CARDUİNO

KAMİL KAPLAN

14542025 – Yazılım Mühendisliği(I. Öğretim)

04.03.2017

FIRAT ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

Yazılım Mühendisliği Bölümü

YMH114 - Yazılım Mühendisliğinin Temelleri Dersi

Proje Uygulaması ve Dokümantasyonu

ARDUINO VE BLUETOOTH İLE ANDROID TELEFON ÜZERİNDEN KONTROL EDİLEN ARABA

Geliştiren

Kamil KAPLAN

Proje Yürütücüleri

Doç. Dr. Murat KARABATAK

Dr. Muhammet BAYKARA

04.04.2017

ÖNSÖZ

Günümüzde sayısal haberleşme teknikleri oldukça ilerlemiştir. Sayısal haberleşme uygulamaları paralel ve seri olmak üzere iki ana başlık altında toplanabilir. Uzak mesafelerde yapılan haberleşme uygulamalarında seri haberleşme kullanılmaktadır. Bu çalışmada bluetooth kullanılarak sayısal haberleşme sisteminin gerçekleştirilmesi ve araç kontrolü sağlanacaktır.

Çalışmalarımız boyunca bize değerli zamanını ayıran ve verdiği fikirler ile bizi yönlendiren hocamız Sayın Doç. Dr. Murat KARABATAK, Sayın Dr. Muhammet BAYKARA
'ya teşekkür ederim.

Ayrıca hayatım boyunca her türlü maddi ve manevi desteklerini hiçbir zaman esirgemeyen ailelerime şükranlarımı sunarım.

Kamil KAPLAN
Mayıs 2017

İÇİNDEKİLER

1. GİRİŞ

ÖNSÖZ

1.1 Projenin Amacı

1.2 Projenin Kapsamı

1.3 Tanımlamalar ve Kısaltmalar

2. PROJE PLANI

2.1 Giriş

2.2 Projenin Plan Kapsamı

2.3 Proje Zaman-İş Planı

2.4 Proje Ekip Yapısı

2.5 Önerilen Sistemin Teknik Tanımları

2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

2.7 Proje Standartları, Yöntem ve Metodolojiler

2.8 Kalite Sağlama Planı

2.9 Eğitim Planı

2.10 Test Planı

2.11 Bakım Planı

3. SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistem İncelemesi

3.1.1 Veri Modeli
3.1.2 Varolan Yazılım/Donanım Kaynakları
3.1.3 Varolan Sistemin Değerlendirilmesi

3.2 Gereksenen Sistemin Mantıksal Modeli

3.2.1 Giriş
3.2.2 Bilgi Sistemleri/Nesneler
3.2.3 Başarım Gerekleri

3.3 Belgeleme Gerekleri

3.3.1 Geliştirme Sürecinin Belgelenmesi
3.3.2 Eğitim Belgeleri
3.3.3 Kullanıcı El Kitapları

4. SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı

4.1.2 Varsayımlar ve Kısıtlamalar

4.1.3 Sistem Mimarisi

4.1.4 Dış Arabirimler

4.1.4.1 Kullanıcı Arabirimleri

4.1.4.2 Veri Arabirimleri

4.1.4.3 Diğer Sistemlerle Arabirimler

4.1.5 Testler

4.1.6 Performans

4.2 Süreç Tasarımı

4.2.1 Genel Tasarım

4.2.2 Modüller

4.2.2.1 Giriş Modülü

4.2.2.1.1 İşlev

4.2.2.1.2 Kullanıcı Arabirimi

4.2.2.1.3 Modül Tanımı

4.2.2.1.4 Modül iç Tasarımı

4.2.2.2 Yönetici Modülü

4.2.2.2.1 İşlev

4.2.3 Entegrasyon ve Test Gereksinimleri

4.3 Ortak Alt Sistemlerin Tasarımı

4.3.1 Ortak Alt Sistemler

4.3.1.1 Yetkilendirme Alt Sistemi

4.4.1.2 Güvenlik Alt Sistemi

4.4.1.3 Yedekleme Alt Sistemi

4.4.1.4 Veri İletişim Alt Sistemi

4.4.1.5 Arşiv Alt Sistemi

4.4.1.6 Dönüştürme Alt Sistemi

5. SİSTEM GERÇEKLEŞTİRİMİ

5.1. Giriş

5.2. Yazılım Geliştirme Ortamları

5.3. Kodlama Stili

5.3.1 Açıklama Satırları

5.3.2 Kod Biçimlemesi

5.3.3 Anlamlı İsimlendirme

5.3.4 Yapısal Programlama Yapıları

5.4. Program Karmaşıklığı

5.4.1 Programın Çizge Biçimine Dönüştürülmesi

5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

5.5. Olağan Dışı Durum Çözümleme

5.6. Kod Gözden Geçirme

5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi

5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular

5.6.2.1 Öbek Arayüzü

5.6.2.2 Giriş Açıklamaları

5.6.2.3 Veri Kullanımı

5.6.2.4 Öbeğin Düzenlenişi

5.6.2.5 Sunuş

6. DOĞRULAMA VE GEÇERLEME

6.1. Giriş

6.2. Sınama Kavramları

6.3. Doğrulama ve Geçerleme Yaşam Döngüsü

6.4. Sınama Yöntemleri

6.4.1 Beyaz Kutu Sınaması

6.4.2 Kara Kutu Sınaması

6.5.Sınama ve Bütünleştirme Stratejileri

6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme
6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme

6.6. Sınama Planlaması

6.7. Sınama Belirtimleri

6.8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri

7. BAKIM

7.1 Giriş

7.2 Kurulum

7.3 Yazılım Bakımı

8. KODLAR

9. SONUÇ

10. KAYNAKLAR

1. GİRİŞ

1.1. Projenin Amacı

Bu proje, bir aracın uzaktan kumada ile hareketini kontrol edilmesini sağlayacak şekilde tasarlanmıştır. Bilgisayar kontrollü araç, batarya ile çalışan, kontrolü bilgisayar veya android işletim sistemi bulunan cihazlarla uzaktan kontrol edilebilen, dört tane tekerleğe sahip, ileri - geri, sağa - sola yön kontrolü edilebilen ayrıca fren sistemi bulunan uzaktan kumandalı bir araçtır. Bu aracın kendini diğer projelerden ayıran özelliği android işletim sahip cihazlarla kontrol edilebilmesi, sensör ve gerekli ekipmanlar takılarak farklı iş alanlarda da pazar payı elde edebilir olmasıdır.

1.2 Projenin Kapsamı

Kablosuz iletişim teknolojisi günümüzde çok yaygınlaşmış durumdadır. Bluetooth en basit ve en çok kullanılan kablosuz haberleşmelerden bir tanesidir.

Bluetooth gerek akıllı telefonlar gerekse bilgisayarlar tarafından kolayca haberleşme aracı olarak kullanılmasından dolayı projede kullanılmıştır. Bluetooth sayesinde aracımızın hem akıllı telefonlar ile hem de bilgisayarlar ile haberleşmesi sağlanacaktır

Araç, mikroişlemciler kullanılarak tasarlanacak olup haberleşmede bluetooth kullanılacaktır. Açık kaynak kodlu olması, kurulumunun kolay olması, tüm işletim sistemleriyle uyumlu çalışabilmesi, kütüphane arşivinin geniş olması, hızlı çalışması, fiyatının uygun olması ve en önemlisi her türlü sensör tipinin bağlanabilmesi mikroişlemci olarak arduino'yu seçme nedenlerimizi oluşturmaktadır.

Aracımızın doğru akım motorları kontrolü sağlamak için motor sürücü devresini kullandık. Araştırmalarımız sonucu iki ayrı doğru akım motorunu sürebildiği için L298N motor sürücü devresinde karar kıldık. Bu projede L298N motor sürücü devresini seçmemiz deki en önemli etken, istediğimiz sayıda giriş çıkış uçlarının olması ve maliyetinin düşük olmasıydı.

Aracımız ilave sensörler takılarak ve mikro işlemciye gerekli programlamalarla farkı alanlarda da kullanılabilir. Örneğin engelli kişilerin kullandığı tekerlekli sandalyelerin yerine uzaktan kumandalı araca gerekli donanımlar yerleştirerek engelli kişilerin engelini biraz olsun azaltmış oluruz. Uzaktan kumandalı aracı geliştirip her sene yurt dışından milyonlar vererek ithal etmemize gerek kalmaz

1.3 Tanımlamalar ve Kısaltmalar

Arduino, **Bluetooth, L298N Motor Sürücü, HC-SR04 Ultrasonik Mesafe Sensörü, Led, Jumper Kablo**

Arduino Uno:

- ✓ Mikrodenetçi: ATmega328P
- ✓ Çalışma voltajı: 5V
- ✓ Giriş voltajı (önerilen): 7-12V
- ✓ Giriş voltajı (limit değerler): 6-20VDijital
- ✓ I / O Pinleri 14 (bunlardan 6'sı PWM çıkışı sağlamaktadır)
- ✓ PWM Dijital I/O Pinleri: 6
- ✓ Analog Giriş Pinleri: 6
- ✓ I/O Pin Başına DC akım: 20 mA
- ✓ 3.3V Pin DC akımı: 50 mA

Bluetooth:

- ✓ Herhangi bir USB Bluetooth adaptörü ile çalışır.
- ✓ Varsayılan Baud Hızı: 9600,8,1, n.
- ✓ Dahili anten.
- ✓ 30 fit'e kadar kapsama alanı.
- ✓ Bluetooth sürümü: V2.0 + EDR
- ✓ Çalışma voltajı: 3.3V
- ✓ Sinyal kapsama alanı: 30ft
- ✓ Öğe Boyutu: 4.3 * 1.6 * 0.7cm

L298N Motor Sürücü:

- ✓ ENA: Sol motor kanalını aktif etme pini
- ✓ IN1: Sol motor 1. girişi
- ✓ IN2: Sol motor 2. gitişi
- ✓ IN3: Sağ motor 1. girişi
- ✓ IN4: Sağ motor 2. girişi
- ✓ ENB: Sağ motor kanalını aktif etme pini
- ✓ MotorA: Sol motor çıkışı
- ✓ MotorB: Sağ motor çıkışı
- ✓ VCC: Besleme voltaj girişi(4.8V-24V)
- ✓ GND: Toprak bağlantısı
- ✓ 5V: 5V çıkışı

HC-SR04 Ultrasonik Mesafe Sensörü:

- ✓ Çalışma Voltajı : 5V DC
- ✓ Çektiği Akım : 15mA
- ✓ Algılama Açısı : 15°
- ✓ Ölçüm mesafesi : 2cm - 4m

Led:

- ✓ 1.5-3V arası gerilimde çalışır.
- ✓ 5V üzeri voltaj değerleri için direnç kullanılması gerekir.

Jumper Kablolar: Bir ucu dişi bir ucu erkek jumper kablolar, breadboard ile yapacağımız çeşitli uygulamalarda kullanabiliriz. Bu kablolar 20 cm uzunluğunda, 40 adet ve 10 ayrı renktedir. 2,54 mm'lik standart pinlere göre üretilmiştir.

2. PROJE PLANI

2.1 Giriş

Bu kapsamda öncelikle inceleme ve analiz yapılacak, bütün sistem bu inceleme ve analiz kısmına dayandırılacaktır. İnceleme kısmında projenin en ince detayına kadar yazılımı yapılacak ve elektronik devre elamanları kurulacaktır. Bundan sonra yapılabilirliğin hesabı yapılacak ve olumlu sonuçlara göre projenin gidişatı belirlenecektir.
Daha sonra izlenecek yolun belirlenmesi, kaynak ihtiyaçları ve gider tahmini, proje süresi ve yazılım geliştirmeye düşen görev ve işlemler analiz edilecek ve sonuçlandırılacaktır.

2.2 Projenin Plan Kapsamı

Projenin plan kapsamında genel olarak mevcut sistem, sistemin gerekliliği ve bu sistemin güvenilirliğinden yola çıkılır.

Teknik Karmaşıklık Tablosu

1	Uygulama, güvenlir yedekleme ve kurtarma gerektiriyor mu?	3
2	Veri iletişimi gerekiyor mu?	5
3	Dağıtık işlem işlevleri var mı?	5
4	Performans kritik mi?	2
5	Sistem mevcut ve ağır yükü olan bir işletim ortamında mı çalışacak?	1
6	Sistem, çevrim içi veri girişi gerektiriyor mu?	5
7	Çevrim içi veri girişi, bir ara işlem için birden çok ekran gerektiriyor mu?	5
8	Ana kütükler çevrim-içi olarak mı güncelleniyor?	1
9	Girdiler, çıktılar, kütükler ya da sorgular karmaşık mı?	5
10	İçsel işlemler karmaşık mı?	5
11	Tasarlanacak kod, yeniden kullanılabilir mi olacak?	5
12	Dönüştürme ve kurulum, tasarımda dikkate alınacak mı?	5
13	Sistem birden çok yerde yerleşik farklı kurumlar için mi geliştiriliyor?	5
14	Tasarlanan uygulama, kolay kullanılabilir ve kullanıcı tarafından kolayca değiştirilebilir mi olacak?	5

0: Hiçbir Etkisi Yok
1: Çok Az etkisi var
2: Etkisi Var
3: Ortalama Etkisi Var
4: Önemli Etkisi Var
5: Mutlaka Olmalı, Kaçınılamaz

Maliyet Kestirim Hesabı

ÖLÇÜM PARAMETRESİ	SAYI	AĞIRLIK	TOPLAM
Kullanıcı Girdi Sayısı	**1**	**4**	**4**
Kullanıcı Çıktı Sayısı	**2**	**4**	**8**
Kullanıcı Sorgu Sayısı	**3**	**3**	**9**
Kütük Sayısı	**6**	**2**	**12**
Dışsal Arayüz Sayısı	**1**	**2**	**2**
Ana İşlev Nokta Sayısı			**35**

İŞLEV NOKTASI HESAPLAMA

İN: İşlev Nokta Sayısı

AİN: Ana İşlev Nokta Sayısı

TKF: Teknik Karmaşıklık Faktörü

İN = AİN x (0.65 x 0.01 x TKF)

İN = 35x (0.65 x 0.01 x 57)

İN = 12,9675

TAHMİNİ OLUŞACAK SATIR SAYISI

Satır Sayısı = İN *20

Satır Sayısı = 259,35

Satır Sayısı Yaklaşık = 260 satır

ETKİN MALİYET MODELİ - COCOMO

İş gücü (K)

K = a x Sb Zaman (T)

T = c x Kd

a, b, c, d: her bir model için farklı olan katsayılar

S = bin türünden satır sayısı

İş gücü (K) K = 3 x 31.2 = 11,21 İş Gücü

Zaman (T) T = 1,10 x 100,35

Zaman Yaklaşık Olarak = 2,5 ay

2.3 Proje Zaman-İş Planı

- Projenin zaman planı kaynak yönetimleri MS Project üzerinden takip edilecektir.
- Zaman planındaki sapmalar sebepleriyle Zaman çizelgesi ve teslim süreci tekrar raporlanır.
- Her biten modül doğrulama ve geçerleme testlerine tabi tutulacaktır.
- Aşağıda projenin **MS PROJECT** de **Gantt Diyagramına** ait bir kısmının görüntüsü verilmiştir.

1			**CARDİNO**	**?126 gün**	**Çar 29.03.17**	**Çar 20.09.17**		
1.1			PROJE BAŞLANGIÇ	0 gün	Çar 29.03.17	Çar 29.03.17		
1.2			**PRPJE PLANLAMA**	**25 gün**	**Çar 29.03.17**	**Sal 2.05.17**		**proje ekibi**
1.2.1			proje çalışanlarınınn belirlenmesi	10 gün	Çar 29.03.17	Sal 11.04.17		proje lideri
1.2.2			proje çalışanlarının tanışması	2 gün	Çar 12.04.17	Per 13.04.17	4	proje ekibi
1.2.3			proje çalışanlarının proje hakkında bilgilendirilmesi	5 gün	Çar 12.04.17	Sal 18.04.17	4	proje lideri
1.2.4			proje ile ilgili egitimin verilmesi	10 gün	Çar 19.04.17	Sal 2.05.17	6	proje lideri
1.3			**PROJE KONTROL**	**11 gün**	**Çar 3.05.17**	**Çar 17.05.17**	**3**	**yazılım test sorumlusu**
1.3.1			proje gidişatını takip etme	3 gün	Çar 3.05.17	Cum 5.05.17		yazılım test sorumlusu
1.3.2			proje muşteri amacına uygunlugunu takip et	2 gün	Pzt 8.05.17	Sal 9.05.17	9	yazılım test sorumlusu
1.3.3			proje analiz test etme	3 gün	Çar 10.05.17	Cum 12.05.17	10	yazılım test sorumlusu
1.3.4			proje sonuç raporu sunma	3 gün	Pzt 15.05.17	Çar 17.05.17	9;10;11	proje ekibi
1.4			**PROJE GEREKSİNİMLERİ BELİRLENMESİ**	**?8 gün**	**Per 18.05.17**	**Pzt 29.05.17**	**8**	**proje ekibi**
1.4.1			proje ihtiyaclarının belirlenmesi	7 gün	Per 18.05.17	Cum 26.05.17		proje ekibi
1.4.2			proje ihtiyaclarının rapor edilmesi	?1 gün	Pzt 29.05.17	Pzt 29.05.17	14	proje ekibi
1.5			**PROJE MALİYET KESTİRİMİ BELİRLENMESİ**	**?11 gün**	**Sal 30.05.17**	**Sal 13.06.17**	**13**	**proje ekibi**
1.5.1			proje ihtiyaclarının maliyetinin tespit edilmesi	7 gün	Sal 30.05.17	Çar 7.06.17		proje ekibi
1.5.2			proje ihttiyacları maliyetinin rapor edilmesi	3 gün	Per 8.06.17	Pzt 12.06.17	17	proje lideri;yazılım test sorumlusu
1.5.3			proje maliyetinin sunulması	?1 gün	Sal 13.06.17	Sal 13.06.17	18	proje lideri
1.6			**YAZILIM GELİŞTİRME**	**?32 gün**	**Çar 14.06.17**	**Per 27.07.17**	**19**	**yazılım ekibi**
1.6.1			yazılım kodu	6 gün	Çar 14.06.17	Çar 21.06.17		yazılım ekibi
1.6.2			yazılım kodlama	25 gün	Per 22.06.17	Çar 26.07.17	21	yazılım ekibi
1.6.3			yazılım birim testi	?1 gün	Per 27.07.17	Per 27.07.17	22	yazılım test sorumlusu
1.7			**PROJE DESTEGİNİN SAGLANMASI**	**?25 gün**	**Çar 14.06.17**	**Sal 18.07.17**	**16**	**proje lideri;yazılım lideri**
1.7.1			proje destegi için destekçi bulunması	8 gün	Çar 14.06.17	Cum 23.06.17		proje lideri
1.7.2			projenin destekçiye anlatılması	3 gün	Pzt 26.06.17	Çar 28.06.17	25	proje lideri
1.7.3			proje destekçisi ile planlama yapılması	5 gün	Per 29.06.17	Çar 5.07.17	26	proje lideri
1.7.4			destekçiye maliyet kestirim sunulması	4 gün	Per 6.07.17	Sal 11.07.17	27	proje lideri
1.7.5			proje destekcisinden sonuç alınması	5 gün	Çar 12.07.17	Sal 18.07.17	28	proje lideri
1.7.6			proje destekçisi ile toplantı yapılması	4 gün	Çar 12.07.17	Pzt 17.07.17	28	proje lideri;tasarımcı;yazılım lideri
1.7.7			proje çalışmalarına başlanması	?1 gün	Sal 18.07.17	Sal 18.07.17	30	proje ekibi
1.8			**PROJE KULLANICILARIYLA BAĞLANTILARININ**	**31 gün**	**Çar 19.07.17**	**Çar 30.08.17**	**24**	**proje lideri;tasarımcı**
1.8.1			proje kullanıcıları ile toplantı yapılması	3 gün	Çar 19.07.17	Cum 21.07.17		proje lideri
1.8.2			proje dokumalarının rapor edilmesi	4 gün	Pzt 24.07.17	Per 27.07.17	33	proje ekibi;yazılım ekibi
1.8.3			proje dokumanlarının sunulması	3 gün	Cum 28.07.17	Sal 1.08.17	34	proje ekibi;tasarımcı
1.8.4			isteklere uygun proje tasarlaması	7 gün	Çar 2.08.17	Per 10.08.17	35	tasarımcı
1.8.5			proje işlemlerinin gerçekleşmesi	7 gün	Cum 11.08.17	Pzt 21.08.17	36	proje ekibi;yazılım ekibi
1.8.6			proje uygun denemeleri yapılıp sunumu	7 gün	Sal 22.08.17	Çar 30.08.17	37	proje ekibi;yazılım ekibi
1.9			**PROJE DEGERLENDİRİLMESİ**	**6 gün**	**Per 31.08.17**	**Per 7.09.17**	**32**	**proje lideri**
1.9.1			proje gidişatının degerlendirilmesi	3 gün	Per 31.08.17	Pzt 4.09.17		proje lideri
1.9.2			proje testi yapılması	3 gün	Sal 5.09.17	Per 7.09.17	40	yazılım test sorumlusu
1.10			**SİSTEM BİTİŞ (GÜNLEME)**	**9 gün**	**Cum 8.09.17**	**Çar 20.09.17**	**41**	**proje ekibi**
1.10.1			yazılım testi ve uygulama	2 gün	Cum 8.09.17	Pzt 11.09.17		yazılım ekibi;yazılım test sorumlusu
1.10.2			donanım testi ve uygulama	2 gün	Sal 12.09.17	Çar 13.09.17	43	proje ekibi
1.10.3			sistem kabul testi ve günleme	3 gün	Per 14.09.17	Pzt 18.09.17	44	proje ekibi;proje lideri
1.10.4			proje teslim	2 gün	Sal 19.09.17	Çar 20.09.17	45	proje lideri

Şekil 2.1 Proje İş-Zaman Çizelgesi

2.4 Proje Ekip Yapısı

- **Proje Yöneticisi:**
  - ✓ **Projenin Bütünlüğünü Sağlama:** Grubumuzun projelerdeki en büyük ve en önemli görevi projenin bütünlüğünün sağlamak olacaktır.
  - ✓ **İzleme ve Raporlama:** Ayrıca projenin en önemli 3 ayağı diyebileceğimiz Kapsam, Zaman ve Maliyet bilgi alanlarını devamlı olarak izlemeli ve buradaki değişimleri raporlayabilmelidir.
  - ✓ **Büyük Resmi Görme:** Ekibi hedeflere motivasyonlu bir şekilde yönlendirirken aynı zamanda resmi büyük çerçeveden görebilmeli ve projenin bütününde neler olup bittiğini görebilmelidir. Bu sayede üst yönetime sunduğu raporlar detaydan çok bütünü ve hedefleri içerecektir. Bu sayede projenin gidişatı daha net izlenebilir.
  - ✓ **Riskleri Görme ve Analiz Etme:** Proje sürecinde oluşabilecek muhtemel riskleri devamlı olarak izlemelidir. Oluşabilecek bu risklere karşı da devamlı olarak Risk Yanıt Planlarını hazırlamalıdır. Risk Yanıt Planı sayesinde risk gerçekleşme durumunda alınacak aksiyon belli olmuş olacaktır. İzlenen riskler üst yönetim ve Proje yönetimi ile devamlı olarak Proje İlerleme Raporunda (Progress Report) paylaşılmalıdır.
  - ✓ **Projeyi Sonlandırma:** Proje Yöneticisi eğer projenin artık hedeflerinden saptığını, gruba yarardan çok zarar sağladığını düşünüyor ve proje amaçlarının gerçekleştirilme olanağının kalmadığını düşünüyorsa bu noktada inisiyatif alabilmeli ve projeyi sonlandırabilmelidir.

- **Kalite Uzmanı:** İşletmenin genel çalışma prensipleri doğrultusunda, araç, gereç ve ekipmanları etkin bir şekilde kullanarak, işçi sağlığı, iş güvenliği ve çevre koruma düzenlemelerine verimlilik ve kalite gereklerini sağlar.

- **Bilgisayar Yazılımcısı:** Değişik konularda ve çok miktardaki bilgiyi, bilgisayar ortamında hızlı ve sistematik bir biçimde çözümlemek ve değerlendirebilmek amacı ile programımızı yazmaktır.
  - ✓ Bilgilerin bilgisayarda amaca uygun olarak sistematik bir biçimde kullanımını sağlayacak bilgisayar programları yazmamız,
  - ✓ Arduino programlama dilini ve elindeki bilgilerin niteliğine en uygun olan programlama dilini seçer ve programı bu dilde yazar,
  - ✓ Programlama dillerinin yazımında kullanılan kodlamaları yapar ve uygular,
  - ✓ Yazdığımız programı test eder,
  - ✓ Sistem analistinin verdiği formları, bilgisayarın kullanım diline kodlar.

- **Yazılım Mühendisi:** Arduino elemanlarıyla hazırlanmış olan devrelerin içine yazılımı atmasını sağlar.
- **Donanım Mühendisi:** Bilgisayar donanım mühendisleri araştırma, geliştirme tasarım ve çeşitli bilgisayar donanımlarını test eder. Bazı güncellemeler yaparak mevcut bilgisayar donanımını yazılımları daha iyi çalıştırmaya yönlendirir.
- **Sistem Çözümleyici:**
- **Sistem Tasarımcı**

Sistem yöneticisi, projenin ihtiyaçlarını analiz ederek bilgisayar sistemlerini tasarlama, kurma, destekleme, geliştirme, sürekliliğini ve güvenliğini sağlama işini yapar.

2.5 Önerilen Sistemin Teknik Tanımları

- ✓ Sistemde, mikroişlemci yazılımı geliştirilecek.
- ✓ Kullanılacak ve kullanılabilirlik açısından olabildiğince basit tasarlanacaktır.
- ✓ Bluetooth sensörü sayesinde uzaktan kullanım sağlayacaktır.
- ✓ Düşük bir maliyet ile herkesin bütçesine uygun kullanım sağlanacaktır.
- ✓ Telefon ve bilgisayar kullanmadan pratik bir şekilde kullanım sağlayacaktır.

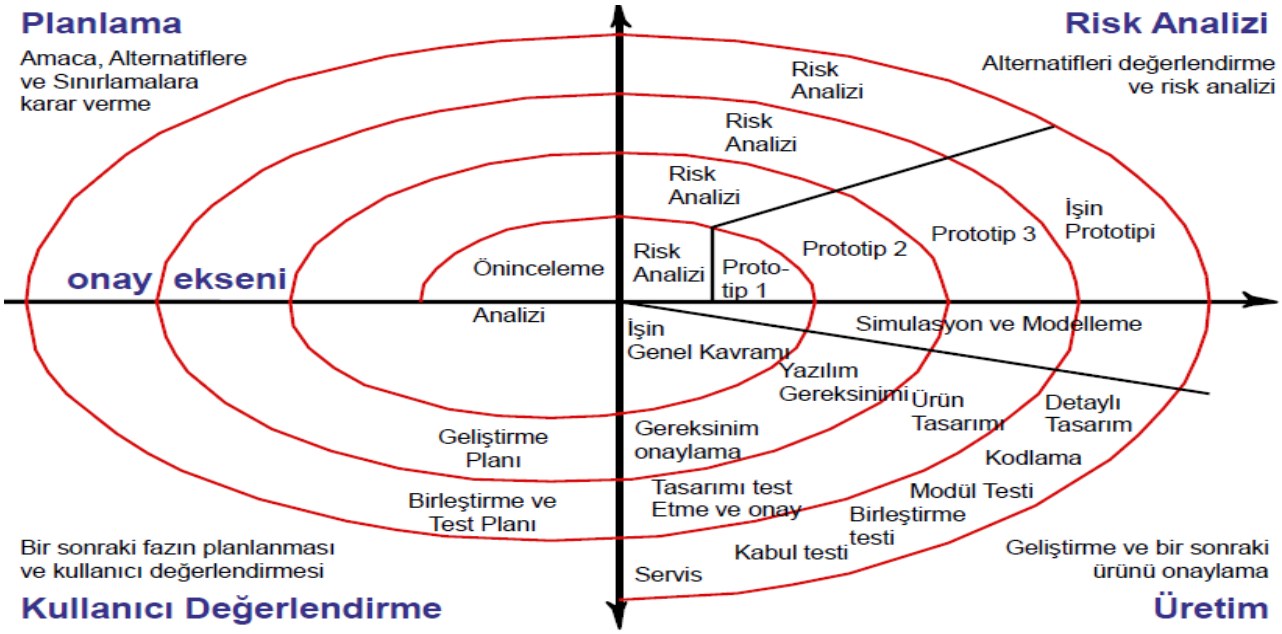
2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

- ✓ İşletim Sistemleri
  - Windows 7
  - Windows 8
  - Windows 10
- ✓ Programlama Dilleri
  - C++
  - Arduino IDE
- ✓ Derleyiciler
  - AVR: Atmel firmasının üretmiş olduğu 8 bitlik RISC mimarisine sahip mikrodenetleyicidir. Ucuz ve hızlı çalışan bir mikrodenetleyici olup, gelişmiş özellikleri bulunaktadır.
  - GCC: GNU Projesi tarafında üretilen ce çeşitli programlama dillerini destekleyen bir derleyicidir.
- ✓ Tasarım
  - Circuits: Görsel arka planını tasarlamak için kullanacağımız programdır.

2.7 Proje Standartları, Yöntem ve Metodolojiler

Proje geliştirme süreci olarak geliştirme süreci boyunca sürekli geri bildirimlerle her yapının test edilerek onaylanması sonucu bir diğer adıma geçilmesi nedeniyle sonuçta elde edilen yazılım en verimli ve doğru sonuç elde edilmesi nedeniyle **Helezonik Model** seçilmiştir.

Geliştirme standartları olarak; *SEI(Software Engineering Institute)*, *EEE Standards*, *ISO* standartları hedeflenerek proje bu standartlar üzerine planlanmıştır ve geliştirilmiştir.



Bu modelde risk analizi ön plana çıkmıştır. Yinelemeli artımsal bir yaklaşım vardır. Her döngü bir fazı ifade eder ve doğrudan adım tanımlama gibi bir faz yoktur. Ayrıca prototip yaklaşımı vardır. Bu modelde süreç 4 gruba ayrılır. Bu süreçler planlama, risk analizi, üretim, kullanıcı değerlendirmesidir.

Planlama üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirmeyi sağlar.
Risk analizi risk seçeneklerinin araştırılması ve risklerin belirlenmesini sağlar.
Üretim ara ürünün üretilmesini sağlar.
Kullanıcı değerlendirmesi ise ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmelerini ele alır.

Bu modelin avantajları vardır. İlki üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından sınanması temeline dayanır. Yazılımı kullanacak personelin sürece erken katılması ileride oluşabilecek istenmeyen durumları engeller.
Diğer bir avantajı gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla proje boyunca karşılaştıkları için daha kolay izleme ve hak ediş planlaması yapılır. Avantajlarından biri de yazılımın kodlanması ve sınanması daha erken başlar.

2.8 Kalite Sağlama Planı

- Kalite Güvence
- Kalite Planlama
- Kalite Kontrol

2.9 Eğitim Planı

- Proje teslimi sonrası sistemi kullanacak olan kişiler arasından seçilen yetkili kişilere proje kullanımı olarak 1 günlük bir seminer verilecektir.

2.10 Test Planı

- Helezonik model gereği her bir işlem sonrası doğrulama ve geçerlilik testine tabi tutulmak içiçn proje Bakım ve Test ekibine verilir. Proje bakım ve test ekibi modülleri germe, aşırı yükleme gibi çeşitli sistem testlerine sokarlar. Bu şekilde her modül oluşturulduktan sonra test edilir eğer doğrulama ve geçerlilik testlerinden başarısız olunursa tekrar başa dönülerek modül için sistem tasarımları tekrardan gözden geçirilir. İsterler tekrardan tanımlanır ve modül gerekirse tekrardan yazılır. Eğer Doğrulama ve geçerlilik testlerinden geçerse bir sonraki adıma geçilir.

2.11 Bakım Planı

- Sistem bakımı için tekrardan bir yazılım yaşam döngüsü oluşturulur ve bu yaşam döngüsü adımları izlenerek sistem bakımı gerçekleştirilir. Sistem bakımı için oluşturulan ekip gereken kullanıcı isterlerini belirler ve buna uygun bir sistem planı çıkartarak yazılım için gerekli geçerleme işlemlerini yaparlar.

3. SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistem İncelemesi

Projenin gereksinimlerin araştırılması, tanımlanması, ortaya çıkarılması ve düzgün bir şekilde, terimsel olarak açıklanması bu bölümde yapılacaktır.

3.1.1 Veri Modeli

Veri tabanı ilişkisel veri modelinde veriler tablolar üzerinden kurulan ilişkiye dayanmaktadır.

3.1.2 Varolan Yazılım/Donanım Kaynakları

- **Microsoft Project:** MS Project, Microsoft tarafından geliştirilen ve satılan, proje yöneticilerine plan oluşturma, kaynakların görevlere atanması, aşama takibi, bütçe yönetimi ve iş yükü analizi gibi konularda yardımcı olması amacıyla tasarlanmış bir proje yönetimi yazılımıdır. Bu projede Gantt Diyagramı çizilmesinde faydalanılmıştır.
- **Microsoft Word:** Microsoft Word, dünyanın en popüler metin kontrol uygulamasıdır. Bu projede dokümantasyonun hazırlanmasında faydalanılmıştır.
- **Microsoft Visio:** Ücretsiz olarak UML diyagramlarını çizmeye, modellemeye yarayan kullanımı kolay bir UML çizim programıdır. Bu projede diyagramların çiziminde faydalanılmıştır.
- **Argo UML:** ArgoUML, ücretsiz olarak UML diyagramlarını çizmeye, modellemeye yarayan kullanımı kolay bir UML çizim programıdır. Bu projede diyagramların çiziminde faydalanılmıştır.
- **SmartDraw:** Use Case Diyagramları hazırlanırken faydalanılmıştır.
- **Adobe Photoshop:** Bu projede belirli resimlerin çizilmesinde faydalanılmıştır.
- **Circuits:** Arduino devre tasarımını bilgisayarda tasarlayıp prototifini görmede kullanılır.

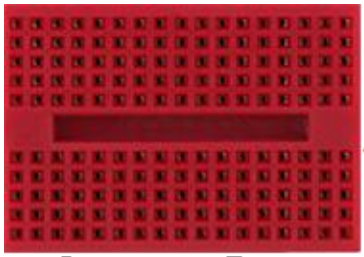
3.1.3 Varolan Sistemin Değerlendirilmesi: Örnek sistemlere bakıp gerekli incelemeler yapıldı ve projemize farklı özellikler eklendi.

3.2 Gereksenen Sistemin Mantıksal Modeli

3.2.1 Giriş

Bu bölümde önerilen sistemin işlevsel yapısı, veri yapısı ve kullanıcı ara yüzünde çözümleme ve arduinonun çalışmasında çözümleme yapılır. Bu model daha çok bilgi sistemini geliştirecek teknik personele yöneliktir. Mantıksal model olarak da tanımlanır.

3.2.2 Bilgi Sistemleri/Nesneler


BREADBOARD TAHTASI


L298N MOTOR SÜRÜCÜ

HC-SR04 ULTRASONİK MESAFE


HC 05 BLUETOOTH

3.2.3 Başarım Gerekleri:

- ✓ Sistem hızlı çalışmalı
- ✓ Mesafeyi doğru ölçmeli
- ✓ Kullanıcıyı led ile bilgilendirmeli
- ✓ Güvenlik sorunları çıkmamalı
- ✓ Kullanıcılardan geri dönüş alınmalı

3.3 Belgeleme Gerekleri

3.3.1 Geliştirme Sürecinin Belgelenmesi

Belgeleme Microsoft Word ile yapılmaktadır. Bu rapor projenin tüm ayrıntılarını içermektedir. Belge içeriği aşağıda listelenen 8 ana konudan oluşmaktadır.

- ✓ Giriş
- ✓ Proje Planı
- ✓ Sistem Çözümleme
- ✓ Sistem Tasarımı
- ✓ Sistem Gerçekleştirimi
- ✓ Doğrulama ve Geçerleme
- ✓ Bakım
- ✓ Sonuç

3.3.2 Eğitim Belgeleri

Çalışan personele sistemin bir parçası olabilmesi için eğitim verilmesi gerekmektedir. Bu eğitimin içeriği; sistemin işleyişi, personelin sistemdeki yeri, kullanacağı program hakkında bilgi verilmesi olacaktır.

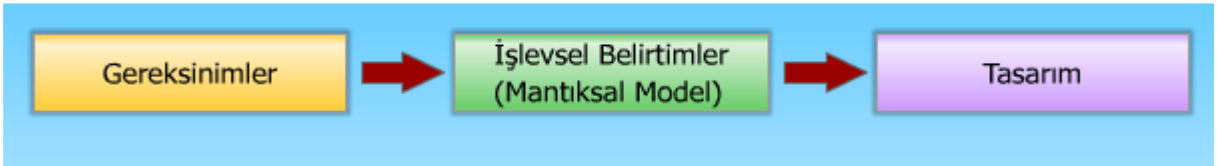
3.3.3 Kullanıcı El Kitapları

Kullanıcı el kitabında sistemin tanıtımı, amacı ve nasıl kullanacağı hakkında bilgiler olacaktır.

4. SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı



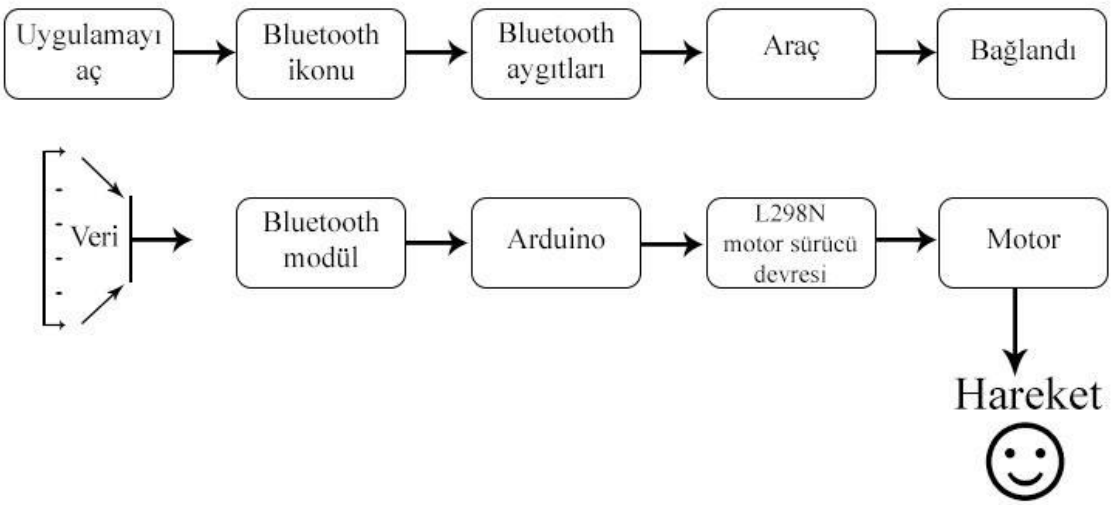
Genel Sistem Tanımı

- Gereksinimler

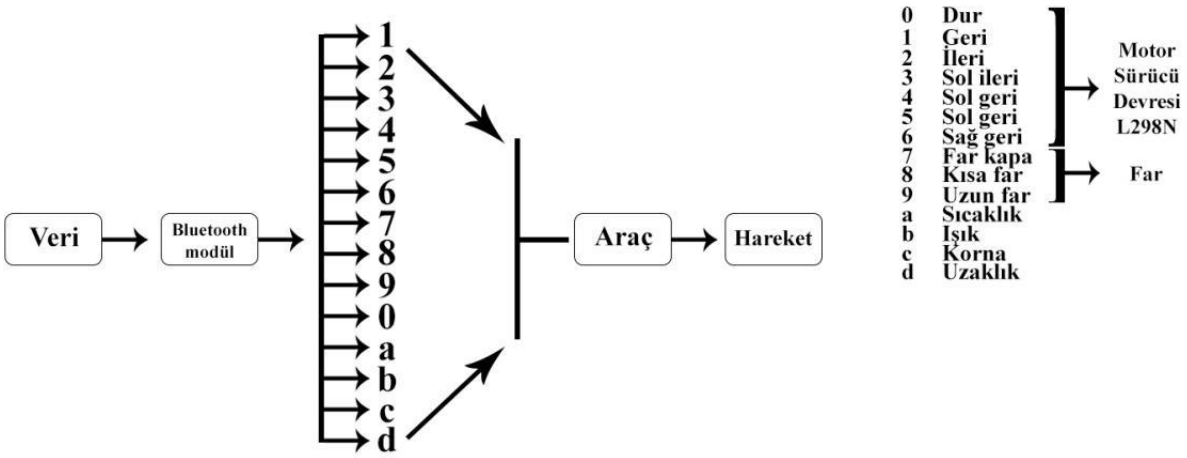
  - ✓ Arduino Malzemeleri
  - ✓ Ardino Uno R3

- İşlevsel Belirtimler: Bu bölümde ilk önce sistemin ne yapacağı sorusuna cevap verelim. Sistem bir aracın uzaktan kumada ile hareketini kontrol edilmesini sağlayacak şekilde tasarlanmıştır. Bilgisayar kontrollü araçtır.

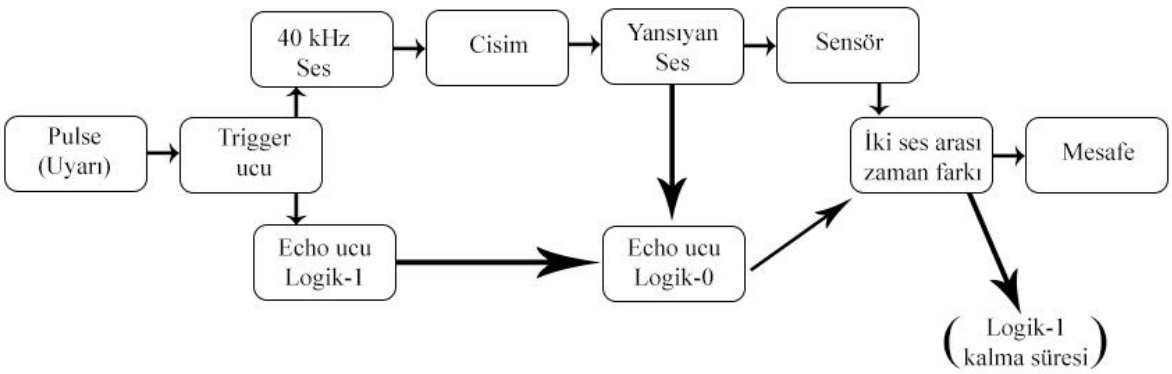
- Tasarım: Tasarımda planladığımız arabamızın devrelerini ve üç boyutlu çizimlerini çizimlerini bilgisayarda gerçekleştirdik. Daha sonra aracın parçalarını bu çizimlerden yararlanarak birleştirdik. İlk önce bilgisayardan veya akıllı telefonumuzdan uygulamayı açıcağız sonra bluetooth teknolojisi ile arabayla bağlantı kuruyoruz arabada ki bluetooth aygıtına bağlandıktan sonra panelimizden aracın hareki ile ilgili komutları verince bluetooth teknolojisi sinyali aracın bluetooth aygıtına gönderir. Bluetooth aygıtı sinyali alıp araçta ki mikro işlemciye (arduino) iletir. Mikro işlemci gelen sinyali değerlendirir ve sinyale karşılık gelen komutu motor sürücü devresine (L298N) gönderir. Motor sürücü devresi gelen komutla motorun hareket etmesini, durmasını, sağa, sola hareket etmesini sağlayacak çıkışlardan sinyal göndererek motoru besler. Böylece motoru uzaktankumanda ile kontrolünü sağlamış oluruz.



Aracın devre planlaması 1

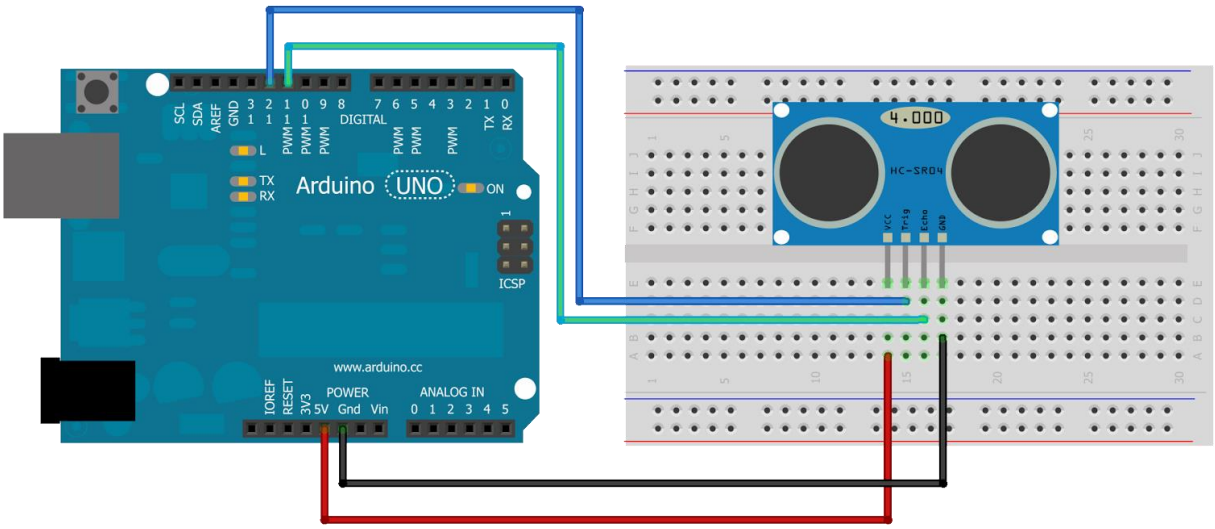


Aracın devre planlaması 2

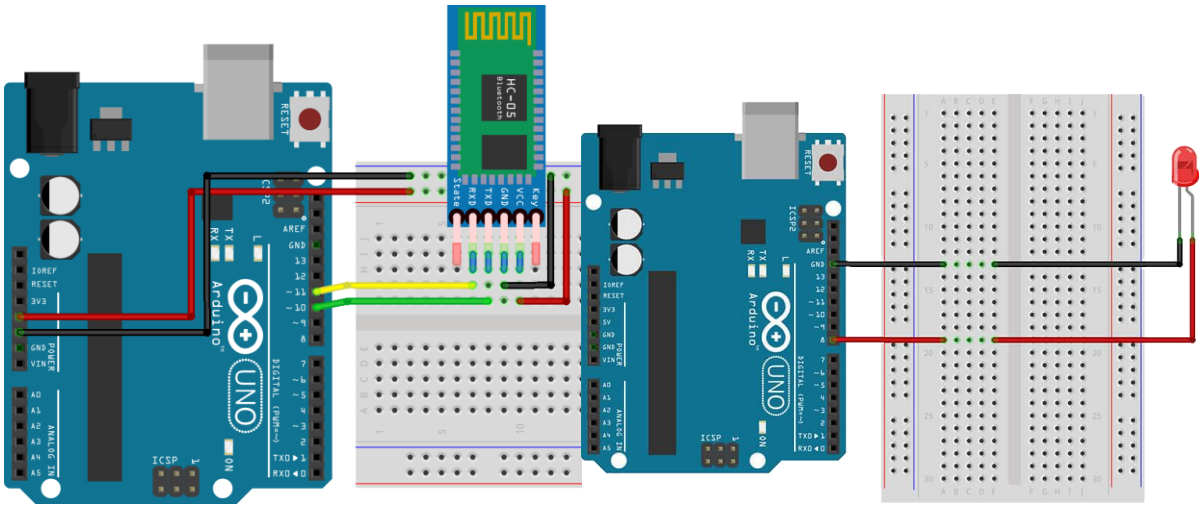


Aracın devre planlaması 3

Uzaktan kumandalı aracın malzemelerini belirleyip, satın aldıktan sonra bu malzemeleri arabanın iskeletinin üzerine doğru yerlere ve doğru şekilde monte etmemiz gerekiyordu. Bu amaçla monte edeceğimiz malzemeleri teker teker şasenin uygun yerlerine bilgisayar programı yardımı ile belirleyip, çizdik.

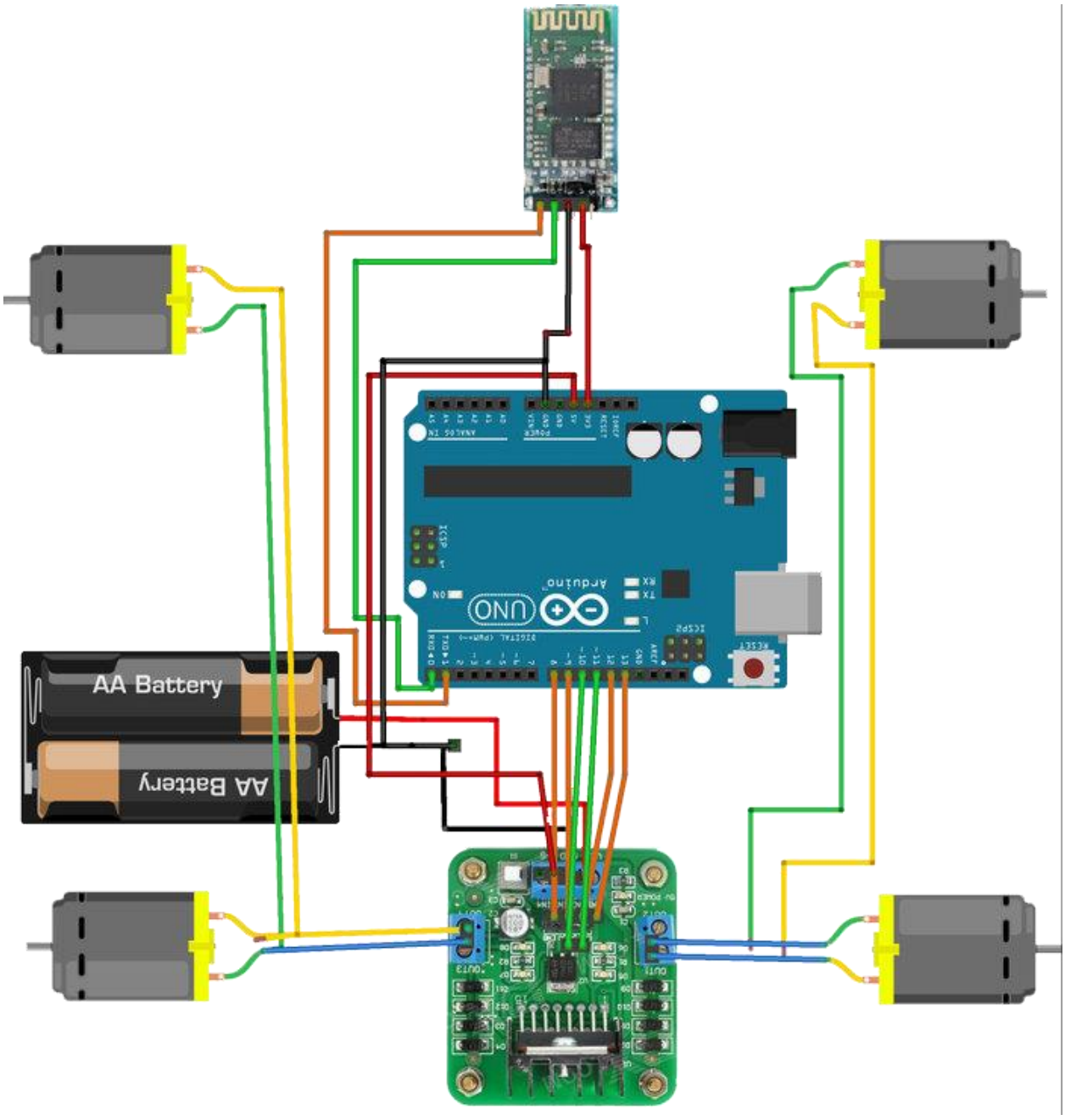


Ultrasonik Mesafe Sensörü Bağlantısı



Bluetooth Sensörü Bağlantısı

Led Bağlantısı

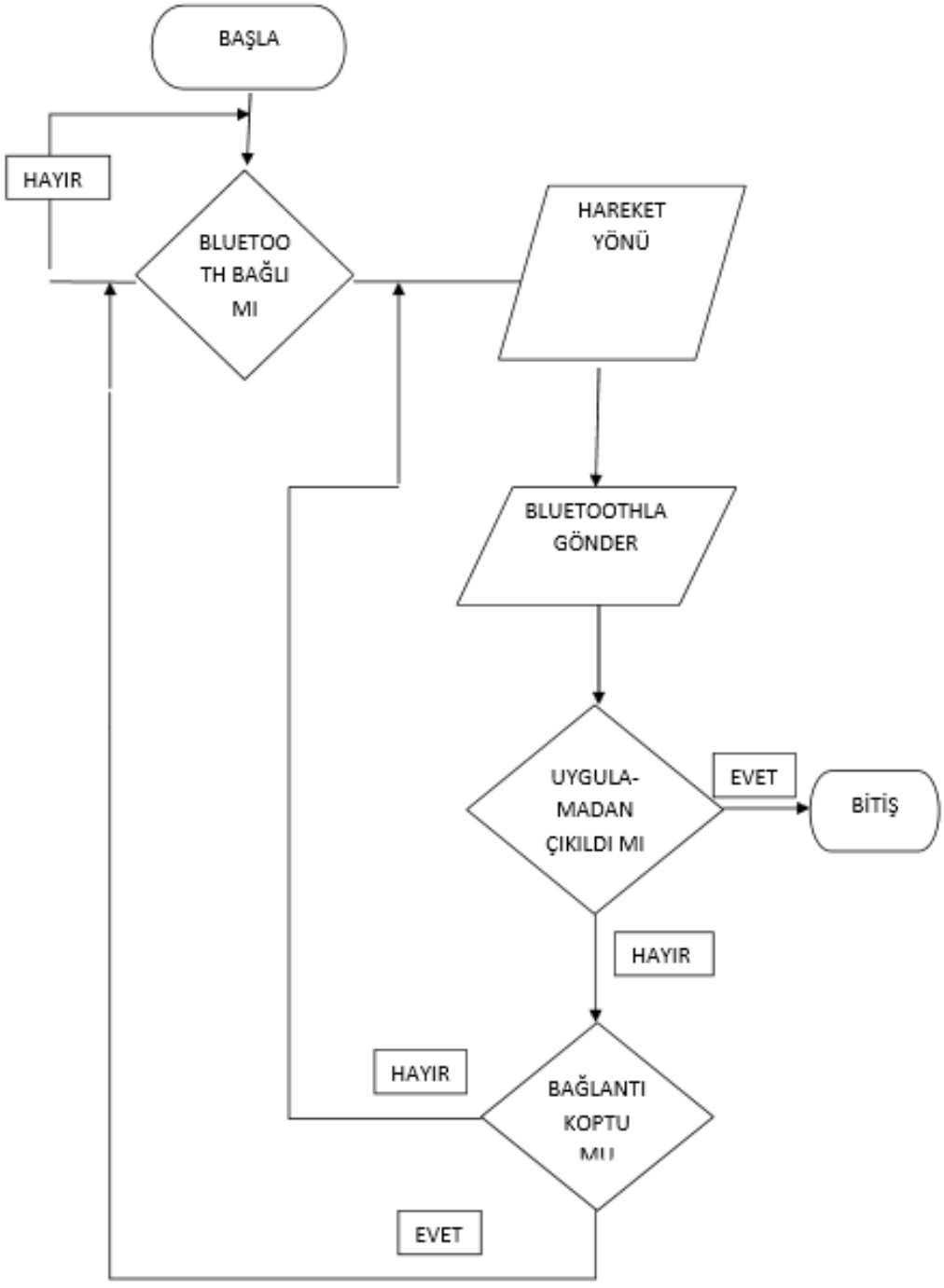


Motor Sürücü Bağlantısı

4.1.2 Varsayımlar ve Kısıtlamalar

- ✓ HC SR04 – Arduino Ultrasonik Mesafe Sensörü
- ✓ HC05 – Arduino Bluetooth Sensörü
- ✓ LED – Arduino arabası için lambalar
- ✓ L298N – Arduino Motor Sürücü

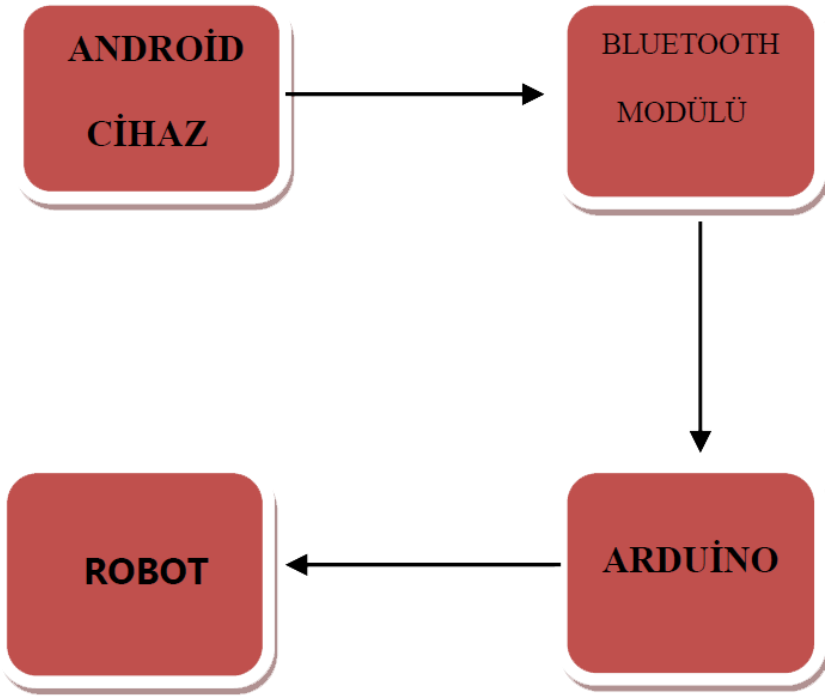
4.1.3 Sistem Mimarisi: Sistemin mimarisinin akış diyagramı şeklinde verilmesinin temel nedeni sistemin işleyiş mantığının nasıl olduğu ve nasıl bir yol çizileceğinin bilinmesidir.Akış diyagramı sistemin temel mantığı hakkında bize fikir verecektir.



Uygulama Akış Diyagramı

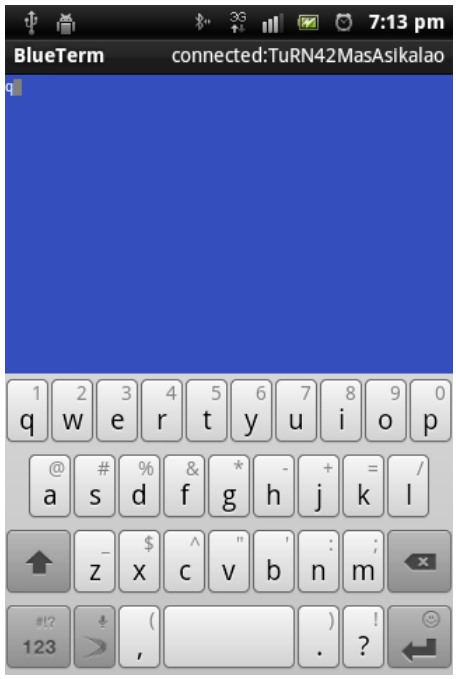
Akıllı telefon uygulaması yapılırken donanım elemanlarıyla telefonun haberleşmesi zorunludur. Bunun için kablosuz iletişim çok daha önemlidir. Projede bu iletişimi sağlamak için HC-05 bluetooth modülü kullanıldı.

Telefon uygulamamız bluetooth vasıtasıyla robotu kontrol edecek bilgileri üzerinde mikrodenetleyicisi bulunan arduinoya HC-05 üzerinden göndermektedir. Kullanılan robotumuz 2 adet DC motora sahip olup ileri,geri,sağ ve sol yönlerde hareket edebilmektedir. Kumandadan yapılan mekanik kontrol yerine arduino tarafından kontrol edilmesi gerekmektedir. Ana kumanda tarafından sağlanan analog sinyallerin bizim tarafımızdan ayarlanması sağlanmıştır.

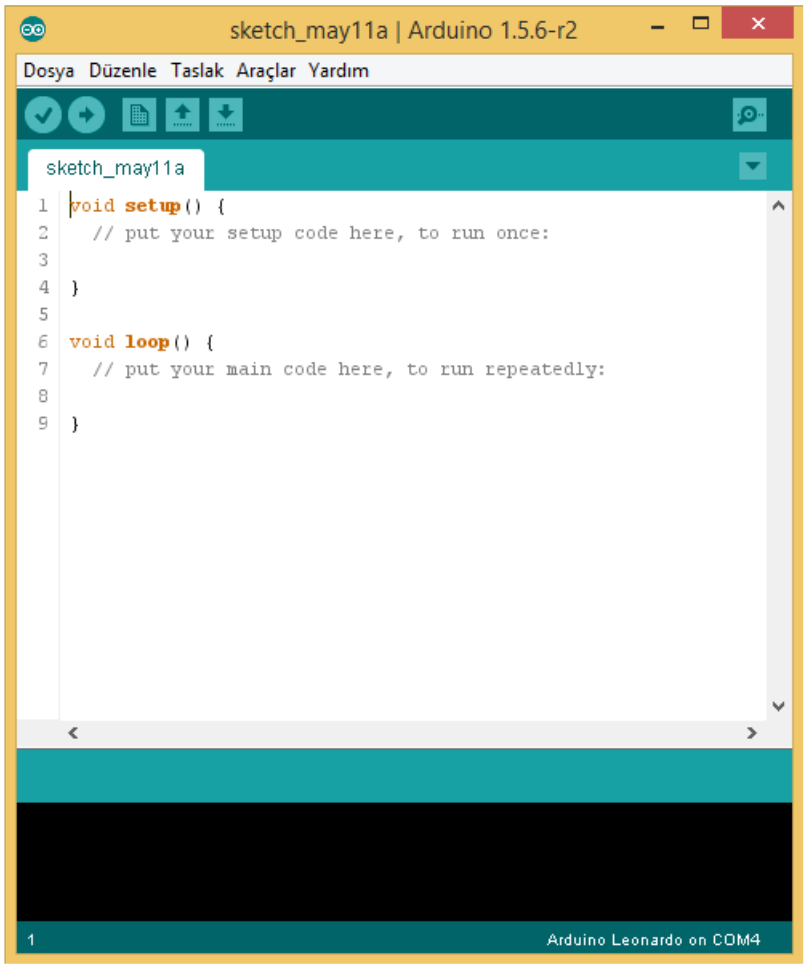


4.1.4 Dış Arabirimler

4.1.4.1 Kullanıcı Arabirimleri: Kullanıcı arabirimlerin ilk başında sistem giriş ekranı bulunacak. Kullanıcı bilgilerini girerek bu sisteme giriş yapmış olacaktır.



4.1.4.2 Veri Arabirimleri: Arduino IDE kodları yazdığımız derlediğimiz ve arduino kartına yüklediğimizJava diliyle yazılmış bir programdır. Herhangi bir programlama dilinde yazılmış olan bir programı derlemekteki amacımız ise, yazdığın bir programı (programlama dilinde), makine diline çevirmek ve çalışan kullanılabilir bir program elde etmektir.



4.1.4.3 Diğer Sistemlerle Arabirimler: Şuan tam bilgi sahibi olmadığımız için bu arabirim kullanılmayacaktır.

4.1.5 Testler

- ✓ Yazılım Testi: Yazılım kodlama aşamasında programcı tarafından oluşabilecek hataları gidermek amacıyla yapılır.

- ✓ Yeterlilik Testi: Yazılımın istenilen şekilde yapılıp yapılmadığını kontrol etmek amacıyla yapılır. Yani yazılım istenenleri tam olarak karşılıyor mu sorusuna cevap olarak yapılır.

- ✓ Sistem Testi: Yoğun veri akışı altında komple yükleme(load) testleri, normal olmayan koşullarda komple sistemin nasıl davranacağını görmek amacıyla germe(stres) testleri, istemli bir şekilde sistemi çökerterek sistemin nasıl davranacağını tespit etmek amacıyla geri kazanım(recovery) testleri, yazılımın geliştirilmesinde birimde yapay verilerle kabul testi, sistemin kullanılacağı yerde asıl verilerle kullanım hattı testleri, ve bundan sonra deneme testleri yapılır.

4.1.6 Performans:

- ✓ Sistemin performansını etkileyen faktörlerin test verileri değerlendirilecek
- ✓ Sistemin Tasarıma Uygunluk Performansı;
- ✓ Tasarımı yapılan sistemin stabilitesi ve işleyiş performansı değerlendirilecek.
- ✓ Veri Yapısının Sistemle Performansı;
- ✓ Veri yapısının sistemle stabilitesi ve çalışma zamanındaki uyumluluk düzeyindeki performansı değerlendirilecek.

4.2 Süreç Tasarımı

4.2.1 Genel Tasarım: Çizimden yola çıkılarak işlem türlerinin bölgeleri tanımlanır ve bu bölgelere karşı düşecek yapısal öğeler ortaya çıkarılmış olur. İstenilen yapı diyagramı kontrol hiyerarşisini de göstermektedir.

4.2.2 Modüller:

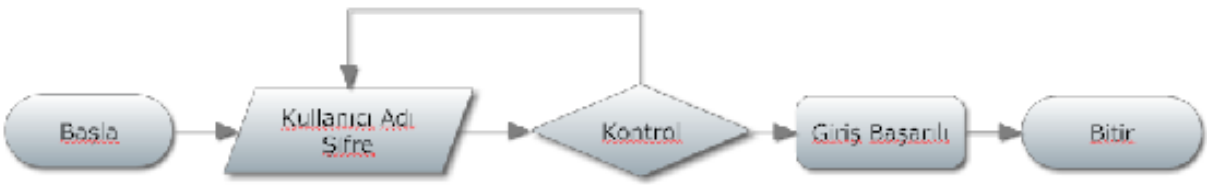
4.2.2.1 Giriş Modülü

4.2.2.1.1 İşlev: Kullanıcının sisteme müdahale edecebileceği ekrana erişmesi için aşması gereken bir modüldür.

4.2.2.1.2 Kullanıcı Arabirimi: Her müşterisinin erişebileceği ve kullanabileceği bir modüldür.

4.2.2.1.3 Modül Tanımı: Admin gibi özel müşterilerin erişebileceği gizli mir modüldür.

4.2.2.1.4 Modül iç Tasarımı



4.2.2.2 Yönetici Modülü

4.2.2.2.1 İşlev: Sistem içindeki tüm arabirimler aslına bakacak olunursa yönetim modülüdür. Arayüzler tamamı ile birbirinin aynısı lakin işlevler farklıdır. Butonların isimleri farklıdır.

4.2.3 Entegrasyon ve Test Gereksinimleri: Yazılımın ayrı ayrı ve birbirleriyle ilişkili şekilde test edilmesi gerekmektedir. Kullanıcı modülü için ayrı ayrı hava durumlarında denenmelidir. Diğer programların tam verimli çalışıp çalışmadığı proje ekibi tarafından detaylı bir şekilde test edilmelidir. Bütün modüller test edilip çalıştığına karar verildiğinde hepsinin arduino kullanılarak test işlemi yapılmalıdır.

4.3 Ortak Alt Sistemlerin Tasarımı

4.3.1 Ortak Alt Sistemler: Herhangi bir bilgi sistemi tasarlanırken, hemen hemen tüm bilgi sistemlerinde ortak olarak bulunan bazı alt sistemlerin dikkate alınması gerekmektedir.

4.3.1.1 Yetkilendirme Alt Sistemi: Uygulamalarda farklı kullanıcıların kullanabilecekleri ve kullanamayacakları özellikleri ifade eder.

- ✓ İşlev bazında yetkilendirme
- ✓ Ekran bazında yetkilendirme
- ✓ Ekran alanları bazında yetkilendirme

4.3.1.2 Güvenlik Alt Sistemi: Arduino açık kaynak kodlu olduğu için paylaşıma açıktır.

4.3.1.3 Yedekleme Alt Sistemi: Her bilgi sisteminin olağandışı durumlara hazırlıklı olmak amacıyla kullandıkları (sistem) yedekleme ve yedekten geri alma işlemlerinin olması gerekmektedir.

4.3.1.4 Veri İletişim Alt Sistemi: Arduino unodaki kodları diğer parçalarla etkileşim halinde olmasıdır.

- ✓ Arduino uno içindeki kodun dağılımı
- ✓ Diğer parçaların bu koda uygun çalışması

4.3.1.5 Arşiv Alt Sistemi: Belirli bir süre sonrasında sık olarak kullanılmayacak olan bilgilerin ayrılması ve gerektiğinde bu bilgilere erişimi sağlayan alt sistemlerdir.

4.3.1.6 Dönüştürme Alt Sistemi: Geliştirilen bilgi sisteminin uygulamaya alınmadan önce dönüştürme (mevcut sistemdeki verilerin yeni bilgi sistemine aktarılması) işlemlerine ihtiyaç vardır.

5. SİSTEM GERÇEKLEŞTİRİMİ

5.1 Giriş

Gerçekleştirim çalışması, tasarım sonucu üretilen süreç ve fiziksel yapısını içeren fiziksel modelin bilgisayar ortamında çalışan yazılım biçimine dönüştürülmesi çalışmalarını içerir. Yazılımın geliştirilmesi için her şeyden önce belirli bir yazılım geliştirme ortamının seçilmesi gerekmektedir.

5.2 Yazılım Geliştirme Ortamları

Yazılım geliştirme ortamı, tasarım sonunda üretilen fiziksel modelin, bilgisayar ortamında çalıştırılabilmesi için gerekli olan:

- Programlam Dil
- Hazır Program Kitapçıkları

5.3. Kodlama Stili

5.3.1 Açıklama Satırları

Bir yazılım geliştirirken kodların tekrar kullanılabilmesi, başkaları tarafından anlaşılabilmesi için kodlara açıklama satırları koyulur. Bunlar genel olarak;

✓ Bir paragraf şeklindeyse;

```
/*Açıklama
   Açıklama
   Açıklama */
```

✓ Tek bir satır ise;

```
//Açıklama
```

Şeklinde ifade edilir.

5.3.2 Kod Biçimlemesi

Kod biçimlenmesi açıklama satırlarına olan ihtiyacı azaltır. Kod biçimlemesinde önemli olan az satır değil kodun okunabilirliğidir. Bu projede bu kriterler göze aldık.

5.3.3 Anlamlı İsimlendirme

Kodların okunabilirliğini ve anlaşılabilirliğini sağlayan önemli unsurlardan biri de kullanılan ve kullanıcı tarafından belirlenen belirteçlerin (Değişken adları, kütük adları, işlev adları, yordam adları vb) anlamlı olarak isimlendirilmesidir.

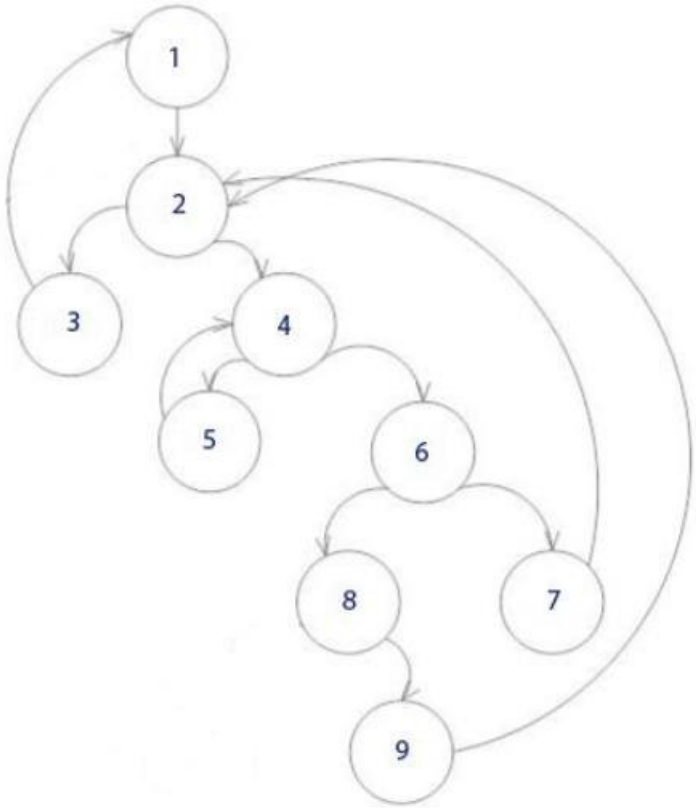
5.3.4 Yapısal Programlama Yapıları

Program kodlarının, okunabilirlik, anlaşılabilirlik, bakım kolaylığı gibi kalite etmenlerinin sağlanması ve program karmaşıklığının azaltılması amacıyla "yapısal programlama yapıları" kullanılarak yazılması önemlidir. Yapısal Programlama "tek giriş ve tek çıkışlı" öbeklerden oluşan yapılardır. Teorik olarak herhangi bir bilgisayar programının, yalnızca Yapısal Programlama Yapıları kullanılarak yazılabileceği kanıtlanmıştır.
Üç temel Yapısal Programlama Yapısı bulunmaktadır:

- ✓ Ardışıl işlem yapıları
- ✓ Koşullu işlem yapıları, Döngü yapıları

5.4. Program Karmaşıklığı

5.4.1 Programın Çizgi Biçimine Dönüştürülmesi



McCabe Çizgi Biçimi

5.4.2 McCabe Karmaşıklık Ölçütü Hesaplama

k = 12 Kenar sayısı

d = 9 Düğüm sayısı

p = 1 Bileşen sayısı

V(G)= k – d + 2p

V(G)=12 - 9 + 2 x 1 = 5

5.5. Olağan Dışı Durum Çözümleme

Olağan dışı durumlar gerek kod yazım sürecinde gerekse testler sırasında gerçekleşebilir. Projede Helezonik Model kullanıldığından dolayı her aşamada test yapılacağından olağan dışı durum anında çözülebilecektir.

5.6. Kod Gözden Geçirme

5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi

- Gözden geçirme sürecinin temel özellikleri;

Hataların bulunması, ancak düzeltilmemesi hedeflenir,
Kalite çalışmalarının bir parçası olarak ele alınmalı ve sonuçlar düzenli ve belirlenen bir biçimde saklanmalıdır. Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir. Gözden Geçirme çalışmasının olası çıktıları biçimindeözetlenebilir. Burada yanıtı aranan temel soru, programın yazıldığı gibi çalışıp çalışmayacağının belirlenmesidir.

5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular

- ✓ Kod doğru bir şekilde build edildi mi? Kaynak kod derlendiğinde hata olmamalı. Kodda yapılan değişikliklerde uyarılar(warning) olmamalı.
- ✓ Kod çalıştırıldığında beklendiği gibi davrandı mı?
- ✓ Kodun sadece çalışırlığına bakılmamalı, kod tasarımı da göz önün de bulundurulmalıdır. Optimizasyon için öneriler takım olarak değerlendirilmelidir.
- ✓ Gözden geçirilen kod anlaşılıyor mu? Gözden geçiricinin kodu anlaması gerekir. Eğer anlaşılmadıysa, gözden geçirme tamamlanmış olmaz veya kod iyi yorumlanabilmiş sayılmaz.
- ✓ Geleneksel kodlama standartlarına uyuldu mu? Değişken isimlendirme, satır başıboşluklar, parantez stilleri vs. takip edilmeli.
- ✓ Telif hakkı bilgisi ve uygun bir başlıkla başlayan kaynak dosya var mı? Her bir kaynak dosyası bu bilgilerle başlamalı, bütün kaynak dosyaları, fonksiyonelliğini anlatan bir dosya içermelidir.**Arduino açık kaynak kodludur.**
- ✓ Değişken deklarasyonlarına yorum satırları eklenmiş mi? Yorumlar, değişkenlerin görevlerini açıklaması gerekir. Özellikle her bir global değişkenin amacı ve neden global olarak tanımlandığı belirtilmelidir.
- ✓ Sayısal verilerin birimleri açıkça belirtilmiş mi? Sayısal verilerin birimleri yorum satırı olarak belirtilmeli. Örneğin, eğer bir sayı uzunluğu temsil ediyorsa, metre mi feet mi olduğu gösterilmelidir.
- ✓ Bütün fonksiyonlar, metotlar ve classlar dokümante edilmiş mi? Her bir fonksiyon, metot ve class'ın tanımlanmasının üstünde bir iki cümle ile açıklaması yer almalıdır. Amacı vurgulamalı ve tasarım gerekliliklerini işaret etmelidir.
- ✓ Fonksiyonların kullandığı input ve output parametreleri açıkça tanımlandı mı?
- ✓ Karmaşık algoritmalar ve kod optimizasyonları yeterli olacak şekilde açıklanmış mı? Karmaşık alanlar, algoritmalar ve kod optimizasyonları için yeterince yorum satırı eklenmelidir. Öyle ki, diğer geliştiriciler kodu anlayabilmeli ve kalınan yerden devam ettirebilmelidir.
- ✓ Kodun çeşitli yerlerinde yorum satırlarıyla açıklamalar var mı? Kodun çeşitli yerlerinde yorum satırlarıyla açıklamalar olmalı. "Ölü Kod"lar çıkarılmalı. Eğer geçici bir kod bloğu ise neden tanımlandığı belirtilmeli. Koddaki eksik işlevsellikler veya çözümlenmemiş sorunlar yorum satırlarında ifade edilmiş mi? Bu ifadeler eksikleri ve yapılacakları açıklamalıdır. Sonradan arandığında bulunabilmesi için de ayırıcı bir işaretleyici kullanılmalı.
- ✓ Her zaman bir fonksiyonun döndürebileceği hatalar düzün bir şekilde handle edilmeli. Fonksiyonun üreteceği her bir sonuç düşünülmeli, her durum kontrol edilmeli ve kodun kalan kısmının yürütülmesini etkileyen hatalar yakalanmış olmalıdır.
- ✓ Alınan hatalardan sonra tüm kaynaklar ve hafıza temizlenip serbest bırakılıyor mu? Bundan emin olunmalı. Bir hata meydana geldiğinde, dosya, soket vb. bağlantı objeleri gibi tüm objeler bağlanmalıdır.
- ✓ Tespit edilen hata kodun başka yerlerini de etkiliyor mu, kontrol edilmeli? Hatanın tüm ekranlarda giderildiğinden emin olunmalı. Kodun değişiklik yapılan yerlerinde, eski halini ve neden yapıldığını mutlaka açıklama olarak eklenmelidir.
- ✓ Kodda yapılmış yorumlar değerlendirilmeli, eğer yorumun uygun/doğru olmadığı düşünülüyorsa geliştirici ile görüşülmeli ve konu tartışıldıktan sonra çözüme kavuşturulmalıdır.

5.6.2.1 Öbek Arayüzü

- ✓ Her öbek tek bir işlevsel amacı yerine getiriyor mu?
- ✓ Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi?
- ✓ Öbek tek giriş ve tek çıkışlı mı?
- ✓ Öbek eğer bir işlev ise, parametrelerinin değerini değiştiriyor mu?

5.6.2.2 Giriş Açıklamaları

- Öbek, doğru biçimde giriş açıklama satırları içeriyor mu?
- Giriş açıklama satırları, öbeğin amacını açıklıyor mu?
- Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu?
- Giriş açıklama satırları, çıktıları ve hata iletilerini tanımlıyor mu?
- Giriş açıklama satırları, öbeğin algoritma tanımını içeriyor mu?
- Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu?
- Giriş açıklama satırları, öbekteki olağan dışı durumları tanımlıyor mu?
- Giriş açıklama satırları, öbeği yazan kişi ve yazıldığı tarih ile ilgili bilgileri içeriyor mu?
- Her paragrafı açıklayan kısa açıklamaları var mı?

5.6.2.3 Veri Kullanımı

- o İşlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı?
- o Değişken adları, işlevlerini yansıtacak biçimde anlamlı mı?
- o Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı? Her değişken tek bir amaçla mı kullanılıyor?
- o Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış?
- o Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı?

5.6.2.4 Öbeğin Düzenlenişi

- ✓ Algoritmalar istenen işlevleri karşılıyor mu?
- ✓ Arayüzler genel tasarımla uyumlu mu?
- ✓ Mantıksal karmaşıklık anlamlı mı?
- ✓ Belirlenen tasarım standartlarına uyulmuş mu?
- ✓ Hata çözümleme tanımlanmış mı?
- ✓ Tasarım, kullanılacak programlama diline uygun mu?
- ✓ İşlerim sistemi ve programlama diline yönelik kısıtlar ya da özellikler kullanılmış mı?
- ✓ Bakım dikkate alınmış mı?

5.6.2.5 Sunuş

- ✓ Her satır, en fazla bir deyim içeriyor mu?
- ✓ Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşılabilirliği kolaylaştıracak biçimde anlamlı mı?
- ✓ Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı?
- ✓ Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı? Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı?
- ✓ Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mı?

6. DOĞRULAMA VE GEÇERLEME

6.1 Giriş

Geliştirilecek bilgi sistemi yazılımının doğrulanması ve geçerlenmesi, üretim süreci boyunca süren etkinliklerden oluşur. Söz konusu etkinlikler:

Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.

Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması.

Projenin bir aşaması süresince geliştirilen anahtar belirtimlerin önceki belirtimlerle karşılaştırılması.

Yazılım ürünlerinin tüm uygulanabilir gerekleri sağladığının gerçeklenmesi için sınamaların hazırlanıp yürütülmesi biçiminde özetlenebilir.

6.2. Sınama Kavramları

Sınama ve Bütünleştirme işlemlerinin bir strateji içinde gerçekleştirilmesi, planlanması ve tekniklerinin seçilmesi gerekmektedir.

- 6.2.1 Birim Sınama: Bağlı oldukları diğer sistem unsurlarından tümüyle soyutlanmış olarak birimlerin doğru çalışmalarının belirlenmesi amacıyla yapılır.
- 6.2.2 Alt-Sistem Sınama: Alt-sistemler modüllerin bütünleştirilmeleri ile ortaya çıkarlar. Yine bağımsız olarak sınamaları yapılmalıdır.Bu aşamada en çok hata arduino devresinde bulunmaktadır. Bu yüzden arduino devresindeki hatalarına doğru yoğunlaşılmalıdır.
- 6.2.3 Sistem Sınaması: Üst düzeyde, bileşenlerin sistem ile olan etkileşiminde çıkacak hatalar aranmaktadır.Ayrıca, belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.
- 6.2.4 Kabul Sınaması: Çalıştırılmadan önce sistemin son sınamasıdır. Artık, yapay veriler yerine gerçek veriler kullanılır.Bu sınama türü alfa sınaması veya beta sınaması olarak ta bilinir.

6.3. Doğrulama ve Geçerleme Yaşam Döngüsü

Gerçekleştirim aşamasına kadar olan süreçlerde doğrulama ve geçerleme işlemlerinin planlaması yapılır. Planlama genellikle;

- alt-sistem,
- bütünleştirme,
- sistem ve
- kabul sınamalarının

tasarımlarını içerir. Gerçekleştirim aşamasının sonunda ise söz konusu plan uygulanır.

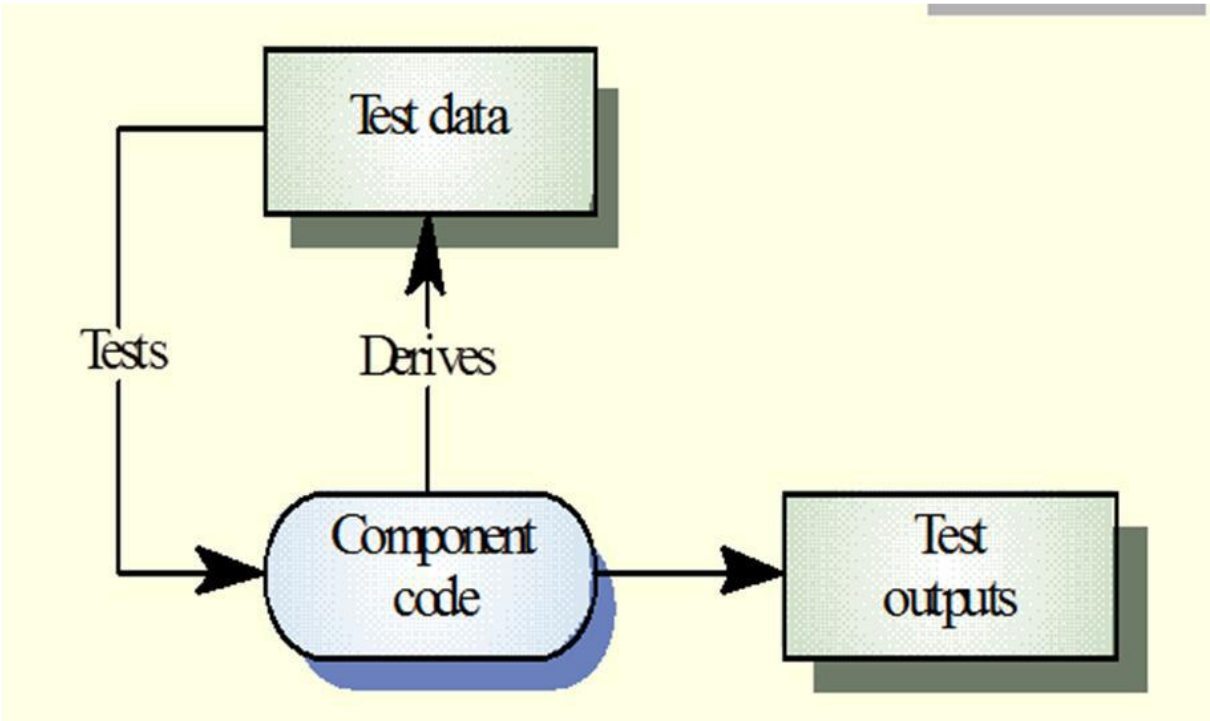
6.4. Sınama Yöntemleri

Sınama işlemi, geliştirmeyi izleyen bir düzeltme görevi olmak ile sınırlı değildir. Bir "sonra" operasyonu olmaktan çok, geliştirme öncesinde planlanan ve tasarımı yapılması gereken bir çaba türüdür.

6.4.1 Beyaz Kutu Sınaması

İç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tabanında sınanmasıdır.

- Bütün bağımsız yolların en az bir kez sınanması gerekir.
- Bütün mantıksal karar noktalarında iki değişik karar için sınamalar yapılır.
- Bütün döngülerin sınır değerlerinde sınanması
- İç veri yapılarının denenmesi



6.4.2 Kara Kutu Sınaması

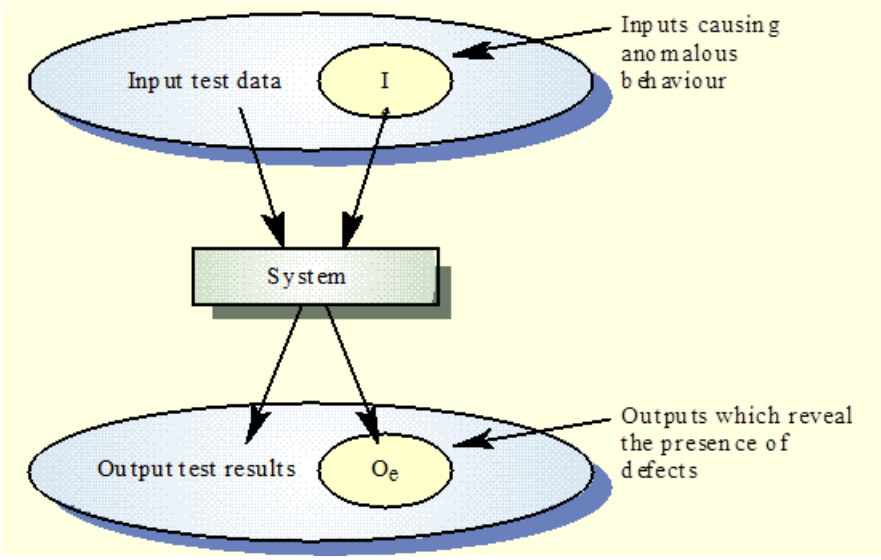
Sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün testidir. Sistem şartnamesinin gerekleri incelenir.

Ürünlerin test edilmesi sırasında kullanılan en ilkel test metodudur. Bir takım test senaryolarının seçilip, yazılım kodundan bağımsız olarak takip edilmesi temeline dayanmaktadır. Bu yüzden ürünlerin fonksiyonel durumları ve inputlara verdikleri tepkilerin gözlenmesi uygulamada kullanılan kara kutu testlerinin kapsamını oluşturmaktadır. Bu noktada, yazılımın kodunda yapılan herhangi bir değişiklik veya data yapısındaki uyarlamalar kara kutu testleriyle kontrol edilen özellikler değildir.

Test edilecek olan uygulamanın kodu hiç dikkate alınmadan, sadece girdilerin ve çıktıların incelenmesi ile gerçekleştirilen test metodudur. 5 ayrı tekniği bilinir. Denklik sınıfı test tekniği, test verileri gruplanır. Gruplar içinde testler yapılır.

- Uç nokta test tekniği, hataların genelde sınırlarda çıktığı varsayılarak sınır değerlerinde test yapılır.
- Karar tablosu test tekniği, çok fazla test yapılması gereken uygulamalarda verilerin matrix haline getirilerek test edilmesi test edilmesidir.

- Sistem durumu test tekniği, farklı durum geçişleri yer alan sistemlerin testleridir.
- İş senaryosu test tekniği, use case dokümanlarının kullanıldığı test tekniğidir.



6.5.Sınama ve Bütünleştirme Stratejileri

Genellikle sınama stratejisi, bütünleştirme stratejisi ile birlikte değerlendirilir. Ancak bazı sınama stratejileri bütünleştirme dışındaki tasaları hedefleyebilir. Örneğin, yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağımlıdır. Ancak işlem yolu ve gerilim sınamaları, sistemin olaylar karşısında değişik işlem sıralandırmaları sonucunda ulaşacağı sonuçların doğruluğunu ve normal şartların üstünde zorlandığında dayanıklılık sınırını ortaya çıkarır.

6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme

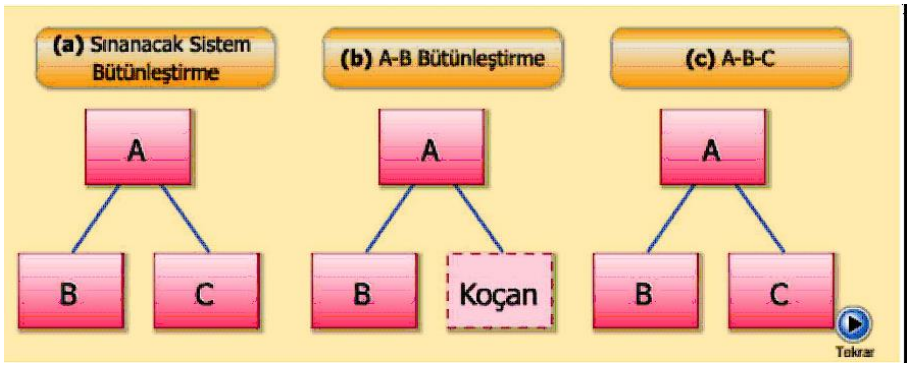
Yukarıdan-aşağıya bütünleştirmede önce sistemin üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeylere ilgili modülleri takılarak sınanması söz konusudur.

En üst noktadaki bileşen sınandıktan sonra alt düzeye geçilmelidir.

Alt bileşenler henüz hazırlanmamışlardır. Bu sebeple Koçanlar kullanılır. Koçan; Bir alt bileşenin, üst bileşen ile ara yüzünü temin eden, fakat işlevsel olarak hiçbir şey yapmayan çerçeve programlardır.

İki temel yaklaşım vardır:

- Düzey Öncelikli Bütünleştirme: En üst düzeyden başlanır ve aynı düzeydeki birimler bütünleştirilir.
- Derinlik Öncelikli Bütünleştirme: En üst düzeyden başlanır ve her dal soldan sağa olmak üzere ele alınır.

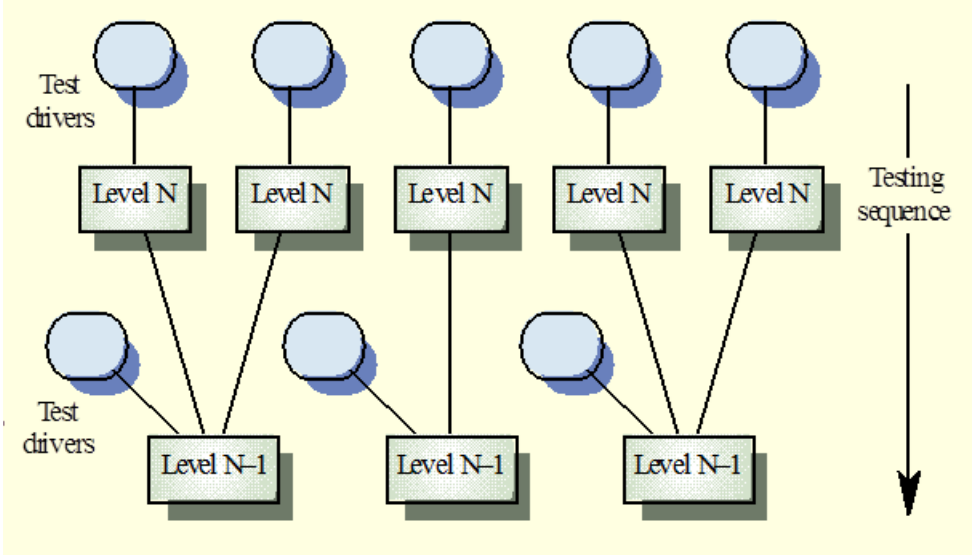


6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme

Önceki yöntemin tersine uygulama yapılır.

Önce en alt düzeydeki işçi birimler sınanır ve bir üst düzey ile sınanması gerektiğinde bu düzey bir sürücü ile temsil edilir.

Bu kez kodlama, bütünleştirme ve sınama, aşağı düzeylerden yukarı düzeylere doğru gelişir.



6.6. Sınama Planlaması

Her sınama planı, sınama etkinliklerinin sınırlarını, yaklaşımını, kaynaklarını ve zamanlamasını tanımlar. Plan neyin sınanacağını, neyin sınanmayacağını, sorumlu kişileri ve riskleri göstermektedir. Sınama planları, sınama belirtimlerini içerir.

1.7			PROJE DESTEGİNİN SAGLANMASI	?25 gün	Çar 14.06.17	Sal 18.07.17	16	proje lideri;yazılım lideri
1.7.1			proje destegi için destekçi bulunması	8 gün	Çar 14.06.17	Cum 23.06.17		proje lideri
1.7.2			projenin destekçiye anlatılması	3 gün	Pzt 26.06.17	Çar 28.06.17	25	proje lideri
1.7.3			proje destekçisi ile planlama yapılması	5 gün	Per 29.06.17	Çar 5.07.17	26	proje lideri
1.7.4			destekçiye maliyet kestirim sunulması	4 gün	Per 6.07.17	Sal 11.07.17	27	proje lideri
1.7.5			proje destekcisinden sonuç alınması	5 gün	Çar 12.07.17	Sal 18.07.17	28	proje lideri
1.7.6			proje destekçisi ile toplantı yapılması	4 gün	Çar 12.07.17	Pzt 17.07.17	28	proje lideri;tasarımcı;yazılım lideri
1.7.7			proje çalışmalarına başlanması	?1 gün	Sal 18.07.17	Sal 18.07.17	30	proje ekibi

6.7. Sınama Belirtimleri

Sınama belirtimleri, bir sınama işleminin nasıl yapılacağına ilişkin ayrıntıları içerir.

Bu ayrıtılar temel olarak:

- Sınanan program modülü ya da modüllerinin adları,
- Sınama türü, stratejisi (beyaz kutu, temel yollar vb.),
- Sınama verileri, Sınama senaryoları türündeki bilgileri içerir.

Sınama verilerinin elle hazırlanması çoğu zaman kolay olmayabilir ve zaman alıcı olabilir. Bu durumda, otomatik sınama verisi üreten programlardan yararlanılabilir.

Sınama senaryoları, yeni sınama senaryosu üretebilmeye yardımcı olacak biçimde hazırlanmalıdır. Zira sınama belirtimlerinin hazırlanmasındaki temel amaç, etkin sınama yapılması için bir rehber oluşturmasıdır. Sınama işlemi sonrasında bu belirtimlere,

- Sınamayı yapan, Sınama tarihi,
- Bulunan hatalar ve açıklamaları türündeki bilgiler eklenerek sınama raporları oluşturulur.

6.8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri

Bütün bu etkinlikleri bir hiyerarşi altında incelemek gerekirse:

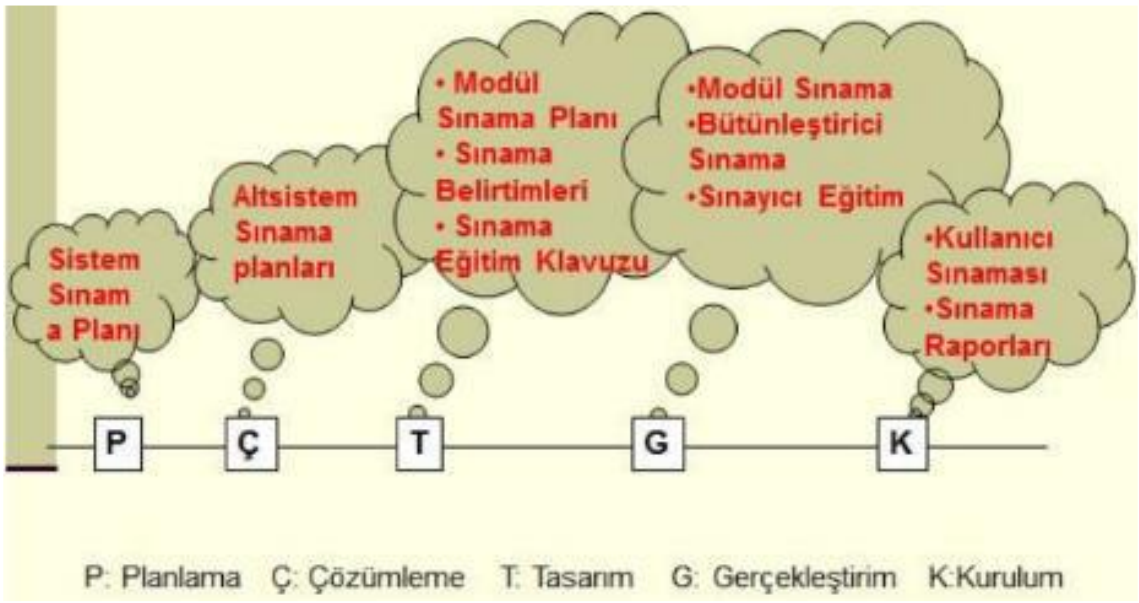
Planlama aşamasında genel planlama sınaması gerçekleştirilir. Bu olan tüm planların basit bir ön hazırlığı niteliğindedir.

Çözümleme aşamasında sınama planı alt sistemler bazında ayrıntılandırılır.

Tasarım aşamasında sınama plana detaylandırılır ve sınama belirtimleri oluşturulur. Bu oluşumlar daha sonra eğitim ve el kitabında kullanılır.

Gerçekleştirim aşamasında teknik sınamalar yapılır sınama raporları hazırlanır ve elle tutulur ilk testler yapılır.

Kurulum aşamasında sistemle ilgili son sınamalar yapılır ve sınama raporları hazırlanır.



7.BAKIM

7.1 Giriş

Sistemin tasarımı bittikten sonra artık seçimden seçime sistemin bakıma sokulması gerekir daha öncede belirttiğimiz gibi sistem hassas ve hata kabul etmeyecek bir sistemden bahsediyoruz.

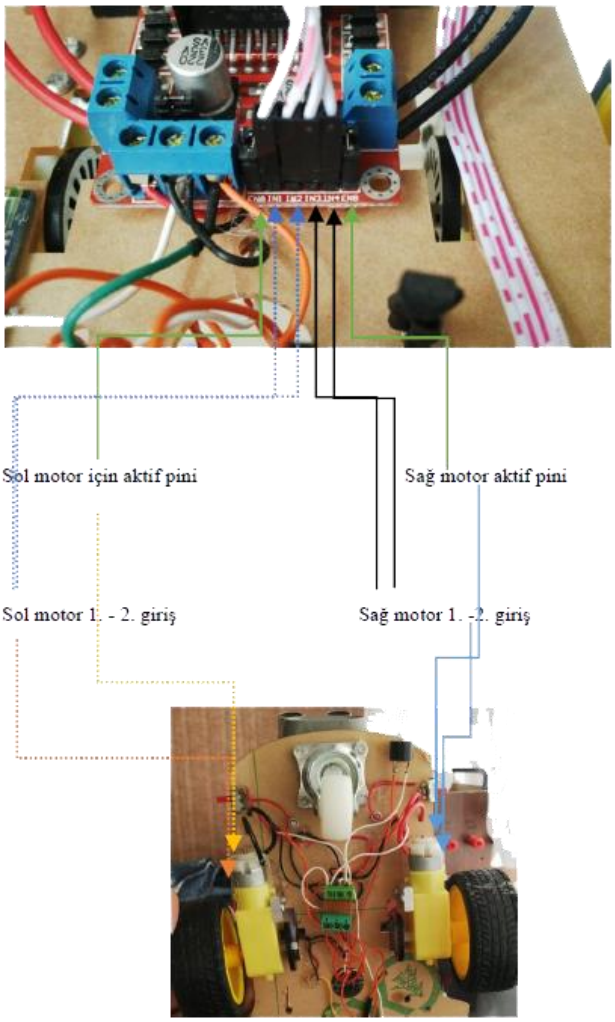
Yazılımın dağıtılması ve kullanıma başlanmasından sonra yazılımda yapılacak değişiklikler yazılımın bakımı (software maintenance) olarak adlandırılır. Bu değişiklikler basit kodlama hatalarının düzeltilmesi (bug-fixes) şeklinde olabileceği gibi tasarımdan kaynaklanan hataların giderilmesi gibi daha kapsamlı değişiklikler şeklinde de olabilir. Yazılımın bakımı aslında yazılımın evrimleşmesidir. Yazılımın yaşamına devam edebilmesi için gerekli değişikliklerin uygulanmasıdır.

7.2 Kurulum

Herhangi bir kurulum yoktur Arduino devreleri karşı tarafa hazır olarak verilir ve Arduino UNO'ya gerekli kodlar atılarak kullanıcıya verilir. Kullanıcın yapması gereken tek şey ürünün kullanacağı zaman on kullanmadığı zaman off yapmasıdır.

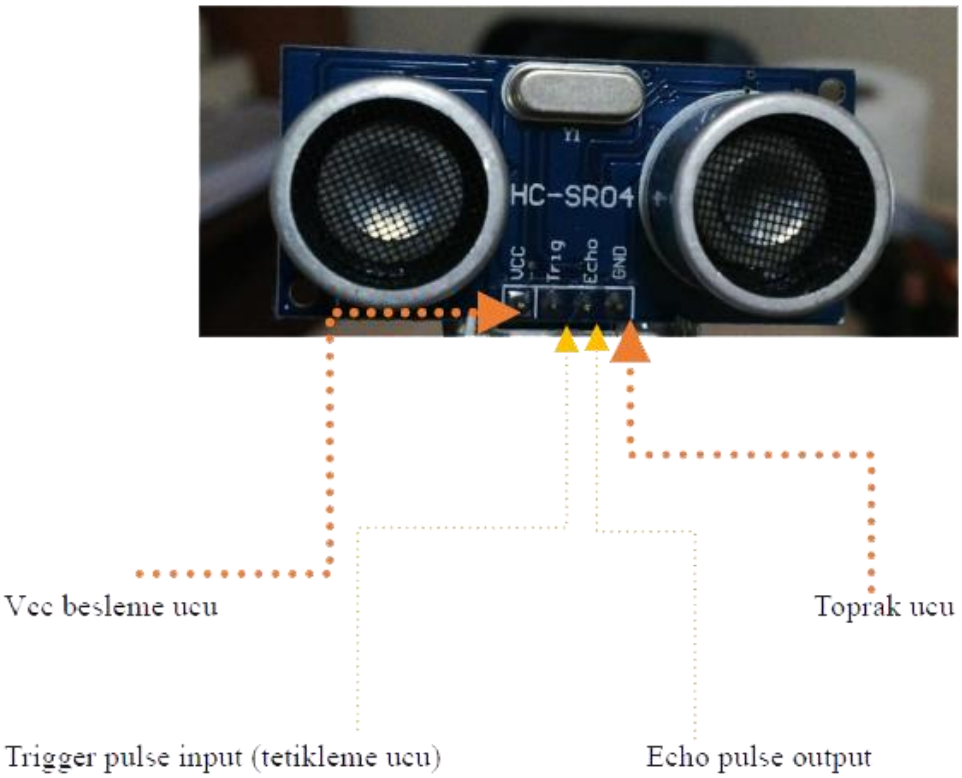
7.2.1 Sürücü Devre Çalışmamız

Aracımızın hareketini sağlayan iki adet doğru akım motoru bulunmaktadır. Bu motorlar L298 sürücü devresi beslemektedir. L298 sürücü devresindeki motorların bağlantı noktaları gösterilmiştir. Bu çıkışlardan aldığımız kabloları doğru akım motorların giriş uçlarına bağladık.



7.2.2 Sensör Çalışmamız

Aracımızda mesafe ölçme amacıyla HC-SR04 modeli ultrasonic sensör kullandık, bu modelin dört bacağı vardır bunlar besleme ucu, tetikleme ucu, echo ucu ve toprak ucudur. Aracımızda kullandığımız ultrasonnic sendörün görünümü aşagıda verilmiştir.



Şekil 39. HC-SR04 sensörü

7.3 Yazılım Bakımı

Yazılım bakımı yapılırken şu programlardan faydalanılacaktır.

Yazılımsal bir sıkıntı olduğu zaman C++ ve Arduino IDE ile gerekli eksiklikler tamamlanacaktır.

8.KODLAR

```
int buzzer = 10;
const int sagileri = 2;
const int saggeri = 4;
const int solileri = 5;
const int solgeri = 6;
const int solenable = 11;
const int sagenable = 3;

int geripin1 = 14;
int geripin2 = 15;
int ileripin1 = 16;
int ileripin2 = 17;
int sagpin1 = 21;
int sagpin2 = 22;
int solpin1 = 19;
int solpin2 = 18;

void setup() {
  pinMode(sagileri, OUTPUT);
  pinMode(saggeri, OUTPUT);
  pinMode(solileri, OUTPUT);
  pinMode(solgeri, OUTPUT);
  pinMode(sagenable, OUTPUT);
  pinMode(solenable, OUTPUT);
  pinMode(buzzer, OUTPUT);

  pinMode(geripin1, OUTPUT);
  pinMode(geripin2, OUTPUT);
  pinMode(ileripin1, OUTPUT);
  pinMode(ileripin2, OUTPUT);
  pinMode(sagpin1, OUTPUT);
  pinMode(sagpin2, OUTPUT);
  pinMode(solpin1, OUTPUT);
  pinMode(solpin2, OUTPUT);

  Serial.begin(9600);
}
```

```
void siren() {
  digitalWrite(ileripin1, LOW);
  digitalWrite(ileripin2, LOW);
  digitalWrite(geripin1, LOW);
  digitalWrite(geripin2, LOW);
  digitalWrite(sagpin1, LOW);
  digitalWrite(sagpin2, LOW);
  digitalWrite(solpin1, LOW);
  digitalWrite(solpin2, LOW);
  for(int i=0; i<=5; i++){
    digitalWrite(buzzer, HIGH);
    delay(500);
    digitalWrite(buzzer, LOW);
    delay(500);
  }
}

void ileri() {
  analogWrite(sagenable, 255);
  analogWrite(solenable, 255);
  digitalWrite(sagileri, LOW);
  digitalWrite(saggeri, HIGH);
  digitalWrite(solileri, LOW);
  digitalWrite(solgeri, HIGH);
  // İLERİ PİNLERİ YAKAR
  digitalWrite(geripin1, LOW);
  digitalWrite(geripin2, LOW);
  digitalWrite(sagpin1, LOW);
  digitalWrite(sagpin2, LOW);
  digitalWrite(solpin1, LOW);
  digitalWrite(solpin2, LOW);
  digitalWrite(buzzer, LOW);
  for(int i=0; i<=5; i++){
    digitalWrite(ileripin1, HIGH);
    digitalWrite(ileripin2, HIGH);
    delay(500);
    digitalWrite(ileripin1, LOW);
    digitalWrite(ileripin2, LOW);
    delay(500);
  }
  digitalWrite(ileripin1, HIGH);
  digitalWrite(ileripin2, HIGH);
}
```

```
void geri() {
  analogWrite(sagenable, 200); //220 degeri pwm olarak motor hizini ayarliyor 255 e kadar
     çikabilir analogWrite komutunu bunun icin kullaniyoruz
  analogWrite(solenable, 200);
  digitalWrite(sagileri, HIGH);
  digitalWrite(saggeri, LOW);
  digitalWrite(solileri, HIGH);
  digitalWrite(solgeri, LOW);
  // GERİ PİNLERİ YAKAR
  digitalWrite(ileripin1, LOW);
  digitalWrite(ileripin2, LOW);
  digitalWrite(sagpin1, LOW);
  digitalWrite(sagpin2, LOW);
  digitalWrite(solpin1, LOW);
  digitalWrite(solpin2, LOW);
  digitalWrite(buzzer, LOW);
  for(int i=0; i<=5; i++){
    digitalWrite(geripin1, HIGH);
    digitalWrite(geripin2, HIGH);
    digitalWrite(buzzer, HIGH);
    delay(500);
    digitalWrite(geripin1, LOW);
    digitalWrite(geripin2, LOW);
    digitalWrite(buzzer, LOW);
    delay(500);
  }
  digitalWrite(geripin1, HIGH);
  digitalWrite(geripin2, HIGH);
}

void sag() {
  analogWrite(solenable, 150);
  analogWrite(sagenable, 255);
  digitalWrite(sagileri, HIGH);
  digitalWrite(saggeri, LOW);
  digitalWrite(solileri, LOW);
  digitalWrite(solgeri, HIGH);
  //SAG PİNLERİ YAKAR
  digitalWrite(ileripin1, LOW);
  digitalWrite(ileripin2, LOW);
  digitalWrite(geripin1, LOW);
  digitalWrite(geripin2, LOW);
  digitalWrite(solpin1, LOW);
  digitalWrite(solpin2, LOW);
  digitalWrite(buzzer, LOW);
  for(int i=0; i<=5; i++){
    digitalWrite(sagpin1, HIGH);
    digitalWrite(sagpin2, HIGH);
```

```
    delay(500);
    digitalWrite(sagpin1, LOW);
    digitalWrite(sagpin2, LOW);
    delay(500);
   }
  digitalWrite(sagpin1, HIGH);
  digitalWrite(sagpin2, HIGH);
}

void sol() {
  analogWrite(solenable, 255);
  analogWrite(sagenable, 150);
  digitalWrite(sagileri, LOW);
  digitalWrite(saggeri, HIGH);
  digitalWrite(solileri, HIGH);
  digitalWrite(solgeri, LOW);
  //SOL PİNLERİ YAKAR
  digitalWrite(ileripin1, LOW);
  digitalWrite(ileripin2, LOW);
  digitalWrite(geripin1, LOW);
  digitalWrite(geripin2, LOW);
  digitalWrite(sagpin1, LOW);
  digitalWrite(sagpin2, LOW);
  digitalWrite(buzzer, LOW);
  for(int i=0; i<=5; i++){
    digitalWrite(solpin1, HIGH);
    digitalWrite(solpin2, HIGH);
    delay(500);
    digitalWrite(solpin1, LOW);
    digitalWrite(solpin2, LOW);
    delay(500);
   }
  digitalWrite(solpin1, HIGH);
  digitalWrite(solpin2, HIGH);

}

void dur() {
  digitalWrite(sagileri, LOW);
  digitalWrite(saggeri, LOW);
  digitalWrite(solileri, LOW);
  digitalWrite(solgeri, LOW);
  //DUR PİNLERİ YAKAR
  digitalWrite(geripin1, HIGH);
  digitalWrite(geripin2, HIGH);
  digitalWrite(ileripin1, LOW);
  digitalWrite(ileripin2, LOW);
  for(int i=0; i<=20; i++){
```

```
    digitalWrite(sagpin1, HIGH);
    digitalWrite(sagpin2, HIGH);
    digitalWrite(solpin1, HIGH);
    digitalWrite(solpin2, HIGH);
    delay(500);
    digitalWrite(sagpin1, LOW);
    digitalWrite(sagpin2, LOW);
    digitalWrite(solpin1, LOW);
    digitalWrite(solpin2, LOW);
    delay(500);
  }
  digitalWrite(geripin1, HIGH);
  digitalWrite(geripin2, HIGH);
  digitalWrite(ileripin1, LOW);
  digitalWrite(ileripin2, LOW);
  digitalWrite(buzzer, LOW);
}

// LOOP FONKSIYONU
void loop() {
  if (Serial.available() > 0)
  {
    char tus = (char)Serial.read();
    Serial.println("KAMIL KAPLAN");
    if ( tus == 'w' ) {
      Serial.println("Araba ileriye hareket etti"); ileri();
    }
    if ( tus == 's' ) {
      Serial.println("Araba geriye hareket etti"); geri();
    }
    if ( tus == 'a' ) {
      Serial.println("Araba Sola dondu");   sol();
    }
    if ( tus == 'd' ) {
      Serial.println("Araba Saga dondu"); sag();
    }
    if (tus == 'x') {
      Serial.println("Araba DURDU"); dur();
    }
    if (tus == 'e') {
      siren();
    }
  }
}
```

9.SONUÇ

Sonuç olarak sistem hayata geçirildiği zaman neler değişeceği gözler önüne serdik. Bunun yansıra basit ama bir o kadarda güvenli olan bu sistemle ek masraflar ortadan kalkacak hataları ortadan kaldırılacaktır.

Proje dokümantasyonunu yazılım mühendisliği kriterlerine uygun olarak yazmaya çalıştık. Projenin gereksinimlerini tam olarak tanımladık. Projemizin örnek proje olarak halka açık bir şekilde tanıtımını yaparak faydalı olabileceğini düşünüyoruz.

10.KAYNAKLAR

- http://arduinom.org/
- Projeler İle Arduino (Erdal DELEBE)
- https://www.arduino.cc/
- https://tr.wikipedia.org/wiki/Arduino
- https://www.youtube.com/channel/UCnD05oNu5qPq_FwF_jfWIlg