**Smart House**

A hőmérséklet változásának

vizsgálata egy bizonyos területen belül

**Szerzők:**

Szabó Andrea  
Mészáros Adél  
Sándor Dévid-Roland

**Software Rendszerek Tervezése**

**Sapientia EMTE, 2018**

Tartalomjegyzék

[1. Bevezetés 4](#_Toc532765152)

[1. Követelmény specifikáció: 5](#_Toc532765156)

[a. Felhasználói követelmények : 5](#_Toc532765157)

[b. Rendszerkövetelmények: 5](#_Toc532765162)

[2. Megvalósítás menete: 5](#_Toc532765176)

[2.2. Raspberry Pi 3 6](#_Toc532765177)

[2.2. Adatbázis: 8](#_Toc532765190)

[2.3. Web alkalmazás 9](#_Toc532765194)

[3. Diagramok: 9](#_Toc532765195)

[3.1. Activity diagram 9](#_Toc532765196)

[4. Következtetések: 9](#_Toc532765197)

1. **Bevezetés**

A projektünk célja, hogy egy olyan weboldal létrehozása, amely segítségével nyomon tudjuk követni a hőmérséklet változását illetve ingadozását otthonaikban, illetve környezetükben. Mivel napjainkban elengedhetetlen eszköz lett a számítógép és mobiltelefon, ezért egy olyan weboldalt fejlesztettünk, amely sokat segítene az emberek mindennapjaiban.

Projektünk megvalósításához egy Raspberry Pi 3 B+ áll a rendelkezésünkre, amely valós hőmérséklet értékekkel szolgál. A weboldalunk az értékeket egy adatbázis segítségével éri el, és az értékek függvényében, diagramok segítségével tudjuk szemléltetni a hőmérséklet ingadozását.

Mindennapjaink megkönnyítése végett hoztuk létre ezt a projektet, hogy azok akiknek felhasználói engedélyük van, könnyen tudják nyomon követni a hőmérséklet ingadozást otthonaikban. A projekt egy tovább fejlesztett változata, az lenne, hogy ha egy felhasználó úgy látja, hogy az utóbbi időben túlságosan lecsökkent a hőmérséklet az otthonában akkor manuálisan be tudja kapcsolni a fűtést, így mire hazaér, a lakás fel lesz melegedve. Viszont idő hiányában nem sikerült ezt megvalósítanunk.

1. **Követelmény specifikáció:**
   1. **Felhasználói követelmények :**
   * A felhasználót a weboldal betöltése után egy bejelentkezési környezet fogadja, ahova előző regisztráció alapján, email segítségével tud belépni.
   * Az alkalmazás számára engedélyezni kell:

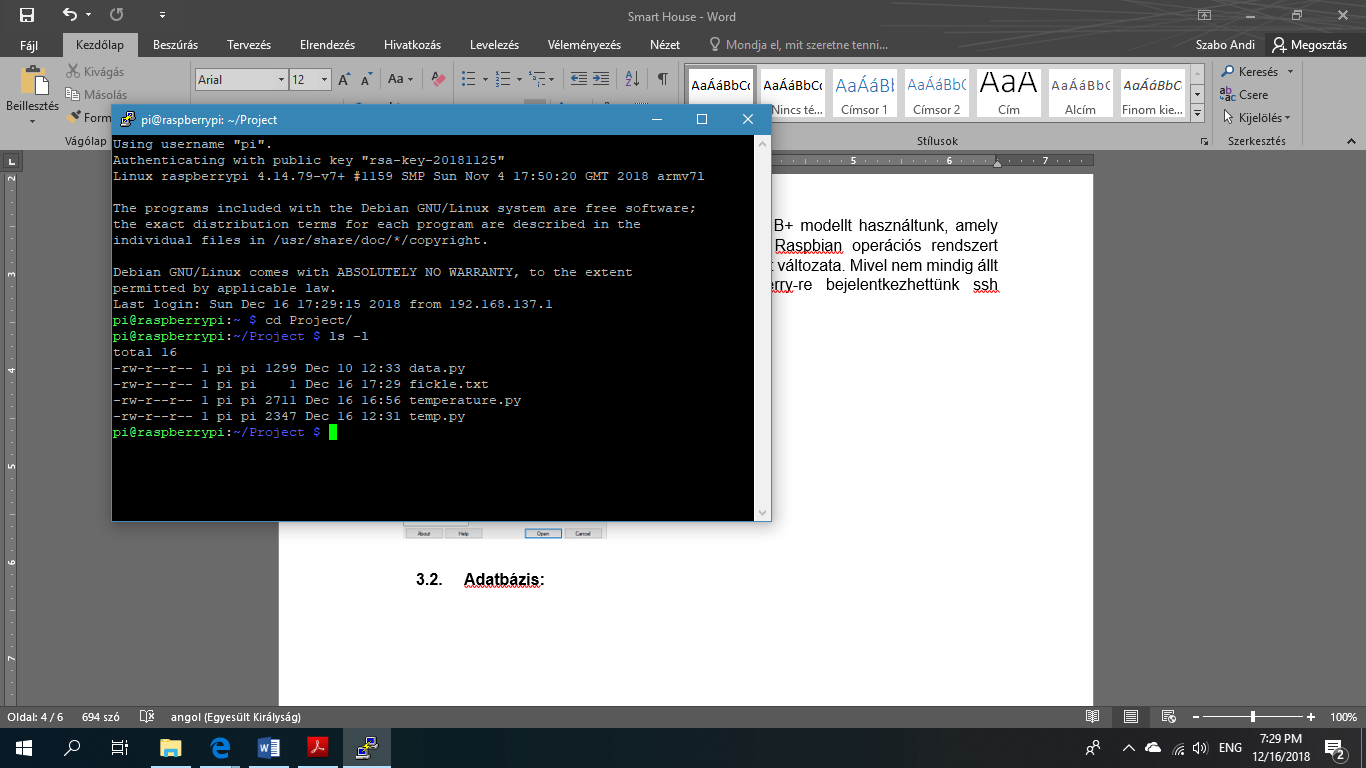
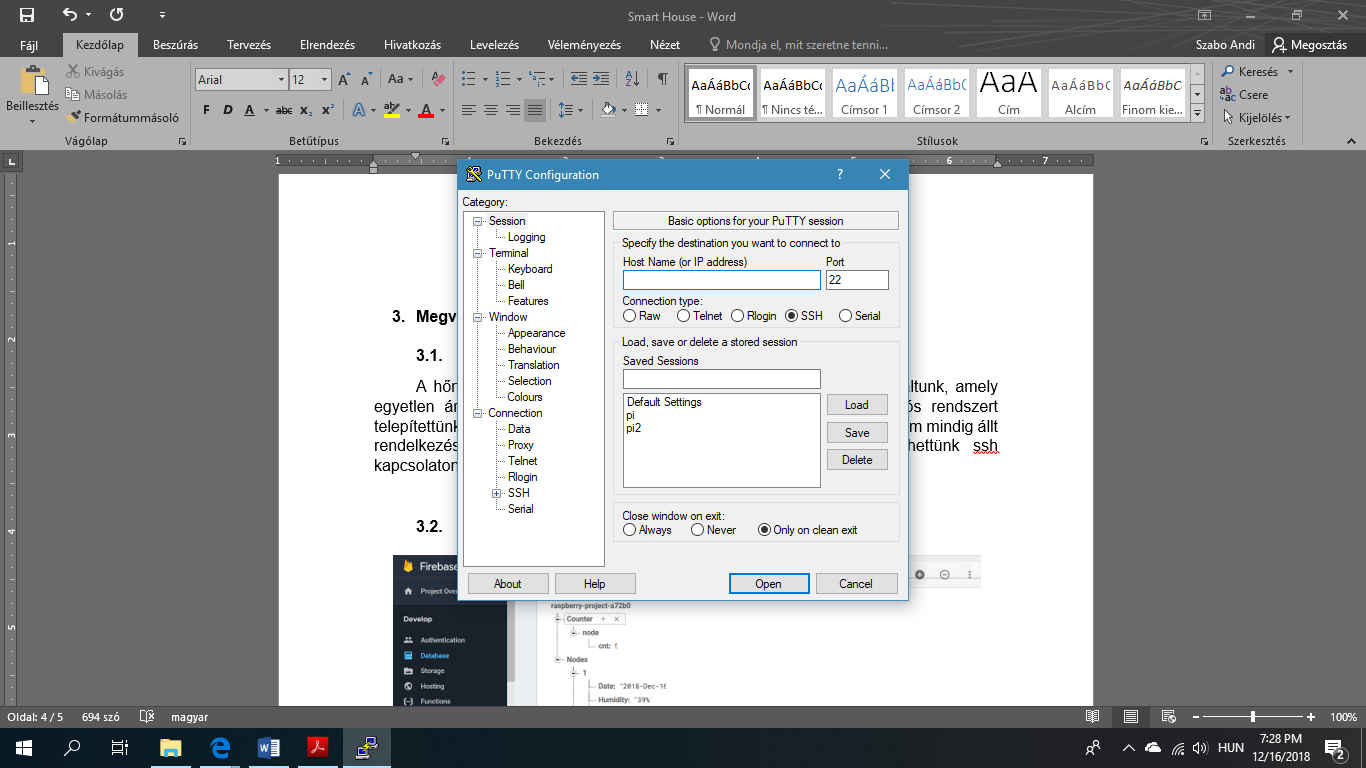
- Internet használatát

- A személyes adatokhoz való hozzáférést

* 1. **Rendszerkövetelmények:**
  + Funkcionális követelmények :
    - Bejelentkezés email alapján
    - Regisztráció
    - Dokumentáció megjelenítése
    - Adott időponthoz a hőmérséklet lekérdezése
    - Adott szenzorhoz tartozó hőmérséklet változásának a lekérése diagram segítségével
  + Nem funkcionális követelmények:
    - Raspberry Pi 3 B+
    - MVC Design Pattern alkalmazása
    - Python
    - Online alapú adatbázis
    - Egyszerű felhasználás
    - Egyszerű feltételeknek tesz eleget

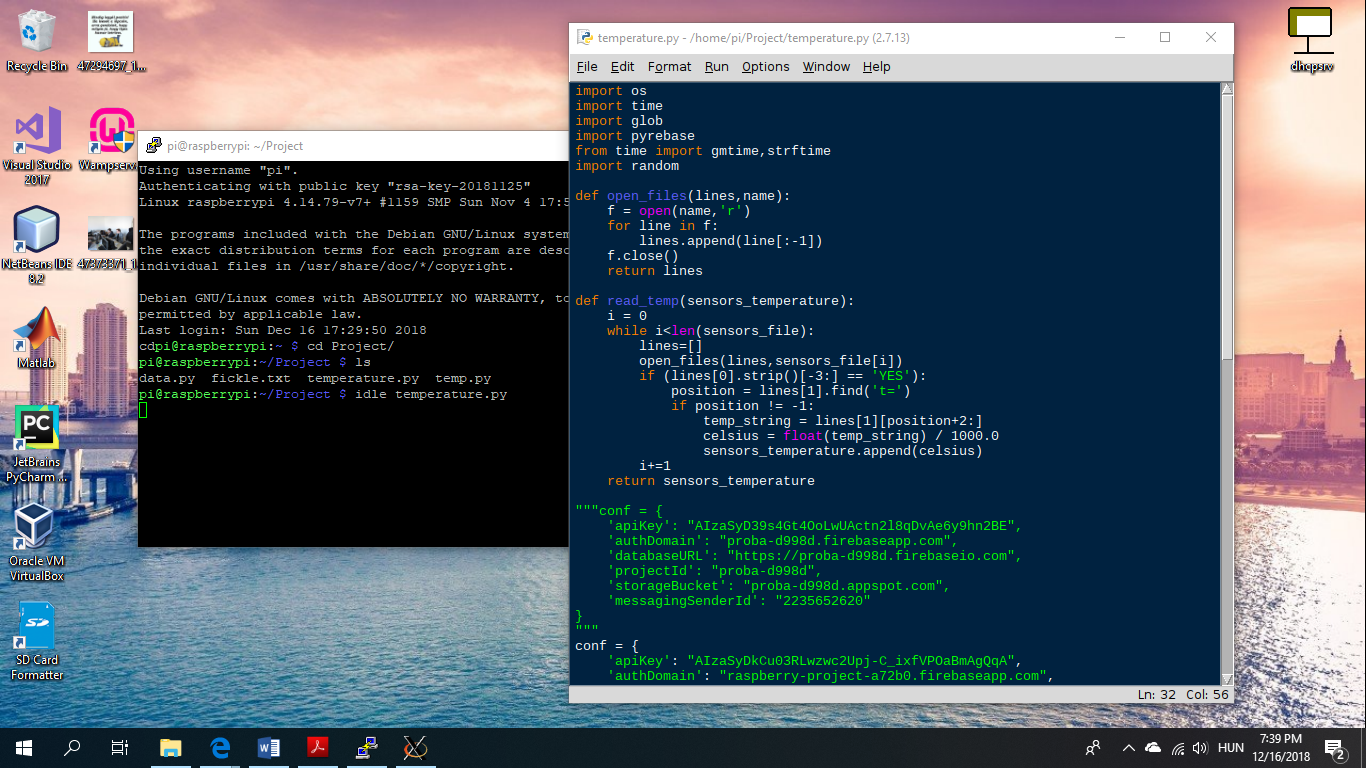
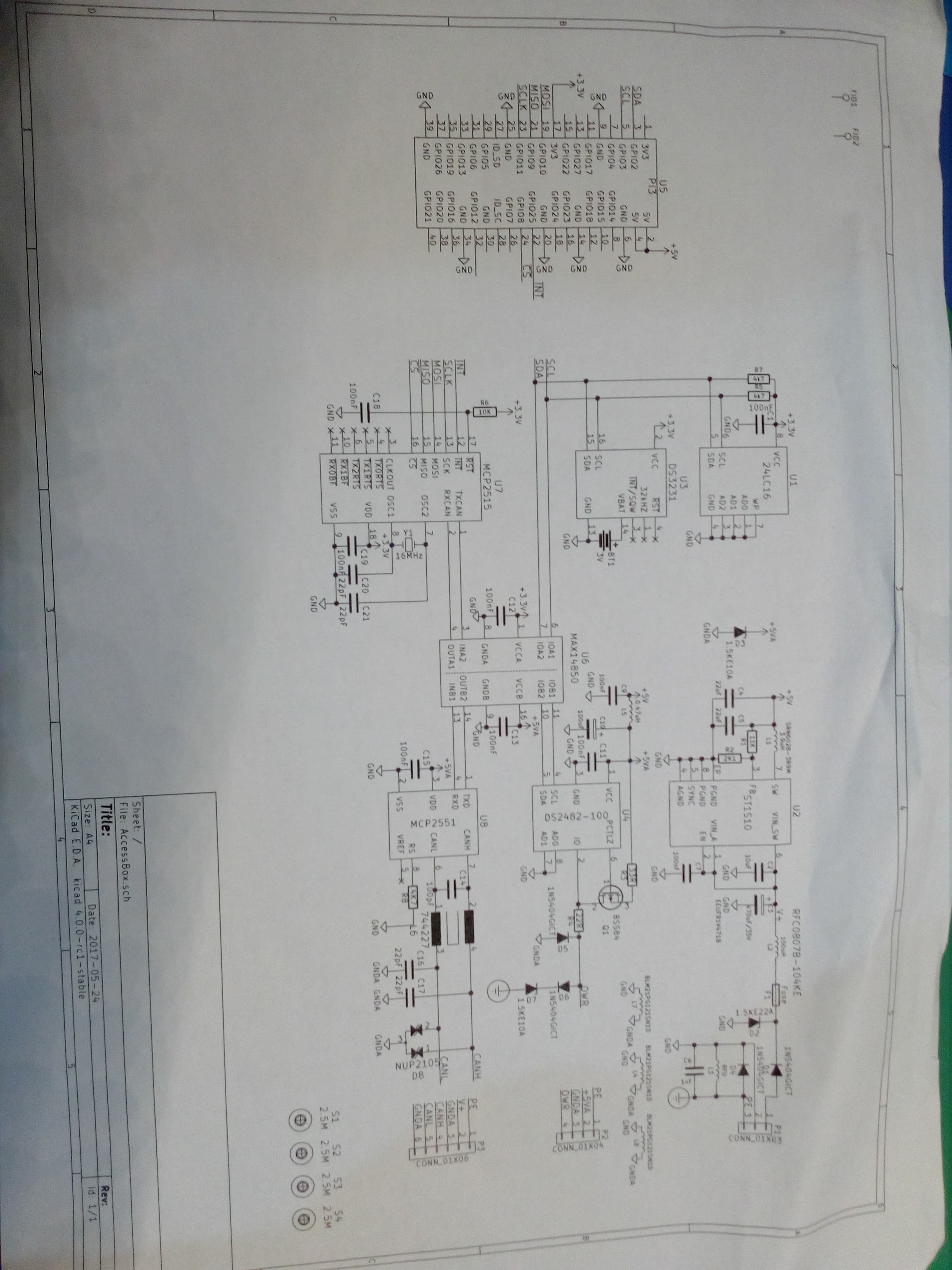
1. **Megvalósítás menete:**
   1. **Raspberry Pi 3**

A hőmérséklet leolvasására egy Raspberry Pi 3 B+ modellt használtunk, amely egyetlen áramköri lapra integrált számítógép. Raspbian operációs rendszert telepítettünk rá, amely a Debian raspberry-re optimalizált változata. Mivel nem mindig állt rendelkezésünk külső HDMI kijelző, így a raspberry-re bejelentkezhettünk ssh kapcsolaton keresztül, egy hálózati kábel segítségével.



1. Ábra – Putty 2. Ábra – Alkalmazási felület

A könnyebb kezelhetőség érdekében egy Xming nevezetű grafikus terminál szervert használtunk, amely lehetővé tette a Raspbian által használt grafikus felhasználói felületek megjelenítését, Windows alatt is, a mi esetünkben a Pythont. Ez nagyban megkönnyítette a programozást.

3.Ábra - Xming

A kapcsolás rajz:

4. Ábra – Raspberry Pi kapcsolási rajza

Mivel tartalmaz egy I2C 1-Wire átalakítót(DS2482-100), ezért szükségünk volt egy parancsra ami segítségével elértük a szenzorjainkat.

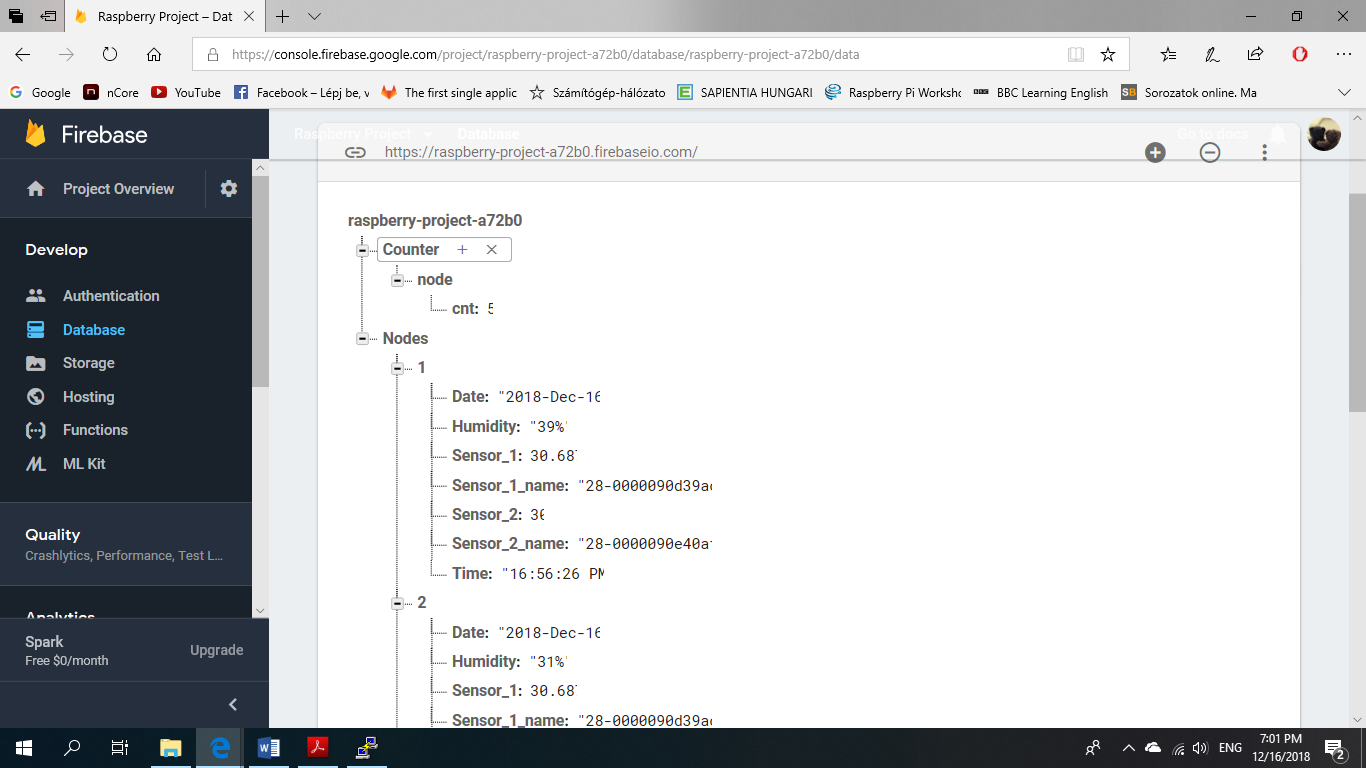
*echo ds2482 0x18 > /sys/bus/i2c/devices/i2c-1/new\_device*

Ezt a parancsot minden induláskor automatikusan inicializálja.

Az alapbeállítások után következett az adatbányászat és az adatbázishoz való kapcsolódás és adatküldés megvalósítása. Ennek megvalósítása érdekében a python programozási nyelvet választottuk, elsősorban mivel dinamikus típusokat és automatikus memóriakezelést használ és mivel a raspberry is támogatja. A nevében a PI is azt jelenti.

A megírt python kódot a raspberry 1 percenkét lefuttatja egy megírt shell szkript segítségével. Így ha a raspberry csatlakoztatva van, minden percben az adatbázis kibővül egy új taggal.

* 1. **Adatbázis:**



5. Ábra – Firebase Database

A Firebase adatbázis bízrosítja a felhasználók számára azokat az eszközöket, melyekkel kiváló minőségű applikációkat fejleszthetnek ki. Ez nem csak egy egyszerű adatbázis, hiszen valós idejű, tehát egy módosítás során szinte azonnal frissül az adatbázis és az új értéket fogja tartalmazni. Nem kell külön szervert írni, mivel gyors/munkát biztosít, és több millió felhasználót képes egy időben kiszolgálni. Több platformmal dolgozik, így biztosítja a webes, iOS és Android applikációk fejlesztését.

Jelen esetben a lekérdezéseket egy C# programmal oldottuk meg. Míg a python esetében pyrebase-t használtunk, addig C# estében FireSharp-ot kellett. A FireSharp tartalmaz sok olyan metódust amire szükség volt a program megírása során. A FireSharp-nak az egyik érdekessége az, hogy aszinkron metódusai vannak.

*pl: FirebaseResponse response = await client.GetTaskAsync("/" + i);*

Az await várakozást jelent, tehát ez a programrészlet azt végzi el, hogy minden aktuális indexű adatot lekér.A metódusoknak beszédes neveik vannak, tehát könnyen megérthetők és használhatók.A fenti példában a klienstől (jelen esetben a firebase adatbázis) kérjük a feladatot aszinkron módon. A ”/” jel megmondja hogy milyen *mappában* keressen. Ha azt adnánk meg, hogy ”Information/”, akkor az information alatti értékeket adná csak vissza.

Ha az ’i’ változónak a 3 értéket adjuk meg, akkor vissza adja a hármas sorszámú adatot, amiben több érték is szerepel.

A client.SetTaskAsync() metódus, az adatbázisba teszi az adatot. Itt is két paraméterre van szükség, az egyik a path a másik pedig az adat. A path parameter tehát azt határozza meg, hogy az adatbázison belül, hova mentődjenek el az adatok.

Ezután meg kell mondjuk, hogy milyen formátumban kérjük az adatot, ami el van mentve az adatbázisunkban.Emiatt kellett létre hozni egy új osztályt aminek a neve Data, ami a következő képpen nézz ki:

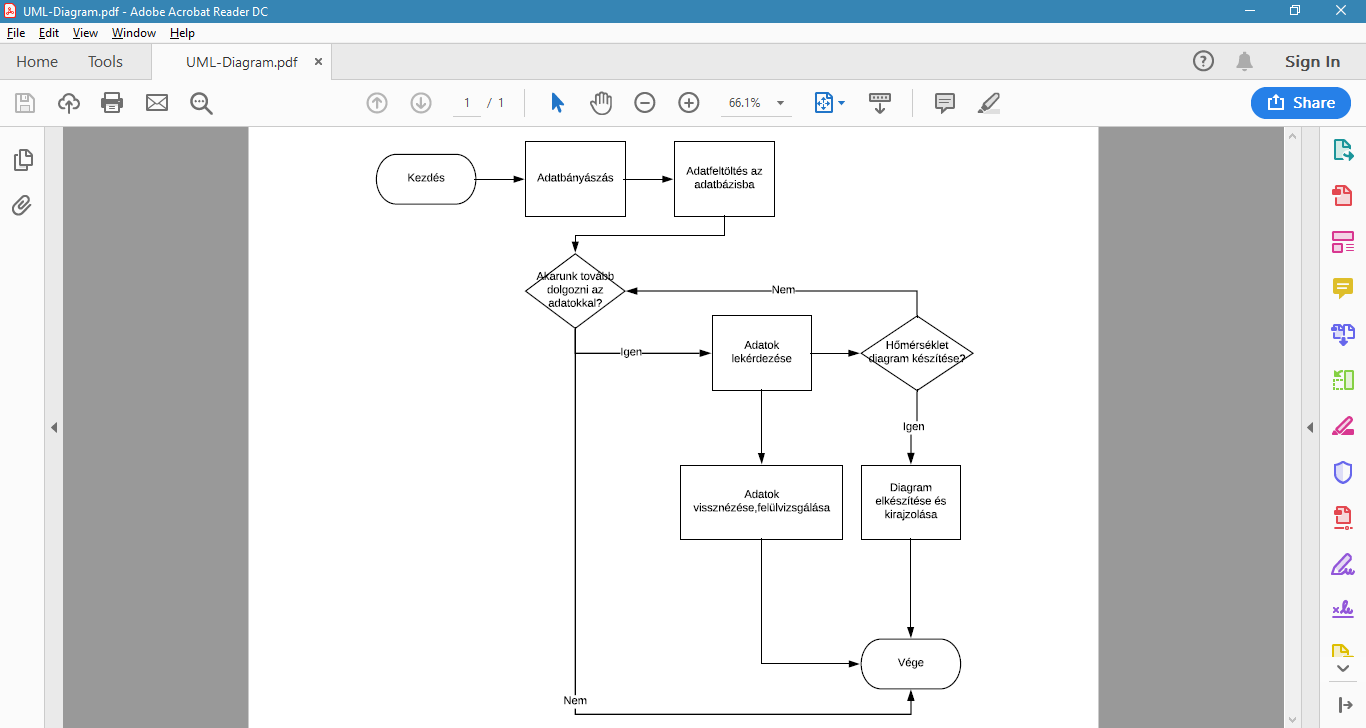
*Data received = response.ResultAs<Data>();*

A fenti metódusnál is lehet érteni, hogy mire való, a *response* amiben megkaptuk már fentebb az értékeket, azt adja át a *recevied* változónak Data típusú ként.

A firebase Authentication modulja tudja kezelni a felhasználókat is. A metódusok nagyban hasonlítanak az adatbázisban használt metódusokhoz. A felhasználó különböző belépési lehetőségek közül tud választani. Mi jelen esteben az email cím, jelszó kombinációt használjuk.

* 1. **Web alkalmazás**

1. **Diagramok:**

**3.1. Activity diagram:**

# **Következtetések:**

Munkánk eredményeként létrehoztunk, egy felhasználóbarát, jól átlátható weboldalt, ahol minden szükséges információ megtalálható, ami egy szoba illetve helység hőmérséklet változásának figyeléséhez szükségesek.

A projekt előrehaladtával számos új dologgal ismerkedtünk meg, ilyen például az MVC struktúrájú weboldal tervezése, a Raspberry Pi illetve a Firebase használata, felépítése.

Jövőbeli terveink közé tartozik újabb funkciók beépítése, azáltal például, hogy több szenzort helyezünk be, így nem csak hőmérséklet változásának a vizsgálatával tudnánk foglalkozni. Illetve, még tovább fejlesztve a projektet, ha egy bizonyos felhasználó úgy látja, hogy a lakásában a hőmérséklet túl lecsökkent akkor ő manuálisan tudja bekapcsolni, így mire hazaér meleg lakás fogja fogadni.