**KARADENİZ TEKNİK ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**HARİTA MÜHENDİSLİĞİ ANABİLİM DALI**

# LOKASYON TABANLI KASİS UYARICI UYGULAMA

**YÜKSEK LİSANS TEZİ**

**Harita Müh. Berna ERGİNÖZ**

**EKİM 2020**
**TRABZON**

**KARADENİZ TEKNİK ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**HARİTA MÜHENDİSLİĞİ ANABİLİM DALI**

# LOKASYON TABANLI KASİS UYARICI UYGULAMA

**Berna ERGİNÖZ**
**ORCID : 0000 - 0002 - 0518 - 7607**

**Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce**
**"HARİTA YÜKSEK MÜHENDİSİ"**
**Unvanı Verilmesi İçin Kabul Edilen Tezdir.**

**Tezin Enstitüye Verildiği Tarih : 09 / 09 /2020**
**Tezin Savunma Tarihi : 14 / 10 /2020**

**Tez Danışmanı : Prof.Dr. Oğuz GÜNGÖR**
**ORCID : 0000 - 0002 - 3280 - 5466**

**Trabzon 2020**

# ÖNSÖZ

"Lokasyon Tabanlı Kasis Uyarıcı Uygulama" başlıklı çalışma, Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Harita Mühendisliği Anabilim Dalı'nda yüksek lisans tezi olarak hazırlanmıştır.

Tez süresi boyunca danışmanlığımı üstlenerek çalışmaların yürütülmesi sırasında araştırmalarımı yönlendiren ve ilgisini esirgemeyen çok değerli saygıdeğer hocam Prof. Dr. Oğuz GÜNGÖR' e en içten teşekkürlerimi sunarım.

Yazılım konusundaki tecrübesine dayanarak fikrine danıştığım Arş. Gör. Alper Tunga AKIN' a ve arkadaşım Emre BİRİNCİ' ye teşekkürlerimi sunarım.

Son olarak tez dönemi boyunca yardımlarını benden esirgemeyen ve hayatımın her anında destekleriyle yanımda olan sevgili nişanlım Hasan SAĞLAM' a ve bugünlere gelmemde büyük katkısı olan, maddi ve manevi desteklerini her zaman hissettiğim canım aileme en içten teşekkürlerimi bir borç bilirim.

Berna ERGİNÖZ

Trabzon 2020

## TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum " LOKASYON TABANLI KASİS UYARICI UYGULAMA" başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Oğuz GÜNGÖR 'ün sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 14/ 10/ 2020

Berna ERGİNÖZ

# İÇİNDEKİLER

**Sayfa No**

Yüksek Lisans Tezi

# ÖZET

## LOKASYON TABANLI KASİS UYARICI UYGULAMA

Berna ERGİNÖZ

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Harita Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Oğuz GÜNGÖR
2020, 65 Sayfa

Trafiği sakinleştirmede kullanılan kasislerin düşük görüş koşullarında göze çarpmama durumları meydana gelmektedir. Bu durumda araçlar yüksek bir hızla kasise yaklaştığında kaza veya yaralanmalar meydana gelmektedir. Bu tez çalışmasında sürücülere kasislere yaklaşmadan önce uyarı veren lokasyon tabanlı mobil bir uygulama yapılması sağlanmıştır. Yazılan mobil uygulama Java programlama dilinde, Android Studio uygulama geliştirme programı kullanılarak oluşturulmuştur. 5 farklı örneklem bölge üzerinde belirlenen kasislerin fotoğrafları çekilip, konumları hakkında bilgi toplanmıştır. Böyle bir ön bilgi edinildikten sonra kasislerin koordinatları, mobil uygulamada kullanılan Google Maps üzerine eklenip, yerleri marker ile belirtilmiş olup, marker içerisine de bilgi penceresi ile kasis uyarı levhası işareti eklenmiştir. Kullanıcının anlık konumunun alınması işlemi de tanımlanmıştır. Daha sonra kullanıcının konumu kasislere yaklaştığı zaman uygulamadan sesli uyarı çıkması sağlanmıştır. Böylece bu uygulama ile sürücülerin kasisler bulunan yollarda araçlarını güvenli bir şekilde kullanmaları sağlanmış oldu.

**Anahtar Kelimeler:** Kasis, Java Programlama Dili, Android Studio, Google Maps

Master Thesis

SUMMARY

LOCATION BASED SPEED BREAKER APPLICATION

Berna ERGİNÖZ

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Geomatics Engineering Graduate Programme
Supervisor: Assoc. Prof. Dr. Oğuz GÜNGÖR
2020, 65 Pages

Inconspicuous situations occur in the low visibility conditions of the speed breakers used to calm the traffic. In this case, accidents or injuries occur when vehicles approach the speed breaker at a high speed. In this thesis, a location based mobile application is given that gives drivers a warning before approaching the speed breakers. The written mobile application was created in the Java programming language using the Android Studio application development program. Photographs of speed breakers determined on 5 different sample regions were taken and information about the positions of speed breakers were collected. After obtaining such a preliminary information, the coordinates of the speed breakers were added to the Google Maps used in the mobile application and their locations were specified with a marker, and the speed breaker warning sign was added to the marker through the information window. The process of taking the instant location of the user is also defined. Then, when the user's location approached the speed breakers, an audio warning came from the application. Thus, with this application, the drivers were provided to use their vehicles safely on the roads that are speed breakers.

**Key Words**: Speed Breaker, Java Programming Language, Android Studio, Google Maps

## ŞEKİLLER DİZİNİ

**Sayfa No**

## TABLOLAR DİZİNİ

**Sayfa No**

## SEMBOLLER DİZİNİ

| | |
|---|---|
| UTM | : Universal Transversal Mercator |
| İBB | : İstanbul Büyükşehir Belediyesi |
| PDA | : Personal Data Asistant (Kişisel Veri Sağlayıcısı) |
| IBM | : International Business Machines (Uluslararası İş Makineleri) |
| IDC | : Uluslararası Veri Şirketi |
| API | : Android Application Progrmaming Interface |
| DVM | : Dalvik Virtual Machine |
| NFC | : Near Field Communication |
| UI | : Kullanıcı ara yüzü |
| IDE | : Entegre Geliştirme Ortamı |
| JDK | : Java Development Kit |
| SDK | : Software Development Kit |

# 1. GENEL BİLGİLER

## 1.1. Giriş

İnsanlar ihtiyaçları doğrultusunda mevcut hizmetleri geliştirilip iyileştirirken, fiziksel bir ürün, yazılım veya teknik süreç gibi birtakım bilgilerden yararlanmaktadır. Bu bilgiler bütünü teknolojinin temelini oluşturmaktadır. Geliştirilen hizmetler sayesinde insanlar dünyada ulaşmakta zorluk çektikleri yerlere gitme imkânı bulmuşlardır. Böylece insanlar dünyada farklı kültürlerle etkileşime geçip, kendi benliklerini farklı açıdan gözlemleyebilme şansını yakalamışlardır. Bu durum da her geçen gün yeni bir teknolojik gelişmenin ortaya çıkmasına neden olmuştur.

Teknolojik gelişme denilince ilk düşünülen bilişim ve iletişim teknolojisidir. Bu gelişme ilk olarak radyo ile hayatımıza girmiş ve ardından televizyona kadar uzanmıştır. 70'li yıllarda bilgisayar teknolojisi çok fazla ön planda olurken, 80'li yıllardan itibaren cep telefonları ortaya çıkmıştır. 90'lı yıllarda ise internetin yayılmasıyla ülkeler arasındaki sınırlar ortadan kalkmış ve artık insanlar farklı coğrafyalardan insanlarla iletişim kurup etkileşim içinde olmaya başlamışlardır. İnternet karşılıklı bilgi akışının yanı sıra insanlığın kendini dilediği yönde geliştirebilmesine de çok büyük bir katkı sağlamıştır. İnternetin gelişip cep telefonlarına girmesiyle ise akıllı telefon dönemi başlamıştır. Böylece sadece insanlarla iletişim kurmada kullanılan cep telefonları akıllı telefon olma özelliğini kazanmıştır (URL-1).

Haritacılık alanı teknolojik gelişmelere açık, çağdaş teknolojiyi en iyi uygulayan ve çağdaş bilimin gelişmesini sağlayan en önemli kaynaklardan biridir (Saraç, 2019). Bilgisayar teknolojilerinin sıklıkla kullanıldığı haritacılıkta, bilgisayar teknolojileri ile verinin üretimi, işlenmesi, analizi ve aktarımı yapılmaktadır. Bunun yanı sıra gerekli ihtiyaçlar doğrultusunda veri üretmede de bilgisayar teknolojisinden yararlanılmaktadır. Böylece yapılacak olan tüm çalışmalar çok daha hızlı ve yüksek doğrulukta bir sonuç vermektedir.

Günümüzde mobil cihazlardaki (cep telefonları ve tabletler) uygulama marketlerinde sıklıkla coğrafi konum bilgilerini kullanarak yer tespiti sağlayan lokasyon tabanlı uygulamalar yer almaktadır. Bu tarz uygulamalarda harita kullanılmasıyla, yeryüzünün tamamında veya bir bölümünde yer alan fiziksel detayların, bu detaylarla ilgili bilgilerin veya bu alanda meydana gelen olguların gösterimi sağlanırken, anlık konum alınmasıyla da

harita üzerinde bireyin bulunduğu yeri görmesi ve bu yer hakkında anlık bilgi elde edinebilmesi sağlanmaktadır. Bütün bunlar ışığında bireyin zamanını doğru ve etkili bir şekilde kullanarak yaşamını kolaylaştırması sağlanmış olur.

Gelişmekte olan ülkelerde yayalara, bisiklet ve motorlu taşıt kullanıcılarına şehir içinde gerekli kullanım alanı bırakılmamaktadır. Bu hem şehrin yaşam kalitesinin düşmesine hem de şehirlerde trafik sıkıntısının meydana gelmesine neden olmaktadır. Taşıt hızının yüksekte seyretmesi, fazla taşıt kullanımının olması gibi sorunları kontrol altına almak için birtakım önlemlerin alınması gerekmektedir (Kaygısız, 2010). Trafiği sakinleştirmeye yönelik bu önlemler içerisinde trafik hızının kontrolünü sağlamak için kullanılan yöntemlerden biri kasislerdir ( hız kesici tümsekler).

Kasisler, karayolu üzerinde bulunan yükseltilere denilmektedir. Genellikle kasisler, motorlu taşıt kullanıcılarının hızlarını arttırmayı uygun gördükleri yerlerde tercih edilmektedir. Bu yerler genellikle yaya trafiğinin yoğun olduğu ve böylece kazaların muhtemel olacağı okul önleri, parklar, bahçeler, hastaneler ve konut alanlarıdır. Ülkemizde de yol ağları üzerinde sıklıkla kasislerden yararlanılmaktadır (Kaygısız, 2010).

### 1.2. Literatür Çalışması

Saraç (2019) yaptığı çalışmada mobil telefonlar için Google Maps ve mobil GNSS kullanılarak bir uygulama geliştirmiştir. Bu uygulama Google Maps üzerinden koordinat verisi kullanılarak alınan koordinatların UTM ( Universal Tranversal Mercator) koordinat sistemine dönüştürülerek kaydedilmesini, istenilen UTM koordinatlarının harita üzerinde belirtilmesini, gerekli durumlarda dışarı aktarılmasını sağlamıştır. Uygulamada kullanıcının anlık konumunu almak içinde mobil telefonlardaki GNSS alıcısından yararlanılmıştır. Böylece alınan konumun istenilen koordinat sistemine dönüştürülmesi sağlanmıştır. Tez çalışmasında yazılan uygulamada Android Studio programı kullanılarak Java dilinde kodlama yapılmıştır. Program içerisinde lokasyonla ilgili kütüphanelerin olmaması harita servis sağlayıcısı verileri ve GNSS verileri için ek matematiksel işlemlerin kullanılmasına yol açmıştır. Bu durumun programda yoğun bilgi karışıklığına sebep olduğu görülmüş, Android Studio'nun eksik bir özelliği olarak nitelendirilmiştir. Ayrıca mobil cihazlarda bulunan GNSS alıcılarının verdiği sonuçların bazı haritacılık alanını ilgilendiren uygulamalar için yetersiz kaldığı belirtilmiştir. Bunun sebebini de bazı telefonların çift frekansa sahip olmaması veya çift frekansa sahip olsa bile her telefonun farklı frekans ve

veri alma özelliğine sahip olmasından kaynaklandığı şeklinde açıklamıştır. Sonuç olarak bu durumun konum değerlerini etkilediğini ortaya koymuştur.

Özcan (2013) yaptığı çalışmasında, toplu taşıma araçlarına ve sistemine ait verileri içeren Android bir mobil uygulama tasarlamıştır. Bu veriler için İstanbul şehrini seçip, uygulamasının ismine "Transistanbul" adını vermiştir. Uygulama ile İstanbul'da ikamet eden insanların ulaşım araçları hakkında doğru ve güvenilir bir şekilde bilgi almaları sağlanmıştır. Böylece tek bir uygulama ile ulaşım araçları hakkında bilgi edinilebilmektedir. Uygulama geliştirmek için Android' in resmi sitesinde yer alan tasarım özelliklerine göre Android Lint aracı ile uygulamanın denetlenmesi yapılıp uygulama tasarlanmıştır. Hem telefon hem de tablet kullanıcıları için tasarımlar yapılmıştır. Dil seçeneği konularak yurt dışından İstanbul'a gelecek olan vatandaşlar için de kolaylık olması sağlanmıştır. Ayrıca uygulamada meydana gelebilecek olası bir hatayı engellemek için hata rapor edilebilecek bir kısım eklenmiştir. Bu uygulama Google Play Store üzerinden yayınlanması için İstanbul Büyükşehir Belediyesi (İBB)'ne verilmiştir. Bu uygulamanın bundan sonraki yapılacak toplu ulaşımla ilgili bütün uygulamalar için bir model olunması hedeflenmiştir.

Kh'tour (2015) çalışmasında Android tabanlı hastaneyle ilgili randevu hizmeti, sıra sorgulama hizmeti, dosya indirme hizmeti ve dosya yükleme hizmeti gibi verileri barındıran mobil bir uygulama yapmıştır. Bu uygulamayı geliştirmek için Java dilini, veri yönetimi için Mysql, veri tabanı yönetimi ve Android uygulamalar arasında haberleşmeyi sağlamak içinde php kullanmıştır. Veri tabanı sürücüsünü kullanırken ücretsiz platformları tercih etmiştir. MySQL ile tasarlanan veri tabanı için gerekli testler yapılmıştır. Uygulama pek çok mobil cihaz için uyumlu hale getirilebilecek nitelikte tasarlanmıştır. İlerleyen aşamalarda uygulamaya gerekli eklemelerle ilaç, eczane, doktor, sigorta vb. hizmetlerin eklenmesi düşünülmektedir.

Kılıç ve Tuncer (2017) çalışmalarında, bir şehrin farklı yerlerinde bulunan otoparklarla ilgili sürücülere bilgi veren Android tabanlı mobil bir uygulama geliştirmişlerdir. Ayrıca otoparkların girişleri için sensörler tasarlayarak otoparklar hakkında bilgi alınması hedeflenmiştir. Bu bilgilerin veri tabanında tutulmasını amaçlamışlardır. Veri tabanı olarak MySQL kullanılmıştır. Oluşturulan Android tabanlı basit ara yüze sahip mobil uygulama ile telefondan otoparkların doluluk oranları görülebilmektedir. Bu sistem akıllı şehirlerle ilgili yapılabilecek her uygulama için kullanılabilmektedir.

### 1.3. Çalışmanın Tanımı

Kentlerin gelişimiyle birlikte, motorlu taşıtların trafikte artışı, kentsel yaşanılabilirliği bazı yönlerden etkilemekte ve kentsel yaşam kalitesi açısından sorunlara neden olmaktadır. Bu tür ulaşım sorunlarının çözümünde trafiği sakinleştirmeye yönelik yöntemlerle motorlu taşıt trafiğinin yarattığı olumsuz etkileri azaltmaya ve kentsel yaşam kalitesini yükseltmeye, dair çalışmalar yapılmaktadır. Türkiye'de bu çalışmalardan en sıklıkla kullanılanı kasislerdir. Ancak kasislerin çoğu zaman tekniğine uygun yapılmamış olması, yol üzerinde belirgin olmaması ve kasislerden önce sürücüyü uyaracak levhaların konulmamış olması seyir halindeki sürücü için ciddi sorunlara neden olmaktadır. Bu nedenle bu tez çalışmasında 5 örneklem bölge üzerinde yer alan kasislerin yerleri tespit edilmiştir. Daha sonra Android Studio programında Java dili kullanılarak araçların kasislere yaklaşmadan önce uyarı vermesini sağlayacak lokasyon tabanlı mobil bir uygulama yazılmıştır.

### 1.4. Çalışmanın Hedefleri

Bu tez çalışmasıyla kullanıcılar mobil uygulamayı cep telefonlarında açtıkları zaman harita üzerinde hem yol güzergâhında bulunan kasislerin konumlarını hem de anlık konum alınması sayesinde kendi konumlarını görebileceklerdir. Aynı zamanda sürücüler seyir halindeyken kasislere yaklaşıklarında gelen uyarı sesiyle kasisin varlığından haberdar olup hızlarını buna göre ayarlamış olacaklardır. Bu sayede bir aracın, eşik hızından daha yüksek bir hızda kasise yaklaşmasını engellenerek kaza veya yaralanma riskinin azaltılması sağlanacaktır. Ambulans ve itfaiye gibi acil durum müdahale araçlarının seyrini zorlaştıran kasisler içinde bu uygulama faydalı olacaktır. Özellikle ambulans içindeki yaralılara ciddi zararlar verebilen bu durumun önüne geçilmiş olunacaktır. Ayrıca geceleyin veya sisli, yağmurlu ve karlı havalar gibi düşük görüş koşullarında göze çarpmayan kasislerin de gelen uyarı sesiyle varlığı bilinip ona göre tedbir alınması sağlanmış olacaktır.

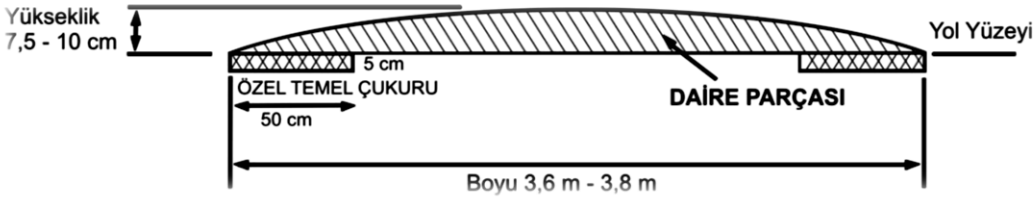
### 1.5. Kasisler (Hız Kesici Tümsekler)

Kasisler, karayolu üzerinde belli bir yüksekliğe ve genişliğe sahip olunan tümseklerdir. Kasisler, genellikle trafiği sakinleştirip taşıtların hızlarını belirli seviyede tutulmasına yardımcı olmaktadır. Yol güzergâhındaki kasislerin üzerinden taşıtlar geçtiği

zaman kasislerin sahip olduğu yükseltiler taşıtta sarsıntılara neden olmaktadır. Taşıt sürücüleri bu sarsıntıları en aza indirmek için kasislere yaklaşınca mecbur hızlarını düşürme gereksinimi duyacaklardır (Gürer, 2006). Standartlara uygun olarak üretilmeyen kasisler, hem taşıt sürücüsün hem de yayaların güvenliğini tehlikeye atarak konforsuz ve kalitesiz kent içi ulaşıma neden olmakta ve ayrıca trafik kazalarına yol açmaktadır.

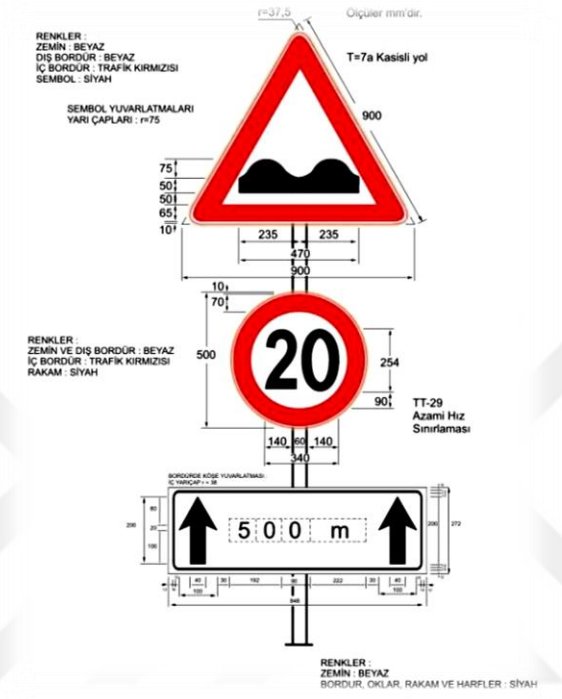
Ulaştırma Bakanlığı, Karayolları Genel Müdürlüğü' nün 24.12.2007 tarihli 2007/115 sayılı genelgesi ile şehir içi yollarda tesis edilecek kasislerin, konu ile ilgili "Yol Sathı Hız Kontrol Elemanları – Tümsekler (Kasisler)" başlıklı TS 6283 standardında belirtilen esaslara uygun bir şekilde yol güzergâhlarına tesis edilmesi gerekmektedir. Yine bu genelgede trafik güvenliği açısından hız kesici set yapılması uygun değildir ifadesi yer almaktadır. Kasislerle ilgili olarak Türk Startları Enstitüsünün (TSE) 1988 yılında kabul etmiş olduğu "Yol Sathı Hız Kontrol Elemanları Tümsekler TS 6283 standardına göre kasisler;

- ✓ 50m – 150 m ara ile birden çok sayıda ve yolun en fazla 800 m' lik kesiminde uygulanabilir.
- ✓ Birinci derece yollarda uygulanamaz.
- ✓ Aracın hızını ayarlayabilmesi için yoldan girişten itibaren ilk tümsek en az 20. metrede olmalıdır.
- ✓ Yapılacağı yolun kaplama maddesi ile aynı cinsten olmalı, taşıt yolunu enine kat etmeli, asfalt yollarda yapılacak ise mastik asfalt kullanılmalıdır.
- ✓ Sürücüler tarafından kolayca görülebilecek şekilde beyaza boyanmalıdır.
- ✓ Uygulamanın yapıldığı çift yönlü yolun başlangıç ve bitişinde, tek yönlü yolun başlangıcındaki, yol/yollara söz konusu tümseklerden en az 50 metre önce, tümseklerle yolda hız tahdidi olduğu ve bu uygulamanın uzunluğunu belirtir uyarıcı trafik işaret ve bilgi levhaları konulmalıdır.
- ✓ Uzunluğu 3.60 ile 3.80 metre arasında, yüksekliği ise 7,5 ile 10 santimetre arasında değişmelidir (Şekil 1).




Şekil 1. TSE'ye uygun kasis en kesiti (TSE, 1988)

✓ Ara yollardan bu tür tümseklerin bulunduğu yollara girişlerde de uyarıcı trafik işaretleri konulmalıdır (Şekil 2).




Şekil 2. Kasis işareti (TSE, 1988)

### 1.5.1. Kasis Tipleri

Kasisler, uzunlukları ve yükseklikleri bakımından 3 gruba ayrılırlar. Kasislerin bu özellikleri dikkate alınarak uygun yol güzergâhlarına eklenmelidir. Böylece araçların hızları istenilen düzeye indirilip, meydana gelebilecek her türlü olumsuzluk ve trafik sıkıntıları giderilmiş olacaktır. (Gürer,2006).

#### 1.5.1.1. Kısa Kasisler

Kısa kasislerin yükseklikleri 75-100 mm, uzunlukları ise 300-900 mm aralığındadır (Şekil 3). Bu kasisler daha çok aracın hızlı gitmesinin sakıncalı olduğu otoparklarda veya özel mülkiyet alanlarında kullanılmalıdır. Bu alanlar dışında kullanımı ise yasaklanmaktadır. Fakat Türkiye'de çoğu yol platformları üzerinde kısa kasislerden yararlanılmaktadır.

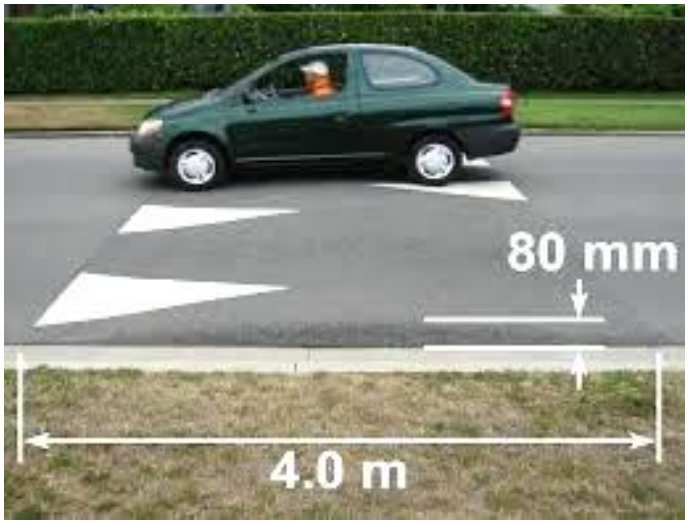
Şekil 3. Kısa kasis örneği (Ölçüler 500 x 465 x 45 mm ) (URL-2)

Bu kasislerin uzunlukları taşıtların ön panellerine oranla çok düşük kalmaktadır. Bu da aracın düşük hızda kasis üzerinden geçmesine rağmen sarsıntı şiddetini azımsanmayacak boyutta hissedilmesine neden olacaktır. Çünkü ani yükseklik değişimleri araçtaki düşey ivmelerin yükselmesine yol açmaktadır. Fakat bazı taşıtlar yüksek hız ile bu kasislerden geçtiklerinde tahmin edilen kadar yüksek bir sarsıntı hissetmemektedir. Bunun nedeni de araçlarda bulunan süspansiyon sistemlerindeki salınımların sıfıra indirilme özelliğinin olmasıdır. (Weber, 1998).

Ülke genelinde yol güzergâhları üzerine yapılan kasisler standart ölçülerde ve uygun malzemelerde yapılmamaktadır. Bu konuda Trabzon şehri incelenecek olursa burada yapılan kasisler mühendislik hizmetlerine dayanmadan uygulanmaktadır. Kasis yükseklikleri standartlarda belirtilenin neredeyse iki katı yüksekliğindedir. Kasislerin bulunduğu yerler için de genelde hızlı ve yoğun trafik olan ana cadde ve benzeri sokaklar tercih edilmiştir. Bu da trafik güvenliği ve taşıt tekniği açısından olumsuzluk doğurmaktadır. Bu olumsuzlukların verdiği en büyük zarar ise kasisin araçlara sert darbe uygulaması sonucu araçların ön takımı ve yürüyen aksamlarının hasar almasıdır. Bu durumda dolayısıyla araçlardaki süspansiyon sistemlerinin arızalanmasına neden olmaktadır.

#### 1.5.1.2. Uzun Kasisler

Uzun kasisler, TSE tarafından hazırlanan TS 6283 standardına göre uzunluğu araçların dingil mesafesine bağlı olarak 3.6-3.8m, tümseğin yüksekliğinin ise 7.5-10 cm olduğu kasislerdir (Şekil 4). Taşıtlar uzun kasislerden geçerken hızlarının belirli bir miktarda düşürmelidirler. Çünkü hızlarını düşürmedikleri takdirde kasisin uzunluğundan ötürü taşıt içerisinde birtakım salınımlara maruz kalacaklardır.




Şekil 4. Uzun kasis örneği (URL-3)

Bir araç bu tarz bir kasise ilk yaklaştığında aracın ön tekerleri kasis üzerindeyken arka tekerleri düz olan yol platformu üzerinde olmaktadır. Bu da araçtaki iki aksın farklı yönlerde harekete geçmesini sağlamaktadır. Çünkü araçlarda bulunan akslar, motorun sağladığı gücü ve dengeyi tekerleklere aktararak, aracın hızlanırken veya yavaşlarken güvenliğini ve konforunu sağlamaya çalışırlar. Bu yüzden kasis üzerinden yüksek bir hız ile geçilmesi ani yükselme ve alçalmalara neden olmaktadır. Bu ani hareketlenmelerde araç içindekilerin konforunu bozmaktadır (Gürer,2006).

#### 1.5.1.3. Düz Tepeli Kasisler

Düz tepeli kasislerin yol platformu üzerindeki enine uzunlukları 6-7 m, tepe düzlük kısımlarının uzunluğu 3 m ve yoldan yükseklikleri 100 mm şeklindedir. (Şekil 5) (Sözen, 2007).

Şekil 5. Düz tepeli kasis örneği (URL-4)

Bir araç düz tepeli kasis üzerinden geçerken aracın ön ve arka tekerlekleri kasisin üzerindeki düzlükte rahatlıkla durabilmektedir. Bunun nedeni kasis üzerindeki düzlüğün uzunluğunun fazla olmasıdır. Böyle bir durum araçlarda bulunan aks sistemlerinin aynı yönde harekete geçmesini sağlamaktadır. Böylece araç içindeki konfor ve güvenliğin sürdürülmesi devam ettirilmiş olur. Yani bu tarz kasislerde araç içinde rahatsızlık hissetmek ancak hız sınırının yükseklerde seyrettiği durumlarda söz konusu olabilmektedir. Ama yine de diğer kasis türlerine göre buradaki hız sınırı daha yüksektedir (Gürer, 2006).

Düz tepeli kasislerin ağır vasıtalardaki durumları ise daha farklıdır. Çünkü ağır vasıtaların süspansiyon sistemleri farklı çalışmaktadır. Ağır vasıta yüksek bir hızla kasise girdiğinde tekerlerin kasis üzerindeki teması kesilebilmektedir. Fakat düz tepeli kasislerde düzlük olan kısmın uzun olmasından dolayı aracın tekerleklerinin kasisten teması kesildikten sonra tekerlekler düz bölüme denk gelecektir. Bu da aracın hızının fazla olmasıyla orantılı olarak sarsıntı şiddetini arttıracaktır (Gürer, 2006).

Sonuç olarak, kısa kasisler araçların hızlarını azaltmak amacıyla kullanılıyor gibi gözükse de sürücü hızını düşürse dahi kısa kasisler üzerinden geçerken ciddi sarsıntılara ve araçların ön akslarında hasarlara neden olabilmektedirler. Bu nedenle kısa kasislerin ortaya çıkardığı olumsuzlukları giderip daha sağlıklı bir önlem için uzun kasislerin kullanılması daha doğrudur. Böylece sürücü hız seviyesini düşürdüğü zaman fazla sarsıntıya uğramadan seyrine devam edebilmektedirler. Düz tepeli kasisler ise üzerindeki düzlükler sayesinde araçların bu düzlük üzerinde durabilmeleri olanağını sağlayıp uzun kasislere göre daha da

az sarsıntı vermektedirler. Şekil 6 ile bu üç farklı kasis türlerinin uzunluk farkları gösterilmiştir.


Şekil 6. Kasis tipleri (Stephens, B.W., 1986)

### 1.5.2. Kasislerin Etkileri

#### 1.5.2.1. Kasislerin Olumlu Etkileri

Kasisler, özellikle kent yol ağları içerisinde motorlu taşıtların hızlarını düşürerek trafiğin sakinleştirilmesi amacıyla kullanılmaktadır. Kasislerin kullanım alanları sadece okul, hastane gibi taşıt trafiğinin yoğun olduğu yerler değil, çocuk bahçesi, konut alanları, parklar gibi yaya trafiğinin de yoğun olduğu yerleri de kapsamaktadır. Burada amaç kent içinde trafiğin daha güvenli bir şekilde seyretmesini sağlayıp, motorlu taşıtların hızlarının 40 km/h ya da daha düşük seviyelerde olmasını sağlamaktır. (Güzel vd., 2019).

Kasislerin trafik yoğunluğuna da olumlu etkileri bulunmaktadır. Mesela kent içlerinde kullanılan kasisler trafik hız limitini düşürdüğünden bir yere daha kısa sürede gitmeyi hedefleyen araç kullanıcıları gideceklere yerlere ulaşmada alternatif yollar aramakta ve bu yolları tercih etmektedirler. Böylece kent içlerinde trafik yoğunluğunun olması engellenmiş olmaktadır. Bu da beraberinde trafik kazalarındaki azaltarak trafikte iyileşmelerin olmasını tetiklemektedir (Gürer, 2006).

#### 1.5.2.2. Kasislerin Olumsuz Etkileri

Kasislerin asıl amacı sürücülere hafif bir sarsıntı hissettirerek taşıt hızlarını belirli seviyede tutmaktır. Ancak burada önemli olan kasislerin uzunlukları, yükseklikleri ve genişlerinin doğru bir şekilde, doğru maddelerden yapılmasıdır. Çünkü uygun olmayan kasis yapılarının kullanılması araç içindeki sürücülere fazla sarsıntı yaşatıp tehlikeli durumların

oluşmasına ve yayaların güvenliğinin olumsuz yönde etkilenmesine neden olmaktadır (Gürer,2006).

Bir aracın kasisi fark etmeden kasis üzerinden hızla geçmesi araçta ve sürücüde ciddi hasarlara neden olabilmektedir. Hatta bu durum trafik kazalarına bile neden olmaktadır. Aynı zamanda gereğinden fazla yükseklikte sert bir maddeden yapılmış olan kasisler araçların ön akslarına da zarar vermektedirler. Kasislerin bu tarz olumsuz özellikleri aslında sadece motorlu taşıtlarda değil bisiklet kullanıcıları içinde sorun teşkil etmektedir (Güzel vd., 2019).

Kasislerin yol güzergâhları üzerinde seyreden acil müdahale araçlarını yavaşlatmak veya ambulans içindeki hastalarda sarsıntılara neden olmak gibi bir olumsuz özelliği daha vardır. Bu durumun önüne geçmek için bazı ülkelerde acil müdahale araçlarının devamlı kullandığı yollar üzerine kasislerin konulması yasaklanmıştır. Kuzey Amerika'da yapılan bir araştırmada kasislerin bu araçlar üzerindeki etkilerini gösteren sonuçlar Tablo 1'de verilmiştir (Gürer,2006).

Tablo 1. Acil durum araçlarının tümseklerde gecikme süreleri (Ewing, R., 1999)

<table>
<tr><th>Bölge</th><th>Austin, Texas</th><th>Berkeley, California</th><th>Boulder, Colorado</th><th>Sarasota, Florida</th></tr>
<tr><td rowspan="4">Tümsek tipi</td><td rowspan="4">3.66 m/ uzun tümsek</td><td rowspan="2">3.66 m/ uzun tümsek</td><td>2,5 m/ uzun tümsek</td><td rowspan="4">3.66 m/ uzun tümsek</td></tr>
<tr><td>3.66 m/ uzun tümsek</td></tr>
<tr><td rowspan="2">6.7 m/ düz tepeli tümsek</td><td>11.3m/ düz tepeli tümsek (yükseklik 150 mm)</td></tr>
<tr><td>12.2 m/ düz tepeli tümsek (yükseklik 150 mm)</td></tr>
<tr><td rowspan="4">Gecikme süresi (saniye)</td><td>2.8 (itfaiye aracı)</td><td>10.7 (itfaiye aracı)</td><td>4.7 (itfaiye aracı)</td><td rowspan="4">9.5 (ambulans)</td></tr>
<tr><td>3.0 ( merdiveni itfaiye aracı)</td><td>9.2 ( merdiveni itfaiye aracı)</td><td>2.8 (itfaiye aracı)</td></tr>
<tr><td>2.3 (hasta olmayan ambulans )</td><td>3.0 (itfaiye aracı)</td><td>3.8 (itfaiye aracı)</td></tr>
<tr><td>9.7 (hasta taşıyan ambulans)</td><td>13.5 ( merdiveni itfaiye aracı)</td><td>3.8 (itfaiye aracı)</td></tr>
</table>

Kasise yaklaşan aracın hızını düşürüp kasisi geçtikten sonra tekrar hızlanmasıyla araçta yakıt tüketiminin artmaktadır. Bu durum hem araçların egzozlardan çıkan gazında fazla olmasına hem de araçtan çıkan sesin fazla olmasına neden olmaktadır. Çıkan ses fazlalığı nedeniyle de çevrede gürültü kirliliği oluşmaktadır (Gürer, 2006).

Bütün bu olumsuz sonuçlar dikkate alındığında kasisler tasarlanırken ve uygulanırken kent yol ağı değerlendirmesi göz önünde bulundurulmalıdır. Böylece hem araçlar, hem yolcular hem de çevre için daha sağlıklı bir kent yaşamı sağlanmış olur.

### 1.6. Akıllı Telefon Teknolojileri

Cep telefonlarına PDA (Personal Data Assistant - Kişisel Veri Yardımcısı) özelliği eklenerek telefonlara akıllı olma yetisi kazandırılmıştır. Bu eklenen özellik sayesinde akla gelebilecek hemen hemen her konuyla ilgili telefon kullanıcılarına fayda sağlanabilmektedir. Bunun içinde yapılacak işle ilgili uygulamalar geliştirilmektedir (URL-5).

İlk akıllı telefonu, dünyanın en büyük bilişim teknolojisi şirketi olan ABD'li IBM (International Business Machines; Uluslararası İş Makineleri) 1994 yılında piyasaya sürmüştür. Bu telefon IBM'nin Simon telefonu olarak bilinmektedir (Şekil 7). Simon telefon dokunmatik bir telefon olmakla beraber çağrı cihazı, faks cihazı ve cep bilgisayarı olarak nitelendirilmektedir. Bu telefonun içerisinde hesap makinesi, not defteri, takvim, oyun gibi özellikler barınmaktadır. Bu özellikleri sayesinde o dönem için bir bilgisayarı aratmayacak şekilde işe yaradığı düşünülmektedir. (URL-5).


Şekil 7. İlk akıllı telefon IBM Simon telefonu (URL-5)

1996 yılında Nokia ilk akıllı telefonu olan "Nokia 9000 Communicator" isimli telefonu piyasaya sürmüştür. Bu telefon ilk QWERTY klavyesini tam olarak barındıran bir

telefondur. Telefonda ilk dikkat çeken özellik telefonun katlanabilir olmasıdır. Bu sayede telefon cebe rahatlıkla sığabilmektedir. İkinci dikkat çeken özellik ise fax tuşunun telefon üzerinde bulunuyor olmasıdır. 2000 yılına kadar Nokia bu telefonun farklı 3 modelini piyasaya sunmuştur (Şekil 8) (URL-5).


Şekil 8. Nokia 9000 Communicator akıllı telefonu (URL-5)

1997 yılında ise Ericsson GS88 modelli bir cep telefonu çıkarmıştır. GS88 ile birlikte piyasaya sürülen bu tarz özelliklere sahip cep telefonlarının "akıllı telefon (smart phone)" olarak nitelendirilmesi literatürlere geçmiştir. Ericsson 2000 yılında piyasaya ikinci akıllı telefonu olan R380'i sunmuştur. (Şekil 9) (URL-5).


Şekil 9. Ericsson GS88 modelli akıllı telefonu (URL-5)

2002 yılında Amerikalı Handspring isimli firma “Palm Treo” adında akıllı bir telefon piyasaya sunmuştur. Yine aynı yıl “BlackBerry 5810” modeli piyasaya sürülmüştür. Bu telefonda çağrı yapmak için gerekli olan kulaklıklar bulunmaktadır. Ayrıca bu modele sahip telefon e-posta alıp göndermek için de gerekli olan özelliği içinde barındırmaktadır. BlackBerry, 2006 yılında “BlackBerry Pearly” adında bir telefonu piyasaya sürmüştür (Şekil 10) (URL-5).


Şekil 10. BlackBerry 5810 modelli akıllı telefon (URL-5)

Daha sonra 2002 yılında akıllı telefon olarak farklı modeller ortaya çıkmıştır. Bu yıl içerisinde Microsoft “Pocket PC” akıllı telefonunu çıkarmıştır. Bu telefon “Windows Pocket PC OS” işletim sistemine sahiptir (Şekil 11) (URL-5).


Şekil 11. Pocket PC akıllı telefonu (URL-5)

2007 yılında ise Apple, Steve Jobs öncülüğünde ilk iPhone modelini çıkarmıştır. iPhone işletim sistemi olarak iOS kullanmaktadır. iPhone ekranın tam anlamıyla dokunmatik olması ve geniş bir ekrana sahip olması nedeniyle piyasada çok ses getirmiştir. Ayrıca bu özellikleri sayesinde kendinden sonra çıkacak her akıllı telefon için bir referans olmuştur. 2008 yılında ise iPhone telefonlara 3G özellik gelmiştir (Şekil 12) (URL-5).

Şekil 12. İlk iPhone (URL-5)

2008 yılına gelindiğinde ise Google ilk akıllı telefonu "HTC Dream" isimli telefonu piyasaya sürmüştür. HTC Dream, ilk Android telefon olma özelliğine sahiptir. Bu telefonun iPhone'dan sonra tuşlu olarak üretilmesi kullanıcılara sıkıntı ve zorluk olarak gelmiştir. Ayrıca telefonun kızaklı olup cebe sığmaması ve kulaklık girişinin olmaması kullanıcılar

tarafından hoş karşılanmamıştır. O döneme göre gelişmiş bir telefon olarak görülmese de günümüzde Android işletim sistemiyle çalışan pek çok telefon bulunmaktadır (Şekil 13) (URL-6).


Şekil 13. Android işletim sistemli ilk akıllı telefon HTC Dream (URL-7)

Telefon satışları 2013 yılı itibariyle dünya genelinde 1 milyar satış yapmaktadır. Bu durum toplumdaki her yaş grubu arasında akıllı telefonun yaygınlaşmasına sebep olmaktadır. Akıllı telefonların her geçen gün gelişmesiyle de bu ilgi devam etmektedir (Saraç, 2019).

Güvenilir istatistik şirketlerinden biri olan Gartner**,** hazırlamış olduğu tablo ile dünya genelinde telefon satışlarının düşüşte olduğunu ortaya koymuştur (Tablo 2). 2019 yılının Ocak, Şubat, Mart aylarındaki tespite göre telefon satışlarında yüzde 2.7 miktarında bir azalma yaşanmıştır. Bu durum toparlanmaya çalışılırken Nisan, Mayıs ve Haziran aylarında yüzde 1.7'lik bir düşüş daha tespit edilmiştir. Kıdemli araştırma direktörü Anshul Gupta, piyasa için orta düzey telefonların daha fazla rağbet göreceğini ve düşüş yaşanan telefon satış miktarlarında da artış yaşanacağını düşünmektedir (URL-8) .

Tablo 2. Akıllı telefon üreticilerinin satış rakamları ve pazar payları (URL-8)

| Vendor | Samsung | Huawei | Apple | Xiaomi | OPPO | Others | Total |
|---|---|---|---|---|---|---|---|
| **3Q19 Units** | 79,056.7 | 65,822.0 | 40,833.0 | 32,271.3 | 30,834.4 | 138,659.9 | **387,477.2** |
| **3Q19 Market Share (%)** | 20.4 | 17.0 | 10.5 | 8.3 | 8.0 | 35.8 | **100.0** |
| **3Q18 Units** | 73,360.1 | 52,218.4 | 45,746.6 | 33,219.7 | 30,563.4 | 153,965.8 | **389,074.0** |
| **3Q18 Market Share (%)** | 18. | 13.4 | 11.8 | 8.5 | 7.9 | 39.6 | **100.0** |

Samsung, Huawei, Xiaomi, OPPO ve diğer telefonlar satış miktarlarını gözlemleyerek telefon kullanıcıları için yeni yöntemler geliştirmeye çalışmaktadırlar. Apple ise bu yıl geçen yıla göre zarar ettiğini görmektedir. Yukarıdaki tabloya göre; Samsung 2018 yılının üçüncü çeyreğinde yüzde 18.9 oranındayken 2019 yılının üçüncü çeyreğinde artış göstererek yüzde 20.4'e yükselirken, Huawei yüzde 13.4'ten yüzde 17'ye yükselmiştir. Apple, yüzde 11.8'den yüzde 10.5'e doğru bir azalma yaşamıştır. Xiaomi yüzde 0.2 oranında azalırken, Oppo yüzde 0.1 oranında bir yükselişe geçmiş, diğer telefonlar ise yüzde 3.8 oranında azalma yaşamıştır. Çıkartılan bu hesaplar göz önüne alındığında Samsung marka telefonun satış zirvesinde olduğu gözlenmektedir (URL-8). Samsung telefonlarının zirvede olmasının en önemli nedeni Google'ın Android işletim sistemini her yıl sürekli olarak yenileyip geliştirmesidir. Hatta bu sebepten dolayı 2017 yılında da Microsoft'u geride bırakmıştır. Bu da Android işletim sistemin en çok kullanılan işletim sistemi yapmakta ve böylece Android işletim sisteminin hızla yayılması sağlanmaktadır (Saraç, 2019).

Günümüzde en çok kullanılan işletim sistemleri Android ve iOS' tur. Diğer işletim sistemleri teknolojik gelişmelerin gerisinde kaldığı için piyasada fazla yer edinememiştir. Uluslararası Veri Şirketi (IDC) yaptığı araştırma ile işletim sistemlerinin pazar paylarını sunmuştur (Tablo 3). Yapılan tabloya göre; Android' in 2020 yılından sonra ufak bir artış göstermesi beklenmektedir. iOS 'da ise durum Android' in tam tersi olup ilerleyen yıllarda düşüşün olması beklenmektedir.

Tablo 3. İşletim sistemlerinin kullanıcı sayıları ve pazar payları (URL-9)

| Yıl | Android | İOS | Diğerleri | Toplam |
|---|---|---|---|---|
| **2017** | %85.1 | %14.7 | %0.2 | %100 |
| **2018** | %85.1 | %14.9 | %0.0 | %100 |
| **2019** | %86.6 | %13.4 | %0.0 | %100 |
| **2020** | %86.6 | %13.4 | %0.0 | %100 |
| **2021** | %86.9 | %13.1 | %0.0 | %100 |
| **2022** | %87.0 | %13.0 | %0.0 | %100 |
| **2023** | %87.1 | %12.9 | %0.0 | %100 |

Android ve iOS arasındaki en belirgin fark yazıldıkları kodlama dilidir. iOS swift programlama diliyle yazılmış olup objective-c üzerine kurulmuştur. Android ise Java programlama diliyle yazılmıştır. Her iki işletim sistemi de açık kaynaklı dillerdir. Fakat iOS kapalı bir sistemdir, bu yüzden dışarıdan erişim zordur. Bu da iOS' u kötü saldırılara karşı daha korumalı yapmaktadır. Ancak Android açık kaynak kodlu ve geliştirilebilen bir işletim sistemi olduğu için Android'e dışardan uygulama yüklenebilmektedir. Bu da dışardaki kötü saldırılara maruz kalma durumunu kolaylaştırmaktadır. Bu iki işletim sisteminden iOS' un ara yüzü daha basit ve kullanımı kolaydır. Android ise sınırlamalar ve esneklik konusunda müdahaleye açık olmasından dolayı daha avantajlı olmaktadır. Ayrıca Android düşük kapasiteye sahip telefonlarda bile çalışabilirken, iOS sadece Apple telefonlarında çalışmaktadır. Sonuç olarak her iki işletim sisteminin de birbirinden üstün olduğu durumların olduğu görülmektedir (URL-10).

### 1.7. Android İşletim Sistemleri

Android Inc. adında bir şirket Andy Rubin, Rich Miner, Nick Sears ve Chris White tarafından kurulmuştur. Google ise 2005 yılında Android'i satın almıştır ve Android gelişmeye başlamıştır (URL-11).

Android işletim sistemi dört ana katmandan oluşmaktadır. Bunlar: Linux çekirdek (Linux Kernel), kütüphaneler (Libraries)ve Android çalışma zamanı (Android Runtime), uygulama çatısı (Application Framework) ve uygulamalar (Applications) katmanlarıdır (Tablo 4).

Tablo 4. Android Sistem Mimarisi ( (Kabakuş vd., 2015)

<table>
<tr><td colspan="5">Uygulamalar</td></tr>
<tr><td>Ana Ekran</td><td>Kişiler</td><td>Öğeler</td><td>Taşıyıcı</td><td>…</td></tr>
<tr><td colspan="5">Uygulama Çatısı</td></tr>
<tr><td>Aktivite Yöneticisi</td><td colspan="2">Pencere Yönetici</td><td>İçerik Sağlayıcılar</td><td>Görüntü Sistemi</td></tr>
<tr><td>Paket Yöneticisi</td><td>Telefon Yöneticisi</td><td>Kaynak Yöneticisi</td><td>Yer Yöneticisi</td><td>Bilgilendirme Yöneticisi</td></tr>
<tr><td colspan="3">Kütüphaneler</td><td colspan="2">Android Yürütücüsü</td></tr>
<tr><td>Yüzey Yöneticisi</td><td>Medya Uyg. Çatısı</td><td>SQlite</td><td colspan="2">Çekirdek Kütüphaneler</td></tr>
<tr><td>OpenGL ES</td><td>FreeType</td><td>Webkit</td><td colspan="2" rowspan="2">Dalvik Sanal Makinası</td></tr>
<tr><td>SGL</td><td>SSL</td><td>libc</td></tr>
<tr><td colspan="5">Linux Çekirdeği</td></tr>
<tr><td>Görüntü Sürücüsü</td><td colspan="2">Kamera Sürücüsü</td><td>Flash Bellek Sürücüsü</td><td>Bağlantı (IPC) Sürücüsü</td></tr>
<tr><td>Tuş Takımı Sürücüsü</td><td colspan="2">Wifi Sürücüsü</td><td>Ses Sürücüleri</td><td>Güç Yönetimi</td></tr>
</table>

Android, Linux çekirdeğini kullanan bir işletim sistemidir. Java kütüphanesini esas almıştır. Android işletim sisteminde uygulamalar API (Android Application Programming Interface) kullanılarak geliştirilmiştir. Uygulamalar tarafından istenen; galeriye ulaşma, konum bilgisi alma ve SMS gönderme gibi konular için izin istemektedir. Kullanıcı izini verdiği zaman Android işletim sistemindeki uygulama bileşenleri içinden istenilen izne uygun olan bilgi işlenerek işlemi gerçekleştirir (Kabakuş vd., 2015). İzne ihtiyaç duyan tüm Android uygulamaların bütün izinleri uygulamanın "AndroidManifest.xml" dosyasında bulunmaktadır. Android uygulamalar Apk uzantılı dosyalarda paketlenmiştir. Bu dosyalar uygulamaların sahip olduğu bütün sınıf ve kaynakları barındırmaktadır (Pocatilu, 2011). Java dilinde yazılan uygulamalarda sınıflar derlenmekte ve Java Virtual Machine (JVM) byte kodu oluşturulmaktadır. Dalvik Virtual Machine (DVM) bu sınıfları özel bir formata dönüştürür ve dex dosyasında tüm sınıfları toplar. Toplanan bu kodlar uygulama ilk kez başlatıldığında ise optimize edilmektedir (Kabakuş vd., 2015).

### 1.7.1. Android Sürümleri

#### 1.7.1.1. Android 1.1 (Petit Four)

Google'ın HTC Dream'i çıkardığı zaman kullandığı ilk Android sürümüdür. Bunun diğer ismi "Petit Four" olmasına rağmen bu ismi duyurmamış sadece kendi aralarında kullanmışlardır (URL-12)(Şekil 14).


Şekil 14. Android 1.0 işletim sistemi ekran görüntüleri (URL-13)

#### 1.7.1.2. Android 1.5 (Cupcake)

Android 1.5'in diğer ismi olan "Cupcake" halka duyurulmuş ilk Android sürümüdür. Bundan sonraki çıkacak olan yeni Android sürümlerinde kullanılan isimlerin alfabetik sıraya göre olacağı söylenmiştir.

Cupcake ile telefonlara ekran geçişi sağlayacak UI (Kullanıcı Ara yüzü) ile otomatik ekran döndürme seçeneği gelmiştir. Ayrıca klavyede yazı yazarken metin tahmini yapma özelliği, ana ekrana eklenebilen Widget pencereleri ve MPEG-4 ve 3GP video formatlarında video kaydı yapabilme özellikleri eklenmiştir. Telefon üzerinden kullanılan tarayıcılardan adres kopyalama ve yapıştırma gibi kolaylıklarda getirilmiştir (URL-12) (Şekil 15).


Şekil 15. Android 1.5 işletim sistemi ekran görüntüleri (URL-14)

### 1.7.1.3. Android 1.6 (Donut)

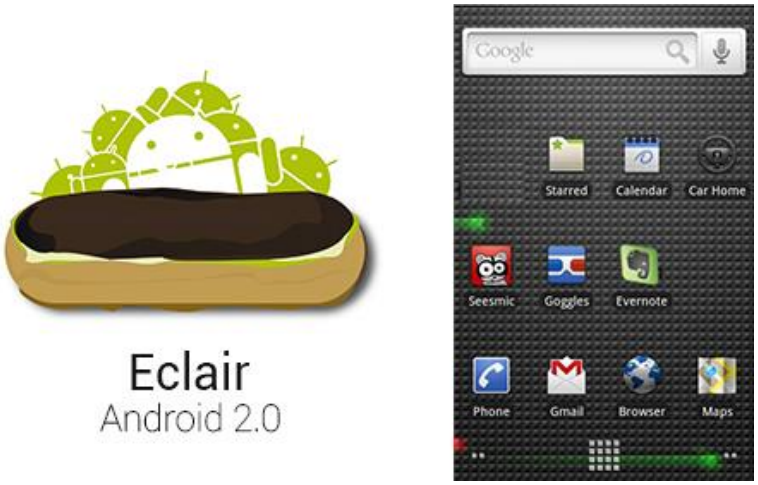
Donut, WVGA ekran çözünürlüğü ile farklı boylardaki cihazlarda ekranın sorunsuz çalışması sağlanmıştır. Android market olan Play Store' da uygulamalar için kategoriler gelmiştir. Böylece ücretli ve ücretsiz uygulamalar kolaylıkla ayırt edilebilir olmakla beraber istenilen uygulamanın adı da arama çubuğuna yazılarak rahatlıkla ulaşım sağlanmıştır (URL-12) (Şekil 16).


Şekil 16. Android 1.6 işletim sistemi ekran görüntüleri (URL-14)

### 1.7.1.4. Android 2.0 ( Eclair)

Eclair ile canlı duvar kâğıtları, navigasyonda yardımcı elemanlar, trafik bilgileri, 3D görünüm, sesli yönlendirme gibi özellikler eklenmiştir. Ayrıca kamera fonksiyonlarına da flaş, yakınlaştırma yapabilme, efekt özellikleri, sahne modu, beyaz dengesi nitelikleri gelerek fotoğraf makinelerine yakın çekimler yapabilme sağlanmıştır.

Android' in bu sürümü az miktarda API değişiklikleri ile hata düzeltmelerini içermektedir. Eclair ile Google ilk Nexus cihazı olan "Nexus One" cihazını tanıtmıştır. Bu telefon aslında HTC tarafından üretilmiş olup ancak 2010 yılında Google tarafından piyasa sürülmüştür. Böylece Android 2.1'i kullanan ilk telefon olmuştur (URL-12) (Şekil 17).




Şekil 17. Android 2.0 işletim sistemi ekran görüntüleri (URL-14)

### 1.7.1.5. Android 2.2 (Froyo)

Android Froyo, 4 inçlik 720p ekranlar gibi yüksek PPI ekranlar (320 ppi'ye kadar) için güçlendirme gelmiştir. Ayrıca hız ve performans bakımından geliştirmeler olmuştur. Telefon belleklerinin daha yüksek seviyede olması için yenilemeler yapılmıştır. Telefonda kullanılan uygulamalardan; arama yapmada, mesaj atmada, yol tarifi almada sesli komut vererek işlemlerin daha hızlı ve kısa yoldan yapılması sağlamıştır (URL-12) (Şekil 18).


Şekil 18. Android 2.2 işletim sistemi ekran görüntüleri (URL-14)

#### 1.7.1.6. Android 2.3 (GingerBread)

GingerBread, büyük ekran boyutlarıyla daha iyi performanslar sağlaması için geliştirilmiştir. Ekrandaki ara yüzün daha basit bir şekilde olmasıyla kullanım kolaylığı getirmiştir. Telefonda uygulamalar kullanırken, telefonun uzun süre uyanık kalmasını sağlamak için güç yönetimi ve kontrol uygulamaları getirilmiş olup bu durum pil ömrünü uzatmıştır. Android telefonlarda oyun oynamanın daha keyifli ve yüksek kalitede olabilmesi için ses, grafik, cihaz kontrolleri ve depolama alanı özellikleri eklenmiştir. Bu ek API dosyasıyla mümkün olmuştur. Çoklu kamera desteği gelerek, ön kameralarda daha fazla sensörün olması sağlanmıştır. Ayrıca NFC (Near Field Communication) yani yeni nesil kablosuz iletişim teknolojisi geliştirilmiş olup başka elektronik cihazlarla yakın mesafeden gerekli iletişimin kurulması sağlanmıştır. (URL-12) (Şekil 19).


Şekil 19. Android 2.3 işletim sistemi ekran görüntüleri (URL-14)

### 1.7.1.7. Android 3.0 (HoneyComb)

HoneyComb, sadece tabletler için tasarlanmıştır. Büyük boyutlu ekranlarda kullanım kolaylığı sağlamaktadır. Google HoneyComb' un özelliklerini daha göstermesi için "Motorola Zoom" tableti model olarak seçmiştir (URL-12) (Şekil 20).


Şekil 20. Android 2.3 işletim sistemi ekran görüntüleri (URL-14)

### 1.7.1.8. Android 4.0 ( Ice Cream Sandwich)

Ice Cream Sandwich, eski Android sürüm olan Android 2.3 ve Android 3.0'ın birleşiminden oluşmaktadır. Böylece hem akıllı telefonlarda hem de tabletlerde çalışmaktadır. Ice Cream Sandwich ile ekranda gezinmeler kolaylaştırılmış, telefona yeni yazı stili olan "Roboto" eklenmiştir. Telefonun kilini açma konusunda da yeni bir boyut getirerek yüz tanıma özelliği olan "Face Unlock" getirilmiştir. Widget'lar için özel bir bölüm yapılıp uygulamalar için sekmeler oluşturulmuştur. İnternet kullanımlarını kontrol etmek ve kullanım limitlerine bakmak için "Veri Kullanım Kontrolü" özelliği getirilmiştir. Android mobil işletim sisteminin bir özelliği olan Android Beam ile NFC donanımlı cihazlar arasında yol tarifi, iletişim bilgileri ve veri alışverişleri sağlanmıştır. Her Android sürümünde olduğu gibi kamera özellikleri tekrardan güncellenmiştir. Bu güncellenmeyle sıfır çekim gecikmesi, zaman atlamalı panorama modları ve 1080p video kaydı gelmiştir. Ayrıca telefonlara kendi fotoğraf editörlerini ekleyerek çekilen fotoğrafların düzenlenmesine olanak sağlamıştır. (URL-12) (Şekil 21).


Şekil 21. Android 4.0 işletim sistemi ekran görüntüleri (URL-14)

### 1.7.1.9. Android ( 4.1 / 4.2 / 4.3 Jelly Bean)

Jelly Bean, kullanıcı ara yüzünde performans iyileştirilmesi sağlamak için Project Butter'ı tanıtmıştır. Google Asistan özelliği ile telefona bir yardımcı tanımlanmıştır. Böylece bulunulan konuma, tarihe ve beğenilen konulara göre bilgi akışı sağlanır. Jelly Bean ile

telefondaki uygulamalardan gelen bildirim özelliklerine harekete geçirilebilir bildirimler getirilmiştir. Ayrıca istenilen uygulamadan bildirim gelmesini onaylamak istenmeyen uygulamanın da bildirimlerini kapatma özelliği getirilmiştir (URL-12) (Şekil 22).


Şekil 22. Android 4.1 – 4.3 işletim sistemi ekran görüntüleri (URL-14)

#### 1.7.1.10. Android 4.4 (Kitkat)

KitKat'ın en çok dikkat çeken özelliği "Ok Google" sesli komutudur. Bu komutu vererek telefonda arama yapma, mesaj yollama, müzik çalma, yol tarifi alma ve hava durumunu öğrenme gibi işlemler yapılmaktadır. Kitkat kullanıcı ara yüzü içinde tasarım konusunda geliştirmeler yapmıştır. Ayrıca Kitkat Android Wear gibi giyilebilir teknolojiler içinde bir sürümünü piyasaya sürmüştür (URL-12) (Şekil 23).


Şekil 23. Android 4.4 işletim sistemi ekran görüntüleri (URL-14)

### 1.7.1.11. Android 5.0 (Lollipop)

Lollipop, ilk defa "Materyal Tasarımı" adı verilen yeni bir tasarım türünü benimseyerek, görünümünü değiştirmiştir. Lollipop ile Android ilk defa televizyon ve arabalarda kullanılmaya başlanmıştır. Ayrıca Lollipop ile bir Android cihazdaki şarkılar, fotoğraflar, uygulamalar ve son aramalar, tüm Android cihazlardan da erişilebilir olmuştur. (URL-12) (Şekil 24).


Şekil 24. Android 5.0 işletim sistemi ekran görüntüleri (URL-14)

### 1.7.1.12. Android 6.0 (Marshmallow)

Marshmallow ile Google Asistan'ın ana ekran tuşuna uzun basılarak açılması sağlanmıştır. Bunun için "Now On Tap" özelliği eklenmiştir. Pil ömrünü uzatmak için kullanılan App Standly ile ekran kapalıyken devreye girip CPU (Central Processing Unit) işlemci hızını azaltan Doze modu birleştirilmiştir. Telefon ekranını açmak için parmak izi desteği getirilerek güvenliğin arttırılması sağlanmıştır (URL-12) (Şekil 25).


Şekil 25. Android 6.0 işletim sistemi ekran görüntüleri (URL-15)

### 1.7.1.13. Android 7.0 (Nougat)

Nougat, kullanıcı ara yüzü (UI) geliştirmeleri ile ekrana çift dokunuşlu uygulamalar sayesinde iki uygulamayı yan yana çalıştırabilme olanağı sağlamıştır. Yeni JIT ( Just in Time ) derleyici ile daha hızlı güncellemelerin olmasını, VR (Sanal Gerçeklik) mod ve Vulkan API desteği ile de daha iyi oyun görüntü deneyimi sağlamıştır (URL-12) (Şekil 26).


Şekil 26. Android 7.0 işletim sistemi ekran görüntüleri (URL-16)

### 1.7.1.14. Android 8.0 (Oreo)

Oreo'nun kullanıcı ara yüzünde (UI) ve tasarım dilinde hiçbir değişiklik yapılmamıştır. Bu Android sürümü ile sağ üstte ilgili uygulamanın sembolü ile bildirim gösterilecektir. Uygulamaların simgelerine belli bir süre basılınca açılan mini bir pencere ile mesajın bir kısmının gözükmesi sağlanmıştır. Google bildirim hatırlatıcı özelliğini genişleterek bildirim uyarılarını yana itip saat ve ayar sembollerine ulaşabilme imkânı sunmuştur (URL-17) (Şekil 27).


Şekil 27. Android 8.0 işletim sistemi ekran görüntüleri (URL-18)

### 1.7.1.15. Android 9.0 (Pie)

Android Pie, yapay zekâyı ön plana çıkarmaktadır. Kullanılmayan uygulamaları algılayarak bunların arka planda çalışmasını engeller ve böylece pil tasarrufu yapılmasını sağlamaktadır. Telefonla konuşmalar esnasında gürültülü bir ortamda olunduğunda sesi artırmak için ekstra ses düzeyi ayarı ile konuşmaların daha kolay anlaşılmasına olanak tanımıştır. Çoklu kamera desteği ile kameralarda farklı deneyimler yaşanmasını sunmuştur. Eklenen "Rahatsız Etmeyin" seçeneği ile uyku anlarında bildirim seslerini sessize alınması sağlanmıştır (URL-19) (Şekil 28).


Şekil 28. Android 9.0 işletim sistemi ekran görüntüleri (URL-20)

### 1.7.1.16. Android 10 (Q)

Beta sürümü sunulan Android Q, Android Ürün Yöneticisi Stephanie Cuthbertson tarafından tasarlanmıştır. Android Q özellikleri arasında dikkat çeken en önemli özellik karanlık moddur. Pil tasarruf modunun açılmasıyla ve hızlı ayarlar panelinden açılarak etkin hale gelmektedir. Telefon karanlık moddayken kullanılan tüm uygulamalarda da bu mod kullanılabilecektir. Android Q özellikleri arasında dikkat çeken bir diğer özellik ise ekran kaydıdır. Telefon ekranındaki görselleri kaydetmek için ekstra bir uygulamaya ihtiyaç duyamadan kolaylıkla kayıt yapılabilmektedir. Bunun için Google'ın makine öğrenme algoritmaları devreye girmektedir. Ayrıca Android Q ile 50 yeni gizlilik ve

güvenlik ayarları da gelmektedir. Bunların en önemlisi sadece uygulama çalışırken konum izlenebilme olayı olmakla beraber kullanıcı uygulamaların konumunu arka planda kullandığına dair bildirimler alabilecektir (URL-21)(Şekil 29).

Şekil 29. Android 10 işletim sistemi ekran görüntüleri (URL-22)

### 1.8. Android Studio

Android Studio, Android uygulama geliştirmede kullanılan resmi IDE (Entegre Geliştirme Ortamı) olarak tanımlanmaktadır. Google'ın resmi Android IDE'si olan Android Studio, akıllı bir kod düzenleyicisini ve hata ayıklayıcısını, performans analiz araçlarını, emülatörleri ve daha pek çok araç dâhil olmak üzere uygulama geliştirmek için ihtiyaç duyulan her şeyi içermektedir. Telefonlar ve tabletler, Android Auto, Android Wear ve Android TV de dâhil olmak üzere her türlü Android cihazda yüksek kalitede çalışan ve başarılı sonuçlar veren uygulamalar geliştirmek için en hızlı araçları sunmaktadır. Android Studio anında çalıştırma özelliği sayesinde kod ve kaynak değişikliklerinin bir cihaz veya emülatör üzerinde çalışan uygulamaya hızlı bir şekilde yansıtılmasını sağlayarak düzenleme, derleme ve çalıştırma süreçlerini hızlandırmaktadır. Gradle tabanlı esnek yapılı derleme sistemi, derleme otomasyonu, bağımlılık yönetimi ve özelleştirilebilir APK, Android uygulama paketi (Android Application Package), oluşturma yapılandırmaları sunmaktadır. Android uygulamasındaki kod şablonları, yaygın olarak kullanılan uygulama özelliklerinin oluşturulmasına yardımcı olmaktadır. Zengin yerleşim düzenleyici ile uygulamaları daha sezgisel bir şekilde oluşturmak için plan modu oluşturur ve sürükleyip bırakarak düzenlemeyi desteklemektedir. Büyük, karmaşık düzenlerin düz ve sadeleştirilmiş bir

hiyerarşiye göre tasarlanması için yeni sınırlama düzeni yöneticisine (Android API seviye 9'dan önceki sürümlerle uyumludur) sahiptir. Her bileşenin boyutuna olan katkısının anlaşılabilmesi için APK'ların içeriğinin incelenmesine olanak tanıyan APK analizcisi bulunmaktadır. Ayrıca Android cihazınızda OpenGL ES komut akışını yakalayan ve analiz için bunu Android Studio'nun içinde yeniden oynatan GPU hata ayıklayıcısını (beta) içermektedir (Android Guides,2020).

### 1.9. Android Studio ’da Kullanılan Programlama Dilleri

Android Studio programı kodlama yapmak için iki dil seçeneği sunmaktadır. Bunlardan birisi Java programlama dili, diğeri ise Kotlin programlama dilidir.

Java, ilk defa 1995 yılında büyük bir konferansla tanıtılmıştır. Yani Java Android’ den çok daha önce vardı. Bu nedenle Java illa Android ile çalışan ve onsuz yapamayan bir dil olmamıştır. Dünyanın en büyük teknoloji şirketlerinden birisi olan Oracle, Java’nın arkasında durmaktadır. Daha sonra Oracle Google’a dava açmıştır. Açılan dava sonrasında Google düzenlediği bir konferansla Android uygulama geliştirmelerde özellikle Android Studio programında Kotlin dilini desteklediğini duyurmuştur. Bu destekle beraber son zamanlarda Kotlin dili kullanımı yaygınlaşmaya başlamıştır.

Android ‘in geliştirilmesinde kullanılan ana yazılım dilinin Java olması, Google Play Store’ da Java ile yazılmış onlarca uygulamanın bulunması ve Android işletim sisteminde milyonlarca Java kodlarının bulunması, hem Android geliştiricileri hem de yazılım geliştiricileri tarafından Java’nın önemini kaybetmeyeceği düşünülmektedir.

#### 1.9.1. Kotlin Programlama Dili

Kotlin, 19 Temmuz 2011 yılında JVM( Java Virtual Machine) Language Summit etkinliğinde duyurulmuş olup, Rus merkezli JetBrains şirketi yazılımcıları tarafından yapılmıştır. JetBrains şirketi ; *“* Performans ve güvenliği riske etmeden Java’dan daha özlü kodlar yazmayı sağlamak için bu dili geliştirdik*”* diyerek Kotlin’nin önemli olduğunu belirtmiştir. Kotlin nesne yönelimli statik olarak Apache 2.0 lisansı altında geliştirilmiş ücretsiz, açık kaynak koda sahip bir dilidir. Kotlin Java diline oranla daha kolay anlaşılır ve özgün olması yönünden avantajlıdır. Java ve Android ile uyumludur. Bu yönüyle Java’nın içinde Kotlin dilini, Kotlin içinde Java dilini kullanmak mümkündür. Ya da Android Studio

içinde Java diliyle yazılmış olan kod Kotlin diline çevrilebilir. Android Studio' da Kotlin diliyle kodlama yaparken Java'nın kütüphanelerini kullanmaktadır.1965 yılından beri yazılan programlarda yazılımcıları zarara uğratan "null" sorununu ortadan kaldırılmıştır (Özel, 2018).

### 1.9.2. Java Programlama Dili

Java, Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış açık kodlu, nesneye yönelik, zeminden bağımsız, yüksek verimli, çok işlevli, yüksek seviye, yorumlanabilen bir dildir. Java'nın sık kullanılan sloganlarından biri olan "write once, run anywhere (bir defa yaz, her yerde çalıştır)" Java kodunun Java'yı destekleyen bütün platformlarda tekrar derlenmeye ihtiyacı olmadan çalışabileceğini anlatmaktadır (URL-23). Yani Java programlama dili ile yazılan kodlar gerekli ortam sağlandığında Linux, IOS, Windows, MacOS, Android gibi pek çok işletim sistemlerinde bir değişiklik yapılmadan kolayca çalışabilmektedir.

#### 1.9.2.1. Java Programlama Dili Temel Değişken Türleri

Algoritmalardaki işlemiş veya işlenememiş bilgilerin ham haline veri denir. Bilgisayarların asıl görevi verileri işlemektir. Fakat bir verinin işlenebilmesi için uygun veri tipinin seçilmesi gerekmektedir. Bilgisayara girdiğimiz her farklı değişkenin birbirlerinden ayırt edilebilmesi için farklı veri tiplerinde olduklarının tanımlanması gerekir. Böylece her değişken kendi kategorisinde saklanmış olmaktadır. Değişkenleri ilk başta tanımlayarak alacakları değer aralığı da belirlenmiş olmaktadır. Java dilindeki temel veri tipleri Tablo 5 'te verilmiştir.

Tablo 5. Java Temel Değişken türleri (Çobanoğlu,2020)

| **İlker Veri Tipi** | **Uzunluk** | **Minimum Değer** | **Maksimum Değer** | **Gömüldüğü Sınıf** | **Varsayılan Değer** |
|---|---|---|---|---|---|
| Char | 16 bit | Unicode 0 | Unicode $2^{16}$-1 | Character | '\u0000' |
| Long | 64 bit | $-2^{63}$ | $2^{63}$-1 | Long | 0l |
| İnt | 32 bit | $-2^{31}$ | $2^{31}$-1 | Integer | 0 |
| Short | 16 bit | $-2^{15}$ | $2^{15}$-1 | Short | 0 |
| Byte | 8 bit | $-2^{7}$ | $2^{7}$-1 | Byte | 0 |
| Double | 64 bit | $-3.4*10^{38}$ | $3.4*10^{38}$ | Double | 0.0d |
| Float | 32 bit | $-1.7*10^{38}$ | $1.7*10^{38}$ | Float | 0.0f |
| Boolean | 1 bit | | | Boolean | False |

Boolean değişken türü, mantık işlemlerinde "true" ve " false" değerleri ile kullanılmaktadır. true doğru anlamını taşırken false yanlış anlamını taşımaktadır. Boolean bir değişkene doğrudan aktarılabildiği gibi başka değişkene de aktarılıp karşılaştırmalar yapılabilmektedir (Çoban, 2001).

Char (harf) değişken türü, harflerin tanımlanmasında kullanılmaktadır. Harfler Java dilinde ISO Unicode kodu ile bilgisayara aktarılmaktadır. Unicode 4 hekzagonal (16 tabanlı) sayının bir araya gelmesiyle oluşmaktadır (Çoban, 2001).

Tam sayı değişken türleri, hafızada işgal ettikleri yere göre byte (8 bit), short (16 bit), int (32 bit) ve long (64 bit) adını almaktadırlar. Bir bitlik hafızaya sadece iki rakamın (0 veya 1) yazılırsa, sekiz bitlik tamsayı değişkenine 256 sayı yazılır. Bitlerden bazıları pozitif bazıları negatif olduğu için byte değişkenleri -128 ile 127 sınır değerleri arasındadır. Bu da toplamda 256 sayı değerini vermektedir. Eğer bir tamsayı değişkenin sadece artı değerleri kullanılmak istenirse "unsigned" terimi kullanılmalıdır. Böylece değişkenlerin sadece pozitif olanları dikkate alınacaktır. Tamsayı değişken türleri içerisinde en sıklıkla kullanılan ise "int" türüdür (Çoban, 2001).

Gerçek sayı değişken türleri (float, double, long double), 0 ve 1 bitleri kullanılarak oluşturulmaktadır. Gerçek sayıların tamsayıdan küçük kısımları 2'li tabanda eksi üstler kabul edilerek oluşturulmaktadır. Gerçek sayı değişkende yeterli hassasiyeti sağlayabilmek için genelde 64 bit uzunluğundaki double değişken türü kullanılır. Java'daki matematik kütüphaneleri de double değişken türü için tanımlanmıştır (Çoban, 2001).

#### 1.9.2.2. Java Nesnesi Olarak Tanımlanmış Temel Değişkenler

Java nesne temelli bir dildir. Java'daki tüm değişken türlerinin nesne kökenli eşdeğerleri mevcuttur. Bunun yanında temel değişken türleri arasında yer almayan String gibi sadece nesne kökenli olarak tanımlanmış değişken türleri de mevcuttur (Çoban, 2001).

Java dilinde değişkenlere tanımlanan isimlerin anlamının ve boyutunun hiçbir önemi yoktur, yani istenilen isim anlamına bakılmaksızın verilebilmektedir. Lakin kodlamayı yazan kişi değişkene verdiği ismi değişkenle bütünleştirip ne anlama geldiğini kendisinin bilmesi yeterli olacaktır. Yalnız Java'da değişkenlere verilen isimlerde büyük küçük harf kullanımı önem arz etmektedir. Aynı zamanda değişkenlerde adlandırma yapılırken boşluk kullanılamaz. Değişkenler harfler ve ($) dışındaki hiçbir karakterle ve rakamlarla başlayamaz. İlk karakter yazıldıktan sonra harf, rakam, alt çizgi ve dolar simgelerinden istenilenlerle devam edilebilmektedir. Örnek vermek gerekirse: b20_ veya $_h07_ gibi yazılabilir ve tanımlamada herhangi bir uzunluk sınırlaması yoktur. Bazen değişken tanımlarda birden fazla kelime kullanılması gerekebilir. Bu gibi durumlarda iki kelime alt çizgi (_) ile birbirlerine bağlanabildiği gibi devenotasyon denilen yöntemde kullanılabilir. Bu yöntemde ise iki kelimelik bir tanımlamada ilk kelimeyi yazıp ikinci kelimenin ilk harfi büyük harfle başlatıp bitişik bir şekilde yazım söz konusu olabilir. Örneğin: uzaktan_algılama veya uzaktanAlgılama şeklinde iki türden yazılabilmektedir (URL-24).

##### 1.9.2.2.1. String Nesne Tipi Değişkeni

String değişkeni metinleri içeren veri tipi oluşturmada kullanılmaktadır. Burada metin yalnızca alfabenin harf, rakam ve işaretleri değil, char tiplerinden de oluşabilecek her diziyi kapsamaktadır. String değişkenlerine atanan metinler çift tırnak (" ") içine yazılırlar. "b" , "0" , "Trabzon" birer String'tir. String aynı zamanda, String a=new String("Trabzon"); şeklinde de tanımlanabilmektedir. Java dilinde String değeri birden çok satıra yazılamaz. Metin satıra sığmadığı zaman ya da metni satırlara bölmek gerektiğinde, yeni satıra geçmek için \n karakteri kullanılır. Örneğin, String deneme= "Adınız nedir? \n Kaç yaşındasınız? " şeklinde oluşturulur. Birden fazla String + işaretiyle bir araya getirilebilirler. Örneğin, String a= "uzaktan"; String b="algılama"; diye iki değişken oluşturup bunları d adında bir değişkene şu şekilde atarsak; String d; d=a+b; d'nin değeri "uzaktan algılama" olarak

gözükür. Oluşturulan String değişkenlerini ekrana yazdırmak için System.out.println ( ); metodu kullanılmaktadır.

#### 1.9.2.2.2. Integer Nesne Tipi Değişkeni

Integer değişkeni basit tam sayıları içeren veri tipi oluşturmak ve bu tam sayıları işlemek için kullanılmaktadır. Integer değişkeni; Integer i; i= new Integer(61); veya Integer i= new Integer(61); şeklinde tanımlanmaktadır. Integer değerini int değerine (temel değişken) çevirmek için int ve Integer değerleri tanımlandıktan sonra Integer.IntegerValue() metodu kullanılır. String değerini int değerine çevirmek içinse String ve Integer değerleri tanımlandıktan sonra Integer.parseInt() metodu kullanılır.

#### 1.9.2.2.3. Double Nesne Tipi Değişkeni

Double değişkeni hesaplamalarda, karmaşık işlemlerde kullanılmaktadır. Double tanımlanması Integer ile aynı şekilde tanımlanmaktadır. Yani, double i; i= new Double(2020.07) veya double i= new Double(2020.07); şeklindedir. Double değerini double değerine (temel değişken) çevirebilmektedir. Bunun için ilk başta double ve Double değerleri tanımlandıktan sonra Double.doubleValue() metodu kullanılır.

#### 1.9.2.2.4. Diğer Nesne Temelli Değişken Türleri

Diğer nesne temelli değişken türleri, Object, Long, Float, Boolean, Character, Vector gibi türlerdir. BigDecimal ve BigInteger nesne tipi türlerin hassasiyeti kullanıcı tarafından belirlenmektedir. Bütün nesne tipi değişkenlerin alt metotları mevcuttur. Bu metotlar kullanılarak istenilen işlemler gerçekleştirilebilir. Lakin bu tür değişkenler çok fazla yer kaplamaktadır. Bu yüzden bunların kullanılması çok tercih edilmemelidir (Çoban, 2001).

### 1.9.2.3. Java'da Operatörler

Operatörler, diğer programlama dillerinde olduğu gibi Java'da da değişkenler ve nesnelerin üzerinde çeşitli değişiklikler yapmak için kullanılan elemanlara denilmektedir. Java'da operatörler yaptıkları işe göre 5 sınıfta toplanırlar. Bunlar; Aritmetik Operatörler, Atama Operatörleri, İlişkisel Operatörler, Binary(İkidelik) Operatörleri, Mantıksal Operatörlerdir (Tablo 6).

Tablo 6. Java'da Operatörler

| **Aritmetik Operatörler** | |
|---|---|
| **Operatör** | **Anlamı** |
| + | Toplama |
| - | Çıkarma |
| * | Çarpma |
| / | Bölme |
| % | Modülüs |
| ++ | Arttırma |
| -- | Azaltma |
| **Atama Operatörleri** | |
| **Operatör** | **Anlamı** |
| = | Basit atama operatörüdür. Operatörün sağındaki değişkenin değeri solundaki değere atanır. |
| += | Operatörün sağındaki değeri solundaki değerle toplayıp tekrar solundaki değere atar. |
| -= | Operatörün sağındaki değeri solundaki değerden çıkartıp tekrar solundaki değere atar. |
| *= | Operatörün sağındaki değeri solundaki değerle çarpıp tekrar solundaki değere atar. |
| /= | Operatörün sağındaki değeri solundaki değere bölüp tekrar solundaki değere atar. |
| %= | Operatörün sağındaki değerin solundakine göre modunu alıp tekrar solundaki değere atar. |

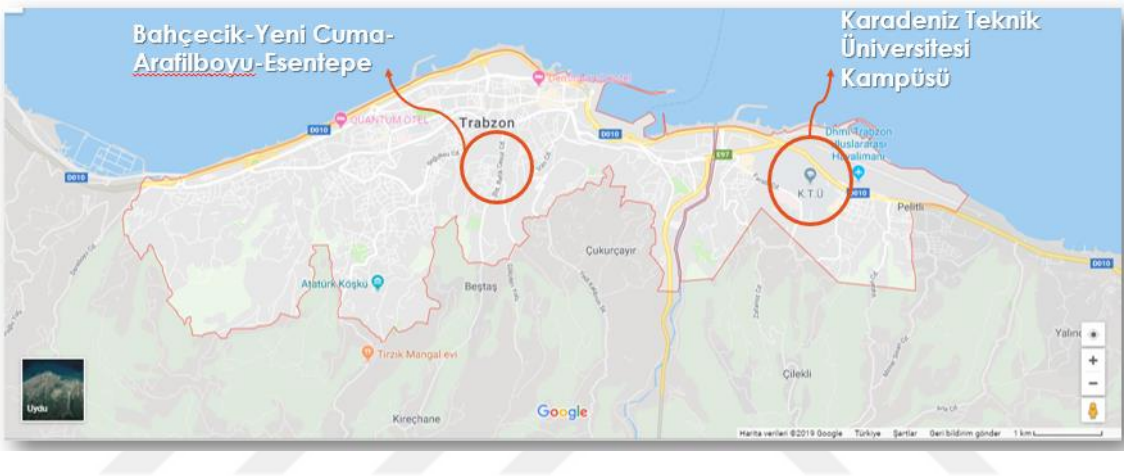
Tablo 6'nın devamı

| **İlişkisel Operatörler** | |
|---|---|
| **Operatör** | **Anlamı** |
| == | Eşit |
| != | Eşit değil |
| < | Küçüktür |
| > | Büyüktür |
| <= | Küçük ya da eşittir |
| >= | Büyük ya da eşittir |
| **Binary(İkidelik) Operatörleri** | |
| **Operatör** | **Anlamı** |
| & | Binary ve( iki tarafta da bulunan biti sonuca kopyalar.) |
| \| | Binary veya ( iki taraftan birinde bulunan biti sonuca kopyalar.) |
| ^ | Binary XOR (bir tarafta bulunup diğer tarafta bulunmayan biti sonuca kopyalar.) |
| ~ | Binary tümler(Sayının bütün 1'lerini sıfır, bütün sıfırlarını 1 yapar.) |
| << | Sola kay |
| >> | Sağa kay |
| >>> | Sağa kay ve sıfırla |
| <<= | Sola kay ve ata |
| >>= | Sağa kay ve ata |
| >>>= | Sağa kay-sıfırla ve ata |
| **Mantıksal Operatörler** | |
| **Operatör** | **Anlamı** |
| && | Mantıksal And (Eğer iki taraftaki değişken de false değilse sonuç true döner.) |
| \|\| | Mantıksal Or (Eğer operatörün iki tarafındaki değişkenlerden birisi true ise sonuç true döner.) |
| ! | Mantıksal Not (Operasyonun mantıksal cevabını terse çevirir. Eğer true değer dönüyorsa, sonucu false değere çevirir.) |

## 2. YAPILAN ÇALIŞMALAR

### 2.1. Çalışma Bölgesi

Bu tez çalışması Trabzon ili için yapılmıştır. Trabzon'da ise kasis yoğunluğunun fazla olduğu 5 farklı bölge tercih edilmiştir. Bunlar: Bahçecik, Yenicuma, Arafilboyu, Esentepe ve Karadeniz Teknik Üniversitesi Kampüsüdür (Şekil 30).




Şekil 30. Çalışma bölgesi

Bu güzergâhlar üzerindeki kasislerin konumlarını tespit etmek için, öncelikle arazi keşfi yapılmıştır. Daha sonra kasislerin fotoğraflarını çekmek için Samsung S7 cep telefonu kullanılmıştır. Fotoğraf çekmeden önce telefonda konum özelliği açılmıştır. Böylece fotoğrafların ayrıntı kısmında bu görsellerle birlikte kasislerin konum bilgileri de eklenmiş olmaktadır. Bütün bu işlemler 5 farklı bölgenin tamamında uygulanmış olup güzergâhlar üzerindeki kasislerin tespit edilme işlemi tamamlanmıştır (Tablo 7).

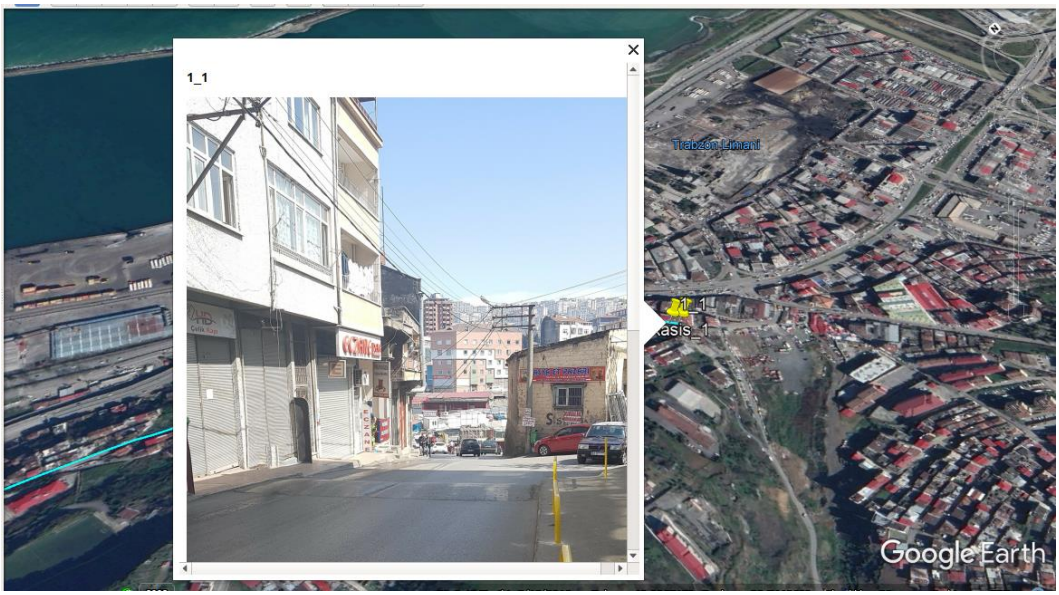
Tablo 7. Kasislerin koordinatları

| **Bahçeçik-Yenicuma- Arafilboyu- Esentepe** | | |
|---|---|---|
| | **Enlem** | **Boylam** |
| **Kasis_1** | 40.997825 | 39.745689 |
| **Kasis_2** | 40.999733 | 39.736747 |
| **Kasis_3** | 40.997406 | 39.722136 |
| **Kasis_4** | 40.996619 | 39.721983 |
| **Kasis_5** | 40.993722 | 39.721633 |
| **Kasis_6** | 40.991261 | 39.721061 |
| **Kasis_7** | 40.994404 | 39.720036 |
| **Kasis_8** | 40.996300 | 39.715781 |
| **Kasis_9** | 40.995544 | 39.713067 |
| **Kasis_10** | 40.993733 | 39.710960 |
| **KTÜ Giriş** | | |
| | **Enlem** | **Boylam** |
| **Giriş_Kasis1** | 40.998032 | 39.764234 |
| **Giriş_Kasis2** | 40.997852 | 39.766635 |
| **Giriş_Kasis3** | 40.997299 | 39.768506 |
| **Giriş_Kasis4** | 40.994939 | 39.767573 |
| **Giriş_Kasis5** | 40.995783 | 39.767427 |
| **Giriş_Kasis6** | 40.997534 | 39.769611 |
| **Giriş_Kasis7** | 40.997429 | 39.771231 |
| **Giriş_Kasis8** | 40.997413 | 39.771725 |
| **Giriş_Kasis9** | 40.995646 | 39.773672 |
| **Giriş_Kasis10** | 40.994664 | 39.774668 |
| **Giriş_Kasis11** | 40.993874 | 39.775471 |
| **KTÜ Çıkış** | | |
| | **Enlem** | **Boylam** |
| **Çıkış_Kasis1** | 40.993277 | 39.776024 |
| **Çıkış_Kasis2** | 40.993953 | 39.775556 |
| **Çıkış_Kasis3** | 40.995727 | 39.773720 |

Tablo 7’nin devamı

| | | |
|---|---|---|
| **Çıkış_Kasis4** | 40.997220 | 39.772257 |
| **Çıkış_Kasis5** | 40.997534 | 39.771730 |
| **Çıkış_Kasis6** | 40.997587 | 39.770499 |
| **Çıkış_Kasis7** | 40.997642 | 39.769650 |
| **Çıkış_Kasis8** | 40.997228 | 39.768362 |
| **Çıkış_Kasis9** | 40.995781 | 39.767413 |
| **Çıkış_Kasis10** | 40.994856 | 39.766923 |
| **Çıkış_Kasis11** | 40.997938 | 39.766670 |
| **Çıkış_Kasis12** | 40.998136 | 39.764456 |

Kasislerin konumlarını bir harita üzerinde güzergâh olarak çizmek için Google Earth programı kullanılmıştır. Google Earth’de “Yer İşareti Ekle” ikonuyla kasislerin enlem ve boylam değerleri girilmiştir. Kasislere numaralandırmalarla isim verilmiştir. Ayrıca her kasisin fotoğrafı yer işaretçisinin içine eklenmiştir (Şekil 31).




Şekil 31. Kasis fotoğrafı eklenmiş yer işaretçisi

Bütün kasislerin konumları ve fotoğrafları Google Earth üzerine eklendikten sonra oluşan güzergâhı harita üzerinde çizmek için “Yol Ekle” butonuna tıklanmıştır. Daha sonra haritadaki yer işaretçileri birleştirilerek yol güzergâhları Şekil 32 ve Şekil 33’ deki gibi çizilmiştir.


Şekil 32. Bahçecik-Yenicuma-Arafilboyu-Esentepe bölgesindeki kasisler


Şekil 33. Karadeniz Teknik Üniversite’sindeki kasisler

## 2.2. Kullanılan Yazılımlar ve Modüller

Bu tez çalışmasında kodların yazımında Java programlama dili kullanılmıştır. Java dilinde yazılan programların derlenip çalıştırılması için, içinde çok sayıda kütüphanenin bulunduğu Java Development Kit (JDK) program paketi çalışmayı yürüttüğümüz Windows 10, 64 bit işletim sistemli bilgisayara yüklenmiştir. Oracle tarafından geliştirilen JDK, Oracle'ın kendi internet sitesinden ücretsiz olarak indirilmiştir.

Daha sonra Google`ın Android işletim sistemi üzerinde uygulama geliştirilmesi için oluşturduğu ve kodların asıl yazılacağı program olan Android Studio'nun kurulumu gerçekleştirilmiştir. Kurulum Android Studio'nun resmi internet sitesinden ücretsiz olarak indirilerek gerçekleştirilmiştir.

Kod yazma ve düzenleme ortamı için birçok IDE bulunmaktadır. Fakat Android tabanlı projelerde "Android SDK (Software Development Kit)" daha kolay kod yazma imkânı sunmaktadır. Android SDK, Android Studio programı içerisinde programla bağlantılı bir şekilde bulunan bir ortamdır. SDK, kullanılan Android' in sürümüne uygun gerekli kod kütüphanelerini, sistemin altyapısını, uygulama için gerekli araçları, Android sürüm kodlarını ve hazır kod şablonlarını barındırmaktadır (Taç, 2013). Bunun için, bu tez çalışmasında da Android Studio programında bulunan ve indirilmesi bir seçenek olarak sunulan SDK indirilip kurulmuştur.

Geliştirilen uygulamalarda kod hatalarını ve uygulamanın kullanılabilirliği test etmek için iki yöntem kullanılmaktadır. İlk yöntem Android telefonlar gibi davranan "Android Emülatör" kullanmaktır. Android Studio programına emülatörü yüklemek için SDK yöneticisinin içinde SDK araçları sekmesinde "Android Emulator" bileşeni seçilmektedir. Android telefonun benzer özelliklerini yansıtmak istediğimiz emülatörler için "AVD (Android Sanal Cihazı)" kullanılmaktadır. İstenilen nitelikte emülatörü üretip, bunları gerekli durumlarda yönetmek için "AVD Manager" sekmesine gidilmesi gerekmektedir. Burada emülatörün ekran boyutu, yan veya dik olarak kullanılma özelliği, emülatörün içinde yüklü olacak Android Sürümü ve daha birçok özellik seçilebilmektedir. Aynı zamanda birden çok farklı özellikte ve boyutta emülatörler de oluşturup programa eklenebilmektedir. Bu tez çalışmasının ilk başlarında emülatör kullanılmış olup, emülatörün programı çok yavaşlatması ve emülatörde yapılacak işlemlerin zaman almasından dolayı emülatör kullanılmaktan vazgeçilmiştir.

Yazılan uygulamaların test edilmesi için kullanılabilecek ikinci bir yöntem ise uygulama geliştiricinin kendi telefonunu programa entegre etmesidir. Bunun için uygulama geliştiricisinin sahip olduğu telefonun Android özellikli bir telefon olması ve telefonda "Geliştirici Seçenekleri" modülünün açık olması gerekmektedir. Normal şartlarda Android telefonlarda "Geliştirici Seçenekleri" gizli olarak telefonda yer almaktadır. Bunu görünür yapmak için telefonun ayarlar kısmına gelip, telefon hakkında bölümde "Yapım Numarası" kısmına yaklaşık 7 veya 8 defa art arda basmak gerekmektedir. Bu işlemin ardından "Geliştirici Seçenekleri" ayarlar kısmında yerini almaktadır. Android Studio uygulamasına girip, cep telefonunun kablosuyla bilgisayara bağlanmalıdır. Ardından cep telefonunda "Geliştirici Seçenekleri" sekmesinde "USB Hata Ayıklama" bölümü seçilip bilgisayarda ve cep telefonunda çıkan şifrenin eşleşmesi sağlanmalıdır. Eşleşme sonrasında ise artık Android Studio programındaki yazılan her uygulamayı çalıştırıldığında uygulama geliştiricinin kendi telefonundan da uygulama kontrol edebilir pozisyona gelmektedir. Yazılan bu tezde, yukarda anlatıldığı gibi adımlar sırasıyla yapılarak Samsung S7 cep telefonu Android Studio programına entegre edilip, uygulama sırasındaki gerekli kontroller yapılmıştır.

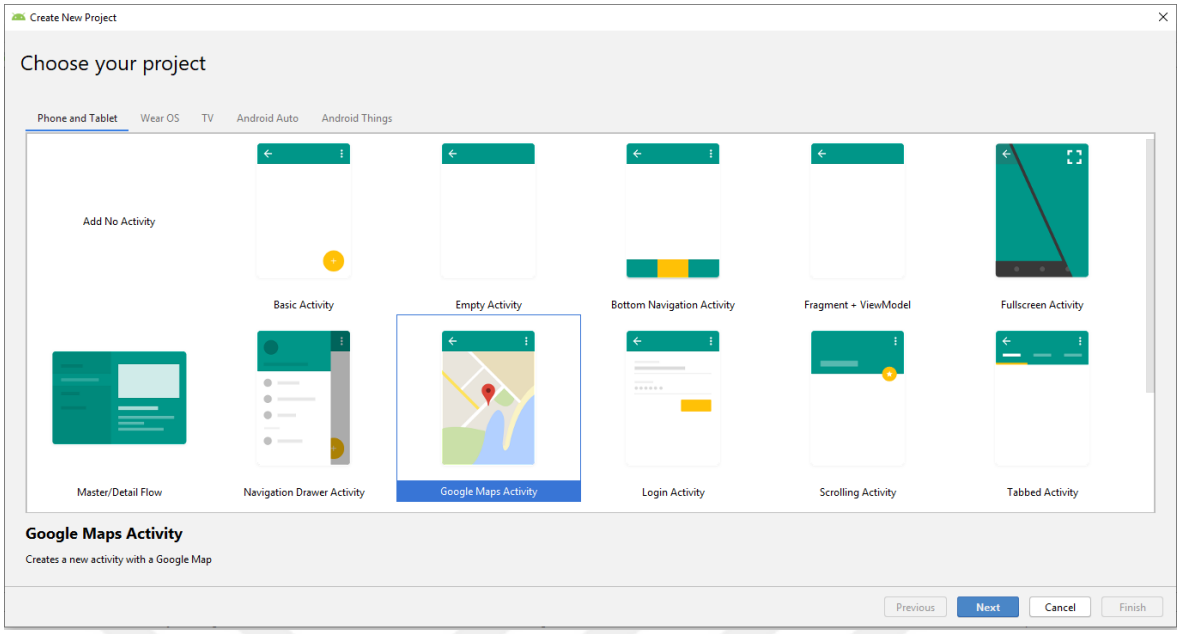
### 2.3. Android Uygulama Geliştirme

Android bir uygulama geliştirirken, iş akışının düzgün bir şekilde yapılması bu sürecin daha planlı ilerlemesini ve uygulama tasarısının daha iyi olmasını sağlamaktadır. Bir Android proje oluştururken genel olarak: programın kurulması ve proje oluşturma, uygulama kodunu yazma, uygulamayı inşa edip çalıştırma, test edip hata ayıklaması yapma ve uygulamayı yayınlama gibi aşamalardan oluşmaktadır. Bu aşamalar Tablo 8'de gösterilmiştir ( Android Guides,2020).

Tablo 8. Android Uygulama Geliştirme Şeması

| | | |
|---|---|---|
| **Setup** | | |
| Set up your environment | Create your project | |
| **Write** | | |
| Write code | Add assests | |
| **Build & Run** | | |
| Connect to a device or emulator | Customize your build | |
| **Iterate** | | |
| Debug | Profile | Test |
| **Publish** | | |
| Version | Sign | |

Android Studio ‘da yeni bir proje oluşturmak için öncelikle proje tasarımını belirlemek gerekmektedir. Bunun için programın içinde yer alan hazır Activity şablonlarından projenin ana yapısına uygun olan tasarım kullanılabilir ya da boş bir Activity seçilerek istenilen şekilde tasarım yapılabilir. Bu tez çalışmasında başlangıç olarak Android Studio’nun bir şablonu olan “Google Maps Activity” kullanılarak geliştirmeler yapılmıştır (Şekil 34).




Şekil 34. Google Maps Activity

Uygulamaya uygun şablon seçildikten sonra gerekli kod blokları yazılmaya başlanmıştır.

Activity'nin modüler bir bölümü olan Fragment'ın kendi yaşam döngüsü bulunmaktadır. Kendi giriş olaylarını (input events) kendisi almakta ve Activity çalışmaya devam ederken ekleme ve çıkartmalar yapılabilmektedir (URL-25). Bu tez çalışmasında kullanılan; Map Fragment, onCreate komutu, onMapReady komutu, LocationManager komutu ve User Permissions komutları ayrı başlık altında incelenecektir.

### 2.3.1. Map Fragment

Map Fragment, uygulama içerisine Google Maps' ten elde edilen harita bileşeninin eklenip activity üzerinden gerekli kodların devamının sağlanabilmesi için view tabanlı bir nesnedir (Saraç, 2019). Uygulamada activity bölümüne eklenen "mapFragment.getMapAsync(OnMapReadyCallback: this)" sınıfı ile harita görünümü başlatılmaktadır. Bu sınıfın eklenmesiyle birlikte uygulama açıldığında haritanın ekranda gözükebilmesi için Google Developer Console (Google Geliştirici Hesabı) giriş yaparak Google Maps API etkinleştirmeli ve API anahtarı alınması gerekmektedir.

#### 2.3.1.1. Google Maps API

Google Maps ve Google Earth internet verisi kullanılarak harita sunumu gerçekleştiren platformlardır. Google Maps Silindirik Projeksiyon (Tranverse Mercator) sistemini ve WGS84 elipsoidini kullanan Google tarafından geliştirilmiş harita hizmeti sunan sistemdir. Belirli bir altlık üzerinde yapılan analog haritaların dijital versiyonlarıdır. Yerküreyle ilgili bilgileri üstten görüntüsü alınmış bir şekilde ölçekli olarak bir düzlem üzerinde sunmaktadır (Bildirici,2015).

Google Maps API (Application Programming Interface), Google'ın Android ortamlar için geliştirdiği Google Maps 'in bütün harita altlıklarını ve geokodlarını barındıran bir ara yüze denilmektedir. Google Maps API'yi kullanabilmek için Google'ın uygulama geliştiriciler için sağladığı bir anahtar şifreye ihtiyaç duyulmaktadır (Saraç,2019). Google Maps API Key alınmadığı takdirde uygulamada harita açılmayacak ve uygulama açılmadan kapanacaktır. Bu nedenle bu adım çok önemlidir. Google Maps API almak için "Console Developers Google" sayfasına bir Google hesabı ile giriş yapılmalıdır. Daha sonra "Create Project" seçeneği ile proje oluşturulmalıdır. "API'ler ve Hizmetler" kısmında "Kitaplık" bölümünden projemize uygun olan kitaplığın seçeneğin aktifleştirilmesi gerekmektedir. Bu tez çalışması için aktifleştirilmesi gereken seçenek " Maps SDK for Android'dir. Bu işlemden sonra kimlik bilgilerini oluşturarak sistemin bir "API Key" vermesi sağlamış olur. Burada eğer ki aktifleştirilen seçenek yanlışsa uygulamada yine harita açılmayacaktır. Alınan Google Maps anahtarı Android Studio programında "res" dosyası altında "values" bölümünün içindeki "google_maps_api.xml" içine aşağıdaki şekilde eklenmiştir.

<string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">AIzaSyCVnpLD5MhVTG-6zqnZ43ievFxlATDUaDo</string>

API Key alırken sisteme girilen bilgileri kullanarak ihtiyaç halinde uygulama geliştiricisine ulaşılabilmektedir. Bu anahtar ile harita uygulanması izlenmekte günlük kullanım sınırları takip edilmektedir. Gerekli görüldüğü durumlarda veya kullanım sınırı aşıldığı durumlarda ek kullanım satın almaları olabilmektedir (Bildirici,2015).

Uygulamanın projesine API anahtarının eklenmesiyle projede Google Maps 'in bütün özellikleri kullanılabilir duruma gelmiş olur. Proje bu haliyle çalıştırıldığında uygulama telefonda Şekil 35'deki gibi gözükmektedir.


Şekil 35. Google Maps görünümü

### 2.3.2. onCreate Komutu

Activity ön planda yer alan sayfa yapısına denilmektedir. Activity başlatıldığında ilk çalışan metot "onCreate" metodudur. Bu metot çalışmadan activity başlatılamaz. Eğer ekran ilk oluştuğunda layout dosyasından ekran tasarımı yüklenmek isteniyorsa o zaman "setContentView" metodu çalıştırılarak layout tasarımı kaynak dosyadan alınmalıdır. Ekran oluşturulurken tanımlanması gereken başka değişkenler ve aksiyonlar varsa, onlar da "onCreate" metodu içinde gerçekleştirilebilmektedirler (Şekil 36).

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified when the map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync( onMapReadyCallback: this);

}
```

Şekil 36. onCreate metodu

### 2.3.3. onMapReady Komutu

Harita hazır olduğunda “onMapReady” metodu devreye girmektedir. Bu kod başlatıldıktan sonra eğer harita null bir sonuç döndürmüyorsa istenilen harita görseli elde edilmektedir. Ayrıca burada haritaların istenilen özelliklerde şekillendirilmesi yapılabilmektedir. Bu tez çalışmasında da onMapReady altında ilk olarak Google’ın kullanıcının anlık konumunu gösteren mavi ikonunun tanımlanması gerçekleştirilmiştir (Şekil 37). Bunun için kod bloğuna “mMap.setMyLocationEnabled(true)” yazılmıştır. Burada mMap diye tanımlanan obje GoogleMap’tir. Yani sonuç olarak uygulama haritası üzerine çoğu navigasyon uygulamalarından aşina olduğumuz kullanıcının kendi konumunu görmesine yarayan mavi ikon eklenmiş olur. Ayrıca bununla birlikte uygulamanın sağ üst köşesine de kullanıcının haritada gezindikten sonra kendi konumuna tek hamleyle geri dönebilmesini sağlamak için gerekli kontrol komutu eklenmiş olur.


Şekil 37. Anlık konum gösteren mavi ikon

İkinci olarak harita üzerinde rahatça yakınlaşıp uzaklaşmak için zoom in ve zoom out özelliklerini içeren “mMap.getUiSettings().setZoomControlsEnabled(true)” kodu onMapReady altında tanımlanmıştır (Şekil 38).


Şekil 38. Yakınlaştırma ve uzaklaştırma ikonları

Android Studio programında “LatLng” enlem ve boylam değerlerini içeren bir yapı oluşturmaktadır. “Location” sınıfı ise enlem ve boylam değerlerini içermektedir. Bu tez çalışmasında 33 tane tane kasis tespiti yapıldığından bunları programa yüklemek için bir liste yapılması gerekmektedir. Bunun için “ArrayList< >” yapısı kullanılmaktadır. ArrayList oluşturulurken listenin hangi tür veri yapılarını içerdiği belirtilmesi gerekmektedir. Bu uygulamada konumla ilgilenildiği için veri tipi “LatLng” belirtilip “locations” adında bir şekilde liste tanımı “ ArrayList<LatLng> locations = new ArrayList<>();” şeklinde yapılmıştır.

Daha sonra tanımlanan bu listeye kasislerin enlem ve boylam değerlerini girmek için “locations.add(new LatLng( V, V1))” kod bloğu eklenmiştir. Burada V ve V1 değerlerine kasislere ait enlem ve boylam değerleri girilmiştir. Bu işlem bütün kasisler için yapılıp kasislerin konum değerleri listeye eklenmiştir. Uygulamada harita üzerine kasislerin konumları bir işaretçi yardımıyla eklenerek harita açıldığında hangi bölgelerde kasislerin olduğu rahatça gözükmesi sağlanıştır. İşaretçiler Android Studio programında “Marker” ve “GoogleMap.addMarker(markerOptions)” yöntemiyle eklenmektedir. Yapılan uygulamada birden fazla işaretçi haritaya ekleneceği için bir döngü kullanarak tek seferde 33 tane noktanın ekrana gelmesi sağlamıştır. Bunun için kullanılan kodlar aşağıda verilmiştir:

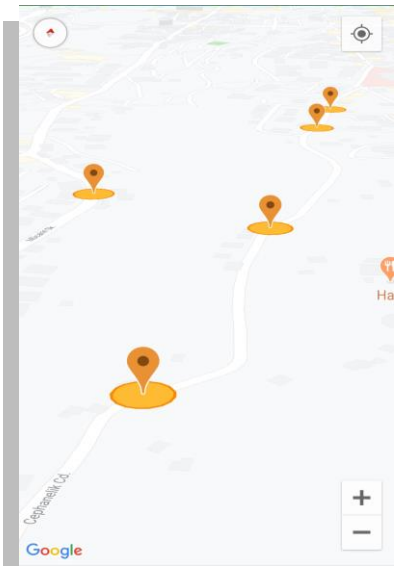
```
for (LatLng location : locations) {
   marker = mMap.addMarker(new MarkerOptions()
         .position(location)
         .icon(BitmapDescriptorFactory.defaultMarker( BitmapDescriptorFactory
```

```
        .HUE_OR_ORANGE)));
}
```

Kullanılan işaretçilerin harita üzerindeki bulunması gereken konumu göstermek amacıyla "position()" kodu eklenmiştir. "BitmapDescriptor" ile işaretçinin görüntüsünün renginin özelleştirilmesi sağlanmıştır. "BitmapDescriptorFactory" ile önceden tanımlanmış bir renk kümesi kullanabilmekte ve işaretçinin rengi ayarlanabilmektedir. Burada kullanılan " HUE_OR_*ORANGE"* ile turuncu renkte bir marker elde edilmiştir.

Uygulama kullanılırken kasise yaklaşmadan 20 metre öncesinde bir uyarı verilmesi istenmektedir. Bunu için harita üzerinde kasislerin bulunduğu marker etrafına 20 metrelik bir alanı gösterecek dairelerin çizilmesi planlanmıştır (Şekil 39). Daireleri çizmek için, "CircleOptions" tanımlanması yapılmıştır. CircleOptions'a ait "center()" ile oluşacak dairenin merkezi belirlenmiştir. Bu çalışmada daire merkezleri kasislere ait işaretçilerin location değerleri girilmiştir. "radius()" ile dairede istenilen yarıçap değeri yazılmıştır. "strokeColor" ile kontür rengi belirlenirken "fillColor" ile dolgu rengi belirlenmiş olup "strokeWidth" ile kontür kalınlığı ayarlanmıştır. Daha sonra özellikleri ayarlanan dairelerin marker etrafına eklenmesi için ".addCircle(CircleOptions)" kodu eklenmiştir. Daire çizme işlemini bütün kasislerde tek tek yapmamak için "for" döngüsü kullanılmıştır. Tezde kullanılan kod bloğu aşağıda verilmiştir:

```
for (LatLng location : locations) {
   CircleOptions = new CircleOptions();
   circleOptions.center(location);
   circleOptions.radius(15);
   circleOptions.strokeColor(ContextCompat.getColor(this,
android.R.color.holo_orange_dark));
   circleOptions.fillColor(ContextCompat.getColor(this,
android.R.color.holo_orange_light));
   circleOptions.strokeWidth(6);
   mMap.addCircle(circleOptions);
}
```




Şekil 39. Kasisler etrafına çizilen daireler

#### 2.3.3.1. Info Window

Info window (bilgi penceresi), işaretçilerin içerisine açılır kapanır pencere oluşturmaktadır. Bu pencere içerisine metin veya görseller eklenebilmektedir. Kullanıcı bu işaretçilere dokunarak bilgi penceresinin açılmasını sağlamaktadır. Tek seferde yalnızca bir bilgi penceresi açılmaktadır. Yani ikinci bir işaretçiye dokunulduğunda bir önceki bilgi penceresi kapatılıp yeni bilgi penceresi ancak bu şekilde açılabilmektedir.

Bu tez çalışmasında Info window ile ilgili tüm çalışmalar “onMapReady” metodu altında yapılmıştır. Bir bilgi penceresi oluştururken “InfoWindowAdapter” çağrılmaktadır. Bununla birlikte “getInfoWindow( Marker )” ve “getInfoContents( Marker )” olmak üzere iki yöntem bu sırayla gelir. Bunlardan ilki tüm bilgi penceresi için kullanılacak bir görünüm sağlarken, ikincisi sadece pencerenin içeriğinin özelleştirilmesini sağlamaktadır. Bilgi penceresinin içeriğini ve tasarımını özelleştirirken ilk olarak tasarım kısmı yapılır. Bunun için “info_window.xml” adında bir tasarım dosyası olan layout oluşturulur ve bu dosya içerisinde istenilen tasarım için gerekli özellikler eklenmektedir. “info_window.xml” layout dosyasına eklenen kodlar aşağıda verilmiştir:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="horizontal"
```

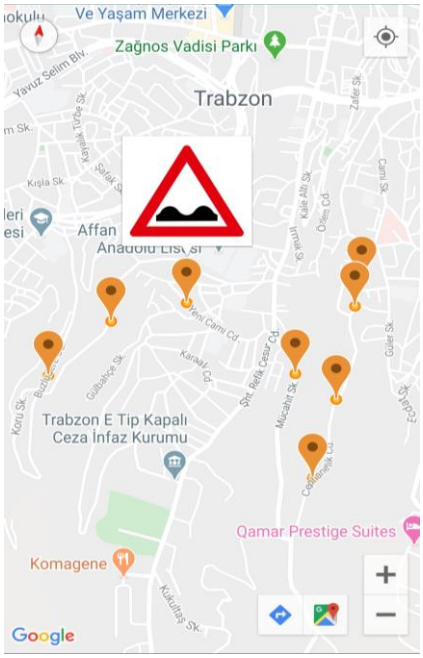
```
   android:layout_width="wrap_content"
   android:layout_height="wrap_content">

   <ImageView
      android:id="@+id/imageView"
      android:layout_width="100dp"
      android:layout_height="100dp"/>

   <LinearLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:orientation="vertical" />
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/locality"
      android:textSize="20sp">

   </TextView>

</LinearLayout>
```

Daha sonra uygulamanın asıl kodlarını yazdığımız sınıf olan “MapsActivity.java” sayfasına gelip “InfoWindowAdapter” altındaki “getInfoContents( Marker )” içerisine eklenmek istenen içeriklerin tanımlanması aşağıdaki gibi yapılmıştır:

```
mMap.setInfoWindowAdapter(new GoogleMap.InfoWindowAdapter() {
   @Override
   public View getInfoWindow(Marker marker) {

      return null;
   }

   @Override
   public View getInfoContents(Marker marker) {
```

```
        View v= getLayoutInflater().inflate(R.layout.info_window,null);
        ImageView r =v.findViewById(R.id.imageView);
        r.setImageResource(R.drawable.kasis);

        return v;
    }
});
```

Bu kodda dikkat edilmesi gereken View ile tanımlanan “v” isminin return kodunun yanında yazan isimle aynı olmasıdır. Aksi halde kod çalışacaktır fakat görsel ve bilgi penceresi gözükmeyecektir. Eklenmek istenen görselin “ImageView” ile tanımlanması yapılıp “info_window.xml” layout dosyası içerisindeki “id” değeri ile buraya gelmesi sağlanmıştır. Daha sonra eklenmek istenen görsel Android Studio programı içinde “res” dosyası altında “drawable” sekmesi içine kopyalanmıştır. Bu görselin buradan çağrılması için “setImageResource” kodu kullanılmıştır. Yapılan bu tezde bilgi penceresi içerisine kasis uyarı levhası görseli eklenmiştir. Böylece harita üzerindeki işaretçilerin ne anlama geldiğini anlamak için işaretçi üzerine basılınca bu bilgi penceresinin açılması sağlanmıştır (Şekil 40).



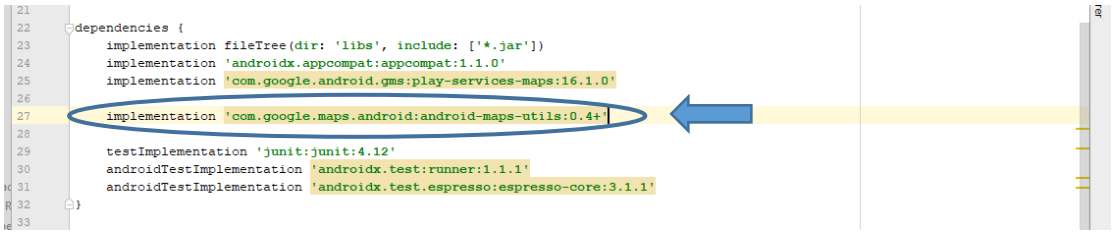

Şekil 40. Info window görüntüsü

### 2.3.3.2. Location Manager

LocationManager, kullanıcının geçerli konumunu alarak ve konum değiştikçe düzenli olarak yeni konum bilgilerini tutmaya yarmaktadır. “locationListener” ise konum güncellendikçe ne yapılması gerektiğini dinleyen bir koddur. Bu kod altında yapılmak istenenler yazılmaktadır.

Bu tez çalışmasında ”locationListener” altında ilk olarak anlık konum güncellendikçe, kullanıcının anlık konumunun alınması için bir “latLng” nesnesi oluşturulmuştur. Oluşturulan bu nesne kullanılarak konum bilgilerinin otomatik alınıp tutulmasını sağlayan “ latLng = new LatLng(location.getLatitude(),location.getLongitude());” kod bloğu eklenmiştir. Konum bilgisi, mobil cihazlarda bulunan GNSS alıcıları aracılığıyla alınmaktadır. Burada göz önünde bulundurulması gereken husus bazen GNSS alıcısından veri alınamamaktadır. Bundan ötürü konumsal hatalar, konumda birkaç metre yanılmalar olabilmektedir. Ayrıca her mobil cihaz farklı frekanslarda değerleri aldığı için konum hassasiyeti her mobil cihazda da farklılık gösterebilmektedir. Bu da yapılan uygulamanın hassasiyetini etkilemekte çıkan sonuçta yanılmaların olmasına neden olmaktadır.

Daha sonra anlık konum ile kasislerin bulunduğu konum arasındaki mesafe hesabının otomatik olarak yapılması sağlamak için “computeDistanceBetween” sınıfı kullanılmıştır. Bu sınıf iki Latitude ve Longitude (LatLng) arasındaki mesafeyi metre cinsinden tanımlamaya yaramaktadır. Bu sınıfı kullanabilmek için gerekli kütüphane build.gradle dosyası içerisine eklenmiştir (Şekil 41).




Şekil 41. Eklenen kütüphane

Eklenen kütüphane ile birlikte bütün kasislerle kullanıcının değişen konumunun arasındaki mesafenin hesaplanması yapılmıştır. Bunun için yine “for” döngüsü kullanılmıştır. Bu işlem için kullanılan kod bloğu aşağıda verilmiştir.

```
for (LatLng k :locations){
   distance.add(SphericalUtil.computeDistanceBetween(latLng,k));
}
```

Sonraki aşamada ise, çıkacak olan uyarı sesinin ayarlanması yapılmıştır. Bunun için eklenmek istenen ses dosyası Android Studio programı içinde “res” dosyası altında “raw” sekmesi içine kopyalanmıştır. Ses dosyalarını kod içerisinde tanımlamak için “MediaPlayer” sınıfında istenilen isimde bir veri oluşturulmuştur. Daha sonra “raw” bölümündeki dosyası kod içerisinde oluşturmak için “ MediaPlayer.*create*(MapsActivity.this,R.raw. dosyaAdı )” kodu kullanılmıştır. Kullanılan ses dosyasından “Dikkat kasis var” şeklinde bir uyarı çıkmaktadır. Anlık konum ve kasislerin konumları arasındaki mesafe değeri 20 metre ve 20 metreden daha az olduğu durumlarda mobil uygulamanın uyarı sesini vermesi için “LocationManager” altında eklenen kodlar aşağıda verilmiştir.

```
for (int m =0; m<distance.size(); m++) {
   if (distance.get(m)<=20){
      sound = MediaPlayer.create(MapsActivity.this,R.raw.sound);
      sound.start();
      sound.stop();
      sound.start();
   }
}
```

Bu kod bloğu çalıştırıldığında uyarı sesi devamlı tekrar edilip sesin ne dediği anlaşılmamaktaydı. Bunu engellemek için aşağıdaki kod bloğu eklendi.

```
      sound.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
         @Override
         public void onCompletion(MediaPlayer sound) {
            MapsActivity.this.sound = MediaPlayer.create(MapsActivity.this, R.raw.sound);
            MapsActivity.this.sound.setLooping(false);
            MapsActivity.this.sound.start();
         }
      });
      break;
   }
```

Eklenen bu kod bloğuyla birlikte seste söylenmek istenen uyarı mesajının anlamı daha rahat bir şekilde anlaşılmakta ve sesin tekrar sayısı düşürülmüş olmaktadır. Bütün bu işlemler sonucunda mobil uygulamayı kullanan sürücü kasislere yaklaşmadan bunun uyarısını alıp, arabanın hız ayarını düşürerek gerekli tedbirlerini almış olacaktır.

### 2.3.4. User Permissions

Uygulamaların kullanımlarında bazı özellik ve bileşenlere ulaşmada erişim sıkıntısı olabilmekte, erişim sorunu kullanıcının gizliliğinin korunması ve güvenliğinin sağlanması sonucu ortaya çıkmaktadır. Bu gibi durumlarda uygulama çalışmadan önce kullanıcıdan izin istenmektedir. İstenilen izinler, kullanıcının konumu, kamera ve ses erişimi, rehbere erişim, galeri ve dosyalara erişim, gibi birçok konuda olabilmektedir. Android Studio' da izinler "Android.Manifest.xml" dosyasına yazılmaktadır. İzin ekleme ve çıkarma işlemleri için manifest dosyasına gidilip ve burada silme ekleme işlemleri yapılmaktadır. Bu tez çalışmasında kullanıcının mevcut konumuna ve internetine erişim için izinler istenmiştir (Şekil 42).

```
-->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET"/>
```

Şekil 42. Konum ve İnternete erişim kod bloğu

İzinler kullanıcıya sunulduğunda kullanıcı onay verdiğinde uygulama çalışıp, izinlerde istenilen özellikler uygulamada kullanılabilmektedir. Eğer kullanıcı izin vermezse uygulama çalışmayı durduracak ve uygulama açılmayacaktır (Şekil 43).


Şekil 43. İzin için sunulan onay penceresi

Android' de işletim sistemlerine göre izinler farklılık göstermektedir. Sdk 23'den küçük olan işletim sistemlerinde kullanıcı uygulamayı indirirken kullanıcıdan izin istenmez uygulama direkt indirilmektedir. Bu durum sistem güvenliği bakımından oldukça sıkıntılar çıkarmaktadır. Sdk 23 ve daha yüksek işletim sistemlerinde kullanıcı uygulamayı indirirken gerekli izinler için kullanıcıya sorular sorulmak zorundadır. Uygulamanın ilk başında iznin olup olmadığı "ContextCompat" sınıfının" checkSelfPermission" metoduyla kontrol edilebilmektedir. Bu metot "PackageManager.PERMISSION.GRANTED" dönerse izin vardır anlamını taşımaktadır. "PackageManager.PERMISSION.DENIED" dönerse izin yoktur anlamını taşımaktadır. İzin istemek için "ActivityCompat" sınıfının "requestPermissions" metodu çağrılmaktadır. Bu metot birden fazla izin çağırmak için de kullanılabilmektedir. "requestPermission metodu" çağrıldığında bir diyalog penceresi çıkmakta ve bu diyalog penceresi onaylandığında "onRequestPermissionResult" metodu çağrılmaktadır. Bu metot uygulama ilk defa çağrıldığında devreye girmektedir. Kullanıcı ilk defa izin verdiğinde yapılacaklar kod bloğu içine yazılmaktadır. Bütün izin isteme işlemlerini içeren kodlar aşağıda verilmiştir:

```
if (Build.VERSION.SDK_INT >= 23) {
    if (checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {    //Eğer izin yoksa
        requestPermissions(new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);   //izin iste
    } else {    // Eğer zaten izin verilmişse
```

```
            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
0, locationListener);

            Location lastLocation =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            latitude = lastLocation.getLatitude();
            longitude =lastLocation.getLongitude();
            userLastLocation = new LatLng(latitude,longitude);
            mMap.setMyLocationEnabled(true); // Anlık konum alma,mavi icon gözüküyor
            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(userLastLocation,
15));
        }
    } else { // Eğer 23 den küçükse bunu yap.İzin sormaz.
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
locationListener);

        Location lastLocation =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
        LatLng userLastLocation = new LatLng(lastLocation.getLatitude(),
lastLocation.getLongitude());
        mMap.setMyLocationEnabled(true);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(userLastLocation, 15));
    }

}

@Override // Kullanıcı ilk defa izin verdiğinde yapılacaklar.
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {

    if (grantResults.length > 0) {
        if (requestCode == 1) {
            if (ContextCompat.checkSelfPermission(this,
```

```
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
0, 0, locationListener);
            Location lastLocation =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            latitude = lastLocation.getLatitude();
            longitude =lastLocation.getLongitude();
            userLastLocation = new LatLng(latitude,longitude);
            mMap.setMyLocationEnabled(true);
            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(userLastLocation,
15));
        }
      }
   }

   super.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

# 3. SONUÇ VE ÖNERİLER

Türkiye'de taşıt trafiğini azaltmak için genellikle yol güzergâhları üzerine kasisler yapılmaktadır. Ancak yapılan kasislerin çoğu tekniğine uygun şekilde yapılmayıp yol üzerinde çoğu zaman belli bile olmamaktadırlar. Ayrıca kasislerden önce sürücüyü uyaracak levhalarında olmaması seyir halindeki sürücü için ciddi sorunlara yol açmaktadır.

Akıllı telefonların gelişmesiyle birlikte akıllı telefonlarda konum belirleme sistem özellikleri de gelişmiştir. Kullanıcıların konumları ve harita üzerinde konumları bilinen nesneler hakkında bilgi edinme isteklerinin yaygınlaşmasıyla yazılımlar geliştirilerek lokasyon tabanlı uygulamalar yapılmaya başlanmıştır. Kullanılan yazılımların kullanıcıya doğru bilgiyi iletmesi için konum verilerinin hassas ve uygun olarak yapılması önem arz etmektedir. Bunun için bu tez çalışmasında, hem trafiği sakinleştirmek için kullanılan kasislerin sürücüleri olumsuz yönde etkilemesini azaltmak hem de son zamanlarda sıklıkla yaygınlaşan yazılım geliştirme kitleriyle mobil uygulama yapılmasını kullanmak amacıyla sürücülere kasislere yaklaşmadan önce uyarı veren lokasyon tabanlı mobil bir uygulama yapılması sağlanmıştır.

Tez kapsamında; Bahçecik, Yenicuma, Arafilboyu, Esentepe ve Karadeniz Teknik Üniversitesi Kampüsü olmak üzere toplam 5 farklı bölge üzerindeki kasislerin konumlarının tespiti sağlanmıştır. Mobil uygulama, Java dili ile Android Studio programı kullanılarak yazılmıştır. Google Maps uygulamaya aktive edilmiştir. Bu harita üzerine kasislerin konum koordinatları aktarılmıştır. Her kasisin bulunduğu yere bir marker atayıp burada bilgi pencereleri oluşturulmuştur. Bu bilgi pencerelerinin içine de kasis uyarı levhası konulmuştur. Aynı zamanda mobil uygulama kullanıcısı araç sürücüsünün de anlık konumunun alınması sağlanmıştır. Kullanıcının konumu ile kasisler arası mesafe 20 metre olduğu zaman uygulamadan "Dikkat kasis var" uyarı sesi çıkarak kullanıcıyı uyarmaktadır. Bu şekilde sürücü aracın hızını düşürerek, aracın kasisin üzerinden güvenli bir şekilde geçmesini sağlamaktadır. Böylece yüksek bir hızla kasise yaklaşmayıp kaza veya yaralanma riski azaltılmış olur. Özellikle ambulans itfaiye gibi acil durum araçları için bu durum daha fazla önem taşımaktadır. Ayrıca havanın karanlık olması, sisli olması gibi düşük görüş koşullarının olduğu durumlarda yollarda belli olmayan kasislerin varlığını anlayıp önceden tedbir alınmasının sağlanması bu mobil uygulamanın önemini göstermektedir.

Haritacılık alanıyla ilgili yapılan konuma dayalı lokasyon tabanlı mobil uygulamalarda konum doğruluğu önemli olmaktadır. Özelliklede alınan anlık konumlarla mesafe hesabı

yapılıyorsa bu durum daha fazla önem taşımaktadır. Ancak mobil cihazlarda yer alan GNSS alıcılarının bazen veri alamamasından ötürü konumsal hatalar, konumda birkaç metre yanılmalar olabilmektedir. Bu da yapılan uygulamanın hassasiyetini etkilemektedir. Ayrıca her mobil cihaz farklı frekanslarda değerleri aldığı için konum hassasiyeti her mobil cihazda da farklılık gösterebilmektedir. Bu nedenle lokasyon tabanlı uygulamalar yapılırken bu durum göz önüne alınmalıdır.

Tez çalışmasında kasis için yapılan bu uygulama geliştirilerek trafiği aksatan veya trafikteki tedbirsizliklerden kaynaklı kazalara yol açılabilecek diğer tüm olguları da kapsayacak şekilde geliştirmeler yapılabileceği gibi bu mobil uygulamanın iOS ve Windows Phone gibi diğer işletim sistemleri için de yazılımının yapılması sağlanabilir.

## 4. KAYNAKLAR


Android Guides, https://developer.android.com/distribute/best-practices/develop/build-with-android-studio?hl=tr , 26.02.2020.

Arık, H.,İ., Genel Anlamda Akıllı Telefon Teknolojisi, https://www.bilgiustam.com/genel-anlamda-akilli-telefon-teknolojisi/ , 27.01.2020.

Bildirici, İ. Ö., 2015. Google Maps API Harita Mashup Hizmeti Kullanarak Internet Yoluyla Harita Yayınlama, Konya

Çoban M., 2001. Java Programlama Dili Kitabı, 2.Baskı, 20-24, Kocaeli.

Çobanoğlu B., 2020. Java ile Programlama ve Veri Yapıları, 5. Baskı, 23, Pusula.

Ewing, R., 1999. Traffic calming state of practice, U.S. Department of Transportation Federal Highway Administration, Washington.

Güzel N., Özdemir Y. ve Özdemir Ş., 2019. Akıllı Ulaşım ve Akıllı Kasis Aydınlatma Projesinin Akıllı Ulaşım Kapsamında Değerlendirilmesi, İstanbul Sabahattin Zaim Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 1,1, 47-52.

Gürer M., 2006. Hız Tümseklerinin Taşıt Titreşim Teknolojisi Açısından İncelenmesi, İstanbul.

Kabakuş A.T., Doğru İ. A. ve Çetin A., 2015. Android kötücül yazılım tespit ve koruma sistemleri, Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 31,1, 9-16.

Kaygısız, Ö., 2010. "Trafiği Sakinleştirmeye Yönelik Önlemler", Karayolu Trafik Güvenliği, Ankara, Sempozyumu Kitabı, 543-562.

Kh'tour M.,2015. Android Tabanlı Hastane Uygulaması, Kırıkkale.

Kılıç T., Tuncer T.,2017. Akıllı Şehir Uygulaması: Android Tabanlı Akıllı Otopark Sistemi, Elazığ.

Özcan M.,2013. Bir Android Uygulama Modeli: İstanbul Toplu Taşıma Bilgi Sistemi, İstanbul.

Özel,H., 2018. Kotlin Nedir? https://medium.com/@halilozel1903/kotlin-nedir43e312d2dca6 , 02.03.2020.

Pocatilu P., 2011. "Android applications security," Inform. Econ., 15, 163–171.

Saraç, S., 2019. Android İşletim Sistemi Üzerinde Çalışan Bir Harita Paket Programının Geliştirilmesi, Ağustos, Zonguldak.

Sözen A., Hız Kesicilerin Trafik Yüklemesi Altındaki Dinamik Simülasyonu, Isparta 2007.

Stephens, B.W., 1986. Road humps for the control of vehicular speeds and traffic flow, Public Roads, 50,3, 82-90.

Taç M., 2013. Android Programlama Kitabı, 3. Baskı, ISBN:978-605-61677-7-5, Dikeyeksen Yayıncılık.

Weber, P.A., 1998. Towards a Canadian standard for the geometric design of speed humps, Master of Engineering Thesis, Carleton University, Department of Civil and Environmental Engineering, Ottawa, Ontario, Canada.

Zafer, H., Android 101, https://gelecegiyazanlar.turkcell.com.tr/konu/android/egitim /101 , 5.03.2020.

URL-1, http://abainnolab.com/teknolojinin-gelisim-sureci/ , 14.01.2020.

URL-2, https://www.trafikburada.com/TrafikMalzemeleri/69-Kaucuk-Hiz-Kesici-6-Bantli-50x60x4-5-cm.aspx , 22.01.2020.

URL-3, http://www.buyukkayseri.com/haber/isyan-ettiren-kasis-29602.html , 23.01.2020.

URL-4, https://twitter.com/jsalomonetv/status/1055626455223361536 , 23.01.2020.

URL-5, https://mediatrend.mediamarkt.com.tr/akilli-telefon-tarihi/ , 27.01.2020.

URL-6, https://www.eniyiandroid.com/ilk-android-telefon-htc-dream-t-mobile-g1.html , 27.01.2020.

URL-7, http://phonesdata.com/tr/smartphones/htc/dream-397/ , 27.01.2020.

URL-8, https://shiftdelete.net/2019-akilli-telefon-satis-rakamlari-dusmeye-devam-ediyor , 27.01.2020.

URL-9, https://www.idc.com/promo/smartphone-market-share/os , 29.01.2020.

URL-10, https://shiftdelete.net/ios-ile-androidi-karsilastirdik-34697 , 27.01.2020.

URL-11, https://en.wikipedia.org/wiki/Android_(operating_system) , 27.01.2020.

URL-12, https://teloji.com/gecmisten-gunumuze-androidin-gelisimi-tum-android-surumleri/ , 30.01.2020.

URL-13, https://www.sonsuzteknoloji.com/androidin-10-yillik-tarihi-tarihcesi/ , 30.01.2020.

URL-14, https://www.yugatech.com/goooooogle/the-flavors-of-android-through-the-years/#sthash.hD30WU8P.dpbs , 30.01.2020.

URL-15, https://www.donanimhaber.com/LG-G3-icin-Android-60-Marshmallow-guncellemesi-dagitilmaya-baslandi--81371, 31.01.2020.

URL-16, https://www.gizrom.com/b520-emui-5-0-huawei-mate-8-android-7-0/ , 31.01.2020.

URL-17, http://www.hurriyet.com.tr/teknoloji/android-8-0daki-yenilikler-iste-bunlar-40630915 , 31.0.2020.

URL-18, https://www.teknoburada.net/2018/03/19/android-8-0-guncellemesi-alacak-samsung-telefonlar-mart-2018/, 31.01.2020.

URL-19, http://www.hurriyet.com.tr/teknoloji/android-pie-nedir-android-9-0-ozellikleri-nelerdir-41063729 , 31.01.2020.

URL-20, https://huaweiadvices.com/download-android-9-0-pie-update-for-huawei-p9-lite-aosp-rom/, 31.01.2020.

URL-21, https://shiftdelete.net/android-10-ozellikleri-neler, 31.01.2020.

URL-22, https://hwp.com.tr/xiaomi-mi-a2-android-10-guncellemesi-yayinlandi-126184 , 31.01.2020.

URL-23, https://tr.wikipedia.org/wiki/Java_(programlama_dili)#cite_note-1, 27.02.2020.

URL-24, https://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/java/ch05/dataTypes.htm, 5.03.2020.

URL-25, https://gelecegiyazanlar.turkcell.com.tr/konu/android/egitim/android-201/fragment-olusturmak, 5.03.2020.

# ÖZGEÇMİŞ

19.07.1994 yılında Trabzon'da doğdu. İlköğrenimini Cudibey İlköğretim Okulu'nda lise öğrenimini ise 88. Yıl Cumhuriyet Anadolu Lisesi'nde gördü. 2012 yılında Karadeniz Teknik Üniversitesi Harita Mühendisliği bölümünü kazandı. 4 yıllık eğitimin ardından 2016 yılında lisans eğitimini tamamladı. 2017 yılında Karadeniz Teknik Üniversitesi'nde yüksek lisans eğitimine başladı. Yüksek lisans eğitiminin ilk döneminde İngilizce hazırlık okudu. Yüksek Lisans eğitimi halen devam etmektedir.