Bu projeyi IoT1929 kampına

murathan1047@gmail.com maili ile kayıtlı Murathan Karoğlu ile ahmetardic.kr@gmail.com – ahmetlelloard.bs@gmail.com kayıtlı

Ahmet Talha Ardıç ortak olarak yapmıştır.

Proje senaryosundan bahsedecek olursak çok genel bir şekilde son kullanıcı için kolaylaştırılmış program yüklemeyi destekleyen ve Android – IOS uygulamasıyla kolay entegre edilebilen, uzak veri tabanında çalışabilen nesnelerin interneti ekosistemi diyebiliriz.

Projemiziz başlangıcında ise odaklandığımız sorun uzaktan doğalgaz kontrolüydü. Sadece kontrol sistemi değil, gaz sensoru ile doğalgaz oda içinde belli bir yoğunluğa ulaştığında otomatik olarak vanayı kapatan bir sistem yapmak istedik. Bununda üstesinden geldik, sadece burada kalmayarak basit ve geliştirilebilir şekilde Android – IOS cross platform çalışabilen bir uygulamada yazdık. İlerde veri tabanı ve API kodlanması öğrenildiğinde kendi veri tabanımızı oluşturabilme imkanımızda olabilir. Esnek bir şekilde hazırlamak bu yüzden çok önemli ve bizde gayet esnek şekilde hazırladık.

Kullanılan kütüphaneler ve ne işe yaradıkları burada açıklanmıştır.

- **WiFi Manager:** Bu kütüphane cihaz ilk açıldığında herhangi bir WiFi 'ye kullanıcı dostu bir şekilde bağlanmamızı sağlıyor.
- **FirebaseESP8266:** Bu sayede cihazımız Firebase ile iletişim kabiliyeti kazanıyor.
- **ESP8266WiFi:** ESP8266'yı kullanabilmemiz için gereken tüm temel komutlar bu kütüphanenin içinde bulunuyor.

Yazılan fonksiyonlar ve tanımlarının açıklamaları:

1. **mylot constructor:** Bu fonksiyon veri tipi **string** olan 2 parametre alır:
    - **ID:** Bağladığımız cihaza verdiğimiz ID
    - **Device:** Cihaza verdiğimiz isim
2. **initialize_firabase:** Bu fonksiyon **setup** içinde tanımlanır ve 2 girdir alır: **User defined** kısmıda yazdığımız firebase secret kodu ve veri tabanı adresi. Bu fonksiyon **static** bir fonksiyon olduğundan class oluştuğunda sadece 1 kez oluşur ve böylece hafıza fazla yer kaplamaz.
3. **initialize_WiFi:** Bu fonksiyonda **setup** içinde çalışır. Bu fonksiyon **static** bir fonksiyon olduğundan class oluştuğunda sadece 1 kez oluşur ve böylece hafıza fazla yer kaplamaz. 2 farklı işlevi vardır:
    - Kart ilk kez açıldığında **WiFi scan modu**na geçer ve içerisine gönderilen **string** ile bir AP adres yayınlar. Bu adrese bağlanıp 192.168.4.1 default adresine giderek bağlanılacak WiFi ağı seçilir.

- Kart bir kez WiFi ’ye bağlandığında o SSID ve şifresini **EEPROM** ‘a kaydeder. Artık her açılışında ilk olarak kayıtlı ağı deneyecek, eğer bağlanamazsa bir üstte açıklananları yapacak, yok eğer bağlanırsa normal işlevselliğine başlayabilecek.

4. **getIntState: int** değer döndüren bu fonksiyon bir **getter** fonksiyonudur. **Private** olan **state** değişkenini okumamız için gereklidir.
5. **set_int_state_to_database:** İçine gönderilen değeri firebase’e göndermekle yükümlüdür. Yükümlü olduğu başka bir konu ise hata sayacıdır. Gönderme işlemi eğer 5 kez hatalı olursa ESP yeniden başlar.
6. **update_int_state_from_database:** Firebase ‘den veriyi çeken fonksiyondur. Çektiği veriyi **state** değişkenimize eşitler. Böylece kontrol mekanizması oluşmuş olur.

Böylece temel kütüphaneleri ve kullanılan fonksiyonları görmüş olduk. Araştırmalarımızda ve kodlamalarımızda ilgili kütüphanelerin github sayfalarından sıkça faydalandık.

KAYNAKÇA

- tzapu/WiFiManager: ESP8266 WiFi Connection manager with web captive portal (github.com)
- mobizt/Firebase-ESP8266: ESP8266 Firebase RTDB Arduino Library (github.com)