A weboldal és a mikrokontroller kapcsolata

Rendszerünk működése nagyvonalakban a következő lépésekből áll. A felhasználó ellátogat a weboldalra és regisztrálja magát, ezek után bejelentkezik és elnavigál egy adott eszköz oldalára. Az oldalon kivárja sorát, majd sorra kerülés esetén megjelennek vezérlő felületek, amelynek segítségével életre keltheti az eszközt. Ezek nagyon egyszerű lépéseknek tűnhetnek, azonban a webalkalmazás és a mikrokontroller közötti kommunikáció korántsem ennyire egyszerű.

Weboldalunk PHP nyelven íródott a használt mikrokontroller, a Nodemcu ESP-32S pedig C++ nyelven programozható. A két eszközben többek között az a közös, hogy mind a kettő csatlakoztatható az internethez. A csatorna így már adott már csak egy közös protokollt kell találnunk, melyen folyhat a kommunikáció, és lehetőség szerint nyíltforráskódú.

Így esett választásunk az MQTT protokollra. Az elnevezés a Message Queuing Telemetry Transport rövidítése. Kifejezetten számítógépek, és egyéb eszközök, mint például mobiltelefonok, mikrokontrollerek és szenzorok működéséhez tervezett protokoll. Előnye, hogy korlátozott sávszélességű hálózatokra lett optimalizálva, továbbá olyan hálózatokra, ahol a kapcsolat időszakos. Az MQTT a TCP/IP protokollon keresztül kommunikál. Az MQTT architektúrájában kétféle szerepkör létezik: az egyik a kliens, a másik az úgynevezett broker. A broker az a szerver, mellyel a kliensek kommunikálnak. A felépítés lényege, hogy a kliensek nem kommunikálnak egymással, minden üzenetet a szervernek küldenek. Az információcsere az ügyfelek között a szerveren keresztül történik. Minden ügyfél lehet adó, vevő vagy mindkettő egyszerre. Ezt a következő képpel szeretném szemléltetni.

Kép…

Az MQTT egy esemény vezérelt protokoll, ami azt jelenti, hogy nincs folyamatos adatátvitel. Ez minimálisra csökkenti a forgalmat a hálózaton. A kliens csak akkor küld adatot, ha valamit szeretne megosztani a többiekkel, a szerver pedig csak akkor küld adatot a klienseknek, ha valaki információt szeretne lekérni.

A protokollon az üzeneteket témákba, úgynevezett ,,topics’’ formában rendezve tehetjük közzé. A témák hierarchikusan vannak felépítve és a témákat egymástól a ,,/’’ jellel tudjuk elválasztani, ez hasonlatos a számítógépes fájlrendszerek felépítéséhez. Ez a felépítés elősegíti, hogy a felhasználó pontosan be tudja azonosítani az általa használandó témát, így nem keverve össze azokat. A témák nincsenek külön definiálva, a témák nevét az üzenetek tárolják, így ha küldünk egy üzenetet, amely egy eddig nem definiált témához tartozik, akkor ehhez már csatlakozhatnak is egyéb kliensek.

A kommunikáció során négy művelet lehetséges

* Feliratkozás, azaz ,,subscribe’’,
* Közzététel, azaz ,,publish’’,
* Ping
* És végül lecsatlakozás, azaz ,,Disconnect’’

Feliratkozás során jelezzük a szerver számára, hogy egy adott témával kapcsolatban információt szeretnénk lekérni. Közzététel esetén egy adattömböt hozunk létre, mely tartalmazza az adott témát, a témához tartozó felhasználói nevet és jelszót, a kliens azonosítóját, a titkosítási módot és végül az üzenetet. Amikor elküldjük a szervernek az adattömböt, megvizsgálja hogy létezik-e ilyen téma. Abban az esetben, ha már létezik, akkor a témához tartalmazó adatot lecseréli az általunk küldött adatra, ellenkező esetben létrejön a téma a hozzátartozó adattal együtt. A Ping funkció esetén a szervernek egy kérést küldünk, ő pedig egy válaszban biztosít minket arról, hogy közöttük él a kapcsolat. A lecsatlakozással pedig azt közöljük a szerverrel, hogy már nem kell a feliratkozott kliensek felé adatot küldenie és a feliratkozott kliensektől már nem fog üzenetet kapni.

A projektünk során a Blue Rhinos által PHP nyelven elkészített könyvtárat használtuk fel, melynek elérhetőségét a források között feltüntettük.

A fentebb feltüntetett négy műveletből kettőt használunk fel, a közzétételt és a lecsatlakozást.

KÓD

Az itt látható kódsorban rendre, fentről lefelé a következő paramétereket kell megadnunk: a szerver elérhetőségét, a használt portot, felhasználói nevet, amennyiben beállítottuk, jelszót, és a kliens azonosítóját. A konstruktorban példányosítjuk az osztályt az átadott paraméterek segítségével. Csatlakozunk a szerverhez, és a közzétételt segítő metódusban paraméterként adjuk át a kívánt témát, az üzenetet és a titkosítási formát. Az elküldés után jelezzük a szervernek, hogy bontjuk a kapcsolatot. Ha a küldés nem sikerült, akkor a ,,connect()’’ függvény visszatérési értéke hamis. Ezt egy logikai elágazással kiértékelve a felhasználó felé jelezhetjük, hogy az üzenet elküldése sikertelen volt.

A mikrokontroller használata során a felsorolt négy műveletből a feliratkozás műveletet használjuk. Az mikrokontrolleren az alábbi kód működik.

kód..

Felhasználjuk az ,,MQTT.h’’ előre elkészített, és beimportáljuk az ,,MQTTDatas.h’’ általunk definiált könyvtárat, melyben a kapcsolathoz szükséges információkat tároljuk, ilyen információ a kliens azonosítója, a felhasználói név és a jelszó.

A topic változóban megadjuk a feliratkozni kívánt téma nevét. A ,,Connect()’’ metódusban másodpercenként ellenőrizzük, hogy sikerült e a feliratkozás, ha nem akkor újra feliratkozunk a témára. Az ,,MQTTLoop()’’ metódus felel azért, hogy ha a kliens még nem csatlakozott a szerverhez, akkor csatlakoztatja azt.

A következő sorokban azt a kódrészletet tekinthetjük meg, mely a mikrokontrolleren fut és feldolgozza a szervertől kapott információkat.

…kód

A ,,setup()’’ nevezetű metódus végzi a mikrokontroller inícializálását. Itt csatlakozatjuk a Wifi hálózathoz és az MQTT szerverhez a mikrokontrollert, majd feliratkozunk egy adott témára. A ,,messageReceived()’’ nevezetű metódus paraméterében megadunk két String típusú változót. A ,,topic’’ nevű a téma nevét, míg a ,,payload’’ a témához kapcsolódó információt fogja tárolni. Ellenőrizzük egy logikai elágazásban, hogy a visszakapott téma neve megegyezik-e a tőlünk elvárt névvel, abban az esetben ha igen, akkor beletöltjük a ,,function’’ nevezetű változóba. A ,,loop()’’ elnevezésű metódus neve is szimbolizálja, hogy azt a kódrészletet tartalmazza, amelyet a mikrokontroller újra és újra végrehajt. Itt a programozási nyelvek többségében fellelhető többirányú elágazást használunk, melynek neve ,,switch’’, mely a kapcsoló angol megfelelőjéből ered. Megvizsgáljuk, hogy a ,,function’’ változó mely adatot tartalmazza, ha valamelyik általunk definiált értékkel megegyezik, akkor a hozzá tartozó kódrészletet fogja a program végrehajtani.

Így jutottunk el egészen attól az állapottól, amikor a felhasználó a weboldalon megnyom egy gombot, addig az állapotig, hogy az általunk fejlesztett eszköz életre kel, és működésbe lép.