztették ki, hogy a tervezők számára egyszerű legyen a különböző alkalmazásokban való alkalmazása. Az Ezurio ezt a smartBASIC beágyazott programozási nyelvvel még egyszerűbbé teszi. A kis adatdarabok küldésének teljes ideje általában kevesebb, mint 6 ms, és akár 3 ms is lehet (szemben a klasszikus Bluetooth 100 ms-os időtartamával). A megnövelt modulációs indexnek köszönhetően a BLE technológia nagyobb hatótávolságot kínál (ideális környezetben akár 200 lábat és még annál is többet), mint a klasszikus Bluetooth.[41] Az alábbi táblázat (7.5. ábra) összehasonlítja a két technológiát.

Application scenario	**Audio stream application** Wireless head-phones Wireless speaker Vehicle-mounted entertainment	**Data transmission application** Sports and fitness equipment Medical and health equipment Peripherals and accessories	**Location services application** Beacon services Indoor navigation service Asset tracking	**Device network application** Control system Monitoring system Automation system
Communication mode	One-to-one	One-to-one	One-to-many (broadcast)	Many-to-many (mesh)
Radio frequency mode	Classic Bluetooth BR/EDR	Bluetooth low energy (Bluetooth LE)		

7.5.ábra: Bluetooth and BLE

(https://blog.nordicsemi.com/the-difference-between-classic-bluetooth-and-bluetooth-low-energy)

A fentebb említett működés előnye összefoglalva az alacsony energiafogyasztás, ami hosszabb akkumulátor-élettartamot eredményez az intelligens otthoni eszközöknél, a konfigurációs eszközök fenntarthatják az internetkapcsolatot, miközben a BLE-n keresztül csatlakoznak, illetve alkalmas az intelligens otthoni eszközök és a webes felület közötti folyamatos adatátvitelre.

8. Kiválasztott eszközök, modulok és környezet

A szakdolgozatomhoz a legfontosabb döntési pont az eszközválasztást tekintve a mikrokontroller kiválasztása, amelyekből összesen négy darabot elemeztem a korábbi pontokban, a fő szempontok a következőek voltak: tartalmazzon beépített Wi-Fi (AP mód) vagy Bluetooth modult, mellyel megvalósítható a webszerver, az összeépítéshez fellelhető legyen hivatalos dokumentáció, már elkészült hasonló projektek és ezek mellett bővíthető legyen a szenzorskálája. Ezen feltételeknek megfelelő választásnak bizonyul az ESP32 mikrokontroller. Az ESP32 ezzel együtt biztosítja az AP üzemmódot, mellyel megvalósítható a webszerverre az adattovábbítás. A következő kiválasztandó eszközök a szenzorok, melyek közül bemutatásra kerültek a hőmérséklet és páratartalom mérő érzékelők, a gázérzékelők, a mozgásérzékelők és egyéb gyakran használt szenzor. A hőmérséklet és páratartalom mérők közül a DHT11 bizonyul megfelelőnek, elegendően széles tartományban mérhető vele a két érték, ezek mellett nem fontos szempont az ára és a mintavételezési ideje. A gáz/levegőminőség szenzorok közül az MQ2 elegendő a bután, propán, metán, alkohol érzékeléséhez. A mozgásérzékelők közül az infravörös mozgásérzékelő (HC-SR501) megfelelő a projektemhez, tervezet szerint elegendő bármilyen mozgást detektálnia, nem szükséges több információ. Ezeken kívül tartalmazni fog egy TM1638 vezérlőpanelt az adatok megjelenítésére.

A fejlesztői környezet, amelyben a szoftvert írom, az Arduino IDE, mely a korábban taglaltaknak megfelelően egyszerűbb fejlesztési módszert nyújt, illetve az interneten fellelhető projektek több százaléka íródott ebben a környezetben.

8.1 *Rendszerterv és folyamatábra*

Rendszerterv

1. Érzékelők adatrögzítése

- A DHT11 érzékelő rögzíti a hőmérséklet és páratartalom adatokat.
- Az MQ2 érzékelő rögzíti a gáz/levegőminőség adatokat.
- A HC-SR501 érzékelő rögzíti a mozgást.

2. Adatok továbbítása

- Az ESP32 mikrokontroller fogadja az érzékelők adatait, majd továbbítja azokat a beépített Wi-Fi modul segítségével.

3. Adatok feldolgozása és elemzése

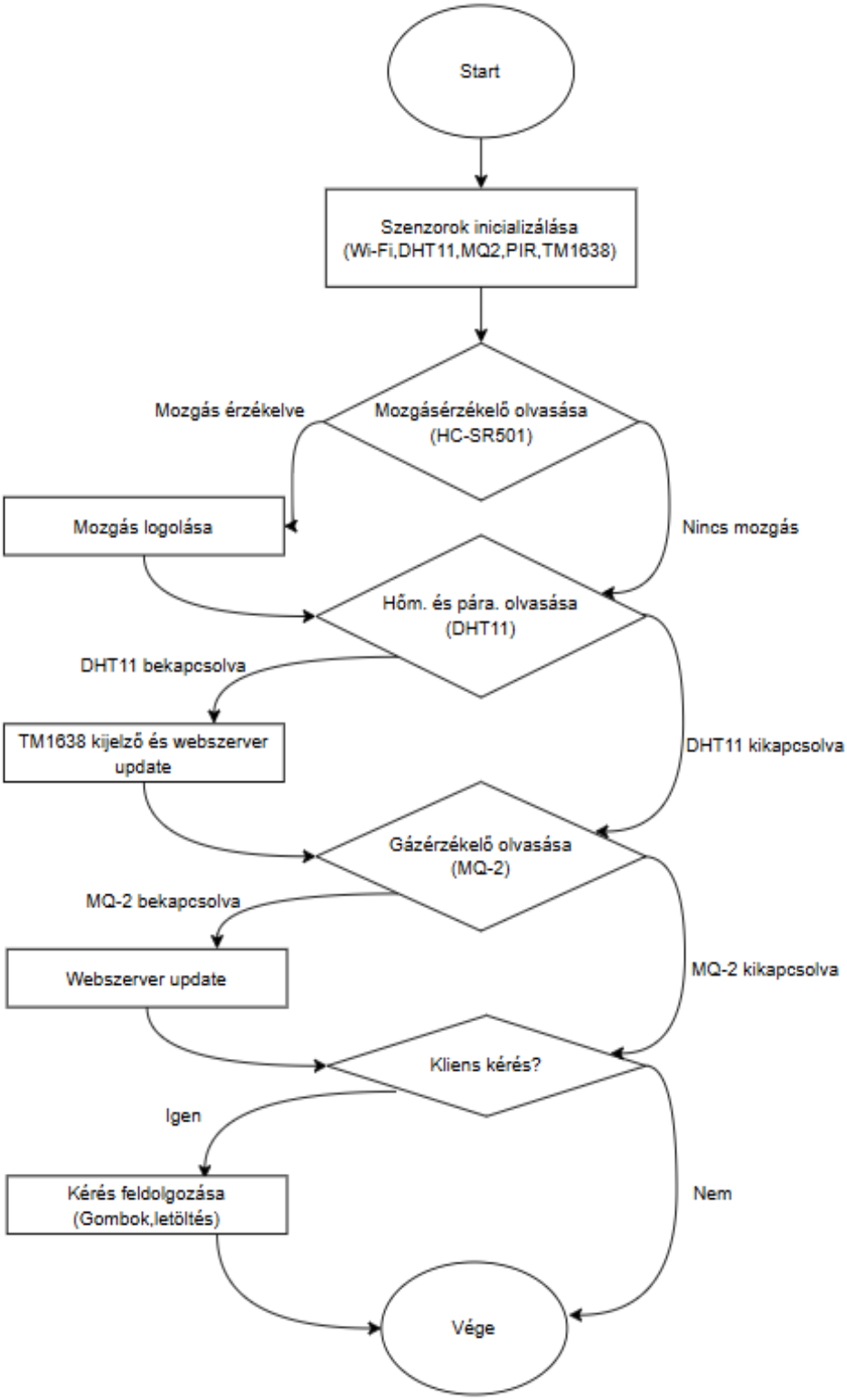
- Az ESP32 mikrokontroller feldolgozza az adatokat (pl. átalakítja a nyers adatokat hasznos információkká).

4. Cselekvés

- A hőmérséklet és páratartalom értékek megjelennek a weboldalon és a TM1638 kijelzőn (pl. „25C 45%”), mely folyamatosan frissíti az értékeket

- Az érzékelt mozgások időbélyeggel rögzítésre kerülnek a naplóban, és megjelennek a weboldalon.
- MQ-2 szenzor méri a gázkoncentrációt. Az értékek megjelennek a weboldalon.
- Grafikonon megjeleníti idősorosan ábrázolva a mért hőmérsékletet és páratartalmat.
- Egyedi gombokkal a DH11, a HC-SR501 és az MQ-2 szenzor ki-/bekapcsolható.
- A HC-SR501 által érzékel mozgások megjelennek a pontos időbélyeggel a webszerveren,ezen felül CSV formátumba letölthetőek.

Folyamatábra



8.1..ábra : A rendszer folyamatábrája

9. Megvalósítás

9.1 *Szükséges eszközök és szenzorok előkészítése*

A rendszer megfelelő működéséhez és a projekt megvalósításához számos eszköz és szenzor előkészítése szükséges. Az alábbiakban részletesen bemutatjuk a projekt alapvető komponenseit, amelyek a hardveres összeállítás és a szoftveres vezérlés kulcsfontosságú elemei :

- ESP32 mikrokontroller
- DHT11 szenzor
- HC-SR501 szenzor
- MQ-2 szenzor
- TM1638 LED&KEY modul
- Csatlakozók és tápellátás
- Nem forraszthtató mező(k)
- Ellenállások

9.2 Szenzorok és modulok bekötése

A referencia projektek és források részletes tanulmányozása lehetőséget nyújtott arra, hogy minden szenzor és modul bekötését az ESP32 gyártói használati utasításának megfelelően végezzek el. Az ESP32 egy rendkívül sokoldalú mikrokontroller, amely különféle kommunikációs protokollokat és GPIO lábakat biztosít, így a projekthez szükséges komponensek könnyen integrálhatók.

Minden egyes szenzor és modul bekötése előtt alaposan áttekintettem a hozzájuk tartozó adatlapokat, hogy pontosan megértsük a működésüket és a szükséges bekötési módokat. Az ESP32 dokumentációja és a gyártói ajánlások alapján kiválasztottom a megfelelő GPIO lábakat a szenzorokhoz és a modulokhoz, figyelembe véve az áramkörök elektromos követelményeit, például a feszültségszinteket és a jelkimenetek típusát (digitális vagy analóg). Az alábbiakban táblázat formájában szemléltetem a szenzorok és eszközök bekötését.

Mozgásérzékelő szenzor

Az alábbi táblázat (9.2. ábra) bemutatja a szenzor bekötési módját.

PIR láb	ESP32 láb	Funkció
VCC	3V3	Tápellátás
GND	GND	Földelés
OUT	GPIO4	Jel

9.2..ábra : mozgásérzékelő szenzor

Páratartalom és hőmérséklet szenzor

Az alábbi táblázat (9.2. ábra) a szenzor bekötési módját ismerteti.

DHT11 láb	ESP32 láb	Funkció
VCC	3V3	Tápellátás
GND	GND	Földelés
DATA	GPIO5	Adatok küldése

9.2.ábra : páratartalom és hőmérséklet szenzor

TM1638 LED&KEY kijelző

Az alábbi táblázat (9.2. ábra) ismerteti a modul csatlakoztatásának módját.

TM1638 láb	ESP32 láb	Funkció
VCC	3V3	Tápellátás
GND	GND	Földelés
STB(CS)	GPIO23	Chip Select (STB)
CLK	GPIO22	Órajel
DIO	GPIO21	Adatok (Data In/Out)

9.2.ábra : TM1638

Zajcsökkentés vagy stabilitás miatt köthető be akár egyes moduloknál pull-up/down vagy soros ellenállás.

9.3 Szoftver

Az új sketch megnyitása után a megfelelő ESP32 eszközt és portot állítottam be, következett a megfelelő könyvtárak importálása a szenzorok és modulokhoz, illetve a Wi-Fi kapcsolat megvalósításához.

Felhasznált könyvtárak:

- WiFi
- Adafruit_Sensor
- DHT
- DHT_U
- time
- TM1638plus

A Wi-Fi kezelését biztosítja a WiFi.h könyvtár, a DHT11 szenzor adatainak olvasásához szükséges a DHT.h és DHT_U.h, valamint a kijelző használatához elengedhetetlen a TM1638plus.h. A pontos idő szinkronizálásához a time.h könytárat használtam.

A könyvtárak importálása után a szenzorok definiálása következik. A bekötés pontban tárgyaltaknak megfelelően kell a kódban jelezni, hogy az egyes érzékelők mely ESP32 lábra vannak kötve a #define paranccsal, például:

A mozgásérzékelő PIR szenzor a GPIO4 bemeneti lábra van kötve, a DHT11 szenzor GPIO5 adatlábra van kötve.

Itt még fontos megadni a Wi-Fi hálózat adatait, melyhez majd az ESP32 kapcsolódik. Ez a lépés biztosítja, hogy a program tudja, melyik GPIO láb felelős az egyes szenzorok és modulok adatainak olvasásáért, illetve vezérléséért. A megfelelő definíciók segítik a hardver- és szoftverkomponensek összhangját. Ezt követi a szenzorok alapértelmetett állapotának beállítása.

A következő lépés a szenzorok és a Wi-Fi szerver definiálása.

Az ESP32 kártya működhet Wi-Fi állomásként, hozzáférési pontként vagy mindkettőként. A Wi-Fi üzemmód beállításához használhatjuk a WiFi.mode() parancsot.

WiFi.mode(WIFI_STA) - Állomás mód

Az ESP32 csatlakozik egy hozzáférési ponthoz.

WiFi.mode(WIFI_AP) – hozzáférési pont mód

Az állomások képesek csatlakozni az ESP32-höz.

WiFi.mode(WIFI_AP_STA)

Hozzáférési pont és egy másik hozzáférési ponthoz csatlakozó állomás.

A dolgozatomban az alapértelmezett WiFi.mode(WIFI_STA) van beállítva, így az ESP32 csatlakozik egy meglévő hozzáférési ponthoz.

A szenzorok inicializálása a setup() függvényben

GPIO módok beállítása

Minden definiált GPIO lábat az Arduino API segítségével be kell állítani, hogy bemenetként vagy kimenetként működjön. A PIR érzékelő és az MQ-2 szenzor bemeneti módra lett konfigurálva, míg a LED kimeneti módban működik. A dht.begin() paranccsal inicializáltam, míg a TM1638 panel működését a tm.displayBegin() paranccsal aktiváltam.

A Wi-Fi kapcsolat létrehozásához itt kell meghívni a WiFi.begin() függvényt, mellyel az ESP32 kapcsolódik a hálózathoz, a kapcsolat állapotát pedig ellenőrzöm folyamatosan, és a soros monitoron megjelenítem az IP-címet, melyen keresztül a webszerver elérhető. Ebben a részben még szükséges az idő szinkronizálása NTP (Network Time Protocol) segítségével, mellyel a rendszer pontosan beállítja a jelenlegi időt. Ez elengedhetetlen a mozgásérzékelési események időbélyeggel való ellátásához. Utolsó lépésként a rendszer egy webszervert hoz létre, amelyen keresztül az adatok megjelenítése és a szenzorok vezérlése történik.

A loop() függvény az ESP32 mikrokontroller működésének központi eleme, amely folyamatosan ismétlődik, és biztosítja a rendszer valós idejű működését. Ez a függvény több egymásba ágyazott folyamatot tartalmaz, amelyek az egyes szenzorok adatainak figyeléséért, feldolgozásáért és az adatok továbbításáért felelősek.

A függvény első lépése a PIR szenzor által érzékelt mozgások figyelése, amely csak

akkor történik meg, ha a mozgásérzékelő funkció engedélyezve van. A szenzor állapotát a GPIO4-es lábon keresztül olvasom be a digitalRead() függvény segítségével. Ha a szenzor mozgást érzékel, az aktuális időbélyeg (timestamp) lekérése történik a getCurrentTime() függvény segítségével. Mozgás hiánya esetén a LED kikapcsolódik, jelezve a nyugalmi állapotot. Ezek a műveletek csak akkor zajlanak le, ha a mozgásérzékelő nincs kikapcsolva, amelynek állapotát az isMotionEnabled változó határozza meg. A mozgásról érkező információt rögzítjük a motionLog változóban, amely egy HTML-listát képez. Ez az információ megjelenik majd a webszerveren. A mozgás észlelésének visszajelzéseként a mikrokontrollerhez kapcsolt LED (GPIO2) felvillan, jelezve a riasztást.

A következő lépésben az MQ-2 gázszenzor által mért adatokat dolgozzuk fel. Az érzékelő a levegő gázkoncentrációjának analóg értékét szolgáltatja, amelyet a GPIO35-ös lábon keresztül olvasom be az analogRead() függvény segítségével.

Ezután a A DHT11 szenzor mérései alapján a hőmérsékleti és páratartalom-értékeket olvassuk be és dolgozzuk fel. Ha a szenzor aktív (engedélyezett állapotban van), a hőmérsékletet és a páratartalmat a readTemperature() és readHumidity() függvények segítségével olvasom be. Ha a mért értékek nem elérhetők, azok helyett alapértelmezett értékeket (0.0) állít be. A mért adatokat továbbá a TM1638 kijelzőn is megjelenítjük, amely valós időben frissül.

Ha a gázérzékelő funkció engedélyezve van, az adatokat eltároljuk a gasValue változóban. Ha a szenzor ki van kapcsolva, a gasValue értéke 0 marad, jelezve, hogy nem végez méréseket.

A negyedik fő rész a kliens kéréseinek kezelése, amely a webszerver funkciókat valósítja meg. Az ESP32 mikrokontroller egy webszervert futtat, amely lehetővé teszi az adatok megjelenítését és a szenzorok vezérlését távolról. A kérések feldolgozása az alábbiak szerint történik:

Szenzorok ki-/bekapcsolása

Ha a kliens kérésében megtalálható egy adott kapcsoló parancs (pl. /toggleDHT, /toggleMotion, vagy /toggleGasSensor), az adott szenzor állapota megváltozik. Ez lehetővé teszi a felhasználó számára, hogy távolról engedélyezze vagy letiltsa a szenzorokat. A változtatás után a rendszer visszajelzést küld a kliensnek a kapcsolás sikerességéről.

Szenzoradatok kiszolgálása

Ha a kliens kérése tartalmazza a /sensor parancsot, a rendszer összegyűjti az aktuális szenzoradatokat (hőmérséklet, páratartalom, gázkoncentráció, mozgásnapló), és JSON-formátumban visszaküldi azokat. Ez az információ a webszerver felületén jelenik meg, és lehetőséget biztosít a felhasználónak az adatok figyelésére.

Mozgásnapló letöltése

A kliens kérésére a mozgásérzékelő által rögzített események CSV formátumban letölthetők.

Az utolsó pont pedig a weboldal tartalmának dinamikus frissítése. Az aktuális szenzoradatok, a mozgásérzékelési napló és a gázérzékelő mérései egyaránt megjelennek a webes felületen. A Chart.js segítségével a hőmérséklet- és páratartalom-adatok grafikonként jelennek meg, amelyeket időbélyeggel látunk el. A rendszer vizuális visszajelzést ad az aktuális állapotról (pl. jó vagy rossz környezetminőség). Az alábbi képek bemutatják a kész és működő webszerver kinézetét.

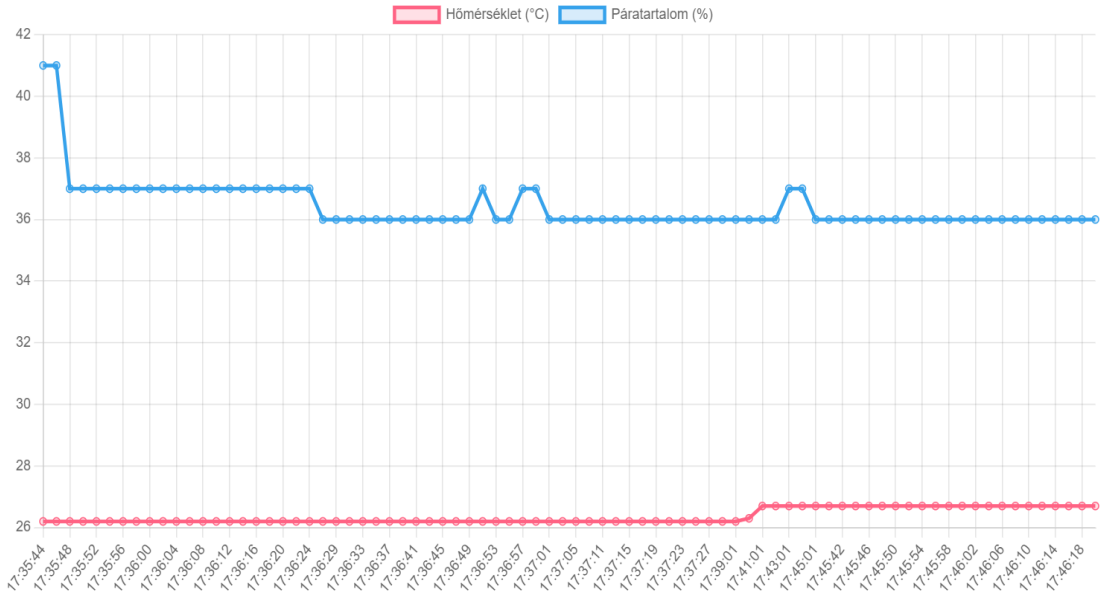
Okosotthon webszerver

Környezetminőség : JÓ

Hőmérséklet: 26.7 °C

Páratartalom: 36 %

Gázérték: 1056



Mozgásnapló

2024-12-09 17:35:37: Mozgás érzékelve!
2024-12-09 17:36:14: Mozgás érzékelve!
2024-12-09 17:37:00: Mozgás érzékelve!
2024-12-09 17:37:44: Mozgás érzékelve!
2024-12-09 17:38:29: Mozgás érzékelve!
2024-12-09 17:39:18: Mozgás érzékelve!
2024-12-09 17:40:02: Mozgás érzékelve!
2024-12-09 17:40:45: Mozgás érzékelve!
2024-12-09 17:41:28: Mozgás érzékelve!
2024-12-09 17:42:11: Mozgás érzékelve!
2024-12-09 17:42:55: Mozgás érzékelve!
2024-12-09 17:43:43: Mozgás érzékelve!
2024-12-09 17:44:27: Mozgás érzékelve!
2024-12-09 17:45:11: Mozgás érzékelve!
2024-12-09 17:45:54: Mozgás érzékelve!
2024-12-09 17:46:37: Mozgás érzékelve!

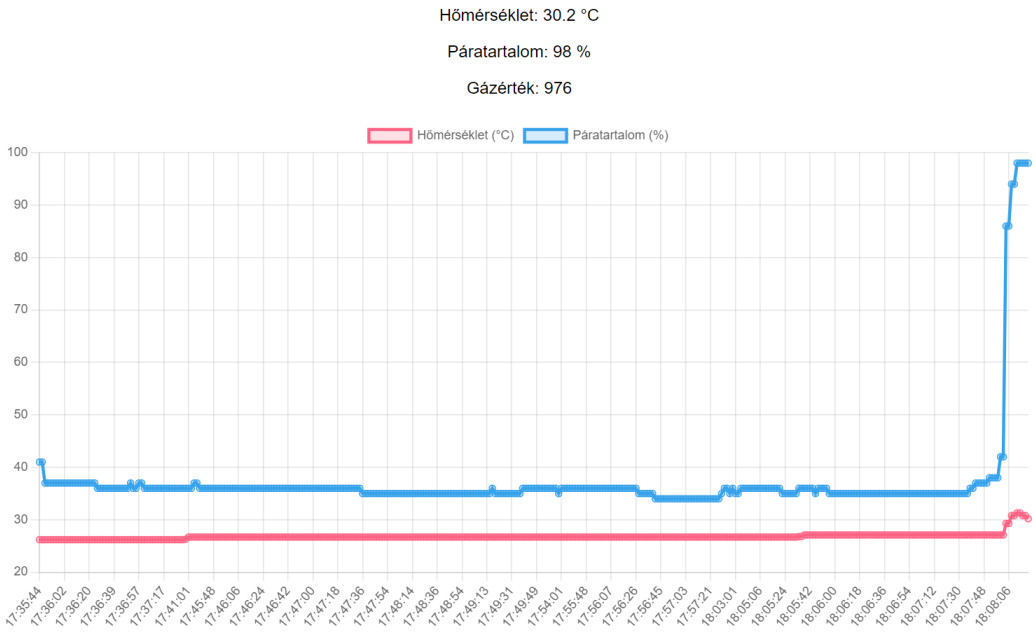
DHT11 Kikapcsolása | Mozgásérzékelő Kikapcsolása | Gázszenzor Kikapcsolása | Mozgásadatok letöltése

10. Tesztelés

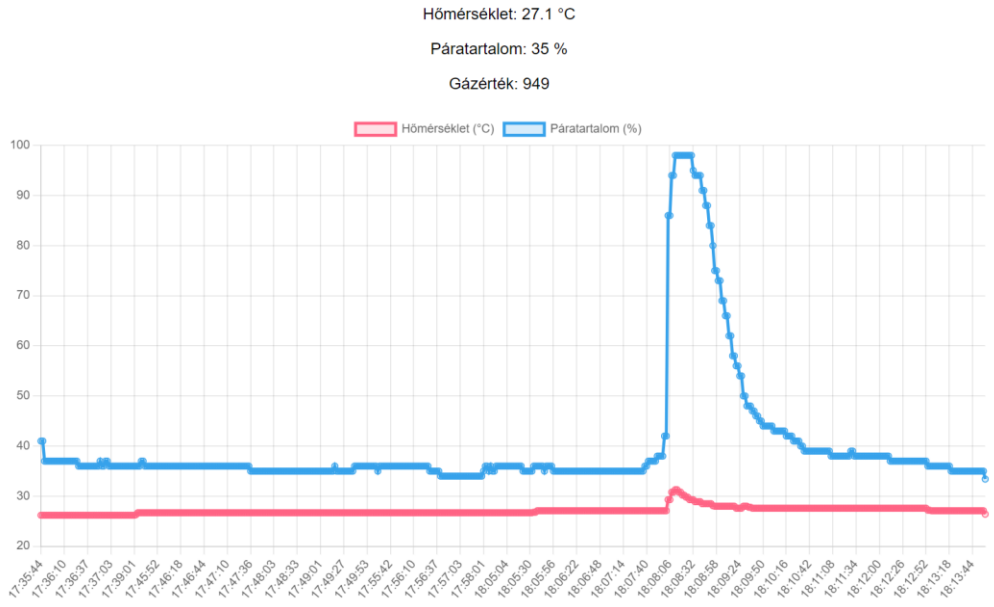
A rendszer teljes körű tesztelése kulcsfontosságú annak biztosítására, hogy minden funkció a tervezettek szerint működjön, és megfeleljen az elvárásoknak. A tesztelés során a weboldal funkcionalitását, a szenzorok helyes működését, valamint a rendszer stabilitását vizsgáltam különböző körülmények között. A cél az volt, hogy az érzékelők által gyűjtött adatok pontosan jelenjenek meg a webes felületen, és a rendszer helyesen reagáljon a beállítások és a környezeti hatások változásaira.

A tesztelési folyamat magában foglalta a hardver és szoftver integráció ellenőrzését, a valós idejű adatok kiértékelését, valamint a weboldal interaktív elemeinek kipróbálását. Emellett külön figyelmet fordítottam a rendszer megbízhatóságára, a hibakezelés hatékonyságára, valamint a felhasználói élmény szempontjaira.

A webszerveren a hőmérséklet- és páratartalom-mérő megfelelően működik, melyet a tesztelések során vizsgáltam. Melegítés hatására a hőmérsékleti értékek növekedése gyorsan és pontosan megfigyelhető, hasonlóképpen a párásítás alkalmazása során a páratartalom értékei emelkednek, ami igazolja a rendszer valós idejű adatközlését.

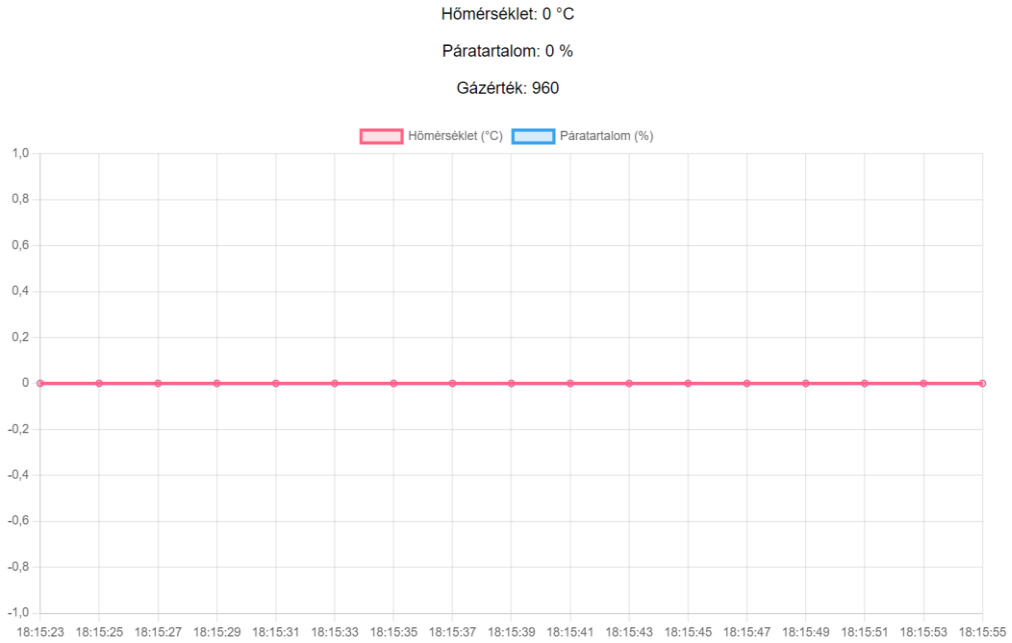


Ezzel szemben, hűtés és párátlanítás hatására az értékek fokozatosan visszaállnak az eredeti állapotba, ami a rendszer adaptív működését és a környezeti tényezőkre való gyors reagálóképességét mutatja. A valós környezeti változások hatása egyértelműen megjelenik a webes felületen és a TM1638 kijelzőn, lehetővé téve a felhasználó számára az azonnali megfigyelést és visszacsatolást.



Mozgás hatására a rendszer azonnal érzékeli az eseményt, és megfelelően kimutatja a riasztást mind a webes felületen, mind a kapcsolódó vizuális indikátorokon keresztül. A PIR mozgásérzékelő gyors válaszidejének köszönhetően a mozgás azonnal rögzítésre kerül, és a LED-fény felvillanása is megerősíti az érzékelés tényét. A mozgásérzékelés időbélyeggel együtt jelenik meg a webes felületen, ahol egy naplóban nyomon követhetők a múltbeli események.

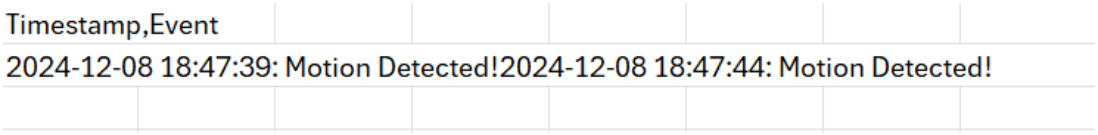
A DHT11 szenzor kikapcsolása után a rendszer a vártnak megfelelően nem rögzít sem hőmérsékleti, sem páratartalom-értékeket. A webes felületen ezek az értékek nullaként jelennek meg, egyértelműen jelezve, hogy a szenzor inaktív állapotban van.



Az MQ-2 gázérzékelő kikapcsolása után a rendszer a vártnak megfelelően 0 értéket jelenít meg, ami azt jelzi, hogy a szenzor inaktív állapotba került, és nem végez méréseket.

Ez a funkció lehetővé teszi a felhasználó számára, hogy szükség esetén kikapcsolja a gázérzékelőt, például energiamegtakarítás, karbantartás érdekében.

A mozgásadatok letöltése után a CSV fájl tartalmazza az addig mért mozgást.



10.1 Tesztesetek

Az alábbi táblázatokban rögzítettem néhány fontos tesztesetet.

Tesztazonosító	T1
Funkció	Hőmérséklet mérés
Tesztelési lépés	Melegítés határásra hőmérséklet növelése
Elvárt eredmény	A hőmérséklet érték növekedik
Állapot	Sikeres

10.1.ábra : T1

Tesztazonosító	T2
Funkció	Páratartalom mérés
Tesztelési lépés	Párásítás alkalmazása a környezetben
Elvárt eredmény	A páratartalom érték növekszik
Állapot	sikeres

10.1.ábra : T2

Tesztazonosító	T3
Funkció	Mozgásérzékelés
Tesztelési lépés	Mozgás szimulálása a szenzor előtt
Elvárt eredmény	Naplóbejegyzés jelenik meg időbélyeggel
Állapot	sikeres

10.1.ábra : T3

Tesztazonosító	T4
Funkció	Szenzorok kikapcsolása
Tesztelési lépés	A DHT11 kikapcsolása a weboldalon
Elvárt eredmény	Nem mér adatot a kikapcsolás után
Állapot	sikeres

10.1.ábra : T4

Tesztazonosító	T5
Funkció	LED&KEY kijelző
Tesztelési lépés	Hőmérséklet és páratartalom frissítése
Elvárt eredmény	Az értékek helyesen jelennek
Állapot	sikeres

10.1.ábra : T5

Tesztazonosító	T6
Funkció	Napló letöltése
Tesztelési lépés	Mozgásérzékelő napló letöltése CSV-ben
Elvárt eredmény	A fájl tartalmazza a naplóbejegyzéseket
Állapot	sikeres

10.1.ábra : T6

10.2 Teszt összegzése

A rendszer tesztelése során a különböző funkciókat alaposan kipróbáltam, és meggyőződtem róla, hogy minden megfelelően működik. A DHT11 szenzor (hőmérséklet és páratartalom mérése) által mért hőmérséklet-páratartalom értékek pontosan megjelentek mind a webes felületen, mind a TM1638 kijelzőn. Az értékek frissítése folyamatos volt, és a mért adatok megfeleltek a valós környezeti változásoknak. A PIR mozgásérzékelő (HC-SR501) érzékenységét és pontosságát ellenőriztem, minden érzékelt mozgást az időbélyeggel együtt rögzített a rendszerem, és a mozgásnapló pontosan megjelenik a webes felületen. Az ESP32 LED megfelelően jelezte a mozgást, és a ki- és bekapcsolása a weben gomb segítségével problémamentesen működött. Az MQ-2 gázszenzor pontos értéket jelenített meg a weboldalon, ki- és bekapcsoló funkciója rendben működött. A webes interfész minden funkciója hiba nélkül elérhető volt, beleértve a grafikonokat, adatok megjelenítését, szenzorok ki- és bekapcsolását és a mozgásnapló letöltését CSV formátumban. A grafikon időszerűen reagált és frissült a Chart.js segítségével. A TM1638 kijelző pontosan mutatta a hőmérsékletet és páratartalmat, az értékek olvashatóak voltak, illetve folyamatosan frissült az aktuális adatokkal.

11. Továbbfejlesztési lehetőségek

Adatgyűjtés és elemzés további érzékelőkkel:

További érzékelők integrálásával hasznosabb és sokoldalúbb rendszerré lehet alakítani, ilyen szenzorok például a fényérzékelő, mellyel a fényintenzitás mérésével beltéri automatikus szabályzást illetve a növényeknek megfelelő mennyiségű fény biztosítása megvalósítható. Emellett nyomás- és magasságérzékelővel időjárási állomásként használva kaphatunk visszajelzést a környezetből. [42]

Automatizálás:

Intelligens világításvezérléssel elérhetjük, hogy az égők automatikusan kapcsolódjanak, ventilátorok vagy legkondicionálók életbe lépjenek a környezetből származó információk alapján, illetve riasztórendszerrel való bővítés is egy továbbfejlesztett megoldás lehet. Ezek mellett a mobilalkalmazás integrációja kiválthatja az email küldés okozta bonyodalmakat azzal, hogy magára a készülékre küldhet értesítést az alkalmazás riasztások esetén.[43]

12. Összegzés

A szakdolgozatom írása alatt rengeteg tapasztalatot szereztem az IoT világáról, a szoftveres és hardveres megvalósításokról.

Az első pár oldalon kitértem az IoT eredetére és szükségességére, majd a projekt céljának bemutatása után folytattam a dolgozat felhasználásához szükséges irodalomkutatással.

A referencia projektek tanulmányozása alatt nagyon sok modern technológiával ismerkedtem meg, ezekből a tanulmányokból és projektekből kiválasztva az előnyöket és a hátrányokat rátértem a hardveres megvalósítás alapjaira.

Ezen a ponton össze kellett hasonlítanom a piacon használt mikrokontrollereket, érzékelőket, fejlesztői környezeteket, kommunikációs technológiákat. Mindegyik megvizsgált modulban megpróbáltam kiválasztani a leggyakrabban haszált eszközöket és módszereket, ezzel indokolva a döntéseket, melyek alapján elkészült a szakdolgozat.

A következő lépés a rendszerterv és folyamatábra megvalósítása volt, ahol ismertettem a funkcionalitást és a folyamatokat.

A megvalósításban részleteztem a kellő eszközöket, szenzorokat és modulokat, majd a szoftver felépítését ábrázoltam.

A tesztelés során minden elemet bemutattam, működésüket vizsgáltam, különböző környezeti hatásokra miként reagált a rendszer.

Továbbfejlesztési módszereket tárgyaltam a végén, melyekkel a projekt komplexitását lehet növelni, javaslatokat tettem a kényelmi és biztonsági funkciók kiterjesztésére.

13. Summary

While writing my thesis, I gained extensive experience in the world of IoT, including both software and hardware implementations.

In the initial sections, I addressed the origins and necessity of IoT, followed by an introduction to the project's objectives. I then proceeded with a literature review essential for the development of the thesis.

While studying reference projects, I became acquainted with many modern technologies. Based on these studies and projects, I evaluated the advantages and disadvantages, eventually transitioning to the basics of hardware implementation.

At this stage, I had to compare the microcontrollers, sensors, development environments, and communication technologies used in the market. In each module I examined, I aimed to select the most commonly used tools and methods, providing justifications for the decisions that formed the basis of the thesis.

The next step involved developing the system design and flowchart, where I described the functionality and processes in detail.

In the implementation phase, I outlined the necessary tools, sensors, and modules, followed by a depiction of the software architecture.

During the testing phase, I demonstrated each element, examined their operation, and evaluated how the system responded to various environmental conditions.

Finally, I discussed methods for further development to increase the complexity of the project and provided suggestions for extending convenience and security features.

14. Irodalomjegyzék

[1] Smart home, https://medium.com/digitalshroud/x10-the-revolutionary-smart-home- device-youve-never-heard-of-48790e272e1f, Utoljára megtekintve (2024.05.12.)

[2] Wi-Fi és BLE, https://randomnerdtutorials.com/esp32-wi-fi-provisioning-ble-arduino/ , Utoljára megtekintve (2024.05.13)

[3] LED, https://randomnerdtutorials.com/esp32-web-server-slider-pwm/ , Utoljára megtekintve (2024.05.13)

[4] DHT11/DHT22, https://randomnerdtutorials.com/esp32-dht11-dht22-temperature- humidity-web-server-arduino-ide/ , Utoljára megtekintve (2024.05.13)

[5] Arduino webszerver, https://lastminuteengineers.com/creating-esp32-web-server- arduino-ide/#google_vignette , Utoljára megtekintve (2024.05.13)

[6] https://randomnerdtutorials.com/esp32-bh1750-ambient-light-sensor/ , Utoljára megtekintve (2024.05.13)

[7] Hercog, D., Lerher, T., Truntic, M., Tezak, O.: Design and Implementation of ESP32-Based IoT Devices, *Sensors* 2023**,** ppt. 1-20.

[8] Niranjana, R., Arvind, S., Vignesh, M., Vishaal, S.: Effectual Home Automation using ESP32 NodeMCU, International Conference on Automation, Computing and Renewable Systems (ICACRS),2022

[9] Mahamud, S., Md., Zishan, S.,Z., Md., Ahmad,S.,I., Rahman,A.,R., Hasan, M., Rahman.,L., Md.: Domicile - An IoT Based Smart Home Automation System, International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST),2019

[10] Arigela,A.,K., Banapuram,C., Venu,N.: Remote based Home Automation with MQTT: ESP32 Nodes and Node-RED on Raspberry Pi, 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC),2024

[11] Haroon, P S.,A.,L., Shafiulla, M., Naveed,S.,M., Ahmed,S., Nawaz, S.,M., Kumar,U.: Home Automation Using Wi-Fi: ESP32-Based System for Remote Control and Environmental Monitoring,Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE),2024

[12] Zare, A., Iqbal, M.T.: Low-Cost ESP32, Raspberry Pi, Node-Red, and MQTT Protocol Based SCADA System,IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS),2020

[13] Mikrokontrollerek áttekintése, https://uk.rs-online.com/web/content/discovery/ideas- and-advice/microcontrollers-guide, Utoljára megtekintve (2024.05.12.)

[14] Arduino Uno adatok, https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel- 7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf Utoljára megtekintve (2024.05.12.)

[15] Chaturvedi, P., Mulla, M.A., Patro, S.K..: Power Electronics Laboratory Education using ARM Cortex M4 32-bit Microcontroller, *IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES)*,2018, pp 1-2.

[16] NodeMCU áttekintése, https://components101.com/development-boards/nodemcu- esp8266-pinout-features-and-datasheet

[17] Maier, A., Sharp, A., Vagapov, Y.: Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things, *Internet Technologies and Applications (ITA)*, 2017, pp. 143-146.

[18] Landaluce, H., Arjona, L., Perallos, A., Falcone, F., Angulo, I., Muralter, F.: A Review of IoT Sensing Applications and Challenges Using RFID and Wireless Sensor Networks, *Sensors* ,2020, pp. 1-18.

[19] Sehrawat,D., Gill, N.,S.: Smart Sensors: Analysis of Different Types of IoT Sensors, 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI),2019

[20] Temperature and Humidity, https://www.renkeer.com/temperature-humidity-sensors- select/ , Utoljára megtekintve (2024.05.12.)

[21] Faroqi, A., Efendi, M.R., Ismail, D.T., Darmalaksana, W.: Design of Arduino Uno Based Duck Egg Hatching Machine With Sensor DHT22 and PIR Sensor, *6th International Conference on Wireless and Telematics (ICWT)*,2020, pp. 1-4.

[22] Debele, G.M., Qian, X.: Automatic Room Temperature Control System Using Arduino UNO R3 and DHT11 Sensor, *17th International Computer Conference on Wavelet Active Media Technology and Information Processing*,2020, ppt. 428-432.

[23] Gomes, J.B.A., Rodrigues, J.J.C., Rabêlo, R.A.L, Kumar, N., Kozlov, S.: IoT-Enabled Gas Sensors: Technologies, Applications, and Opportunities, *J. Sens. Actuator Netw.* 2019, ppt. 1-29.

[24] Romadhon, A.S., Widyaningrum, V.T.: Application of Sensors in Arduino as a control in Smart Home, *IEEE 8th Information Technology International Seminar (ITIS)*,2022, ppt. 130-133

[25] Sahu, P., Dixit, S., Mishra, S., Srivastava, S.: Alcohol Detection based Engine Locking System using MQ-3 Sensor, *International Research Journal of Engineering and Technology (IRJET)*,2017, ppt. 979-981.

[26] Daugela, I., Visockiene, J.S., Kumpiene, J., Suzdalev, I.: Measurements of Flammable Gas Concentration in Landfill Areas with a Low-Cost Sensor, *Energies*,2021, ppt. 1-15.

[27] Rani, S.U., Rajarajeswari, S., Jaimon, J.G., Ravichandran, R.: Real-time air quality monitoring system using MQ-135 and thingsboard, *Journal of critical reviews*,2020, ppt. 4107-4116.

[28] Yam, F.K., Hassan, Z.: Innovative advances in LED technology, Microelectronics Journal Volume 36, Issue 2, 2005, pp.129-137.

[29] ESP-IDF, https://www.espressif.com/en/products/sdks/esp-idf, Utoljára megtekintve (2024.05.14.)

[30] Oner, V.O.: Developing IoT Projects with ESP32: Automate your home or business with inexpensive Wi-Fi devices.*Packt Publishing Ltd*.,2021

[31] Webszerver,https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server, Utoljára megtekintve (2024.05.13)

[32] Szinkron és aszinkron, https://nordicapis.com/the-differences-between-synchronous-and-asynchronous-apis, Utoljára megtekintve (2024.12.10.)

[33] Tomiša, M., Milković, M., Čačić, M.: Performance Evaluation of Dynamic and Static WordPress-based Websites,23rd International Computer Science and Engineering Conference (ICSEC),2019

[34] Kommunikáció, https://www.electronicshub.org/connect-esp8266-to-wifi, Utoljára megtekintve (2024.12.10.)

[35] Yan, Y.: Design of an Intelligent Vehicle Environment Monitoring System based on WiFi, Journal of Robotics, Networking and Artificial Life,2022, ppt. 249-252

[36] MQTT, mqtt.org, Utoljára megtekintve (2024.12.10.)

[37] Guan,S., Hu,w., Zhou,H.: Real-Time Data Transmission Method Based on Websocket Protocol for Networked Control System Laboratory, Chinese Control Conference (CCC),2019

[40] Bluetooth, https://www.ezurio.com/resources/blog/bluetooth-low-energy-vs-bluetooth-classic-what-s-the-difference, Utoljára megtekintve (2024.12.10.)

[41] Heydon, R.: Bluetooth Low Energy: The developer's handbook,*Prentice hall*,2012

[42] Qasim,H.,H., Abbas,F.,W., Tawfeeq,H.,N., Jasim,H.,N.,Hamza, A.,E.,Hussein,W.,Altmemi, J.,M.: Enhancing Weather Monitoring: A Comprehensive Study Utilizing IoT, ESP32, Sensor Integration, and Blynk Platform, IEEE 10th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA),2024

[43] Lellapati,A.,R., Pidugu,B.,K., Rangari,R.,S.,M.,Acharya,P.,G.,Jayaprakasan,V., Development of Advanced Alerting System using Arduino interfaced with ESP32 CAM, IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA),2022