T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

NESNELERİN İNTERNETİ ve UYGULAMALARI ÖDEV RAPORU

AKILLI PARK UYGULAMASI

B191210008 - Gamze CEYLAN

SAKARYA

Aralık, 2021

Nesnelerin İnterneti ve Uygulamaları Dersi

AKILLI PARK UYGULAMASI

Özet

Uygulamanın değerlendirilmesinde bir IoT uygulaması olması (internet üzerinden kontrol), uygulamanın çalışması, derste gösterilenin dışında platform kullanımı ya da derste gösterilen platformlardan bir proje yapmamız istendi.

Ben de bu ödevde bir akıllı park uygulaması gerçekleştirdim. Otoparkın dolu oldup olmadığı, hangi alanların dolu ya da boş olduğu bir mobil uygulama aracılığıyla kullanıcı tarafından görülebilecek. Kişi otoparka geldiğinde uygulama üzerinden kapı aç butonuna bastığı zaman kontroller yapılıp kapı açılacak. İşlemci tarafında ise otoparkta boş olan yerler ultrasonik sensorler ile control edilecek, araç giriş çıkışları realtime bir veritabanına kaydedilecek ve mobil uygulamada güncelleme yapılacak.

1) TASARIM ve GERÇEKLEŞTİRME

Uygulamayı seçerken günlük hayatta yaşadğımız problemleri düşündüm. Çok sık karşılaştığımız otopark sorununu ele almak istedim. Arabamızı park edebilecek bir yer ararken bazen çok zaman kaybedebiliyoruz. Özellikle kalabalık şehirlerde bu sorun oldukça fazla. Zaman ise birim en önemli değerlerimizden biridir. Bu yüzden böyle bir soruna bir çözüm getirmek istedim.

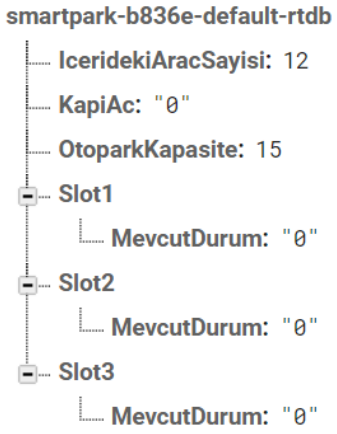
Ben aklımdaki fikrin sadece bir kısmını gösterebildim. Fikir çok daha geliştirilebilir bir fikirdir. Proje içerisinde Firebase veritabanını, Mit app inventor2, nodemcu, ultrasonic sensör servo motor, teknolojilerini kullandım

1.1)Veri Tabanı Seçimi

Projede Firebase Real Time Database'i kullandım. Bunun sebebi gerçek zamanlı bir veri tabanı olması. Çünkü otoparka sürekli giriş çıkışlar olacağı için bilgilerin veritabanında sürekli güncellenmesi gerekir. Bu yüzden Firebase'i tercih ettim.

Firebase nesnelerin internet alanında da çokça kullanılan bir veritabanıdır. Ayrıca kullanımı kolaydır ve arayüzü basittir. Nodemcu ile de kolayca bağlantı kurulur.

Uygulamamdaki veri tabanı gerçeklemesi ise aşağıdaki gibidir:



Yandaki şekil veri tabanının uygulama çalışmadan önceki halidir. İlgili slotlara araçlar giriş/çıkış yaptıkça bilgiler eklenir ve güncellenir.

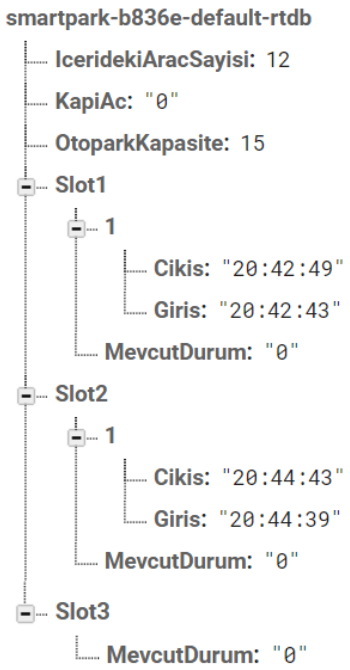
Veri alanları:

-IceridekiAracSayisi: otoparkta olan araç sayısını gösterir. Ben uyuglama ile senkronize olaması açısından 12 atadırm. Bu değer içeri araba girdikçe ve çıktıkça güncellenir.

-KapiAc: Bu uygulama tarafı için tutulmuştur. Kullanıcı uygulamadan kapı aç butonuna basarsa değer güncellenir ve 1 yazılır.

-OtoparkKapasite: otoparktaki kapasiteyi tutar. Ben yine uygulama ile paralel olması açısından 15 olarak atadım
-Slot1, Slot2, Slot3: Uygulama içinde kullandığım park alanları. İçeride araç varsa veri alanları altında araçlar aşağıdaki gibi tutulur.
-MevcutDurum: ilgili slotun boş olup olmadığını gösterir. 0 ise slot boş, 1 ise slot soludur.

Aşağıdaki tablo şekil ise veri tabanı çalışırken ki görüntüsüne aittir:



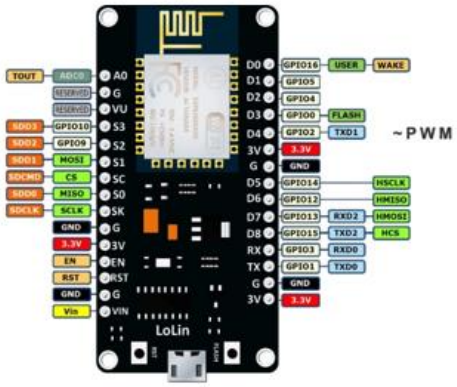
Slot1 ve Slot2'ye araçlar park etmiş ve oradan ayrılmışlardır. Araçların giriş/çıkış saatleri de yandaki gibi tutulur. Ayrıca mevcut durum da her seferinde güncellenir

Ayrıca uygulamada kapı aç butonuna basıldığı zaman eğer boş slot varsa yani otopark hala araç alabiliyorsa kapı servo motor ile açılır.

1.2)Mikrodenetleyici Seçimi

Projemde NodeMCU kartını kullandım. Yapılacak işlemler için yeterli olacağını düşündüm.

NodeMCU kartında kullanıdığım pinler ve protokoller aşağıdaki gibidir:



Ultrasonik Sensör 1:

- D5 - Echo
- D6 – Trig

Ultrasonik Sensör 2:

- D1 – Echo
- D2 – Trig

Ultrasonik Sensör 3:

- D4 – Trig
- D3 – Echo

Servo Motor:

- D8

Tüm elemanlar için de VCC – 3V ve GND – G bağlantısını yaptım.

1.3)Geliştirilen Yazılım

```
// kütüphaneler
#include <Servo.h>
#include "FirebaseESP8266.h"
#include <ESP8266WiFi.h>
#include <NTPClient.h>
#include <WiFiUdp.h>

//Firebase veritabanı adresi, Token bilgisini ve ağ adresi bilgileri
#define FIREBASE_HOST "smartpark-b836e-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "3gP01DpgiKBV9vsWvFwhlmESDP775OQqWlz9svBM"
#define WIFI_SSID "VodafoneNet-BNX3AG"
#define WIFI_PASSWORD "4LFCQYXLQN"

// tarih almak için
const long utcOffsetInSeconds = 3600;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday"
String months[12]={"January", "February", "March", "April", "May", "June", "July",
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds);

// kullanılacak değişkenler
int distance1;
int distance2;
int distance3;
int aracNo_S1 = 1; // 1. slota kaç araba girdi?
int aracNo_S2 = 1; // 2. slota kaç araba girdi?
int aracNo_S3 = 1; // 3. slota kaç araba girdi?
int angle=0; // servo motor açı

//int otoparkKapasitesi=15; // atanan otopark kapasitesi
int iceridekiAracSayisi=12; // içeride olduğu varsayılan araç sayısı

FirebaseData veritabanim;
Servo servo;
```

Kullanılan kütüphaneler yandaki gibidir.

- Servo motoru çalıştırmak için Servo kütüphanesi kullanılır.
- Firebase üzerinden bağlantı sağlamak için FirebaseESP8266 kullanılır.
- ESP8266, modülün internete çıkması için kullanılan kütüphanedir.
- NTC, realtime saati almak için kullanılır.
- UDP ise bir bağlantı kurmadan verileri göndermek için kullanılan bir protokoldur. Bu kütüphaneyi yine saati çekmek için gerekli olan kütüphanelerden biridir.

Firebase'e bağlanmak için gerekli tanımlamaları yaptım.

```
void setup() {

  Serial.begin(115200);

  // wifi bağlantısı
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Ağ Bağlantısı Oluşturuluyor");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  // zaman başlat
  timeClient.begin(); // saati alma
  timeClient.setTimeOffset(10800); // türkiye GTM +3

  // firebase bağlantısı
  Serial.println();
  Serial.print("IP adresine bağlanıldı: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  //Firebase bağlantısı başlatılıyor
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  //Ağ bağlantısı kesilirse tekrar bağlanmasına izin veriyoruz
  Firebase.reconnectWiFi(true);


void loop() {

  timeClient.update(); // zaman güncelleme

  // slot kontrolleri
    Slot1();
    Slot2();
    Slot3();

  delay(1000);
  Serial.print("icerideki AracSayisi ");
  Serial.println(icerideki AracSayisi);

 // Firebase.setInt(veritabanim, "OtoparkKapasite", 15);
  Firebase.setInt(veritabanim, "Icerideki AracSayisi",icerideki AracSayisi );

// kapı durumu okunuyor, kosul sağlanırsa açılıyor
  Firebase.getString(veritabanim,"KapiAc");
  String durum = veritabanim.to<String>();
  if(durum=="1" && icerideki AracSayisi <15){
       ServoHareket();
       Firebase.setString(veritabanim, "KapiAc", "0");

   }
}
```

Zamanı almak için NTC'nin kullanımı için gerekli gün ve ay dizilerini tuttum, uygulama içinde kullanacağım değişkenleri tanımladım. Her slota kaç araba giriş/çıkış yapacağını tutmak için ayrı ayrı aracNo_S değişkenlerini tuttum.

Bu kısımda otopark kapasitesini ve içerideki araç sayısını yazılımda verdim.

Setup içinde Wifi, Firebase, zaman için gerekli kodları yazdım

Daha sonra kullanacağım elemanların pinlerini ayarladım

Slot1, Slot2 ve Slot3 içinde slotlar için gerekli olan kodlar yazdım. Bu fonksiyonlardan biri bir sonraki resimdeki gibidir. üç fonksiyon da temel olarak aynı işlemleri yaptığından dolayı sadece birini ekledim.

İçerideki araç sayısını yukarıda da belirttiğim gibi kendim yazılım kısmında belirlerim ve veri tabanına ekledim.

Kapı Aç komutunun gelip gelmediğini ise yandaki gibi kontorl ettim. Komut geldiğinde Servo motor çalışıyor ve kapı açılıyor.

Slot fonksiyonlarından biri aşağıdaki gibidir:

```
void Slot1(){

  digitalWrite(D4,LOW);
  delayMicroseconds(2);
  digitalWrite(D4,HIGH);
  delayMicroseconds(2);
  digitalWrite(D4,LOW);

  long timedelay1 = pulseIn(D3,HIGH);
  distance1 = timedelay1*0.034/2;

  String number = String(aracNo_S1);

  if(distance1<=5){
       Serial.println("Slot1 dolu");
       Firebase.getString(veritabanim,"Slot1/MevcutDurum");
       String durum = veritabanim.to<String>();

       if(durum == "1"){}
       else if(durum == "0"){
            Firebase.setString(veritabanim, "Slot1/MevcutDurum", "1");
            Firebase.setString(veritabanim, "Slot1/"+number+"/Giris", timeClient.getFormattedTime());
            iceridekiAracSayisi++;
        }
        delay(15);
    }
```

Öncelikle LOW ve HIGH değerleri okunuyor. Ve sensorörün ölçtüğü mesafe hesaplanıyor.
Eğer mesafe 5 cm den küçükse araç var olarak Kabul ettim ve veri tabanını buna göre güncelledim.
Loop sürekli bir döngü halinde çalıştığı için ve veri tabanı sürekli güncelleneceği için giriş çıkış saatini tutmak için veri tabanının bir önceki durumuna bakarak karar verdim. Eğer sürekli 5 ten küçük değer ölçüyorsa ve zate veri tabanına araç var olarak kayıt edilmişse zaman gücellenmiyor.

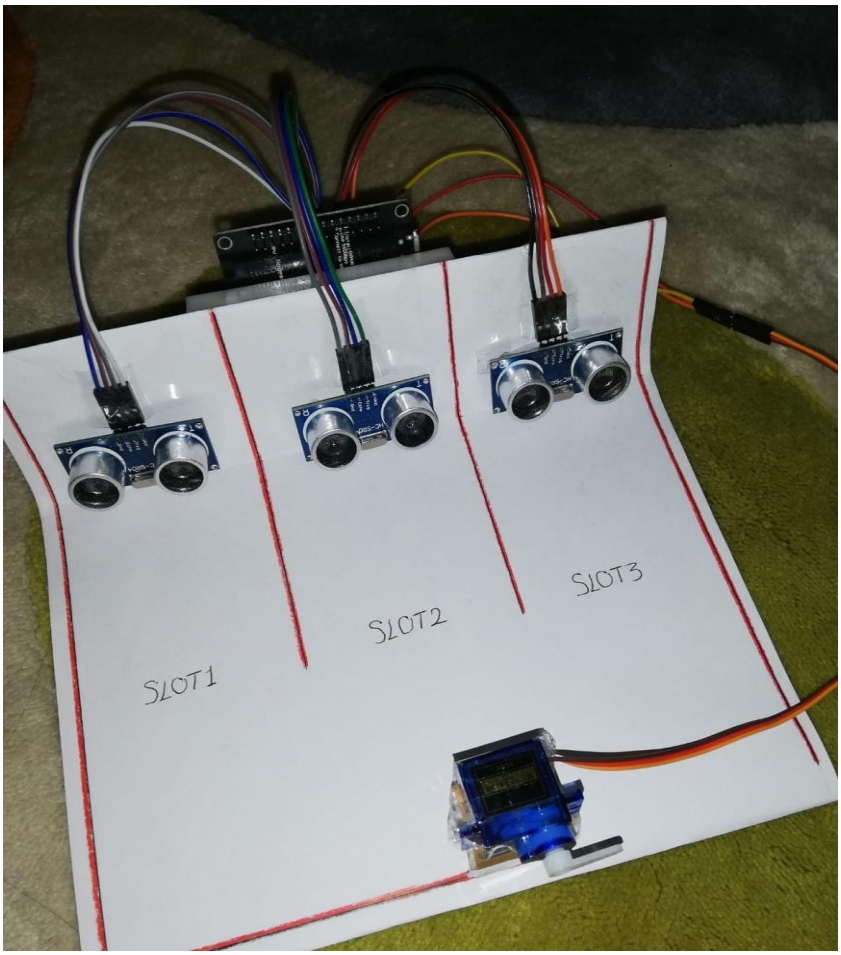
```
  else{
       Serial.println("Slot1 bos");
       Firebase.getString(veritabanim,"Slot1/MevcutDurum");
       String durum = veritabanim.to<String>();

        if(durum == "1"){
            Firebase.setString(veritabanim, "Slot1/MevcutDurum", "0");
            Firebase.setString(veritabanim, "Slot1/"+number+"/Cikis", timeClient.getFormattedTime());
            iceridekiAracSayisi--;
          aracNo_S1++;
        }
        else if(durum == "0"){ }
   }
   delay(15);
}
```

Veriler sadece senörün ilk değerleri ölçtüğü zaman güncelleniyor ve veri tabanına kaydediliyor. Bu şekilde sadece giriş/çıkış saatlerini çekebildim.

Servo motorun hareket etmesi için yazdığım fonksiyon:

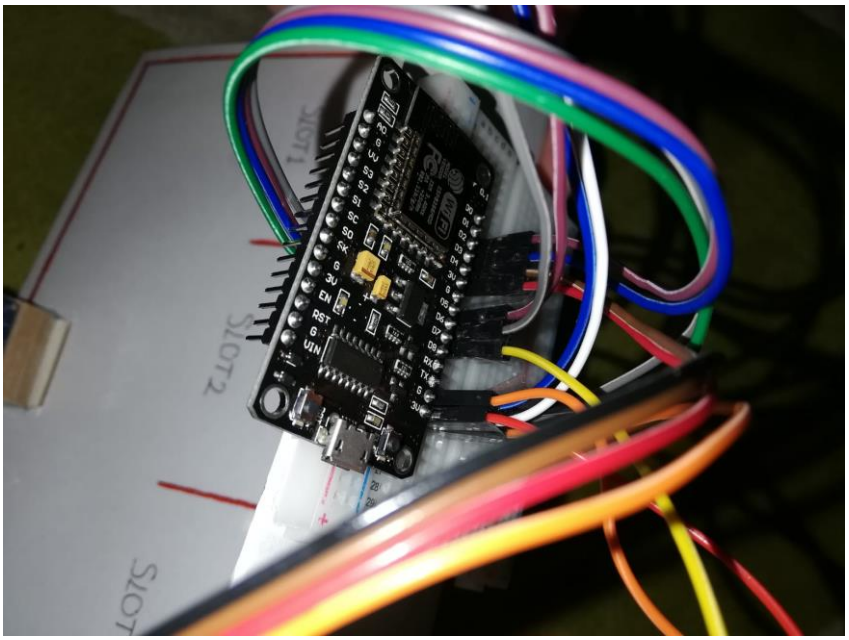
```
void ServoHareket(){

  for(angle = 90; angle < 180; angle += 1)
      {
        servo.write(angle);
         delay(5);
      }
      delay(2000);
      for(angle = 180; angle>=90; angle-=1)
      {
        servo.write(angle);
        delay(15);
      }
       delay(15);
}
```

1.4) Proje Görünümü



Projeyi yandaki gibi tasarladım. 3 slot boş yer var ve araçlar sensörlerin önüne parkediyor.

Servo motorun olduğu yer ise kapıdır. Kullanıcı kapı aç butonuna bastığında controller yapılır ve servo motor çalışır



Bağlantılar yandaki şekildeki gibidir.

1.5)Mobil Uygulamanın Gerçekleştirilmesi

Mobil uygulama için Mit App Inventor2 ‘ yi seçtim. Bunun sebeni basit bir arayüzü olması, işlemler için kod gerektirmemesi, firebase ile kontrolünün kolay olması. Yapmak istediğim işlemler için de yeterli oldu.

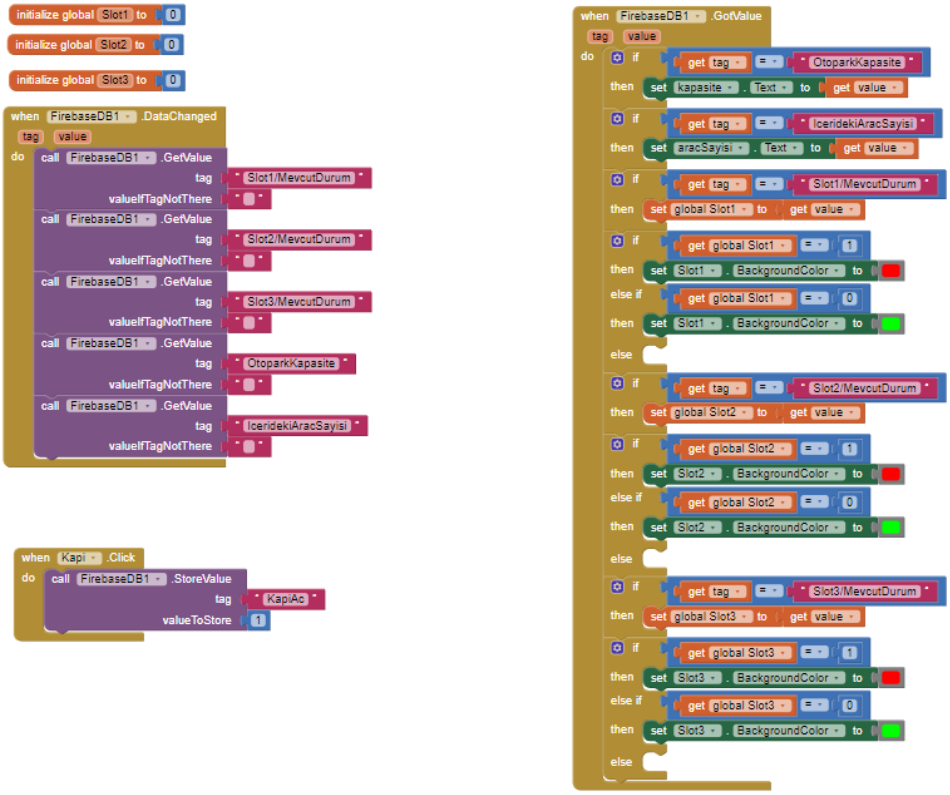
Uygulama arayüzü yandaki gibidir. Otopark kapasitesi ve içerideki araç sayısını uygulama çalıştığı yazılım ile veritabanına gönderiyorum ve uygulmaya veri tabanından çekiyorum. Bu şekilde içerideki araç sayısını uygulamada güncel tutabiliyorum.

Kapasiteyi uygulamaya Slotları bu şekilde eklediğim için 15 olarak ayarladım. 3 sensörüm olduğu için de 3 park yeri kurguladım.

Her slot otoparktaki park alanını gösterir. Slotlar dolu ise kırmızı, boş ise yeşil yanar. Park yerlerine araç park ettikçe sensörlerden bilgi gelir yazılım ile veri tabanına bu bilgi gönderilir ve veritabanına MevcutDurum “1” olarak yazılır. Bu şekilde uygulama da 1 gördüğü zaman kırmızı yanar, 0 gördüğü zaman yeşil yanar.

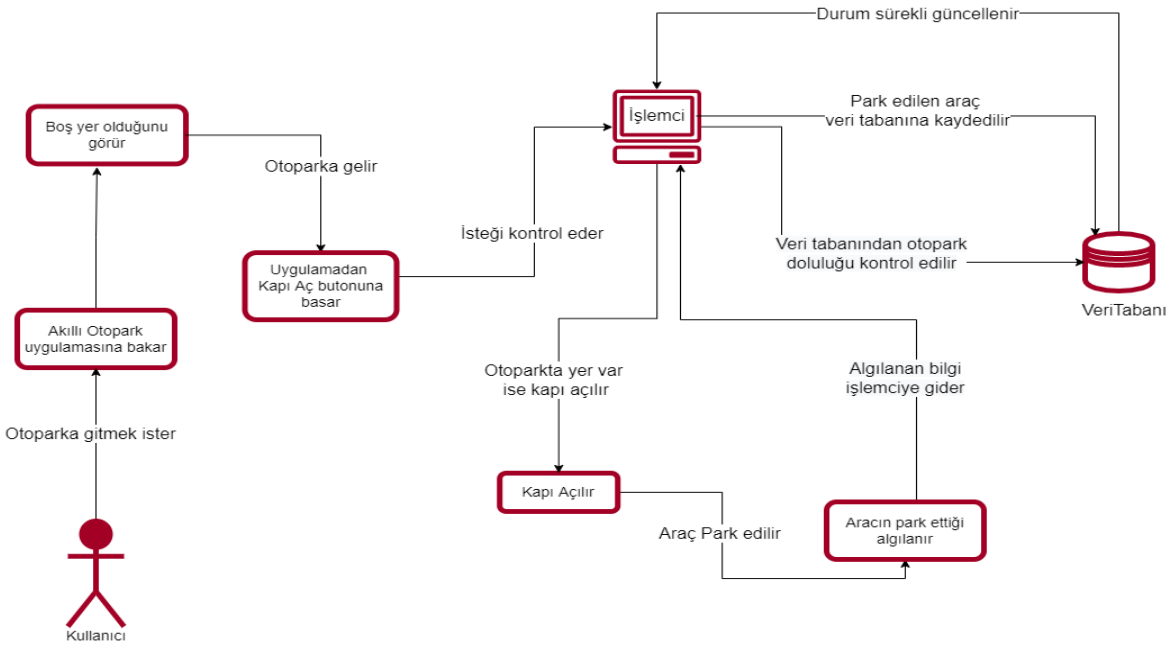
Bir de kapıyı aç butonu ekledim. Kullanıcı otoparka geldiği zaman bu butona basarak kapıyı kendi açabilir. Kapı açıldıktan sonra kendiliğinden kapanır.

Aşağıda ise uyuglamayı yazılım olmadan basit bloklarla gerçekleştirilme şekli vardır

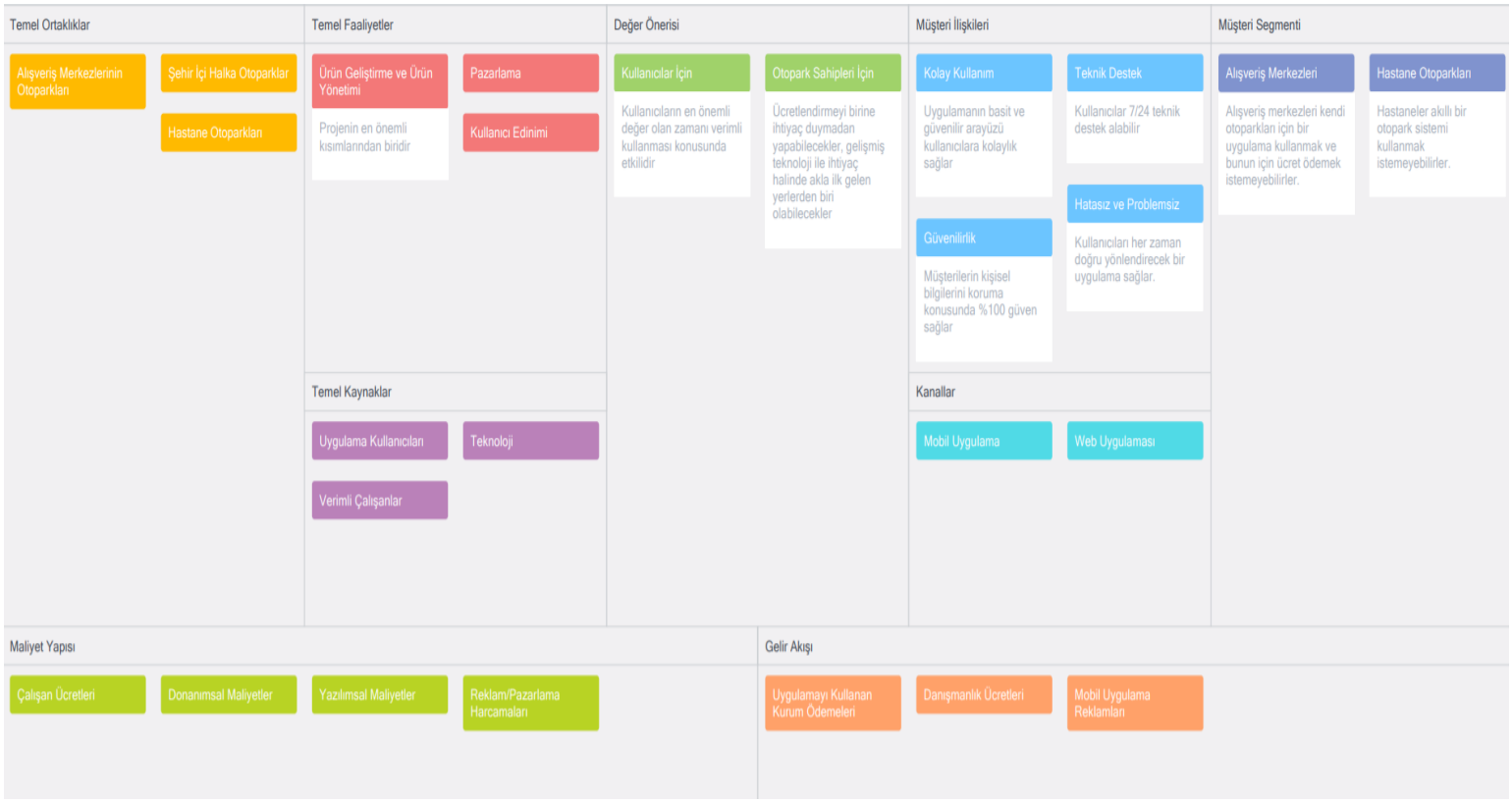


Veriler, veri tabanında güncellendikçe DataChanged bloğu ile uygulamada güncellenir. Kapı aç butonuna bassıldığında Kapi.Click çalışır. Veri tabanından gelen güncel bilgilerin kontrolü yapılarak uygulama içinde gerekli işlemler GotValue bloğunda yapılır

1.6) UML DİYAGRAMI



1.7)İş Modeli Kanvası



1.8) BÜYÜK VERİ ANALİZİ

Günümüzde sosyal medyanın, internetin kullanımının yaygınlaşmasından dolayı büyük veri daha çok hayatımıza girmiştir ve popülerlik kazanmıştır. İnsanlar inanılmaz büyüklükte veriler üretir hale gelmiştir. Teknolojinin gelişmesi ile de bu kadar büyük verileri depolamak kolaylaşmış ve ucuzlaşmıştır. Büyük veri analizinin önemi de gittikçe artmıştır.

Öncelikle büyük verinin neden bu kadar önemli olduğunu anlamamız gerekir. Büyük veri analizi bir çok işin iyileştirilmesinde, strateji geliştirilmesinde kullanılabilir. Büyük veri analizi sonrasında yapılacak değişikler sosyal hayatımızı doğrudan etkileybilir hatta kolaylaştırabilir. Çünkü yapılan analizler biz insanlara yöneliktir.

Bu projeyi yaparken aşağıdaki soruları sordum:

- Projeden büyük veri elde edebilir miyim?
- Büyük veriyi nasıl elde edebilirim?
- Elde ettiğim veri yararlı bir veri mi? Nasıl kullanılabilir?

Projenin aktif kullanılıyor olması, insanlara yönelik olması, internetin ve depolamanın işin içinde olmasından dolayı ilk soruya cevabım evetti. Projeden büyük veri elde edebiliirim.

Diğer sorum olan Nasıl elde edebiliirim? Sorusunu düşünürken ise aklıma tarih tutmak geldi.

Her aracın giriş çıkış saati elimizde olduğundan dolayı bu büyük veri bir şekilde yararlı kullanılabilir.

Büyük veriyi elde ettikten ve yararlı kullanılabilceğine karar verdikten sonra ki aşama veriyi analiz etmektir. Projemde ise analizlerim aşağıdaki gibidir:

- Otoparka kaç araç giriş/çıkış yaptı
- Araçlar ne kadar süre kalmak için otoparkları tercih ediyor
- Günlerin, mevsimlerin insanların otoparkları tercih etmesinde etkisi var mı
- Uygulama içinde yapılan otopark aramaları da depolanarak, en çok hangi bölgelerde yoğunluk yaşanıyor

Gibi analizler yapılabilir. Analizlerin sonuçları yorumlanarak hizmet politikaları değiştirilebilir, güncellemeler yapılabilir.

Projede Hadoop kullanılabilir. Haadop verilerin çok fazla olmasıyla birlikte gelen bilgiyi işleme sorunu ve verinin yedeklenip saklanması ihtiyacına yönelik çıkmıştır.