Derlenen Yol Kenarı Ünitesi, Araç CANBus ve Konum Verilerinin Android Tabanlı Cihazlarda Uyarı Mesajı Olarak Gösterimi

[1]Gülsüm Çiğdem Çavdaroğlu ve [2]Erdem Ergen
[1]KoçSistem, İstanbul, Türkiye, cigdem.cavdaroglu@kocsistem.com.tr
[2]KoçSistem, İstanbul, Türkiye, erdem.ergen@kocsistem.com.tr

Abstract

In this study, a graphical user interface which executes on a computer with android OS is developed for warning of vehicle drivers. Application consists of two main sections. First is frontend application which presents data of visual warnings. Second is backend application which receives data by connecting to the Obe 102 device. Application system is designed with client-server architecture. Connection between android based graphical user interface and Obe 102 device is provided on TCP/IP stack. IP address and port number is given to the android gui application at runtime. Android Gui application requests connection establishment with the provided information. Obe 102 device accepts connection request and then starts transmission of message data. Android gui application presents warnings corresponding to the transmitted data.

Özet

Bu çalışmada, araç sürücüleri için Android işletim sistemli bilgisayar üzerinde çalışacak bir önyüz uygulaması geliştirilmiştir. Uygulama iki ana bölümden oluşmaktadır. Birincisi, verilerin sunumunu gerçekleştiren görsel öğeleri içeren önyüz uygulamasıdır. İkincisi ise Obe 102 donanımına wifi üzerinden bağlanarak veri dinleme işlemini gerçekleştiren arkaplan uygulamasıdır. Uygulama için tasarlanan sistem, istemci – sunucu mantığı ile çalışmaktadır. Android GUI uygulama ile Obe 102 donanımı arasındaki bağlantı TCP/IP protokolü üzerinden sağlanmaktadır. Uygulamaya çalıştırılma sırasında Obe 102 donanımının IP adresi ve port numarası bilgileri verilir. Uygulama bu bilgileri kullanarak wifi internet erişimi üzerinden Obe 102 donanımına bağlantı kurma isteği iletir. Obe 102 donanımı gelen bağlantı talebini kabul eder ve verilerin iletimine başlar. Android uygulama bu verileri alarak ekranda görsel olarak sunumunu gerçekleştirir.

Anahtar kelimeler: CANBus, Gömülü Sistem, Araç Dinamikleri

1. Giriş

Bu çalışma Eureka [1] kümesi olan Celtic-Plus [2] programı kapsamında, 2012 – 2015 yılları arasında yürütülen CoMoSeF [3] projesinde oluşturulmuştur. CoMoSeF projesiyle Avrupa Birliği Akıllı Trafik Sistemleri aksyon planı ve Hedef 2023 Ulusal Aksiyon Planını destekleyen Akıllı Trafik Sistemlerine yönelik geleceğin hizmetlerinin yaratılması için ihtiyaç duyulacak uygulama ve donanımların geliştirilmesi adreslenmektedir. Bu kapsamda pazar ihtiyaçları çerçevesinde hizmet ve iletişim çözümlerini içeren iş modelleri yaratılacaktır. Projede, araçların diğer araçlar ile ve yol kenarı üniteleri ile haberleşmesi sağlanarak elde edilen verilerden sürüş kalitesini ve güvenliğini arttıracak ve trafik yönetimini destekleyecek bilgi sağlanması gerçekleştirilmektedir. Bu verilerin işlenmesi ile elde edilen bilgiler mobil uygulamalar ve servisler aracılığı ile son kullanıcı olan sürücüler, yayalar ve trafik otoritelerinin hizmetine sunulmaktadır. Projenin Türkiye Konsorsiyumu tarafında yürütülen çalışmalarda İstanbul'da ve Sakarya'da bir pilot proje gerçekleştirilmesi planlanmıştır. Projeye 9 ülkeden 23 iş ortağı katılmaktadır.

Corresponding author: Address: KoçSistem, İstanbul, Türkiye, cigdem.cavdaroglu@kocsistem.com.tr

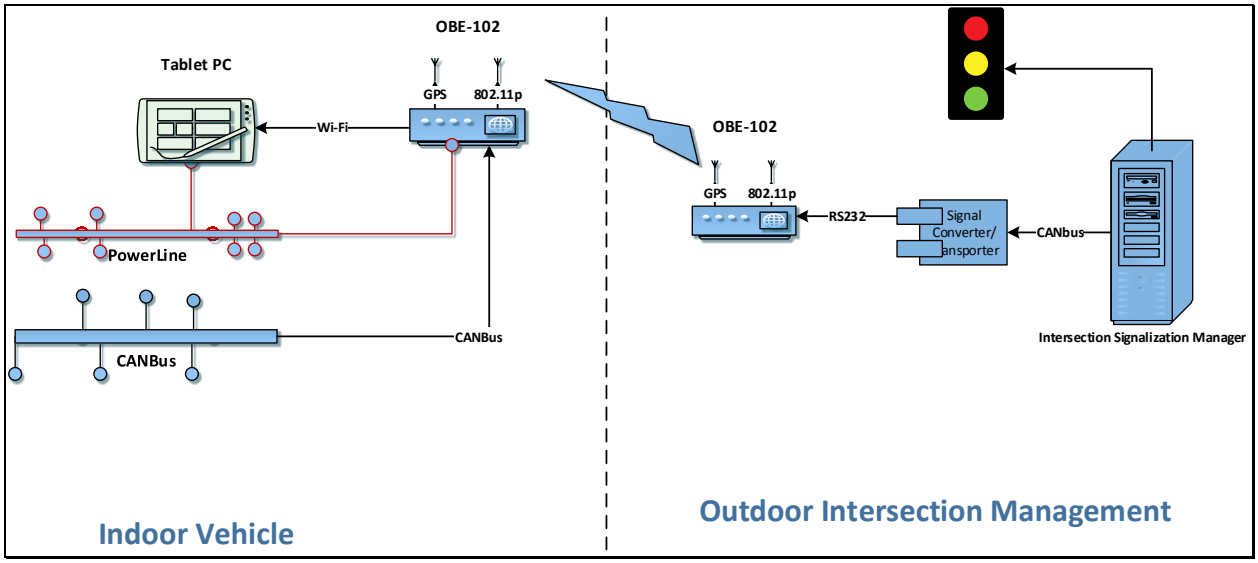
Proje çerçevesindeki kullanım senaryolarında; yol kenarı ünitesinden gelen verilerin araç içi donanımda işlenerek uyarı mesajlarının üretilmesi ve üretilen uyarı mesajlarının Android tabanlı ekranlar üzerinde gösterilmesiyle sürücünün bilgilendirilmesi amaçlanmıştır. Sunulan çalışmada ise istenilen sistemin oluşturulabilmesi için gerçekleştirilen görevler anlatılmıştır.

2. Yöntem

Çalışmada geliştirilen yöntem, "araç içi ünitede uyarı mesajlarının oluşturulması ve Android tabanlı cihazlara iletilmesi" ve "Android tabanlı mobil cihazlarda mesajların gösterilmesi" olmak üzere iki ana başlık altında ele alınmıştır.

2.1. Araç içi ünitede uyarı mesajlarının oluşturulması ve android tabanlı cihazlara iletilmesi

Sunulan çalışmada, araç içi ünitede iki ana birimin çalışma prensibi gösterilmiştir. Bunlardan birincisi "uyarı mesajlarının oluşturulması" birimidir. Bu birimde, yol altyapısından/yolkenarı ünitelerinden gelen bilgiler, yığın yapısında kritik bölüm mekanizmaları ile kullanıcı arayüzü için üretilmekte ve istemciler ile de tüketilmesi sağlanmaktadır. "Uyarı mesajlarının android tabanlı cihazlara iletilmesi" biriminde istemcilerin üretilen mesajları tüketebilmesi için istemci özel iletişim kanalı açılması sonrasında ise uyarı mesajı verilerinin yığın yapısından çekilerek bu kanaldan iletilmesi sağlanmaktadır. Şekil 1'de sistemin genel mimarisi gösterilmiştir.



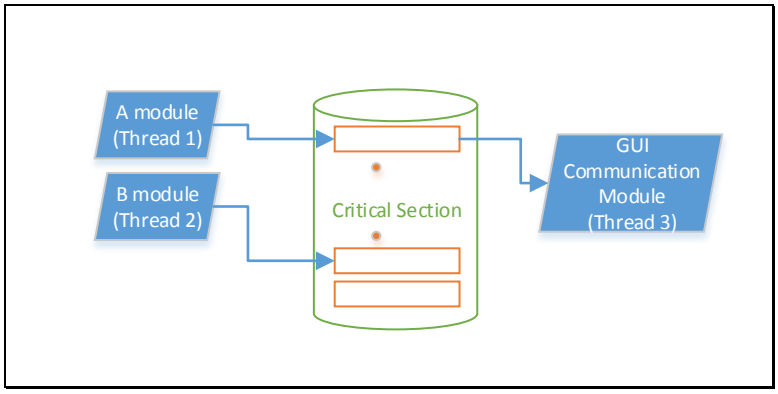
Şekil 1. *Sistem genel mimarisi*

2.1.1 Uyarı mesajlarının oluşturulması

Yazılım birimlerinin ürettikleri veriyi diğer yazılım birimleri ile paylaşabilmesi için üretilen verinin saklandığı ve kullanıldığı bir yapı oluşturulmuştur. Bu yapı "üretici-tüketici" tasarım deseni ile oluşturulmuştur. Üretilen veri yazılım birimine has veri yapılarında işletim sistemi "mutex"lerinden yararlanılarak belirlenmiş boyutlardaki yığınlarda saklanmaktadır. Saklama işlemi iki farklı biçimde yapılabilmektedir:

- Yığın dolana kadar ve dolduğunda tüketilene kadar yazılamayacak şekilde,
- En son üretilen verinin kuyruktaki son verinin üzerine yazılarak güncellenebilmesi ile.

Üretilen verinin alınması işlemi ise yine yığın mantığına göre en güncel verinin yığından çekilmesi ile yapılmaktadır.



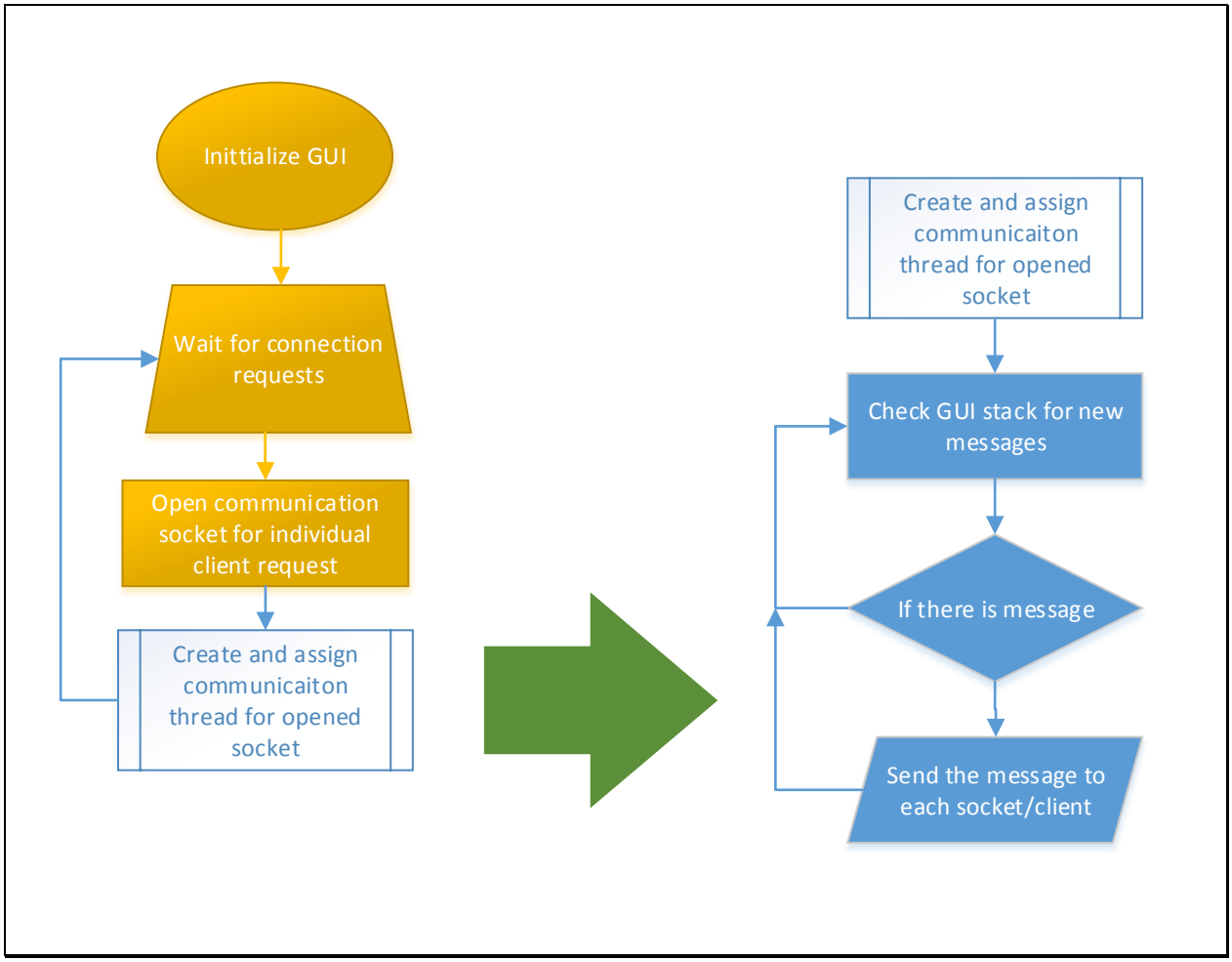
Şekil 2. Kanal (thread) ve üretim tüketim diyagramı

Yazılım birimlerinin her biri birer kanal (thread) olarak tasarlandığı için veriye ulaşım işlemlerinin her birinde "mutex" kullanılarak kritik bölümler oluşturulmuş ve veri okuma-yazma işlemleri bu bölümlerde yapılarak, veri yarış durumlarının önüne geçilmiştir. Yine yazılım birimlerinin kanal olarak oluşturulması sayesinde farklı veri üretim süreçleri paralel ve birbirinden bağımsız olarak gerçekleştirilmiştir.

2.1.2 Uyarı mesajlarının android tabanlı cihazlara iletilmesi

İlklendirilen iletişim birimi ile android tabanlı istemcilerden gelecek olan bağlantı istekleri beklenmeye başlanır. Bağlantı isteği geldiğinde kabul edilir ve bu işlem sonucunda o bağlantıya özel yeni bir soket oluşturulur. Sonraki süreçte bağlantıya ilişkin işleri koşacak olan fonksiyon ile yeni bir iş parçacığı oluşturulur. Oluşturulan iş parçacığına başlatılması sırasında parametre olarak, yine yeni oluşturulmuş soket veilir. Böylece bağlantı işlemlerini koşacak iş parçacığı istek yapan istemci özelinde çalışmaya başlar. Yığın veri yapısında her mesaj için en fazla olabilecek iş parçacığı sayısı kadar alan vardır. Bu alanlar iş parçacıkları tarafından mesaj çekileceği zaman iş parçacığı kimliği ile damgalanarak; hem mesajların tüm iş parçacıkları tarafından okunduğunda yığından silinmesini hem de aynı iş parçacığı tarafından birden fazla kez çekilmesini önler. Böylece farklı istemciler tek bir yığından aynı mesajı ayrı ayrı alabileceklerdir.

Sonraki adımda iş parçacıkları yığından çekilmiş mesajları kendi istemcilerine ait soketlere yazarak o mesaj için gönderme işlemini tamamlamış olacaklardır. Iş parçacıkları yığından tekrar mesaj çekerek süreci bu döngü içerisinde devam ettireceklerdir.



Şekil 3. Her bir istemci için iletişim kanalı oluşturulması ve veri iletimi

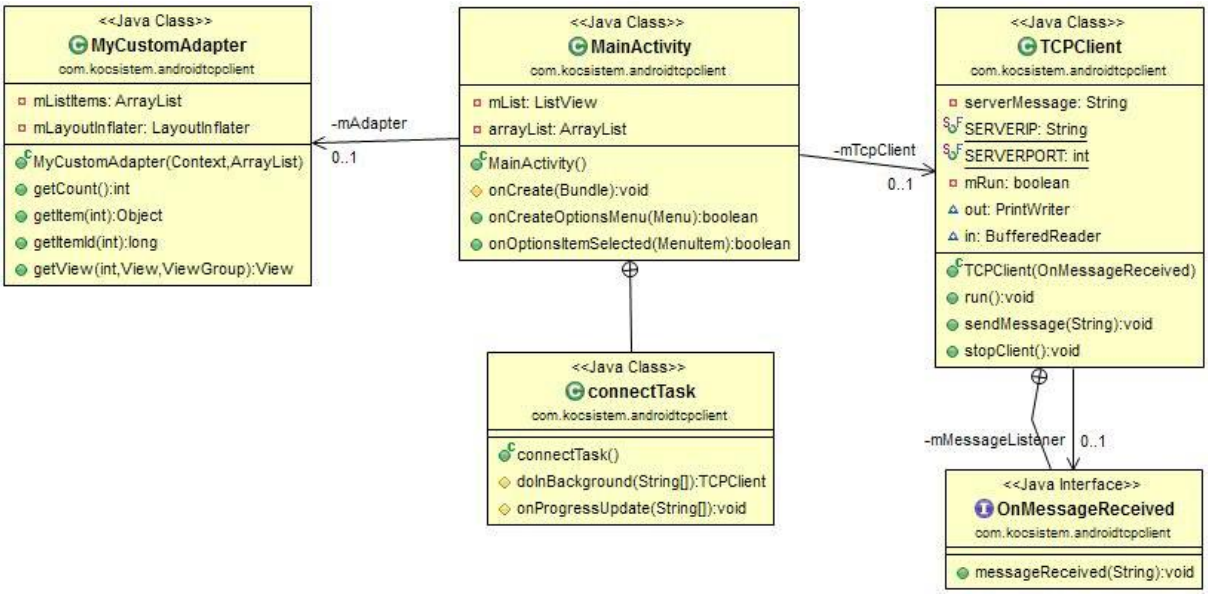
2.2. Android tabanlı mobil cihazlarda mesajların gösterilmesi

Araç içi donanımdan elde edilen bilgilerin sürücülere Android tabanlı cihazlar üzerinden sunumunun sağlanabilmesi için Android işletim sistemi üzerinde iki farklı uygulama geliştirilmiştir:

- GUI Uygulama: Verilerin sunumunu gerçekleştiren görsel öğeleri içermektedir.
- Yönetim Uygulaması: Obe 102 donanımına wi-fi üzerinden bağlanarak veri dinleme işlemini gerçekleştirmektedir.

Geliştirilen sistem, istemci – sunucu mantığı ile çalışmaktadır. Android GUI uygulama ile araç içi donanımı arasındaki bağlantı TCP/IP protokolü üzerinden sağlanmıştır. Uygulamaya, çalıştırılma sırasında araç içi donanımının IP adresi ve port numarası bilgileri verilmektedir. Uygulama bu bilgileri kullanarak wi-fi internet erişimi üzerinden araç içi donanımına bağlantı kurma isteği iletmektedir. Araç içi donanım, gelen bağlantı talebini kabul edip verilerin iletimine başlamaktadır. Android uygulama bu verileri alarak ekranda görsel olarak sunumunu gerçekleştirmektedir. Uygulama Window 7 – 64 Bit işletim sisteminde Eclipse editörü kullanılarak Java ve Android programlama dilinde geliştirilmiştir. Android sürümü 4.4.2, API

seviyesi 19'dur. Yönetim uygulaması, üç sınıftan oluşmaktadır. Bu sınıflar bağlantıyı gerçekleştirip sunucu tarafından iletilen verileri alarak saklanma işlemlerini gerçekleştirmektedirler. Yönetim uygulamasının sınıf diyagramı şekil 1'de görülmektedir.



Şekil 4. Yönetim uygulaması sınıf diyagramı

Android kullanıcı grafik ara birimi, tek kanallı model (single threaded model) üzerine kuruludur. Sistem, Android uygulaması çalışmaya başladığında ana kanal üzerinde (main thread, UI thread) çalışmaya başlamaktadır. Grafik ara birimi bileşenleri ve bu bileşenlere ait olayların ele alınması bu kanal üzerinde gerçekleştirilmektedir. Ağ (network) erişimi, veritabanı bağlantıları ve veri çekme, bellek okuma gibi uzun zaman alan işlemler tek bir kanal içerisinde gerçekleştirildiğinde, tek kanallı model yapısı performans sorunlarına neden olabilmektedir. Kanal, uzun süren işlemlerden cevap dönülmesini bekler ve bu süreçte hiçbir grafik ara birimi olayını işleyemez. Bunun sonucu olarak kullanıcılar ekranda yer alan hiçbir eleman ile bağlantı kuramaz ve uygulama donmuş izlenimi oluşur. Arka planda devam eden işlemler yaklaşık olarak 5 saniyeden daha uzun sürdüğünde, Android işletim sistemi tarafından kullanıcıya otomatik olarak ANR mesajı (Application Not Responding) sunulmaktadır. Android geliştirme ortamında bu tür uzun zaman alıcı işlemlerin derlenmesi de derleyici tarafından yasaklanmıştır. Android grafik ara birimi kanal – güvenli (thread-safe) değildir, bu nedenle yapılan işlemlere dikkat edilmesi gerekmektedir. Kanal – güvenli olmayan uygulamalarda, farklı kanallar birbirlerinin alanlarına erişebilmekte ve birbirlerinin işleyişlerini bozabilmektedirler. Android işletim sisteminde grafik ara birimi, kanal – güvenli olmadığı için grafik ara birimi ile yapılacak olan etkileşimlerin sadece ana kanal üzerinden yapılması sağlanmıştır. Diğer tüm kanalların grafik ara birime erişimleri engellenmiştir.

Android işletim sisteminde asenkron fonksiyon çalıştırmayı sağlamak için *AsyncTask* ve *Thread&Handler* yöntemleri olmak üzere iki farklı yöntem kullanılmaktadır. Bu çalışmada *AsyncTask* yöntemi tercih edilmiştir. Obe 102 donanımına bağlanma ve veri çekme işlemleri, uygulama başlatıldıktan sonra farklı bir kanal içerisinde gerçekleştirilmektedir. *AsyncTask* sınıfı içindeki metodlar yardımıyla arka planda farklı bir akış üzerinde gerçekleştirilen işlem, ön yüzü

etkileyen ana akışa entegre edilmiştir. Arka planda Java Thread sınıfıyla çalıştırılan iş parçacıkları, ön yüz elemanlarına müdahale edememektedir. Bunun için *Handler* adında bir sınıf kullanılmıştır. *AsyncTask* sınıfı soyut (abstract) bir sınıftır, başka bir sınıf üzerinden genişletilerek (extend) kullanılmaktadır. *AsyncTask* sınıfında bulunan doInBackground metodunda arka planda gerçekleştirilecek olan işlemler yer almaktadır. Bu sınıf içerisinde yer alan diğer metotlar şunlardır:

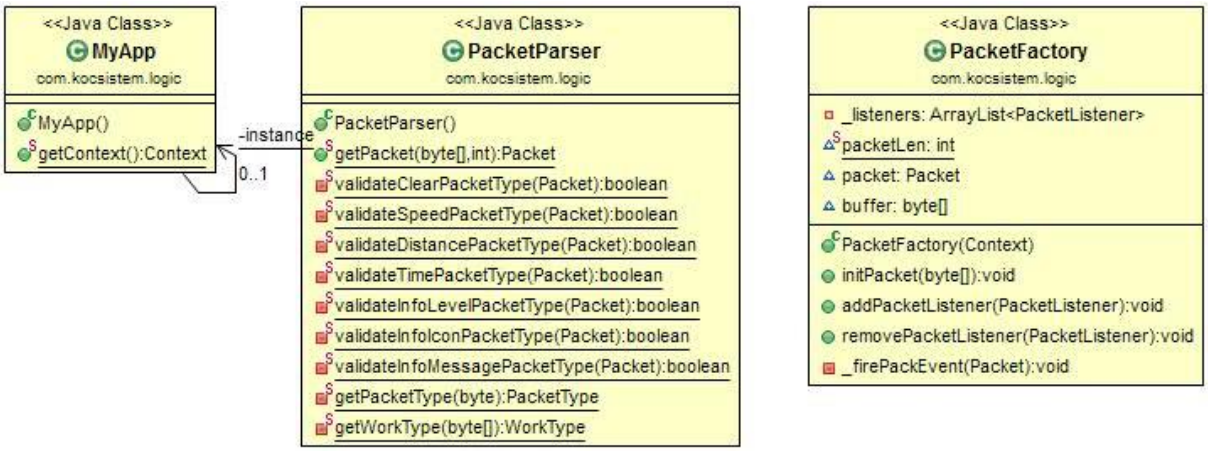
- onPreExecute: Ana işlem başlamadan önce ön yüzde yapılması gereken değişiklikler burada gerçekleştirilmektedir.
- doInBackground: Arka planda yapılması istenen işlemler burada yer almaktadır. Bu metot içerisinde yapılan işlemler ön yüzün donmasını engellemektedir.
- onPostExecute: doInBackground metodu tamamlandıktan sonra işlemlerin sonucu bu metoda result değişkeni ile gönderilmektedir. Buradaki işlemler ana akışı herhangi bir hataya sebep olmadan etkileyebilmektedir.
- onProgressUpdate: doInBackground metodu içerisinde yapılan işlemlerin ilerleme durumlarının kullanıcıya bildirilmesi gerektiğinde kullanılmaktadır.
- onCancelled: AsyncTask herhangi bir sebepten dolayı iptal edilirse bu metot tetiklenmektedir. Bu noktada kullanılan kaynaklar temizlenmektedir.

Araç içi donanımına asenkron metot içerisinde bağlantı kurmayı sağlayan kod bloğundan bir kesim şekil 2'de gösterilmiştir.

```
public class connectTask extends AsyncTask<String,String,TCPClient> {

    @Override
    protected TCPClient doInBackground(String... message) {

        //we create a TCPClient object and
        mTcpClient = new TCPClient(new TCPClient.OnMessageReceived() {
            @Override
            //here the messageReceived method is implemented
            public void messageReceived(String message) {
                //this method calls the onProgressUpdate
                publishProgress(message);
            }
        });

        mTcpClient.run();

        return null;
    }
```

Şekil 5. Obe 102 donanımına bağlanma

Yönetim uygulamasında yer alan sınıflara ilişkin sınıf diyagramı aşağıdaki şekil 3'te gösterilmiştir.



Şekil 6. Yönetim uygulaması sınıfları

MainActivity uygulamanın giriş noktası ve tüm işlemlerin yönetildiği sınıftır. PacketFactory sınıfı ile araç içi donanımına erişim, veri bekleme, veri alma işlemleri gerçekleştirilmektedir. PacketParser sınıfı ile alınan veri paketlerinin kontrolleri, gerekli veri bölme işlemleri gerçekleştirilmekte ve gelen veriye uygun olacak şekilde sonuçlar ana sınıfa iletilerek grafik ara birimde gösterilmesi sağlanmaktadır.

3. Sonuçlar

Geliştirilen uygulama, proje kapsamında oluşturulan senaryonun uygulanması ile gerçek trafik ortamında test edilmiştir. Testler Kağıthane İstanbul'da gerçekleştirilmiştir. Şekil 7'de test senaryosunun pilot bölgesi gösterilmiştir.

Şekil 7. Test senaryosu pilot bölgesi

Test yolu üzerinde bulunan trafik ışığı ile ilgili bilgi, İSBAK kampüsü yanında yer alan yol kenarı ünitesinden alınmaktadır. Obe 102 donanımı üzerinde koşan yazılım, yol kenarı ünitesi ile haberleşerek trafik ışığının durumu hakkında bilgiyi almaktadır. Aracın hız bilgisini de canbus'tan alarak aracın kırmızı ışığa yakalanmadan geçebilmesi için en uygun hangi hızda gitmesi gerektiğini hesaplamaktadır. Araç haritada gösterilen konumda iken bu bilgileri alarak ilgili hesaplamayı gerçekleştirir ve aracın gitmesi gereken uygun hızı hesaplar. Eğer aracın o anki hızı, uygun hız civarında ise sürücüye herhangi bir uyarıda bulunulmaz. Ancak araç daha düşük bir hızda ya da daha yüksek bir hızda seyrediyorsa uygun olan hız sürücüye uyarı mesajı olarak bildirilmektedir. Aşağıdaki şekilde test senaryosuna ilişkin görüntüler gösterilmiştir.

Şekil 8. Test anı görüntüleri

Yapılan test sonucunda, geliştirilen uygulamanın doğru zamanda doğru hız bilgisini verebildiği sonucu görülmüştür.

4. Tartışma

Bu çalışmada istemci ve mesaj önceliği için herhangi bir çalışma yapılmamıştır. Çoklu istemci bağlantılarında sürücüye ait istemcinin daha öncelikli olması işlem yoğunluğuna göre bir gereksinim oluşturabilir.

Ayrıca görece önemli mesajların daha önce yığından çekilmesini ve iletilmesini sağlayacak bir sistem de sürücü uyarılarının acil durumlarda daha verimli çalışmasını sağlayabilecektir.

Testlerin gerçekleştirildiği bölge trafik yoğunluğu az olan bir bölgedir. Trafik yoğunluğunun yüksek olduğu bölgelerde sürücünün uygun hız açısından bilgilendirilmesi öncesinde önerilecek olan hızın o anki trafik yoğunluğuna uygun olup olmadığı bilgisinin de değerlendirilmesi yoluyla hesaplanması sistemi daha akıllı ve gerçekçi hale getirebilecektir.

Çıkarımlar

Sunulan çalışmada kullanım senaryosunun geliştirilen sistem ile seçilen pilot bölgede başarılı bir şekilde gerçekkelebildiği görülmüştür. Sistemde kullanılan parametrelere trafik ve araç ile ilgili anlık verilerin de eklenmesi ile sistemin gerçek trafik ortamında daha akıllı davranabileceği öngörülmektedir.

Teşekkür

Çalışmanın test süreci, İSBAK ve Otokar firmaları ile birlikte gerçekleştirilmiştir. İSBAK ve Otokar firmalarına işbirlikleri için teşekkürlerimizi sunarız. Geliştirme sürecindeki işbirlikleri ve çalışmalarından ötürü tüm CoMoSeF proje ortanlarına da teşekkürlerimizi sunarız.

Referanslar

[1] http://www.eureka.org.tr/
[2] https://www.celticplus.eu/
[3] http://comosef.com/