



T.C.
YENİ YÜZYIL ÜNİVERSİTESİ
— *İstanbul* —
MÜHENDİSLİK MİMARLIK FAKÜLTESİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
BİTİRME PROJESİ

Mikroişlemci Tabanlı Nesnelerin İnterneti Uygulamaları

MURAT DEMİRTAŞ

120103002

Danışman: Doç.Dr. Mehmet Sağbaş

İSTANBUL, 2016



T.C.
YENİ YÜZYIL ÜNİVERSİTESİ
— *İstanbul* —
MÜHENDİSLİK MİMARLIK FAKÜLTESİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
BİTİRME PROJESİ

Mikroişlemci Tabanlı Nesnelerin İnterneti Uygulamaları

MURAT DEMİRTAŞ
120103002

Danışman: Doç.Dr. Mehmet Sağbaş

İSTANBUL, 2016

Bu çalışma... / ... / 20... tarihinde aşağıdaki juri tarafından Elektrik-Elektronik Mühendisliği Bölümünde Lisans Mezuniyet Projesi olarak kabul edilmiştir.

Mezuniyet Projesi Jürisi

Proje Danışmanı		
Üniversite	T.C. Yeni Yüzyıl Üniversitesi	
Fakülte	Mühendislik Mimarlık Fakültesi	

Jüri Üyesi		
Üniversite	T.C. Yeni Yüzyıl Üniversitesi	
Fakülte	Mühendislik Mimarlık Fakültesi	

Jüri Üyesi		
Üniversite	T.C. Yeni Yüzyıl Üniversitesi	
Fakülte	Mühendislik Mimarlık Fakültesi	

ÖNSÖZ

Günümüzde gelişen akıllı sistemler sayesinde, günlük hayatımızda kullandığımız birçok aletin kullanım kolaylaşmış ve bu aletlere yeni bir özellikler eklenmesi daha kolay hale gelmiştir. Programlama yeteneklerinin açık kaynak olarak paylaşılması ve günümüz teknolojisi sayesinde güçlenen mikroişlemci mimarileri kullanılarak, bu bitirme tezinde günlük hayatını kolaylaştırmayı amaçlayan iki tane bitirme tezi uygulaması tasarlanmıştır. Bunlardan ilki tablet bilgisayar ile kontrol edilen bir keşif robotudur. Bu keşif robotu çalışmasında kablosuz haberleşme teknolojisi ile Windows işletim sistemine sahip tablet PC tarafından kontrol edilen, kamera ve sensörlerle donatılmış bir mobil robotun tasarımını yapıp gerçeklenmiştir. Bu uygulamalardan digeri ise kablosuz hasta izleme sistemidir. Bu çalışmada hastaya ve çevreye bağlı sensörlerden alınan veriler mikrodenetleyici sayesinde alınarak Wi-Fi modüller ile sunucuya ulaştırılmış ve yazılan C++ tabanlı programlar sayesinde bu hasta verilerini (ECG, SpO₂, sıcaklık) görüntülenmesi gerçekleşmiştir.

Bu tez çalışmasında adı geçen uygulamaların makaleleri, gerek ülke içinde gerek yurtdışında yaylanması için yazılmıştır. Bu tez çalışmasındaki uygulamaların gömülü sistemler ve IoT (Internet of Things) konularında araştırma yapmak isteyen herkese yardımcı olacağını düşünüyorum.

Lisans bitirme tezimin başlangıcından bu yana kadar bana göstermiş olduğu ilgi, hoşgörü ve yardımlarından ötürü değerli tez danışmanım Sayın, Doç.Dr. Mehmet SAĞBAŞ'a teşekkürü bir borç bilirim.

Aynı şekilde bu bitirme tezimin yapıp aşamasından beri her konuda fikir danışabildiğim eğitmenim Sayın Yrd. Doç.Dr Gokalp TULUM'a teşekkür ederim.

Her zaman yanında olan, anlayışlarını benden esirgemeyen ve maddi-manevi beni destekleyen aileme sonsuz teşekkürlerimle.

Mayıs, 2016

Murat DEMİRTAŞ

İÇİNDEKİLER

ÖNSÖZ.....	iii
İÇİNDEKİLER.....	v
SİMGE LİSTESİ.....	viii
KISALTMA LİSTESİ.....	ix
ŞEKİL LİSTESİ.....	xiii
ÇİZELGE LİSTESİ.....	xv
ÖZET.....	xvi
ABSTRACT.....	xviii
BÖLÜM 1.....	1
GİRİŞ.....	1
1.1 Literatür Özeti.....	1
1.2 Tezin Amacı.....	4
1.3 Orijinal Katkı.....	5
BÖLÜM 2.....	7
RASPBERRY PI	7
2.1 Raspberry Pi Nedir?	7
2.2 Raspberry Pi Kurulumu İçin İhtiyaç Duyulan Minimum Donanımlar.....	9
2.3 Raspberry Pi için SD kart kurulumu.....	10
2.4 Raspberry Pi'de GPIO.....	16
BÖLÜM 3.....	20
ARDUINO MİKRODENETLEYİCİSİ	20
3.1 Arduino Yazılımının (IDE) Yüklenmesi.....	20
3.2 Arduino geliştirme ortamının kullanımı.....	21
3.3 Arduino GPIO Pinleri.....	23

3.4	Arduino'da ADC Kullanımı.....	25
	BÖLÜM 4.....	29
	DARBE GENİŞLİK MODÜLASYONU.....	29
4.1	Raspberry Pi'de PWM Kullanımı.....	29
4.2.1	WiringPi kütüphanesinin kurulumu.....	30
4.2.2	ServoBlaster kütüphanesinin kurulumu.....	33
	BÖLÜM 5.....	36
	Tablet Bilgisayarla Kontrol Edilen Kablosuz Keşif Robotu.....	36
5.1	Keşif Robotu birimi.....	36
5.2	Kontrolör Birimi.....	36
5.3	Keşif Robotunun Çalışma Prensipleri.....	38
5.3.1	Sistemin Güç Kaynağı Olan Batarya ve DC-DC Çeviriciler.....	41
5.3.2	Motor Sürücüler ve Keşif Robotunda Bulunan Motorlar.....	42
5.3.3	Optokuplör devreleri	47
5.3.4	Sensörler.....	48
5.3.5	Wi-Fi Ağ Adaptörleri.....	51
5.3.5.1	Raspberry Pi Üzerinde Kablosuz Ağ Kurulumu.....	52
5.3.6	Raspberry Pi Mini Bilgisayarı.....	53
5.4	Windows Tabanlı Kontrolör Tablet Çalışma Prensibi.....	72
	BÖLÜM 6.....	95
	KABLOSUZ MULTİ PLATFORM HASTA İZLEME SİSTEMİ	
6.1	Elektrokardiyografi	95
6.1.2	Tarihsel Gelişimi.....	95
6.1.3	Elektrokardiyogram İşareti.....	98
6.2	Pulse Oksimetresi	99
6.2.1	Pulse Oksimetrenin Çalışma Prensibi.....	99
6.2.2	Pulse Oksimetrenin Kullanımını Etkileyen Nedenler.....	101
6.3	Hasta Üzerindeki Gömülü Sistem.....	103

6.4	Sunucu birimi.....	112
6.5	Monitör Birimi.....	118
BÖLÜM 7.....		127
7.1	SONUÇ VE ÖNERİLER.....	127
EK-A		132
QT ÇALIŞMA ALANI.....		132
A.1	QT Creator IDE'sinin Windows Sistemlere Kurulumu ve Örnek Program.....	132
A.2	QT Creator IDE'si Üzerinde Örnek Program.....	136
A.3	QT Creator IDE'sinin Raspbian Sisteme Kurulumu ve Örnek Program	139
EK-B.....		143
SOKET HABERLEŞME.....		143
B.1	TCP/IP Protokol Yapısı.....	144
B.1.1	Uygulama Katmanı.....	144
B.1.2	Taşıma Katmanı.....	144
B.1.3	İnternet Katmanı.....	146
B.1.4	Ağ Erişim Katmanı.....	147
EK-C.....		148
KEŞİF ROBOTU KAYNAK KODLARI.....		148
C.1	Raspberry Pi Üzerinde Çalışan Kodlar.....	148
C.1.1	Motor kontrol programı.....	152
C.1.2	Servo motor kontrol programı.....	152
C.1.3	Sensör programı.....	160
C.1.4	Ping Programı.....	160
C.2	Kontrolör Tablet Üzerinde Çalışan Kodlar.....	161
C.2.1	Motor Kontrol Yazılımı (Windows WPF).....	161
C.2.2	Servo Kontrol Yazılımı (Windows WPF).....	166

C.2.3	Sensor Kontrol Yazılımı (Windows WPF).....	171
C.2.4	Ana arayüz programı Yazılımı (Windows FORM).....	175
C.2.4	Ping Kontrol Yazılımı (Windows WPF).....	182
C.3	Arduino Yazılımı.....	185

SİMGELİSTESİ

Thigh HIGH sinyal periyodu

Tlow LOW sinyal periyodu

T PWM periyodu

KISALTMA LİSTESİ

ADT	Android Developer Tools (Android Geliştirme Araçları)
API	Application Programming Interface (Uygulama Programlama Arayüzü)
ARM	Acorn RISC Machine (Acorn RISC Makinesi)
ARP	Adress Resolution Protocol (Adres Çözümleme Protokolü)
CSI	Camera Interface Specifications (Kamera Arayüz Özellikleri)
CPU	Central Process Unit (Merkezi İşlem Birimi)
DC	Direct Current (Doğru Akım)
EKG	Electrocardiography (Elektrokardiyografi)
FAT	File Allocation Table (Dosya Yerleşim Tablosu)
GHz	Gigahertz (Gigaherts)
GPIO	General Pursose Input -Output (Genel Amaçlı Giriş/Çıkış)
GPU	Graphics Procesing Unit (Grafik İşleme Ünitesi)

HD	High Definition (Yüksek Çözünürlük),
HDMI	High Definition Multimedia Interface (Yüksek Çözünürlüklü Çoklu Ortam Arayüzü)
HTTP	Hyper-Text Transport Protocol (Hiper Metin Transfer Protokolü)
Hz	Hertz (herts)
ICMP	Internet Control Message Protokol (İnternet Kontrol Mesajı Protokolü)
IDE	Integrated Development Environment (Entegre Geliştirme Ortamı)
IEEE	Institute of Electrical and Electronical Engineers (Elektrik ve Elektronik Mühendisleri Entitüsü)
IPv4	Internet Protocol Version 4 (İnternet Protokolü Versiyonu 4)
IPv6	Internet Protocol Versiyon 6 (İnternet Protocol Versiyon 6)
JPEG	Joint Photographic Experts Group (Birleşik Fotoğraf Uzmanları Grubu)
LAN	Local Area Network (Yerel Alan ağı)
LCD	Liquid Crystal Display (Sıvı Kristal Ekran)
MHz	Megahertz (Megaherts)
MJPG	Motion Joint Photographics Experts Group (Hareketli Birleşik Fotoğraf Uzmanları Grubu)

MJPEG	Motion Joint Photographic Expert Group (Hareketli Birleşik Fotoğraf Uzmanları Grubu)
OS	Operating System (İşletim Sistemi)
PAN	Personal Area Network (Kişişel Alan Ağı)
PWM	Pulse Width Modulation (Darbe Genişlik Modülasyonu)
RAM	Random Access Memory (Rastgele Erişim Belleği)
RCA	Radio Corporation of America (Amerika Radyo Birliği)
RTP	Real-Time Transport Protocol (Gerçek zamanlı Transfer Protokolü)
SD	Secure Digital (Güvenli digital)
SDK	Software Development Kit (Yazılım Geliştirme Aracı)
SPI	Serial Peripheral Interface Bus (Seri Çevresel Arayüz Veriyolu)
TCP	Transmission Control Protocol (İletişim Kontrol Protokolü)
TCPMP	The Core Pocket Media Player (Temel Paket Medya Oynatıcısı)
UART	Universal Asynchronous Receiver/Transmitter (Evrensel Asenkron Verici-Alıcı)
UDP	User Datagram Protocol

	(Kullanıcı Veri Bloğu Protokolü)
USB	Universal Serial Bus
	(Evrensel Seri Veriyolu)
Wi-Fi	Wireless Fidelity
	(Kablosuz Bağlantı alanı)
WLAN	Wireless Local Area Network
	(Kablosuz Yerel Alan Ağısı)
WPF	Windows Presentation Foundation
	(Windows Presentation Foundation)

ŞEKİL LİSTESİ

Şekil 2.1 Raspberry Pi 2 Model B'nin üstten görünümü.....	8
Şekil 2.2 Raspberry Pi 2'de bulunan çevresel birimler.....	9
Şekil 2.2.1 SD kart formatlama aracının Windows sürümünün indirilmesi.....	11
Şekil 2.2.2 SD kart formatlama aracında gereken ayarların yapılması.....	12
Şekil 2.2.4 Raspbian Jessie işletim sisteminin indirilmesi.....	13
Şekil 2.2.5 Raspbian Jessie işletim sisteminin zip'ten çıkartılması.....	14
Şekil 2.2.6 Win32 disk imager programı.....	14
Şekil 2.2.7 Putty SSH protokolü ile Raspberry Pi terminaline erişilmesi.....	15
Şekil 2.2.8 Putty SSH protokolü ve Xming ile sistem görüntüsünün elde edilmesi.....	15
Şekil 2.4.1 Raspberry Pi'nin GPIO pinleri gösterimi.....	16
Şekil 2.4.2 Raspberry Pi'nin GPIO pinleri diagramı.....	17
Şekil 3.1 arduino.cc web sitesi download bölümü.....	20
Şekil 3.2 Arduino versiyonu seçme bölümü.....	21
Şekil 3.3 arduino.cc donation bölümü.....	21
Şekil 3.4 Arduino IDE programı arayüzü.....	22
Şekil 3.5 Arduino geliştirme platformu arayüzü ve açıklamaları.....	22
Şekil 3.5 ATmega328 pin ayrıntıları.....	23
Şekil 3.7 Arduino uno ön yüzü gösterimi.....	24
Şekil 3.8 ADC bloğu yapısı.....	25
Şekil 3.9 Arduino Nano geliştirme kartı.....	26
Şekil 3.10 Arduino Pro Micro geliştirme kartı.....	27
Şekil 4.1 PWM sinalinde periyot, darbe genişliği ve doluluk boşluk oranı gösterimi..	28
Şekil 4.1 Raspberry Pi BCM2836 çipinin PWM blok diyagramı.....	29
Şekil 4.2 Raspberry Pi'de superuser yetkisine sahip kullanıcı oluşturma.....	31

Şekil 4.3 WiringPi'nin Yazılımsal pinlerinin Raspberry Pi terminalinde görüntülenmesi.....	31
Şekil 5.1 Keşif robottu kısımının bileşenleri ve bağlantıları.....	36
Şekil 5.2 Kontrolör tablet PC kısmının bileşenleri ve bağlantıları.....	37
Şekil 5.3 Keşif robotunun tasarlanan tüm pin diyagramları.....	39
Şekil 5.3.1 LM2596 Step down düşürücü elektronik kart.....	40
Şekil 5.3.2 11.1VDC 5000mAh 35C 3 Hücreli Li-Po pil.....	41
Şekil 5.3.3 L298N Motor sürücü shieldi.....	42
Şekil 5.3.4 R260 6V DC Motorlar.....	43
Şekil 5.3.5 Kontrolör arayüzde parazitlenmeler.....	43
Şekil 5.3.6 Pan-Tilt Mekanizmasında kullanılan SG90 servo motor.....	44
Şekil 5.3.7 Pan-Tilt Mekanizmasında pan-tilt.....	45
Şekil 5.3.8 Pan-Tilt Mekanizmasının keşif robottu üzerindeki son hali.....	45
Şekil 5.3.9 TLP521-4 Entegresinin iç yapısı.....	46
Şekil 5.3.10 TLP521-4 entegresinin dıştan görüntüsü.....	47
Şekil 5.3.11 LM35 sıcaklık sensörü.....	48
Şekil 5.3.12 HCSR04 ultrasonik mesafe sensörü.....	49
Şekil 5.3.13 MQ-7 CO sensörü.....	49
Şekil 5.3.14 LB-link RTL8188CUS chipsetli Wi-Fi Ağ adaptörü.....	50
Şekil 5.3.15 lsusb komutu ile Raspberry Pi'ye bağlı cihazların görünümü.....	54
Şekil 5.3.16 Logitech C310 5 MP 720p HD webcam.....	54
Şekil 5.3.17 Raspberry NoIR kamerası.....	55
Şekil 5.3.18 sudo raspi-config komutunun ekran görüntüsü.....	56
Şekil 5.3.19 sudo raspi-config komutundan dsi kameranın aktif edilmesi.....	56
Şekil 5.3.20 HD kamera web yayınının aynı ağdaki başka bilgisayarda görüntülenmesi.	59
Şekil 5.3.21 NoIR kamera web yayınının aynı ağdaki başka bilgisayarda görüntülenmesi.....	60
Şekil 5.3.22 CH340G chipsetli USB-TTL dönüştürücü.....	67

Şekil 5.3.23 İletişim bağlantı noktalarının terminalde görüntülenmesi.....	67
Şekil 5.4.1 Keşif robotu kısmının bileşenleri ve bağlantıları.....	72
Şekil 5.4.2 Kontrolör birimi kontrol arayüzü.....	73
Şekil 5.4.3 Arayüz giriş ekranı.....	74
Şekil 5.4.3 NoIR kameradan alınan MJPEG stream akışından bir ekran görüntüsü.....	77
Şekil 5.4.4 NoIR kameradan alınan MJPEG stream akışından bir ekran görüntüsü.....	78
Şekil 5.4.5 HD kameradan alınan MJPEG stream akışından bir ekran görüntüsü.....	79
Şekil 5.4.6 HD kameradan görüntü.....	80
Şekil 5.4.7 Sensör arayüzü formu.....	85
Şekil 5.4.8 Mesafe arayüzü formu.....	85
Şekil 5.4.9 Motor kontrol formu.....	86
Şekil 5.4.10 Görüntü işleme kütüphanelerinden bir örnek.....	86
Şekil 5.4.11 İki kare farkı ile hareketli nesne takibi.....	87
Şekil 5.4.12 İki kare farkı ile hareketli nesne sayımı.....	87
Şekil 5.4.13 İki kare farkı ile iyileştirilmiş hareketli nesne takibi.....	88
Şekil 5.4.14 Karakter algılama ile ileri otonom kontrol arayüzü.....	88
Şekil 5.4.15 Karakter algılama ile sola otonom kontrol arayüzü.....	88
Şekil 5.4.16 İki kamera görüntüsü aynı anda.....	89
Şekil 5.4.17 Kuşbakısı kontrol arayüzü HD kamera ekranı.....	90
Şekil 5.4.18 Kuşbakısı kontrol arayüzü NoIR kamera ekranı.....	90
Şekil 5.4.19 Hareket halinde kontrol arayüzü.....	91
Şekil 5.4.20 İlk görüntülü işlemeli arayüz.....	91
Şekil 5.4.21 İlk prototip keşif robotu.....	92
Şekil 5.4.22 Son prototip keşif robotu.....	93
Şekil 6.1 İlk insan kardiyografisi.....	95
Şekil 6.2 Standart göğüs derivasyon noktaları.....	96
Şekil 6.3 EKG cihazı.....	96
Şekil 6.4 Örnek EKG grafiği.....	97

Şekil 6.5 Pulse oksimetre cihazı.....	98
Şekil 6.6 Pulse Amped Sensor.....	101
Şekil 6.7 ds18b20 sıcaklık sensörü.....	101
Şekil 6.8 DHT11 ısı ve nem sensörü.....	102
Şekil 6.9 LDR ışık sensörü.....	102
Şekil 6.10 ESP Wi-Fi modül.....	103
Şekil 6.11 Elektronik sistem çalışma prensibi.....	110
Şekil 6.11 Elektronik sistem pin bağlantıları.....	111
Şekil 6.12 Sunucu birimi blok diyagramı.....	112
Şekil 6.13 Monitör birimi arayüzü.....	118
Şekil 6.14 Hasta listesi formu.....	119
Şekil 6.15 Monitör arayüzünden alınan EKG grafiği örneği.....	121
Şekil 6.16 Monitör arayüzünden alınan SpO2 grafiği örneği.....	121
Şekil 6.17 Aktarım halinde monitör arayüzü.....	122
Şekil 6.18 Aktarım halinde monitör arayüzü.....	122
Şekil 6.19 Aktarım halinde multiplatform arayüzler.....	123
Şekil 6.20 Oda kontrol arayüzü.....	123
Şekil 6.21 Mobil cihazlar için tasarlanılan arayüz.....	124
Şekil A.1 QT web sayfası.....	132
Şekil A.2 QT Creator online web yükleyicisi ekranı.....	132
Şekil A.3 QT web yükleyici kullanıcı girişи ekranı.....	133
Şekil A.4 QT Creator yükleme dizini seçim ekranı.....	133
Şekil A.5 QT versiyonu seçim ekranı.....	134
Şekil A.6 Lisans koşullarını kabul ekranı.....	134
Şekil A.7 QT Creator IDE yardım arayüzü.....	135
Şekil A.8 Proje türü seçim ekranı.....	135
Şekil A.9 Proje ismi seçme ekranı.....	136
Şekil A.10 Proje için derleyici seçim ekranı.....	136

Şekil A.11 Proje için derleyici seçim ekranı.....	137
Şekil A.12 İlk örnek uygulamanın çalıştırılması.....	137
Şekil A.13 QT Creatorun bulunması.....	138
Şekil A.14 QT Creator Remote Linux özelliğinin devre dışı bırakılması.....	139
Şekil A.15 Qt qmake versiyonunu seçme ekranı.	140
Şekil A.16 Qt gcc derleyici seçim ekranı.....	140
Şekil A.17 Örnek uygulamanın çalışması.....	141

TABLO LİSTESİ

Tablo 2.1 Raspberry Pi 2 Model B'nin Özellikleri.....	8
Tablo 2.2 Raspberry Pi 2'de bulunan çevresel birimler.....	9
Tablo 2.3 GPIO pin dizilimi ve açıklamaları.....	18
Tablo 4.1 ServoBlaster kullanılabilen PWM pinlerinin bilgisi.....	33
Tablo 5.1 Logitech C310 birkaç kamera parametreleri.....	55
Tablo 5.2 Raspberry Pi birkaç kamera parametreleri.....	57
Tablo 6.1 ESP8266 teknik özelliklerı.....	103
Tablo 6.2 AT komut takımı.....	104

ÖZET

Mikroişlemci Tabanlı Nesnelerin İnterneti Uygulamaları

Murat DEMİRTAŞ

Mühendislik-Mimarlık Fakültesi

Elektrik-Elektronik Mühendisliği Bölümü

Lisans Bitirme Tezi

Tez Danışmanı: Doç.Dr. Mehmet SAĞBAŞ

Bu çalışmada, günümüzde gelişen nesnelerin interneti teknolojisi üzerine değişik disiplinlerde mikrodenetleyici ve mikroişlemciler kullanılarak 2 farklı uygulama gerçekleştirilmiştir. Bunlardan ilkinde üzeri sensörler ve farklı özellikte iki kamera ile donatılmış, kablosuz haberleşme teknolojisi ile Windows işletim sisteme sahip olan tablet veya bilgisayardan kontrol edilebilen bir keşif robottu tasarlanıp gerçekleştirilmiştir. Keşif robottu, üzerinde açık kaynak Linux işletim sistemi kullanılan ARM mikroişlemci mimarisi tabanlı bir gömülü sistem barındırmaktadır. Bu tasarım kapsamında gerçekleştirilen keşif robotunun temel özellikleri; C++ ve C# tabanlı olması, HD kalitede gece-gündüz video ve fotoğraf çekebilmesi, üzerindeki sensörler ile hedef ortamı analiz etmesi, geliştirilebilir olması, üzerinde isteğe bağlı görüntü işleyebilmesi ve gerçek zamanlı çalışabilmesi olarak özetlenebilir.

Keşif robottu ve kontrolör birimi olan tablet, birbirleriyle Wi-Fi teknolojisi üzerinden soket haberleşmesi ile haberleşmektedirler. Keşif robottu bulunan gömülü sistem, sensör verilerini, HD kamera ve NoIR gece görüş kamerası görüntülerini gerçek zamanlı olarak kontrolör tabletin arayüzüne aktarmaktadır. Gerçek zamanlı video aktarımı debian tabanlı Raspbian işletim sisteminde MJPEG ile elde edilmektedir. Performans için bu video kaydı Raspberry Pi sisteminde depolanmaz. Keşif robottu kamera bakış açılarını değiştirebilmek ve farklı yönlere bakabilmek üzere bir kamera servo pan-tilt mekanizmasına sahiptir ve bu mekanizmayı Raspberry Pi kontrol etmektedir.

İstemci olan tablet pc keşif robotunun kontrolü için gereken tüm kontrol arayüzleri ve ortam sensörlerini görüntüleyebilmek için gerekli görsel barları içermektedir. Kontrol arayüzü C# programlama dili kullanılarak Windows Form ve WPF yapıları kullanılarak arayüz basit ve anlaşılır bir hale getirilmiştir.

İkinci uygulama ise hastaya bağlı olan biyomedikal sinyallerden ve hastanın odasına bağlı olan sensörlerden, mikrodenetleyici sayesinde elde edilen verilerin anlık olarak depolanarak ilgili programlarda görüntülenebilmesini sağlayan bir tasarım gerçeklenmiştir. Bu tasarım Arduino Nano mikrodenetleyicisi ve ESP8266 Wi-Fi modülü içermektedir. Bu tasarımda sensörlerden alınan veriler 10 saniyede tamponlanarak sunucuya yollanır. Sunucu programı bu elektronik karttan gelen verileri ilgili veritabanı tablolarına kaydeder ve hasta için gerekli olan hayatı veriler bu sensör verilerine şifreli olarak kaydedilir. Bu uygulama sayesinde doktorlar hastalara ait tüm biyomedikal verileri istediği yerde istediği zaman C++ tabanlı yazılan programlar aracılığı ile platform farkı gözetmeksiz (Windows, Linux, MacOs, IOS, Android) erişebilecektir.

İstemci programlarında hastaya ait veriler zamana göre grafiksel olarak ilgili sağlık personeline gösterilebilmektedir.

Anahtar Sözcükler: Keşif Robot, Soket Haberleşme, Kablosuz Haberleşme, Windows, Kablosuz Hasta Takip, C++,C#

ABSTRACT

Tablet Computer Wirelessly Controlled Reconnaissance Robot Design

Murat Demirtaş

Faculty of Engineering and Architecture

Department of electrical engineering

Undergraduate Thesis

Supervisor: Doç.Dr. Mehmet SAĞBAŞ

In this study, the object of the present developed using a microcontroller and microprocessor technology in various disciplines on the Internet was carried out in 2 different applications. These sensors mounted on the first and equipped with two cameras with different properties, a reconnaissance robot that can be controlled from the tablet or computer with the Windows operating system has been designed and realized by wireless communication technology. Discovery robot, based on the open source Linux operating system used ARM microprocessor architecture contains an embedded system. The main features of the scope of this design carried out reconnaissance robots; C ++ and C # is based, HD-quality day-night video and photo shoot to the sensors and analyze the target environment on, can be improved on to handle optional display and can be summarized as to work in real time.

Reconnaissance robot and a controller unit, a tablet computer, with each other Wi-Fi technology to communicate with via socket communication. Reconnaissance robots, embedded systems, sensor data, real-time footage and Night Vision HD Camera Noir is transferring to the controller in the tablet interface. Real-time MJPEG video transmission with Debian based raspbian operating system are obtained. This Raspberry Pi for video recording is not stored in the system performance. Exploration the robot to be able to change the perspective camera to be able to look in different directions and a camera, servo Pan-Tilt mechanism, and this mechanism Raspberry Pi to control it. Tablet PC with the client interfaces and ambient sensors to control the robot all you need is a reconnaissance view of the control bar includes useful to be able to a visual. Control

interface the C# programming language using Windows Forms and WPF using simple and straightforward interface structures has been made.

The second application which is connected to the patient's room and patient's biomedical signals and sensors that are connected to the microcontroller and stored as a snapshot of the data from a design that allows to be viewed in the related programmes are implemented. This design Arduino Nano microcontroller and ESP8266 Wi-Fi module included. In this design the data received from the sensors is sent to the server in 10 with buffer. The incoming data from the server corresponding records into the database tables program this electronic card of the patient, and the vital data that is necessary for the data from this sensor are stored in encrypted form. Thanks to this application, all biomedical doctors that belong to the patients data anywhere, any time through C based programs that are written, regardless of Platform (Windows, Linux, Mac OS, IOS, Android) will be able to access. Of patients relevant to health personnel in the client program data over time can be shown graphically.

Keywords: Robot, Exploration, Socket Communications, Wireless Communications, Windows, Wireless Patient Monitoring, C, C#

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Günümüzde gelişen teknolojiye paralel olarak ve günlük hayat şartları nedeniyle oluşan gereksinimleri karşılayabilmek için sürekli olarak yeni ve geliştirilebilir akıllı sistemler hayatımıza giriş yapmıştır. Bu tür yeni akıllı sistemler günümüzde halen yoğun biçimde tasarlanıp geliştirilmektedirler. Bu akıllı sistemler öncelikle sadece belli bir amaca yönelik olarak tasarınlarken daha sonradan daha farklı sistemler birleştirilmiş ve birçok işlem aynı anda yapılabilir hale gelmiştir. Bu sistemleri birleştirmek için tasarlanan elektronik geliştirme platformlarının üretilmesi ile gömülü sistemler hayatımıza girmiştir. Programlama yeteneklerinin açık kaynak olarak paylaşılması ile birlikte gömülü sistemlere olan ilgi artmıştır. Bu ilgiye en çok akademisyenler ve şirketler tarafından yapılan yatırımlar sayesinde her geçen gün farklı farklı çalışmalar hayatımıza giriş yapmışlardır.

İnternetin kullanımının yaygınlaşması ile birlikte istenilen bilgiye daha çabuk ve etkin olarak ulaşılabilmesi ile birlikte gömülü sistem tasarlayacak kişinin tasarlamak istediği proje için konuları araştırıp gerekli bilimsel çalışma yapması hızlanmıştır. Bu sayede her gün yeni yeni sistemler oluşmakta ve oluşmaya devam etmektedir.

Gömülü sistemlere şirketler ve devletler tarafından yapılan yatırımlar ile birlikte bu sistemlere katkılar çok önemli bir büyüklükte artmıştır. Ayrıca bu gömülü sistem platformu üreten firmaların kaynak kodlarını açık kaynak olarak ücretsiz paylaşması sayesinde bu tür sistemlerin geliştirilmesini ve yeni buluşların önünü açmıştır. Bununla birlikte her bütçeye göre tasarlanan ve kullanıcının gerek duyabileceği donanımları gömülü sistem platformlarına dahil edilmesiyle bu tür çalışma yapan bireylerin çalışmalarına büyük destek sağlanmaktadır. Ayrıca bu geliştirme platformlarına

eklenecek olan donanımların ve entegrelerin daha fazla küçültülmesi ile birlikte mobil olarak taşınabilirlikleride arttırlılmıştır.

Her gün daha da kolaylaşan ve gelişen programlama sektörünün sayesinde bilimsel çalışma yapmak isteyen çoğu kimseye, gerek duyduğu kod, program ve teknik bilgi çok hızlı bir şekilde ulaşabilmektedir. Bilimsel çalışma yapmak isteyen çoğu kimsenin programlama bilgisi gerektirmeyecek seviyede ihtiyaç duyduğu yazılım, geliştirme platformları sayesinde yazılan kodlar kaynak bilgisayar sisteminde derlenip simulasyon yapılmaktadır. Bu yazılım geliştirme platformları ile birlikte kullanıcılara kendi kafasında yapmak istediği tüm algoritmaları kolay bir şekilde tasarlayıp ilgili gömülü sisteme programlarını gömelmektedirler.

Programlama yeteneklerinin ve donanımların teknik bilgilerinin açık kaynak ve ücretsiz olarak dağıtılması ile birlikte kullanıcı, kendisine ihtiyaç duyduğu tüm sistem elemanlarını şebekelemekte ve programlayabilmektedir. Gömülü sistem programcılığının gelişmesinde şüphesiz olarak açık kaynak paylaşımının önemi bu yüzden çok büyütür. Bu sayede araştırmacıların kendi sistemlerine müdahale edebilmesi ve de bu bilgilere ücretsiz olarak ulaşabilmesi büyük bir teşvik kaynağıdır. Bu açık kaynak platformlarının en başında Arduino olmak üzere ve Linux tabanlı gömülü sistemler ön plana çıkmaktadır.

Mobil tabanlı akıllı sistemlerin ön plana çıkması ile mobil sistemler günlük hayatımızda daha çok ortaya çıkmaktadırlar. Böylece amaca yönelik farklı mobil sistemler, firmalar tarafından ücretli veya ücretsiz olarak tasarlanmaktadır. Bununla birlikte birçok amatör veya profesyonel yazılım geliştiriciler yapmak istediği bilimsel araştırmada bu sistemleri kullanmakta ve yayılmaktadır.

Günümüz şartlarında ilgili gömülü sistemleri, hareketli platformlara entegre edilerek bu platformları yazdığımız algoritmalarla göre kontrolünü sağlayabiliyoruz. Akıllı telefon veya tabletimizden istediğimiz bir nesneyi kontrol etmek oldukça basitleşmiştir. Örneğin evimizde bulunan aydınlatma, bulaşık makineleri, çamaşır makinesini veya televizyonlarımızı internete bağlı olduğumuz her yerden kontrol edebilmekteyiz. Bununla birlikte evimize gerekli yerlere gömülü sistem ile birlikte amaca yönelik sensörler kullanırsak, kolaylıkla bir akıllı ev otomasyonu sistemi kurabiliriz. Bu sayede evde olusabilecek her türlü yanık, su basması gibi durumlarda gerekli işlemleri yapmamızı ve önlemler alabilmemizi sağlayacak sistemler tasarlayabiliriz.

İnsanların yaklaşmasının bile tehlikeli olduğu koşullarda oylara müdahale edebilmek için tasarlanan mobil robotlar, uzun yıllar boyunca bilim insanların dikkatini çekmiştir. Otonom hareket yapabilen mobil robotlar başta fabrikalarda olmak üzere evde

kullandığımız elektronik eşyalara kadar birçok alanda kullanılmaktadır. Bu yüzden hayatımıza kolaylaştmak için mobil robotlara ihtiyaç duyuyoruz.

Bu bitirme tezinin ilk uygulaması yukarıda belirtilen amaçlar için tasarlanmış ve gerçekleşmiştir.

Günümüzde milyonlarca insan, başta yüksek tansiyon olmak üzere kalp-damar hastalıkları, şeker hastalığı ve çeşitli kronik hastalıklarla yaşamaktadır. Bu sayı artan dünya nüfusuna doğru orantılı olarak artmaktadır. Bu sebeple hastaların takibi zorlaşmaktadır. Erken teşhis gerektiren birçok sağlık problemi, sağlık personeli tarafından tam olarak izlenemediği için hastaya geç müdahale edilmesi gibi sorunlar ortaya çıkmaktadır. Bu nedenle ritim bozukluğu veya obstrüktif uyku apne sendromu rahatsızlığı bulunan hastalar için erken teşhis inanılmaz derecede önem taşımaktadır. Çünkü bu hastalıklara sahip kişilerde ölüm riski yüksektir. Buna benzer kronik hastalıkları, hastalık ilerlemeden önce tespit etmek ve hastayı sürekli kontrol altında tutabilmek için dünya çapında mobil kablosuz hasta takip sistemleri önem kazandırmıştır.

Bu gibi takip sistemlerinde temel amaç, gerek hastanede gerekse hastane dışında kalan hastalara takılacak biyomedikal sensörlerden (Pulse oksimetre, EKG, sıcaklık vb.) alınan verinin kaydedilip istenildiği zaman bu verilerin sağlık personeli tarafından erişiminin sağlanmasıdır. Bu sayede bu gibi rahatsızlıklara erken teşhis ve anında müdahale imkânı sağlanabilir.

Günümüzde çoğu mobil takip sistemi gerçek zamanlı veri aktarımı yapamamaktadır. Örneğin, günümüzde sık kullanılan mobil takip sistemlerinden Holter cihazı, gün boyunca alınan EKG verisini gerçek zamanlı olarak iletilememektedir.

Bu çalışmada, sağlık personelinin platform farkı gözetmeksızın, hastaya bağlı sensörlerden anlık olarak ölçüm yapabilmesi sağlanmıştır. Temel amaç sağlık personelin, hastaya ait bütün verileri -sisteme bağlı olan hastanın nerede olduğuna bakmaksızın- anlık olarak ulaşabilmesini sağlamaktır. Bu sayede sağlık personeli, hastasına tedavisi için yapılması gerekenleri önceden belirleyebilecek ve hastayı gözetim altında tutabilecektir. Bununla birlikte bütün hastaların verileri kayıt altına alınıp daha sonra incelenmesi için depolanabilecektir.

Bu bitirme tezinin ikinci uygulaması yukarıda belirtilen amaçlar için tasarlanmış ve gerçekleşmiştir.

1.2 Tezin Amacı

Mobil teknoloji kullanan ürünlerin hayatımıza girmesi ile birlikte bu sistemleri uzaktan kontrol etme isteği doğmuştur. Bu sayede kendi tasarladıkları sistemleri uzaktan kontrol edebilmek amacıyla araştırmacılar, kendi yaptıkları arayüz programları vasıtası ile telefon, bilgisayar, tablet ve akıllı saatler kullanarak bu sistemleri internete bağlı oldukları her yerden kontrol edebilmektedirler. Günümüzde ev sahipleri evine kurdukları akıllı ev sistemleri ile işyerlerinde evlerinde olup biten kolaylıkla izleyebilmekte, elektronik eşyaları kontrol edebilmekte ve bilgi sahibi olabilmektedirler. Hızlıca gelişen teknoloji ve programlama yeteneklerinin açık kaynak olarak paylaşılması ile birlikte bu sistemler daha kolay kontrol edilebilir hale gelmiştir. Bu tez çalışmasında kablosuz haberleşme teknolojisine sahip bir robotun, uzaktan bir mobil sistemin kontrol edebildiği bir sistem ve sinyal aktarımı üzerine kurulu kablosuz hasta takip sistemi tasarlanmıştır ve gerçeklenmiştir.

Günlük hayatı oluştıracak gaz kaçağı, deprem, heyelan, terör gibi olaylarda sivil vatandaşların bu olayın olduğu bölgeye girmesi çok tehlikelidir. Bu durumlarda insanın erişemeyeceği veya giremeyeceği bölgeler olabilmektedir. Örneğin içinde patlayıcı olan bir çantanın bulunduğu ortamdan insanlar uzaklaştırılarak patlayıcılar konusunda uzman bir bomba imha uzmanının olay yerine gelmesi beklenilir. Bu gibi durumlarda uzaktan kumanda edilerek ve üzerine gerekli ekipman yerleştirilerek çantanın içeriğinin görüntülenmesi veya çantanın robotun bırakacağı fünye ile patlatılması bu tür tehlikeleri kolay ve güvenli hale getirmektedir.

Bu üstte bahsedilen durumlara karşı can ve mal güvenliğini tehlikeye atmadan amaca yönelik sensör ve ekipman takılabilicek bir mobil sistemin alt yapısının oluşturulması bu tez çalışmasının kapsamında bulunmaktadır. Bu bitirme tezi çalışmasında gerçekleşen sistem sayesinde uzaktan kontrol edilen, hedef alanda gece veya gündüz HD çekim yapabilme yeteneği olan, üzerine takılan sensörler ile hedef alanda bilgi almamızı sağlayan, isteğe bağlı görüntü işleme yapabilen, istenilen yöne hareket edebilen bir mobil keşif robottu hazırlanmıştır.

Bu robot güvenlik amacıyla ev, işyeri, fabrika gibi yerlerde hareketli olması sebebi ile ve isteğe bağlı görüntü işleme yapabilmesi sayesinde hırsızlık gibi olaylarda kullanıcıyı bilgilendirebilir ve de güvenlik amaçlı devriye atmak için kullanılabilir. Üzerinde bulunan gece ve gündüz görüntü alabilen ucuz maliyetli kameralar ve tüm sistem sayesinde güvenlik masrafları en az seviyeye inebilir. Görüntü işleme opsiyonlarının değiştirilmesi ile istenilen yerlerde otonom olarak devriye atabilir.

Bu çalışma prensiblerine sahip robotların kullanılabilmesi bir başta maliyet ve zaman açısından çok büyük avantajları vardır. Kullanıcı ister işyerinden ister robotun yanında kablosuz olarak bu keşif robotunu hareket ettirebilmektedir. Bu sayede istediği yerden görüntü alabilicektir. Ayrıca bu keşif robotunda oluşturulan sistemde veriler aynı anda birden çok bilgisayara aktarılabilmektedir.

Keşif robotunun, uzaktan kontrol edebilecek donanıma sahip herhangi bir Windows işletim sistemine sahip kontrolör sistemden komuta edilebilmesi bu sistemin kullanılabilirliğini ve de taşınabilirliğini kolaylaştırmaktadır.

Bu bitirme tezinin ikinci uygulaması olan kablosuz hasta izleme sisteminde uzaktan hastaya bağlı olan sensörlerden alınan verilerin uzak bir monitörde izlenilmesi amaçlanmıştır.

Günümüzde biyomedikal ve sağlık alanında, zamandan ve mekâna ihtiyaç duymadan, gerekli olan processlerin uzaktan karşılama gereksinimi, kablosuz mobil sistemlerin kullanılmasını kaçınılmaz hale getirmiştir. Gelişmiş ülkelerde hasta sayısına oranla doktor sayısının artmaması ve hasta-doktor iletişiminin zorlaşmasından ötürü erken teşhis gerektiren tüm hastalıkların tespiti zorlaşmaktadır. Bu nedenlerle ikinci uygulama olan kablosuz hasta izleme sistemi tasarlanmıştır. Bu sayede hastaların hayatı verileri alınıp işlenildikten sonra kablosuz olarak iletilip veritabanlarında depolanacaktır. Ayrıca sensör verileri tasarlanan monitör uygulamaları sayesinde anlık veya dönemsel olarak görüntülenebilicek vede hastalar için gerekli olan tüm veriler incelenebilicektir.

1.3 Orijinal Katkı

Bu bitirme tezi çalışmasının ilk uygulaması olan keşif robotunda, hedef ortamda üzerinde bulundurduğu 2 kameraldan görüntü alabilicek ve sensör verilerini kontrolör sisteme aktaracak ve kontrolör tabletten alınacak komutlara göre istenildiği yere gidebilicek bir keşif robottu gerçeklenmiştir. Keşif robotunun beyni, ARM tabanlı bir mikroişlemci barındıran Raspberry Pi anabilgisayarıdır. Ayrıca sensörleri okumak için bir tane Arduino ATmega328 mikrodenetleyicisi bu sisteme konulmuştur. Tasarlanan kontrolör ise Windows işletim sistemi üzerinde çalışan tabletterdir. Bu sayede bu keşif robotunun kontrolü Windows tabanlı herhangi bir bilgisayar, tablet veya telefondan yapılabilmektedir.

Bu projenin oluşturulmasında aşağıda belirtilen referanslardan yararlanılmıştır. Keşif robotunda alınan görüntünün aktarılması için [1] numaralı referanstaki çalışmada yapılmıştır. [2]numaralı referansta ise LEGO mindstorms NXT robot platformunun farklı geliştirme ortamlarında denemeleri yapılmıştır. Bir başka gömülü sistem platformu üzerinde görüntü ve video aktarımını akıllı sınıflarda kullanılmak üzere bir projeksiyon cihazına aktarmak için yapılan başka bir çalışmada da verilmektedir.[3].

Bitirme tezinin ikinci uygulaması olan kablosuz hasta takip sisteminde ise aşağıda belirtilen referanslardan yararlanmıştır.

Reza Fazel-Rezai'nın çalışmasında ECG sinyali, Bluetooth kablosuz iletişim teknolojisi sayesinde alıcı üniteye aktarılmış ve labVIEW programı kullanılarak görselleştirme yapılmıştır[4].

K. Kaya, hastaların günlük aktivitelerini engellemeden kullanabilecekleri ve verileri kaydedip daha sonra e-posta yolu ile bir kardiyoloji uzmanına göndermeye çalışmıştır. Bu iletimi Radyo Frekans (RF) modülleri ile sağlamıştır. Farklı hastaların EKG verilerini kayıt altına alınabilmekte ve e-posta ile dünyanın herhangi bir yerindeki uzmana gönderebilmektedir [5].

S. Can, düşük maliyetli bir cihaz tasarlayarak elde edilen verileri telefona göndermeye çalışmıştır. Bluetooth modülü ile EKG işaretlerini cep telefonuna ileterek telefonda görüntülenmesini sağlamıştır. Bluetooth yolu ile yaklaşık 10 metre mesafedeki başka cihazlara veya Global System for Mobile Communications (GSM) operatörleri üzerinden uzak mesafelere veri akışı sağlanmıştır. Fakat bu çalışmada GSM üzerinden veri gönderilmesi için şebeke alanı dışına çıkılmaması gerekmektedir [6].

Son olarak ise Cosmonescu ve ekibinin çalışması incelenmiştir. Cosmonescu ve ekibi yaşamsal ECG ve EMG sinyallerinin RF (433,92 MHz) standartı ile alıcı birime aktarmışlardır[7].

Bu bitirme projesinde yukarıda referans belirtilen kaynaklardan faydalılmıştır.

BÖLÜM 2

RASPBERRY PI

Bu bitirme tezinde ilk uygulama olan keşif robotunun beyni olan Raspberry Pi biraz daha ayrıntılı olarak açıklanmaktadır.

2.1 Raspberry Pi Nedir?

Raspberry Pi, ARM mimarisine sahip mikroişlemci tabanlı gömülü sistemlerde eğitim ve hobi amaçlı olarak kullanılması için tasarlanmış, ucuz, hızlı ve muadillerine oran ile daha fazla performansa ve çevresel donanıma sahip bir mini bilgisayardır[8]. Bu bitirme projesinde kullanılan Raspberry Pi “Model 2 B” (Tip B) olarak kullanıcılarla sunulan sürümdür[9]. Bu ARM tabanlı mikroişlemci barındıran sistem ile günümüz bilgisayarlarının yapmakta olduğu çoğu işlemi yapabilme kapasitesi bulunmaktadır. Örnek olarak temel ofis programları, web tarayıcılar ve yazılım geliştirme platformları verilebilir. Bununla birlikte Raspberry Pi içerisinde bulunan 4 çekirdekli GPU sayesinde yüksek çözünürlüklü (HD) videoları oynatabilme ve uzak bilgisayarlara yayın akışı yapabilmektedir.

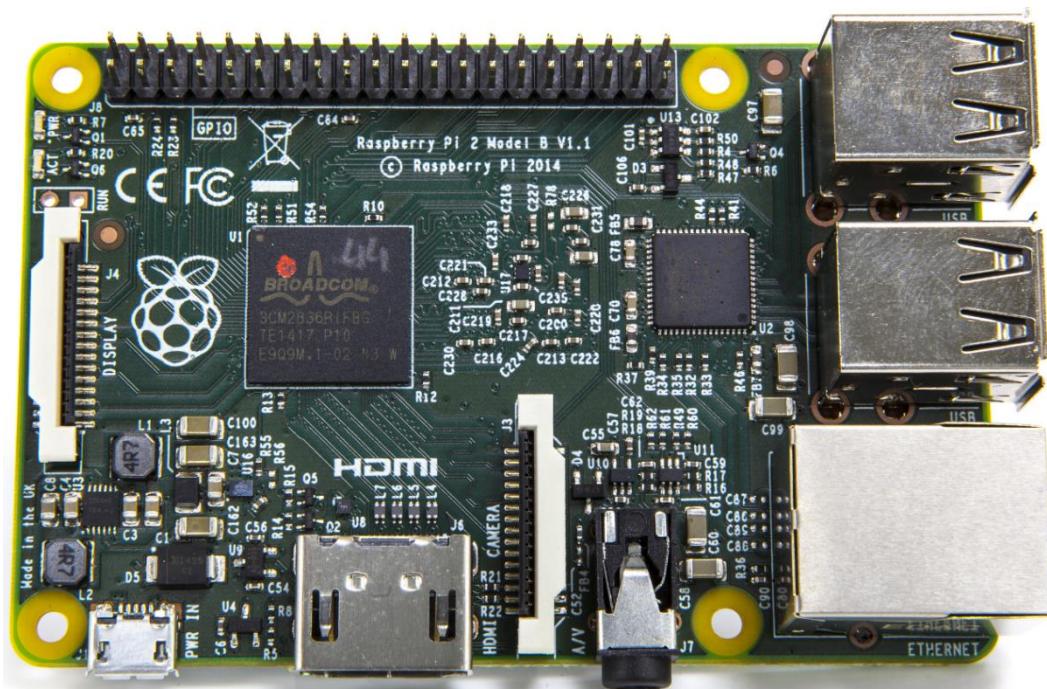
ARM Cortex-A7 mikroişlemcine sahip bu bilgisayar, 900 MHZ hızda ve 1GB RAM belleğe sahiptir. Bu geliştirme kartı üzerinde bulunan SD karta başta desteklenen Linux tabanlı dağıtımlar olmak üzere Windows 10 IoT ve Android sürümleri sorunsuz kurulabilmektedir.

Bu kart üzerinde 3,5 mm ses ve kompozit video jackı, Full HDMI video çıkışı, Ethernet portu, CSI Kamera arayüzü, DSI ekran arayüzü, Broadcom BCM2836 CPU, VideoCore IV Grafik işlemci birimi, 40 Tane GPIO pin, Mikro SD kart soketi ve 4 tane USB portu bulunmaktadır.

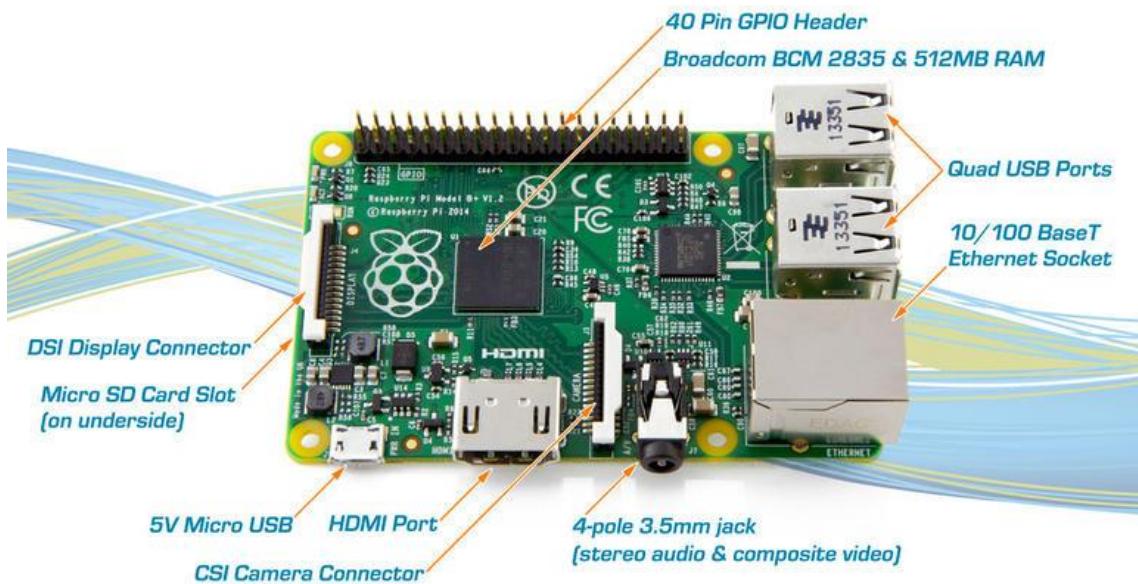
Raspberry Pi'nin özellikleri ve ayrıntıları Tablo 2.1'de verilmektedir. Raspberry Pi'nin üstten görünümü Şekil 2.1'de verilmektedir. Şekil 2.2'de ise çevre birimlerin nerelerden bağlanıldığı gösterilmektedir.

Tablo 2.1 Raspberry Pi 2 Model B'nin Özellikleri

Özellikler	Ayrıntıları
Boyutu	85.60 mm × 56.5 mm
GPU	Broadcom VideoCore IV @ 250 MHz
CPU	900 MHz quad-core ARM Cortex A7
Hafiza	1GB SDRAM
Video	HDMI ve 3.5mm jack üzerinden analog video
Audio	HDMI ve 3.5mm jack üzerinden stereo analog
USB	4 x USB2.0(Model B)
Depolama	SD Kart Üzerinden (Max 64GB SD Kart)
Güç	5V Mikro USB (En son sınır 2.1 Amper)



Şekil 2.1 Raspberry Pi 2 Model B'nin üstten görünümü



Şekil 2.2 Raspberry Pi 2'de bulunan çevresel birimler

2.2 Raspberry Pi Kurulumu İçin İhtiyaç Duyulan Minimum Donanımlar

Raspberry Pi kurulumu yapmak için bir takım çevresel donanıma ihtiyaç bulunmaktadır[10]. Bunun nedeni ise Raspberry Pi kurulumu için görüntüyü almak ve bu mini PC'yi kullanabilmek içindir. İhtiyaç duyulan donanımlar ve gerekli açıklamalar Tablo 2.2'de verilmektedir.

Tablo 2.2 Raspberry Pi 2'de bulunan çevresel birimler

Maddeler	Minimum özellik gerektiren şartlar
SD Kart	<ul style="list-style-type: none"> Minimum 4 GB ve Class 4 sınıfında olmalıdır.
HDMI Kablosu	<ul style="list-style-type: none"> HDMI çıkış destekleyen monitörler için görüntü çıkışını alabilmek için gereken kablodur.
HDMI'dan VGA'ya veya HDMI'dan DVI'ya dönüştürücü	<ul style="list-style-type: none"> HDMI çıkış bulunmayan monitörler için görüntü çıkışını alabilmek için gereken dönüştürücüdür.
3.5MM Kompozit kablosu	<ul style="list-style-type: none"> HDMI çıkış kullanılmıyorsa, analog görüntü bu kablo üzerinden alınabilir.

Klavye ve Fare	<ul style="list-style-type: none"> Raspberry Pi mini bilgisayarının kullanımı için gerekli standart PS2 klavye ve faredir.
İnternet(Ağ Kablosu) Tercihen	<ul style="list-style-type: none"> Eğer herhangi bir video çıkıştı alınamıyor ise bu bilgisayarın görüntüsünü başka bir cihazdan alınabilmesi için SSH yöntemi kullanılır. SSH ile kaynak Raspberry Pi bilgisayarının ekran görüntüsü putty ve xming gibi programlar üzerinden alınabilir.
Güç Adaptörü	<ul style="list-style-type: none"> Raspberry Pi'yi çalıştırabilmek için minimum 5v 700 mA güç kaynağı gerekmektedir. Bu şartları sağlayan akıllı telefon şarj cihazları veya powerbank üzerinden çalışabilir. Akım sınırı önerilen seviyede en fazla 2.1 mA'dır.
Audio Girişi	<ul style="list-style-type: none"> Eğer HDMI üzerinden kullanılabilirse, digital yolla bu ses alınabilir veya 3.5MM jackinden stereo olarak alınabilir. SSH teknigi ile bağlı olunan Raspberry Pi'den ses alınabilir.

2.3 Raspberry Pi İçin SD kart kurulumu

Raspberry Pi'yi kullanabilmek için bu mikroişlemci mimarisine uygun bir işletim sistemi SD kart üzerine kurulmalıdır[10]. Bu bitirme tezinde ana sağlayıcı olan Raspberry vakfı tarafından sunulan ve sitesinden indirilen Raspbian işletim sistemi kullanılmıştır. Bu işletim sistemi sayesinde Raspberry Pi'nin ihtiyaç duyduğu tüm temel yazılımlar ve programlar çalışabilicektir.

Bu kısımda, Raspberry Pi'ye Windows, Linux ve Mac işletim sistemi ile çalışan bilgisayarlar yardımıyla, SD kart aracılığıyla işletim sistemi kurmanın yolları anlatılacaktır. Bu yöntemler için [11] de anlatılan yollar takip edilmektedir.

Bu anlatımı maddeler halinde basitçe ifade eder isek:

1. 4GB veya bu kapasiteden daha büyük SD kartta işlem yapması istenen kaynak bilgisayara bağlanır. Bu tez çalışmasında programlar ve işletim sisteminin daha

fazla alana ve hızda ihtiyaç duyabileceği düşünüldüğünden 8GB Class 10 SD kart kullanılmıştır. [12]

2. Kaynak bilgisayarda SD karta işletim sistemi kurulabilmesi için öncelikle bu SD kartın İngilizce: Extended File System (Türkçe: Genişletilmiş Dosya Sistemi) formatında biçimlendirilmesi gerekmektedir. Bunlar işletim sistemlerine göre şu şekilde açıklanabilmektedir. [13]
 - A. Windows İşletim Sistemlerinde,
 - I. SD kart ile ilgili SD formatlama aracı [14] ‘de verilmekte olan bağlantından indirilir.

SD Interface Devices

The following interface devices can be used to access SD/SDHC/SDXC memory cards:

- SD slot on computer
- USB SD reader
- PC Card, CardBus or ExpressCard SD adapter

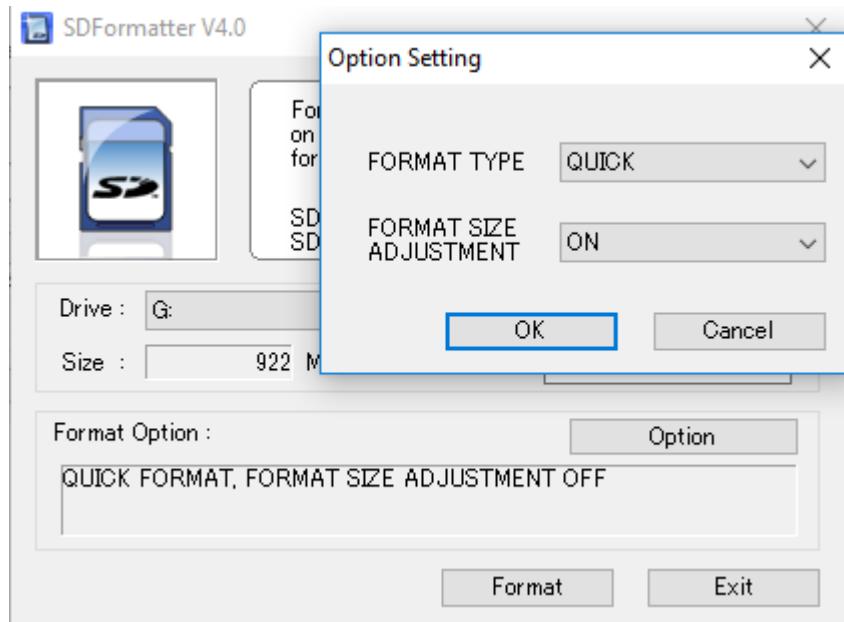
Always confirm that the device is compatible with the SD, SDHC or SDXC memory card before formatting.

SD Formatter 4.0 for Windows and Mac



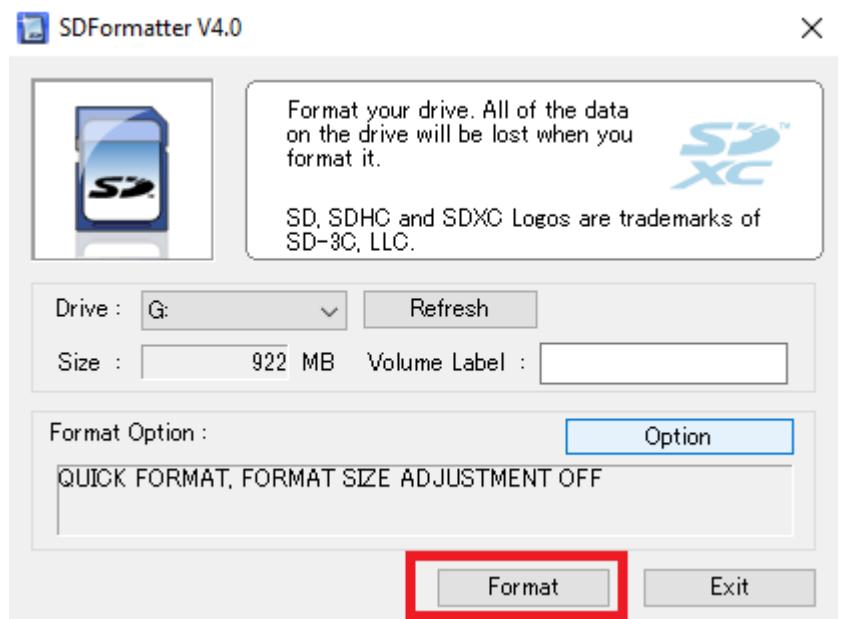
Şekil 2.2.1 SD kart formatlama aracının Windows sürümünün indirilmesi

- II. Formatlama aracı bilgisayara indirilir ve çalıştırılır ve options menüsünden Format size Adjustment Ayarı ON olarak ayarlanır.



Şekil 2.2.2 SD kart formatlama aracında gereken ayarların yapılması

- III. SD Kartın, Windows işletim sistemi tarafından tanımlanan konumu otomatik olarak programdan algılanabilmektedir ama genede bu araç tarafından seçildiği kontrol edilir ve format butonuna basılır.



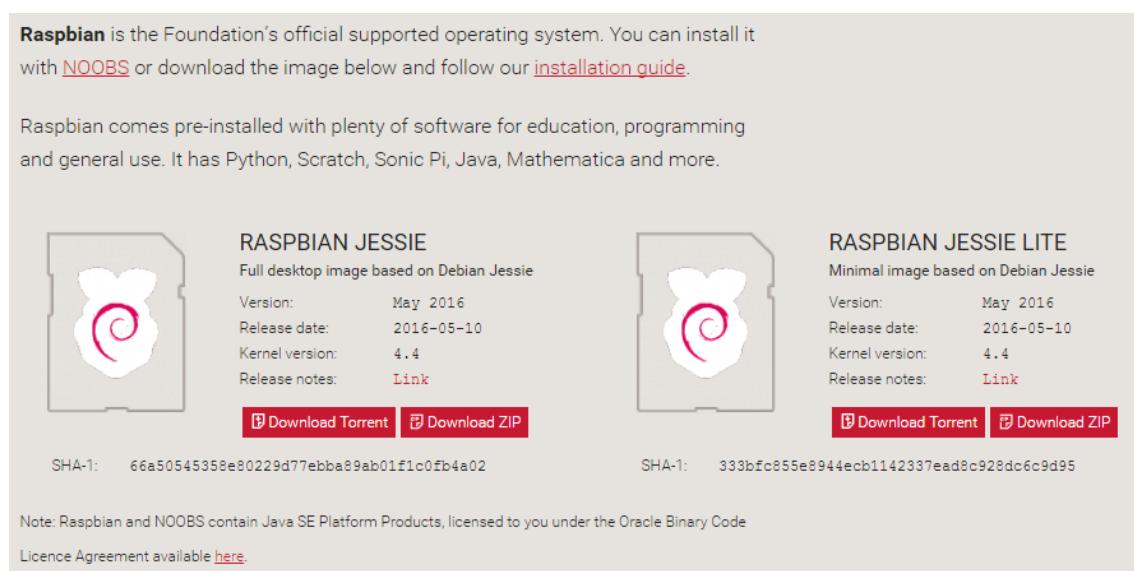
Şekil 2.2.3 SD kart formatlama aracında gereken ayarların yapılması

B. Mac İşletim Sistemlerinde,

- I. SD kart ile ilgili formatlama aracı [15] de verilen bağlantından indirilmelidir.
- II. Formatlama aracı bilgisayara kurulu ve çalıştırılır.
- III. “Overwrite Format (Üstüne yazma formatı)” seçilir.
- IV. SD kartın araç tarafından algılanıp, seçildiği kontrol edilir ve format tuşuna basılır

C. Linux İşletim Sistemlerinde

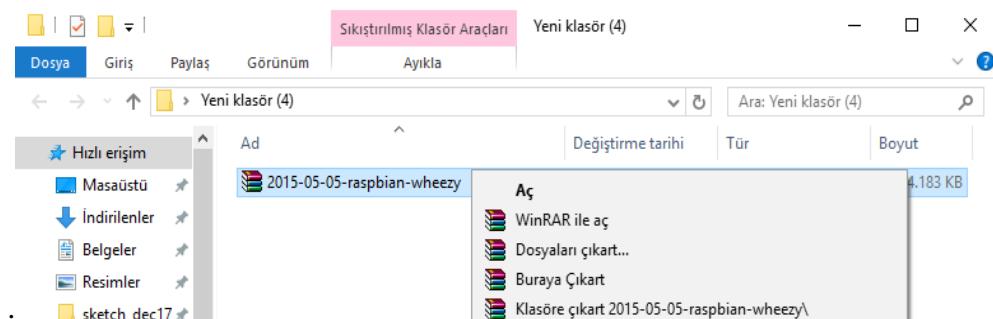
- I. “gparted” kullanılması tavsiye edilir. Yada “parted” komut satır versiyonu kullanılabilir.
 - II. Takılan SD kart FAT olarak formatlanır.
3. Raspbian işletim sisteminin en son sürümü [16]’te verilen adresten indirilir.



Şekil 2.2.4 Raspbian Jessie işletim sisteminin indirilmesi

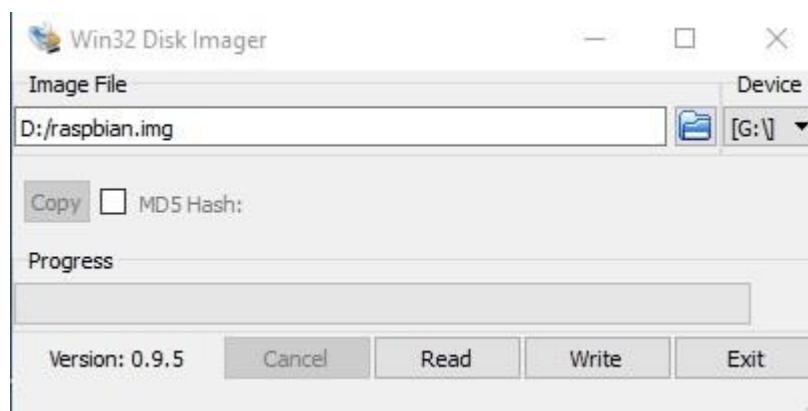
4. İndirilen bu dosya sıkıştırılmış haldedir ve öncelikle açılması gereklidir.

A. Windows İşletim Sistemi: Sağ tuşa tıklanarak “hepsini çıkar” denilir



Şekil 2.2.5 Raspbian Jessie işletim sisteminin zip'ten çıkartılması

- B. Mac İşletim Sistemi: dosya üzerine çift tıklanır ve dosyalar görüntülenir.
 - C. Linux İşletim Sistemi: komut satırında “unzip [indirilen dosya adı] çalıştırılır.
5. Çıkarılan dosyalar win32 disk imager aracına tanıtılr ve SD karta kopyalanır. Bu program bu kaynak numarasına ait referanstan indirilebilir. [17]



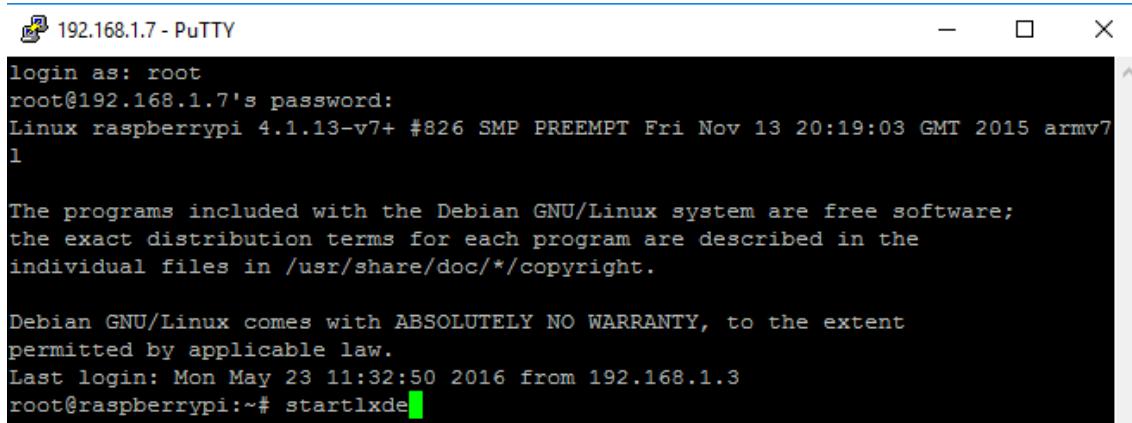
Şekil 2.2.6 Win32 disk imager programı

Kopyalamadan sonra SD kart Raspberry Pi'nin SD kart soketine takılır ve mikro USB portundan Raspberry Pi'ye güç verilir. Görüntü alınmak isteniyor ise HDMI kablosu da Raspberry Pi'ye takılabilir.

Raspberry Pi artık çalıştırılmaya hazırdir. Hızlı bir önyüklemeden sonra grafiksel işletim sistemi arayüzü ekrana gelmektedir. Grafik arayüzde şifre soracaktır. Kullanıcı adı pi, şifre ise pi olarak yazılır ve oturum açılır.

Bu bitirme tezinde Raspberry Pi bilgisayarının ekranı Ethernet üzerinden ssh yöntemi ile elde edilmiştir. Bu uygulama için Putty ve Xming programlarının bu kaynak referansından kullanılan işletim sistemine uygun mimari (x86, x64) indirilmesi

gerekmektedir. [18][19]. Bu ekranda kullanıcı oturumu açıldıktan sonra “startlxde” yazılarak Raspberry Pi ekranı elde edilmiş olur.

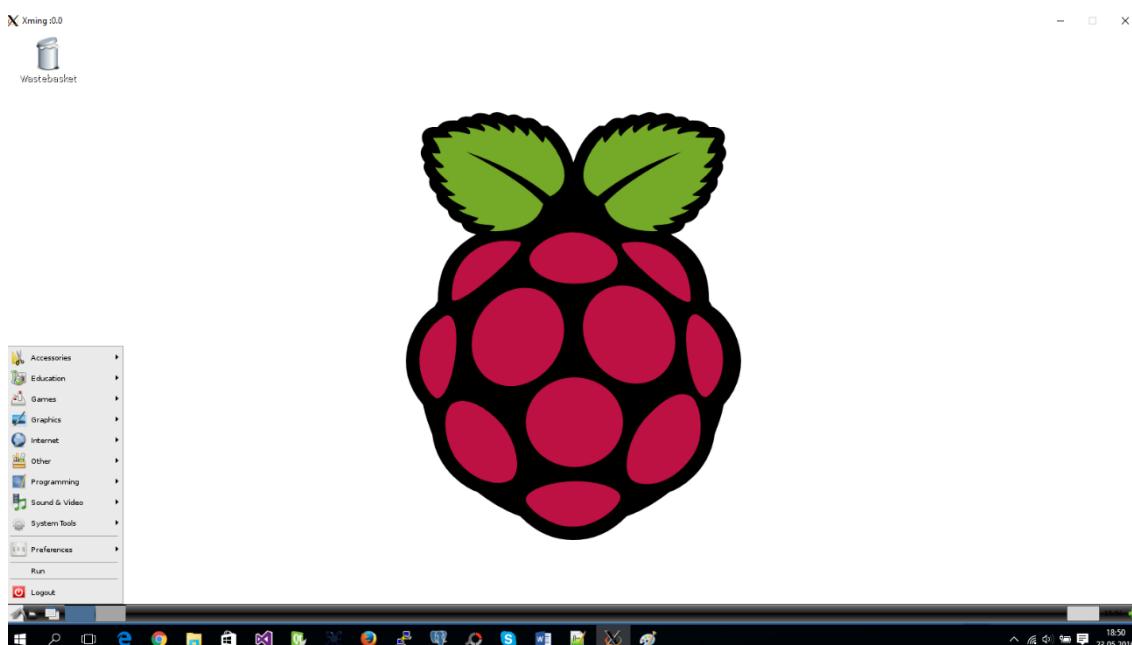


```
192.168.1.7 - PuTTY
login as: root
root@192.168.1.7's password:
Linux raspberrypi 4.1.13-v7+ #826 SMP PREEMPT Fri Nov 13 20:19:03 GMT 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon May 23 11:32:50 2016 from 192.168.1.3
root@raspberrypi:~# startlxde
```

Şekil 2.2.7 Putty SSH protokolü ile Raspberry Pi terminaline erişilmesi



Şekil 2.2.8 Putty SSH protokolü ve Xming ile sistem görüntüsünün elde edilmesi

2.4 Raspberry Pi'de GPIO

GPIO, "General Purpose Input/Output" kısaltmasıdır ve Türkçe "Genel Amaçlı Giriş Çıkış" anlamına gelir. Raspberry Pi CPU birimi olan BCM2836 mikroişlemcinin sahip olduğu pinlerdir.[20] Lojik seviyesi 3.3VDC'dir. Hangi amaçla kullanılacağı kullanıcı tarafından belirlenir ve yazılımlar tarafından programlanır.

Bu tez çalışmasının ilk uygulamasında hem Raspberry Pi hem de Arduino pinleri kullanılmaktadır. Raspberry Pi'nin GPIO pinlerinin board üzerinde gösterimi ve pinlerin işaretlenmiş gösterimi aşağıdaki şekilde verilmektedir.

Raspberry Pi'nin her sırası 20 pinden oluşan 2 sırası olmak üzere toplamda 40 pini vardır. GPIO pinleri üzerindeki ARM mikroişlemciden dolayı 3.3VDC toleransa sahiptirler. Yüksek akım veya voltaj için herhangi bir koruma bulunmamaktadır. Pinler 3.3V ile beslenmelidir. Bu voltaj değerinde maksimum akım 50mA'dır.



Şekil 2.4.1 Raspberry Pi'nin GPIO pinleri gösterimi

Raspberry Pi2 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)		DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I ² C ID EEPROM)		(I ² C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 1
26/01/2014

<http://www.element14.com>

Şekil 2.4.2 Raspberry Pi'nin GPIO pinleri diagramı

Raspberry Pi üzerindeki pinlerin herhangi bir koruması yoktur. Herhangi bir kısa devrede Raspberry Pi'nin zarar görebilir. Ayrıca Raspberry Pi'nin elektronik bir geliştirme kartı olmasından ötürü, üzerindeki pinlere sürekli bağlantılarının sökülpük takılması, SD karta ve diğer donanımlara zarar verebilir.

Tüm GPIO'lar alternatif kullanımlarda SPI, PWM, I2C ve başka sekillerde tekrar rekonfigüre edilebilir. Tablo 2,3'de tüm GPIO pinlerinin kullanımı ve fonksiyonları verilmiştir.

Tablo 2.3 GPIO pin dizilimi ve açıklamaları

Tanım	Fonksiyon	Pin No	Pin No	Fonksiyon	Tanım
DC besleme	3.3VDC	1	2	5VDC	DC Besleme
I2C SDA pini veya GPIO	GPIO02, I2C SDA	3	4	5VDC	DC Besleme
I2C SCL pini veya GPIO	GPIO03, I2C SCL	5	6	GND	Toprak
Saat sinyali veya GPIO	GPIO04, GPIO_CLOCK	7	8	GPIO014, TXD0	UART Verici Pini
Toprak	GND	9	10	GPIO015, RXD0	UART Alıcı Pini
GPIO	GPIO017	11	12	GPIO018	GPIO
GPIO	GPIO027	13	14	GND	Toprak
GPIO	GPIO022	15	16	GPIO023	GPIO
DC besleme	3.3VDC	17	18	GPIO024	GPIO
GPIO veya SPI MOSI	GPIO10, SPI MOSI	19	20	GND	Toprak
GPIO veya SPI MISO	GPIO9, SPI MISO	21	22	GPIO025	GPIO
GPIO veya SPI Clock pini	GPIO11 CLK	23	24	GPIO08	GPIO
Toprak	GND	25	26	GPIO07	GPIO
Toprak	GND	15	28	ID_SC	EEPROM I2C Erişim

EEPROM I2C Erişim Pini	ID_SD (EEPROM)	17	30	GND	Toprak
GPIO	GPIO05	19	32	GPIO12	GPIO
GPIO	GPIO06	25	34	GND	Toprak
GPIO	GPIO013	15	36	GPIO016	GPIO
GPIO	GPIO019	17	38	GPIO020	GPIO
GPIO	GPIO026	19	40	GPIO021	GPIO

Bu GPIO pinlerinden 19, 21, 23, 24 ve 26 Serial Peripheral Interface (SPI) (Seri Çevresel Arayüz) olarak kullanılabilir. Raspberry Pi'de 5 adet SPI pini bulunmaktadır. SPI sayesinde Raspberry Pi diğer cihazlarla haberleşebilir. Tabi ki Raspberry Pi ile haberleşirilecek devrenin TTL seviyesinin 3.3VDC olması gerekmektedir. Aksi takdirde Raspberry Pi zarar görücektir.

Pin 8 ve Pin 10 Universal Asynchronous Receiver/Transmitter UART (Evrensel Asenkron Alıcı/Verici) olarak çalışabilmektedir. UART, serial bus bağlantısıdır. Bu pinler 3.3VDC TTL seviyesinde çalışmaktadır.

Tüm GPIO pinleri değişimlere sahiptirler ve hepsi olmasada bazı pinler 6 farklı fonksiyona kadar çalışabilirler. [21, 22, 23, 24].

BÖLÜM 3

ARDUINO MİKRODENETLEYİCİSİ

Arduino, Atmel mikrodenetleyicileri tabanlı Atmel firmasının piyasaya sürdüğü geliştirme kartıdır[25]. Bu geliştirme kartı üzerinden 8, 16, 32 bit mikrodenetleyiciler veya bazı sürümlerinde Linux işletim sistemi çalıştırabilen mikroişlemcilerde bulunur. Bu geliştirme kartı üzerinde ADC, SPI, PWM, I2C, UART, Timer, USB, DAC, EEPROM vb, yapılar bulundurur. Bu yapılar sayesinde gerçekleştirilmek istenen elektronik projeler gerçekleştirilebilir.

Arduino platformunun dünyada en çok tercih edilmesinin sebebi ucuz ve kolay programlanabilir olmasıdır. Arduino kendi üzerindeki mikrodenetleyiciyi programlamak için ekstradan bir programlama kartına ihtiyaç duymaz. Bu sayede boyutları küçülmüş ve fiyatları bazı modellerde 1.80 \$ dolara kadar düşmüştür. [26]

Bu bitirme tezinde Arduino Atmega328p 8 bitlik mikrodenetleyici barındıran nano ve Arduino pro mini kullanılmıştır.

3.1 Arduino Yazılımının (IDE) Yüklenmesi

Arduino programlamak için gerekli yazılımı yüklemek için öncelikle www.arduino.cc web sayfasını açıp, sekmeler bölümünden indir yani (“Download”) butonuna tıklıyoruz[27].



Şekil 3.1 arduino.cc web sitesi download bölümü

Daha sonra ise açılan sayfadan hangi işletim sistemini kullanıyor ise ona ait Arduino IDE sürümü seçilir ve indirilir. Bu bitirme tezinde kodlama yapmak için Windows 10 64 bitlik işletim sistemi kullanılmıştır. Bu yüzden buna uygun Arduino sürümü indirilmiştir.



Şekil 3.2 Arduino versiyonu seçme bölümü

İşletim sisteminize uygun seçenekleri seçtikten sonra önünüze bağış sayfası gelecektir. Arduino geliştirme kartları Atmel Corporation tarafından üretilen açık kaynak kodlu bir elektronik platformdur. Sadece indir("Just download") butonuna basarak program indirilir.

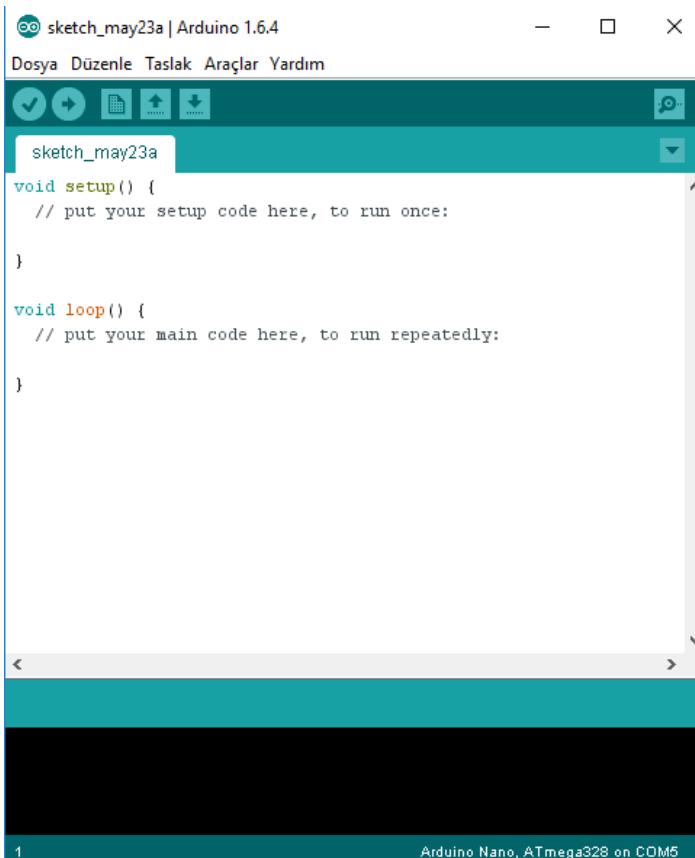


Şekil 3.3 arduino.cc donation bölümü

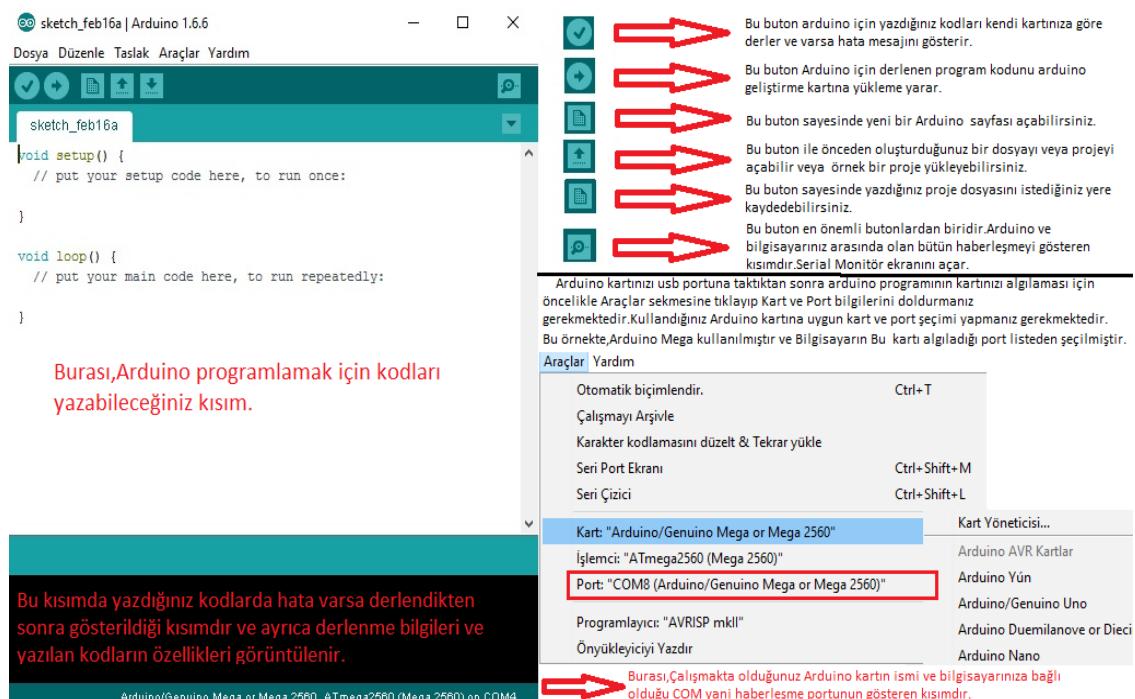
Arduino programı sunucudan inmeye başlayacaktır. İndirme bitikten sonra Arduino programını kurulur. Kurduktan sonra masaüstüne gelecek olan Arduino simgesine tıklayarak programı açılır.

3.2 Arduino Geliştirme Ortamının Kullanımı

Arduino programını açtıktan sonra karşınıza Arduino IDE ekranı gelecektir. Arduino kartınızı bilgisayarınıza herhangi bir USB girişinden bağlanabilir.



Şekil 3.4 Arduino IDE programı arayüzü



Şekil 3.5 Arduino geliştirme platformu arayüzü ve açıklamaları

Yukarda belirtilen gerekli adımları yaptıktan sonra artık Arduino programlanmaya hazırır.

3.3 Arduino GPIO Pinleri

Arduino Uno'nun üzerinde bulunan ATmega328P mikrodenetleyicisinin Şekil 3.6'da görülebileceği üzere 14 tane dijital giriş/çıkış pini vardır.[28] Bunlardan 6 tanesi PWM çıkış olarak kullanılabilir. Ayrıca 6 adet analog giriş, bir adet 16MHZ kristal osilatörü, USB bağlantısı, power jaki, ICSP başlığı ve reset butonu bulunmaktadır. Arduino Uno bir mikrodenetleyiciyi desteklemek için gerekli bileşenlerin hepsini içerir. Arduino Uno'yu bir bilgisayara bağlayarak, bir adaptör ile ya da pil ile çalıştırılabilir.

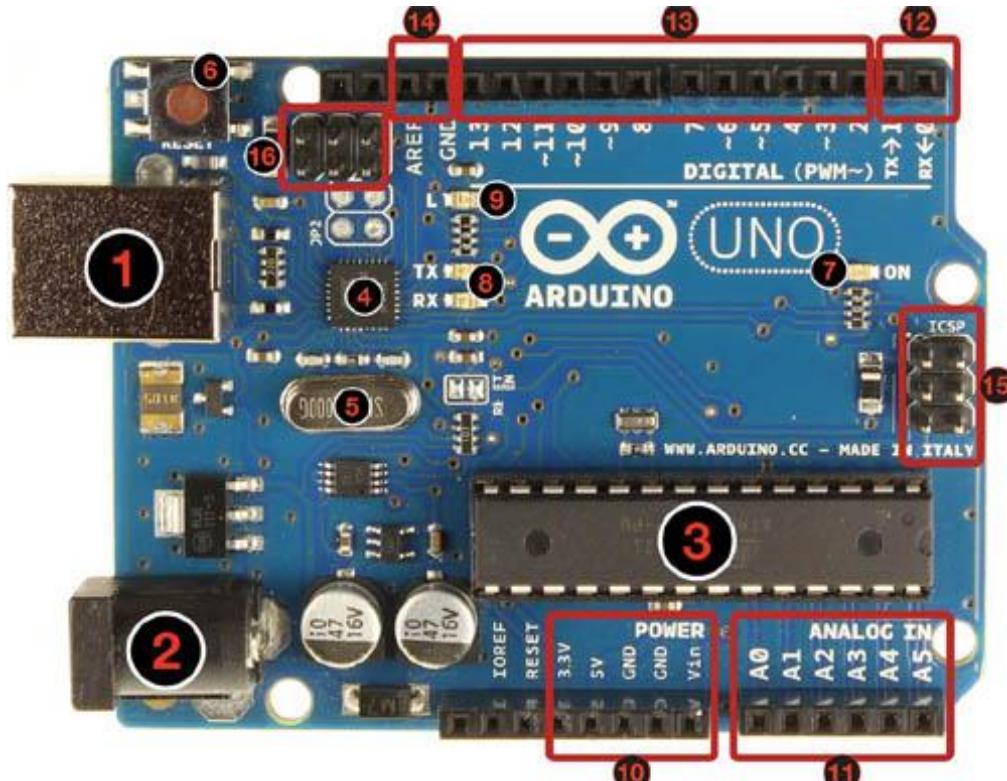
ATmega328 Pin Mapping

Arduino function		Arduino function
reset	(PCINT14/RESET) PC6	20 PC5 (ADC5/SCL/PCINT13) analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	21 PC4 (ADC4/SDA/PCINT12) analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	22 PC3 (ADC3/PCINT11) analog input 3
digital pin 2	(PCINT18/INT0) PD2	23 PC2 (ADC2/PCINT10) analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	24 PC1 (ADC1/PCINT9) analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	25 PC0 (ADC0/PCINT8) analog input 0
VCC	VCC	26 GND GND
GND	GND	27 AREF analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	28 AVCC VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	29 PB5 (SCK/PCINT5) digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	30 PB4 (MISO/PCINT4) digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	31 PB3 (MOSI/OC2A/PCINT3) digital pin 11 (PWM)
digital pin 7	(PCINT23/AIN1) PD7	32 PB2 (SS/OC1B/PCINT2) digital pin 10 (PWM)
digital pin 8	(PCINT0/CLK0/ICP1) PB0	33 PB1 (OC1A/PCINT1) digital pin 9 (PWM)

Digital Pins 11, 12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega 168 pins 17, 18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

Şekil 3.6 ATmega328 pin ayrıntıları

Aşağıdaki Şekil 3.7'de Arduino Uno geliştirme kartı görüntülenmektedir. Bu kart üzerinde bulunan kısımlar aşağıda maddeler halinde açıklanmıştır.



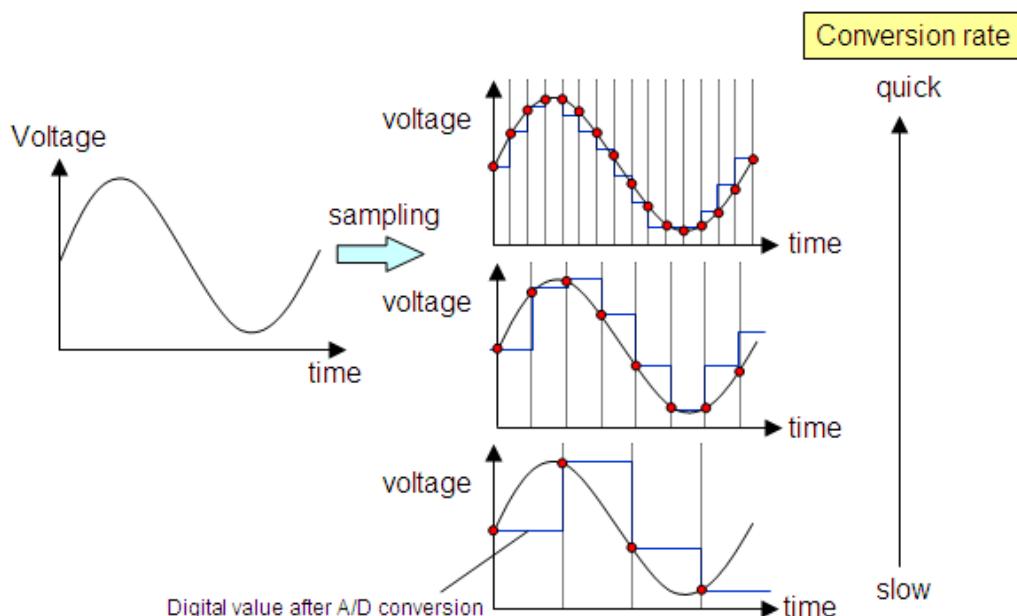
Şekil 3.7 Arduino uno ön yüzü gösterimi

1. Arduino'nun bilgisayara bağlandığı USB girişidir.
2. Arduino'nun güç kaynağı girişidir. DC jackten 5VDC ile 12VDC arası giriş uygulanabilir.
3. Uno geliştirme kartında bulunan 8 bitlik Atmega328p mikrodenetleyicisidir.
4. Atmega328p mikrodenetleyicisini programlamaya yarayan Atmel firmasının USB-TTL dönüştürücüsüdür.
5. Atmega328p'nin çalışması için gereken 16 MHz kristal osilatördür.
6. Arduino geliştirme kartını resetlemeye yarar.
7. Arduino kartının çalıştığını bildiren led ışıkları.
8. Atmega328p'nin UART pinlerine bağlanmış pinleridir. Bu sayede seri iletişim varsa ledlerden gözlemlenebilir.
9. 13 numaralı pine bağlanmış led ışıkları.
10. Arduino geliştirme kartının güç pinleridir.
11. Atmega328p mikrodenetleyicisinin analog pinleridir.
12. Atmega328p mikrodenetleyicisinin UART (seri haberleşme) pinleridir.

13. Atmega328p mikrodenetleyicisinin digital ve PWM(~) destekleyen pinleridir.
14. Atmega328p mikrodenetleyicisinin ADC referans voltajı gerilimi girişidir.
15. Atmega328p mikrodenetleyicisinin SPI ve ICSP (in circuit serial programming) pinleridir.
16. Atmel116u2 serial programlayıcı mikrodenetleyicisinin ICSP (in circuit serial programming) pinleridir.

3.4 Arduino'da ADC Kullanımı

Keşif robotu ve hasta takip sistemi uygulamalarında Arduino'nun ADC birimi kullanılmıştır. ADC (Analog Digital Converter) adından da anlaşılabilceği üzere analog sinyalleri yani sürekli zamanlı sinyalleri ayrık zamanlı sinyallere yani digital sinyallere dönüştürür[29]. Bu sayede bizim mikrodenetleyicimiz olan Arduino'nun bu sinyalleri okumasına imkan verir. Aşağıdaki Şekil 3.7'de ADC bloğu yapısı görülebilir.



Şekil 3.8 ADC bloğu yapısı

Dünyada çeşitli ADC entegreleri bulunmaktadır ama gelişen teknoloji ile artık bu entegreler mikrodenetleyicilerin içinde de yer almaktadırlar. Arduino kendi içindeki ATmega328 mikrodenetleyicinde bir adet ADC entegresi barındırmaktadır.

ADC entegreleri okuma yapabilmek için bir referans voltaja ihtiyaç duyarlar. Arduino'da default durumda referans voltajı 5 Volta yani VCC'ye çekilmiştir ama analogReference()

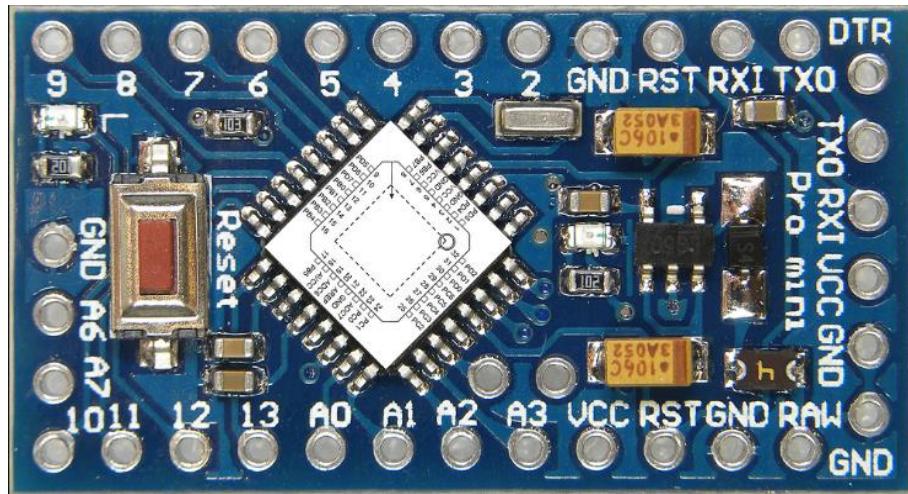
komutu ile değişik voltajlara (5.0V, 3.3V, 2.56V, 1.1V) gibi seviyelere çekilebilir. Arduino içindeki ADC 10 bitlidir[30]. 10 bit $2^{10}=1024$. Referans voltajımızı 5V olarak belirlediğimizde, ADC modülü 1024 değerinin her birine bu voltajı eşit olarak dağıtır. Yani ADC modülü okuduğumuzda 0 volt= 0, 5 V=1023 değeri ile bize döner. Her bir bit değerinin alacağı voltaj değerine bakarsak $5/1024=0.00488$ volt olarak karşımıza çıkmaktadır. Yani diyelimki ADC modülden okuma yaptığımızda 500 değerini alıyor isek yazılımda $500*0.00488$ işlemini yaparak ADC mızın girişindeki voltajın 2.44 volt olduğunu hesaplamış oluruz.

Ayrıca arduino üzerinde bulunan AREF pinine uygulanacak (0-5VDC) arası gerilimler yardımıyla referans voltajı düşürmemiz mümkündür. Bunun için analogReference (EXTERNAL) komutu kullanılmalıdır.

Arduino'nun üzerinde ADC çevrim yapabilmek için ayrılmış analog sinyal okuyucu pinleri bulunmaktadır. Bu pinlerin sayısı kullanılan Arduino geliştirme kartına göre değişmektedir. Arduino uno üzerinde A0'dan A5'e kadar 6 tane olmak üzere analog pin vardır. Bu pinler en fazla 5V ile çalışabilirler. 5V üzeri gerilimlerde pinler hasar görebilir veya Arduino kartı yanabilir. Bu yüzden işlem yaparken dikkatli olunması gerekmektedir.



Şekil 3.9 Arduino Nano geliştirme kartı

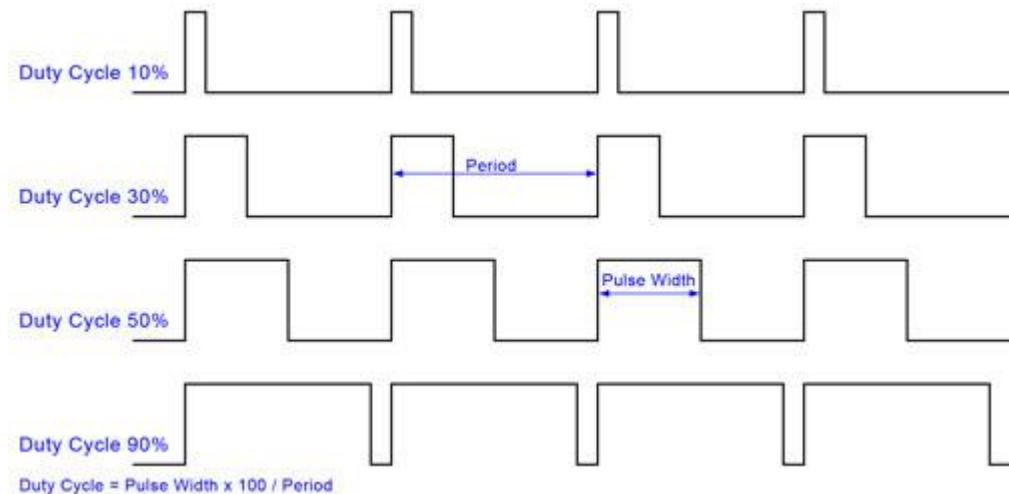


Şekil 3.10 Arduino Pro Micro Geliştirme kartı

BÖLÜM 4

DARBE GENİŞLİK MODÜLASYONU

Darbe Genişlik Modülatörü (PWM-Pulse Width Modulation), darbe genişlik modülasyonu anlamına gelmektedir. PWM sinyali frekansı ve genliği ayarlanabilen bir kare dalga formunda bir digital dalgadır[31].



Şekil 4.1 PWM sinyalinde periyot, darbe genişliği ve doluluk boşluk oranı gösterimi

Bu şekilde gösterilen örnek PWM sinyali bir kare dalga formu olup, belirli bir T zamanı içinde hem LOW hem de HIGH olabilmektedir. Bu periyot süresini değiştirmeden T süre içerisinde HIGH ve LOW süreleri değiştirilerek PWM sinyal üretilir.

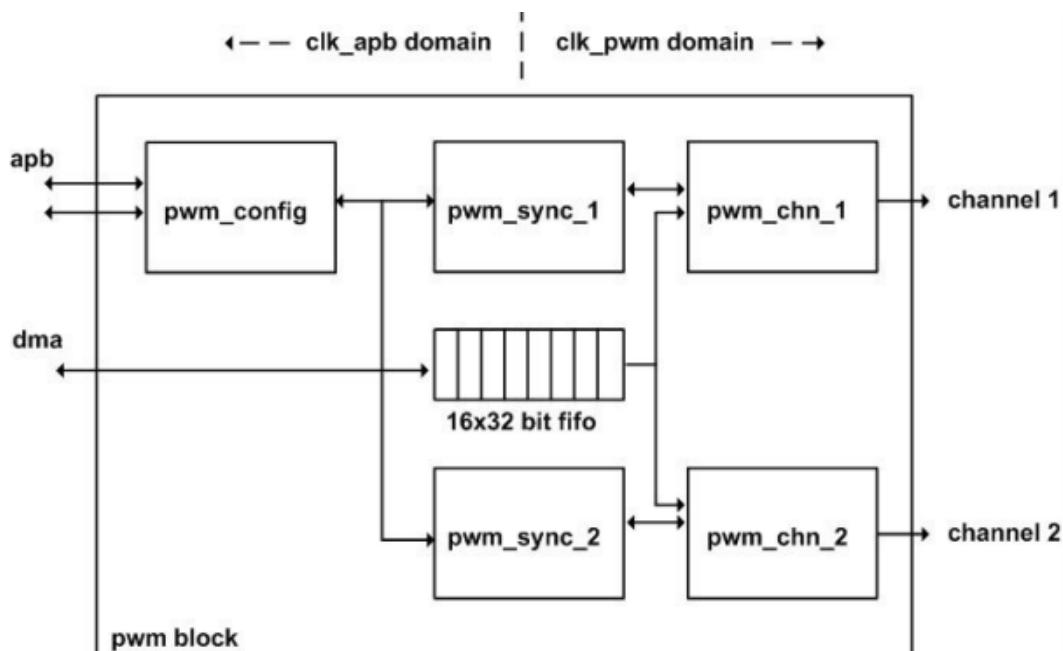
Örneğin $T=100$ us olan bir kare dalganın doluluk boşluk oranı %40 ise $T=T_{low} + T_{high}$ şeklindedir. Burdan anlaşılacağı üzere HIGH süresi 40us, LOW süresi ise 60 us'dır. Başka bir örnek verir ise gene aynı periyot da boşluk oranı %20 olan aynı kare dalga sinyalinin HIGH süresi 20 us, low süresi ise 80 us olucaktır. Periyot ise değişmemiştir.

PWM sinyaller günümüzde en çok ledlerin aydınlatığının yapılmasında, DC motorların sürülmüşinde ve güç kaynaklarında kullanılır. Bu gibi sistemlerde gerek duyulan analog sinyaller PWM sinyaller kullanılarak digital sinyallerden elde edilmektedir. Bu sayede örneğin DC motorların hız kontrolleri yapılabilmektedir.

4.1 Raspberry Pi'de PWM Kullanımı

Raspberry Pi'den donanımsal ve de yazılımsal olarak PWM sinyal üretmek mümkündür. Raspberry Pi, bu PWM sinyalini isteğe göre kendi mikroişlemcisi olan BCM2836 mikroişlemcisinde ilgili pinler için oluşturmaktadır. Bu donanımsal PWM sinyal üretebilmek için gerekli açıklamalar ilgili mikroişlemcinin datasheetinde açıklanmıştır. [16]. Bu bitirme tezinde yazılımsal olarak bir PWM sinyal oluşturulmuştur.

Raspberry Pi 2'nin Broadcom BCM2836 mikroişlemci tarafından sağlanan donanımsal PWM'in blok diyagramı aşağıdaki Şekil 4.1'de verilmiştir.



Şekil 4.1 Raspberry Pi BCM2836 çipinin PWM blok diyagramı[33]

4.2 Raspberry Pi'de Yazılımsal PWM Kullanımı

Keşif robottu üzerinde bulunan DC ve servo motorları hareket ettirmek için PWM sinyale ihtiyaç duyulmuştur. Bu tez çalışmasında Raspberry Pi'nin PWM pin sayısı yetmediğinden ötürü diğer GPIO pinlerinden PWM sinyal üretmek amacı ile yazılımsal PWM kullanılmıştır. Bu sayede bu motorların kontrolleri sağlanmış bulunmaktadır. Raspberry Pi ürettiği bu sinyal aracılığı ile aslında motorların çalışması için gereken sinyali oluşturmaktadır.

Raspberry Pi'de PWM sinyal aşağıdaki yöntemler ile elde edilebilir. Bunları sıralar isek:

- Üzerindeki mikroişlemci donanımı ile: Donanımsal olarak Raspberry Pi'nin kendi mikroişlemcisi üzerinden PWM sinyal üretmek mümkündür. Bu PWM sinyal yalnızca GPIO18 numaralı pin vasıtası ile üretilebilir. Eğer bu aktif edilirse ses çıkışına engel olacaktır. Bu işlem için Raspberry Pi üzerindeki kernelin tekrardan derlenmesi gerekmektedir.
- Yazılımla: Yazılımsal olarak PWM sinyal gerekli kodlar veya programlar ile hızlı bir şekilde üretilebilmektedir. Donanımsal PWM'den pek fazla bir farkı bulunmamaktadır. Ama üretilen her sinyal nedeniyle CPU da %0,5 veya %1 lik fazla bir kullanım söz konusu olabilmektedir.
- Harici entegreler yardımıyla: Raspberry Pi'ye dışardan bağlanabilecek ve Raspberry Pi ile SPI veya başka bir haberleşme birimi ile haberleşebilen bir entegre veya mikrodenetleyici ile bu sinyali oluşturmak mümkündür.

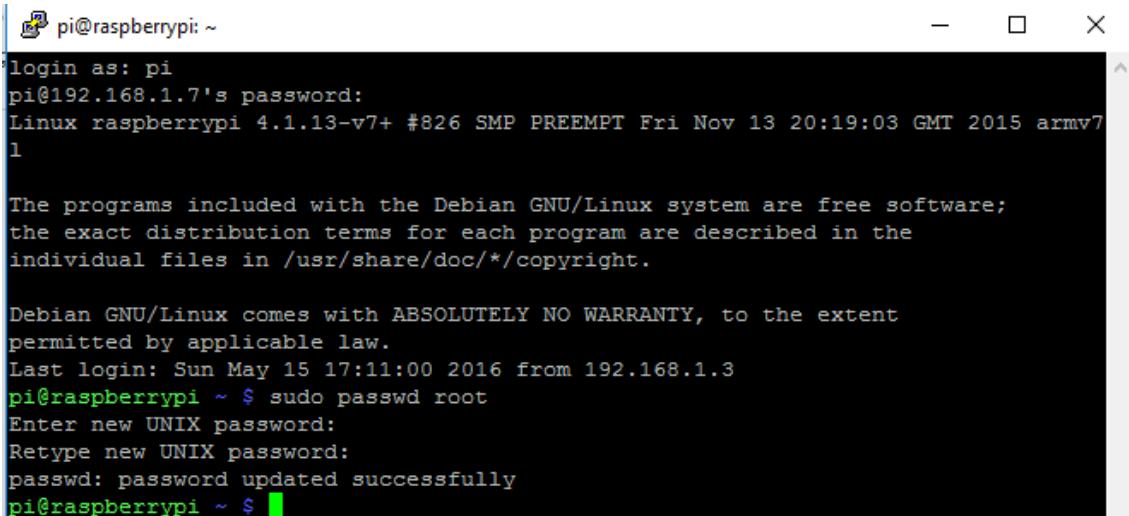
Bu bitirme projesinin ilk uygulaması olan keşif robottunda 2 tanesi DC motor ve de 2 tanesi servo motor olmak üzere 4 motor kullanılmıştır. Bu yüzden 4 tane Raspberry Pi GPIO pini yazılımsal PWM sinyal üretmek için kullanılmıştır. Bu yazılımsal PWM sinyali üretmek için C++ programlama ortamında WiringPi ve ServoBlaster kütüphaneleri kullanılmıştır.[33] [34] Bu kütüphaneler Raspberry Pi'nin komut satırı olarak adlandırılan terminalinde aşağıdaki komutlar aracılığıyla indirilip kurulmuştur.

4.2.1 WiringPi Kütüphanesinin Kurulumu

Raspberry Pi'de yazılımsal olarak PWM sinyal oluşturabilmek için öncelikle standart kullanıcı modundan çıķıp SuperUser moduna (root) girilmesi gerekmektedir.[35] Bu ise aşağıdaki komutlar aracılığı ile yapılır. Bu işlem için öncelikle normal kullanıcı gibi girilir ve ardından aşağıdaki komutlar uygulanır.

```
$ sudo passwd root
```

Bu aşamadan sonra ise bu kullanıcı için şifre atanması gerekmektedir. İki kez şifre girilmesi tekrarlanır ve enter tuşuna basılır. Gerekli işlemler aşağıdaki Şekil 4.2'de görülebilir.



```

pi@raspberrypi: ~
login as: pi
pi@192.168.1.7's password:
Linux raspberrypi 4.1.13-v7+ #826 SMP PREEMPT Fri Nov 13 20:19:03 GMT 2015 armv7l

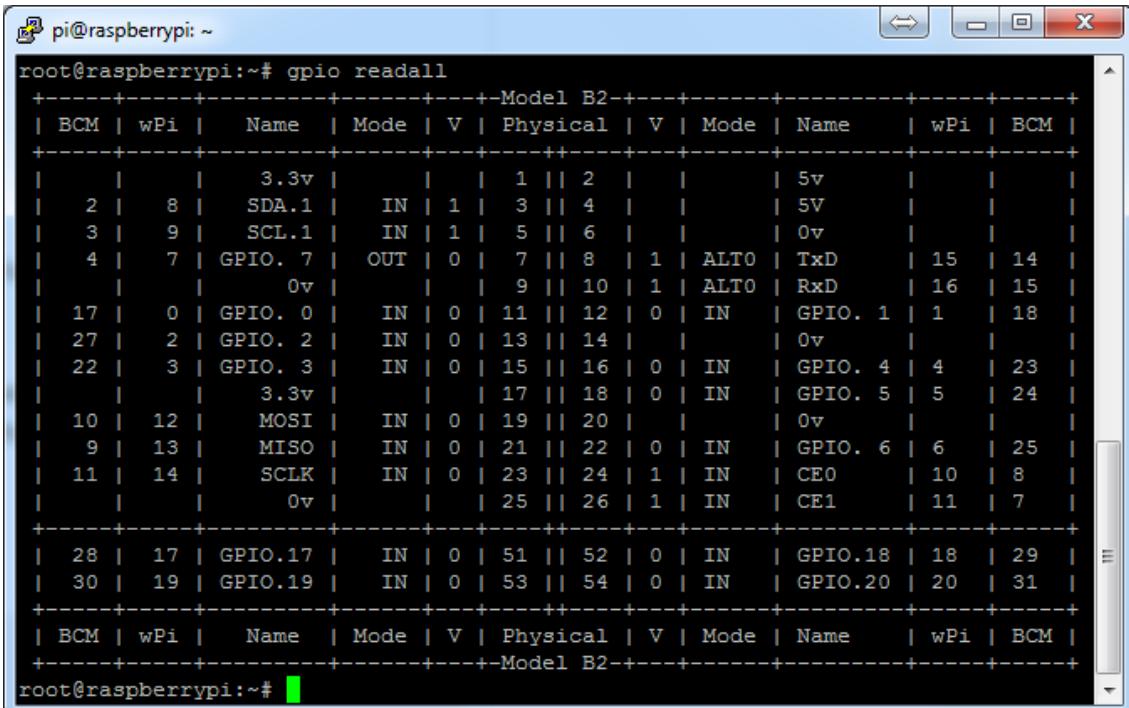
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 15 17:11:00 2016 from 192.168.1.3
pi@raspberrypi ~ $ sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
pi@raspberrypi ~ $

```

Şekil 4.2 Raspberry Pi’de superuser yetkisine sahip kullanıcı oluşturma

Bundan sonra artık Raspberry Pi’ye kullanıcı adı: root, şifre: toor yazılarak superuser yetkisi ile girilecektir. Bu işlemlerden sonra WiringPi kütüphanesi Raspberry Pi içine kurulmuştur. WiringPi kütüphanesi ile Raspberry Pi GPIO numara bakımından farklıdır. Örneğin WiringPi’de 17 numaralı pin Raspberry Pi’nin GPIO28 numaralı pinine karşılık gelmektedir. Aşağıdaki Şekil 4.3’te bu pin atamaları gösterilmektedir.



```

root@raspberrypi:~# gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     | 3.3v |     |   |     |   |     | 5v  |     |   | |
| 2   | 8   | SDA.1 | IN  | 1  | 3  | 4  |     | 5V  |     |   |
| 3   | 9   | SCL.1 | IN  | 1  | 5  | 6  |     | 0v  |     |   |
| 4   | 7   | GPIO. 7 | OUT | 0  | 7  | 8  | 1  | ALTO | TxD  | 15  | 14 |
|     |     | 0v   |     |   | 9  | 10 | 1  | ALTO | RxD  | 16  | 15 |
| 17  | 0   | GPIO. 0 | IN  | 0  | 11 | 12 | 0  | IN   | GPIO. 1 | 1   | 18 |
| 27  | 2   | GPIO. 2 | IN  | 0  | 13 | 14 |     | 0v   |     |   |
| 22  | 3   | GPIO. 3 | IN  | 0  | 15 | 16 | 0  | IN   | GPIO. 4 | 4   | 23 |
|     |     | 3.3v |     |   | 17 | 18 | 0  | IN   | GPIO. 5 | 5   | 24 |
| 10  | 12  | MOSI  | IN  | 0  | 19 | 20 |     | 0v   |     |   |
| 9   | 13  | MISO  | IN  | 0  | 21 | 22 | 0  | IN   | GPIO. 6 | 6   | 25 |
| 11  | 14  | SCLK  | IN  | 0  | 23 | 24 | 1  | IN   | CE0   | 10  | 8  |
|     |     | 0v   |     |   | 25 | 26 | 1  | IN   | CE1   | 11  | 7  |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 28  | 17  | GPIO.17 | IN  | 0  | 51 | 52 | 0  | IN   | GPIO.18 | 18  | 29 |
| 30  | 19  | GPIO.19 | IN  | 0  | 53 | 54 | 0  | IN   | GPIO.20 | 20  | 31 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+
root@raspberrypi:~#

```

Şekil 4.3 WiringPi’nin Yazılımsal pinlerinin Raspberry Pi terminalinde görüntülenmesi

Bu sayede birçok Raspberry Pi GPIO pinleri PWM çıkış olarak ayarlanabilecektir. Örnek bir PWM sinyal üretmek için aşağıdaki C++ kodları yazılabilir.

İlk önce WiringPi için C++ dilinde şu komutlar yazılır.

```
wiringPiSetup () ;
pinMode (25, OUTPUT) ;
pinMode (24, OUTPUT) ;
pinMode (23, OUTPUT) ;
pinMode (22, OUTPUT) ;
softPwmCreate (29,0,250);softPwmWrite (29,0) ;
softPwmCreate (27,0,250);softPwmWrite (27,0) ;
```

Ardından ise çalıştırılmak istenen PWM pinler için gerekli ayarlamalar yapılır. Örneğin aşağıdaki alınan C++ fonksiyonunda TCP/IP üzerinden gelen x ve y (bunlar sırası ile sağ ve sol motorun PWM frekans değerleridir.) değerlerine göre ilgili pinlerden PWM işaret üreten fonksiyonun kaynak kodudur.

```
void motorileri(int x) {
    softPwmWrite(29,x);
    softPwmWrite(27,x);

    digitalWrite(25,LOW );
    digitalWrite(24,HIGH);
    digitalWrite(23, LOW );
    digitalWrite(22, HIGH);
    delay(30);
}
```

4.2.2 ServoBlaster Kütüphanesinin Kurulumu

Bu kütüphane kamera pan-tilt sistemini kontrol eden pan-tilt servo motorları sürmek için hazırlanmıştır. WiringPi kütüphanesinin servo motorları sürmesindeki zayıflıklarından ötürü bu kütüphane seçilmiştir. Bu yüzden öncelikle Raspberry Pi komut satırından Servoblaster kütüphanesi aşağıdaki komutlar aracılığı ile indirilip ardından derlenmesi gerekmektedir.

```
pi@raspberrypi ~ $ git clone https://github.com/richardghirst/PiBits.git
pi@raspberrypi ~ $ cd PiBits/ServoBlaster/user/
pi@raspberrypi ~ $ make servod
pi@raspberrypi ~ $ ./servod
```

Bu işlemlerden sonra Raspberry Pi derlenmiştir. Servoblaster bir kez çalıştığı zaman sürekli çalışır. Raspberry Pi için bu kütüphane tarafından kullanılan pinler aşağıdaki Tablo 4.1'de görülebilir.

Tablo 4.1 ServoBlaster kullanılabilen PWM pinlerinin bilgisi

Kanal Numarası	GPIO Numarası	Pin Başlık Numarası
0	4	P1-7
1	17	P1-11
2	18	P1-12
3	21	P1-13
4	22	P1-15
5	23	P1-16
6	24	P1-18
7	25	P1-22

Bu kütüphane sayesinde Raspberry Pi üzerinde 8 kanallı PWM sinyali aynı anda üretilebilir. Komut satırı üzerinde deneme yapmak ister isek pin 1’i %40 doluluk oranı olarak PWM sinyali üretmek istersek şu komutu çalıştırabiliriz.

- Pin 1’i %40 PWM olarak ayarlamak için terminale:

```
sudo echo "1=40" > /dev/servoblaster
```

- Pin 1’i tamamen kapatmak için terminale:

```
sudo echo "1=0" > /dev/servoblaster
```

- Pin 1’i tamamen açacak olur isek terminale:

```
sudo echo "1=100" > /dev/servoblaster
```

Bu komutlarla istenilen frekans değerinde PWM sinyal üretilebilir.

Aşağıdaki komut Raspberry Pi üzerinde yazılan servo motorların sürülmesi için gerekli bir C++ tabanlı programdır.

```
fp = fopen("/dev/servoblaster", "w");
QString komut = "echo 1="+QString::number(y_ekseni)+"% >
/dev/servoblaster";
system(qPrintable(komut));
QString komut2 = "echo 0="+QString::number(x_ekseni)+"% >
/dev/servoblaster";
system(qPrintable(komut2));
```

Sinyallerin kesilmesi istendiğinde aşağıdaki kod kullanılabilir. Çünkü Servoblaster bir kere çalışıldığında sürekli olarak çalışır.

```
root@raspberrypi:~# sudo killall servoblaster
```

Bu sayede gelen TCP/IP protokolü üzerinden kontrolör tabletten kablosuz bir şekilde gelen yön bilgisine göre servo motorların sürülmesi için gerekli sinyaller üretilmiş olur.

Bu uygulamada bütün motorların sürülmesi için gerekli kodlar eklerde verilmiştir.

Bu sayede yazılan kodlar sayesinde üretilicek PWM sinyaller ile DC motorun hız kontrolü ve servo motorlara bağlı olan kamera pan-tilt sisteminin pozisyonları yer değiştirebilicektir. Bu tezde DC motorları ve servo motorları sürmek için farklı farklı iş parçacığına sahip C++ tabanlı programlar yazılmıştır.

BÖLÜM 5

Tablet Bilgisayarla Kontrol Edilen Kablosuz Keşif Robotu

Bu bitirme tezinin ilk uygulaması olarak tasarlanan ve gerçekleştirilen keşif robotu sistemi 2 kısımdan oluşmaktadır. Bu kısımlar gezgin birim olan keşif robotu ve bu keşif robotunu uzaktan kontrol eden mobil tablet birimidir. Bundan sonra gezgin birimi olan keşif robotunu kumanda eden mobil tablete “kontrolör tablet” ifadesi kullanılacaktır.

5.1 Keşif Robotu Birimi

Bu uygulamada tasarlanan robot şu birimlerden oluşmaktadır.

- Batarya ve DC-DC çeviriçi
- Motorlar ve sürücüsü
- Optokuplör devreleri
- Raspberry Pi mikrobilgisayarı
- Kamera Servo Pan-Tilt mekanizması
- Kameralar
- Sensörler
- WiFi ağ adaptörü

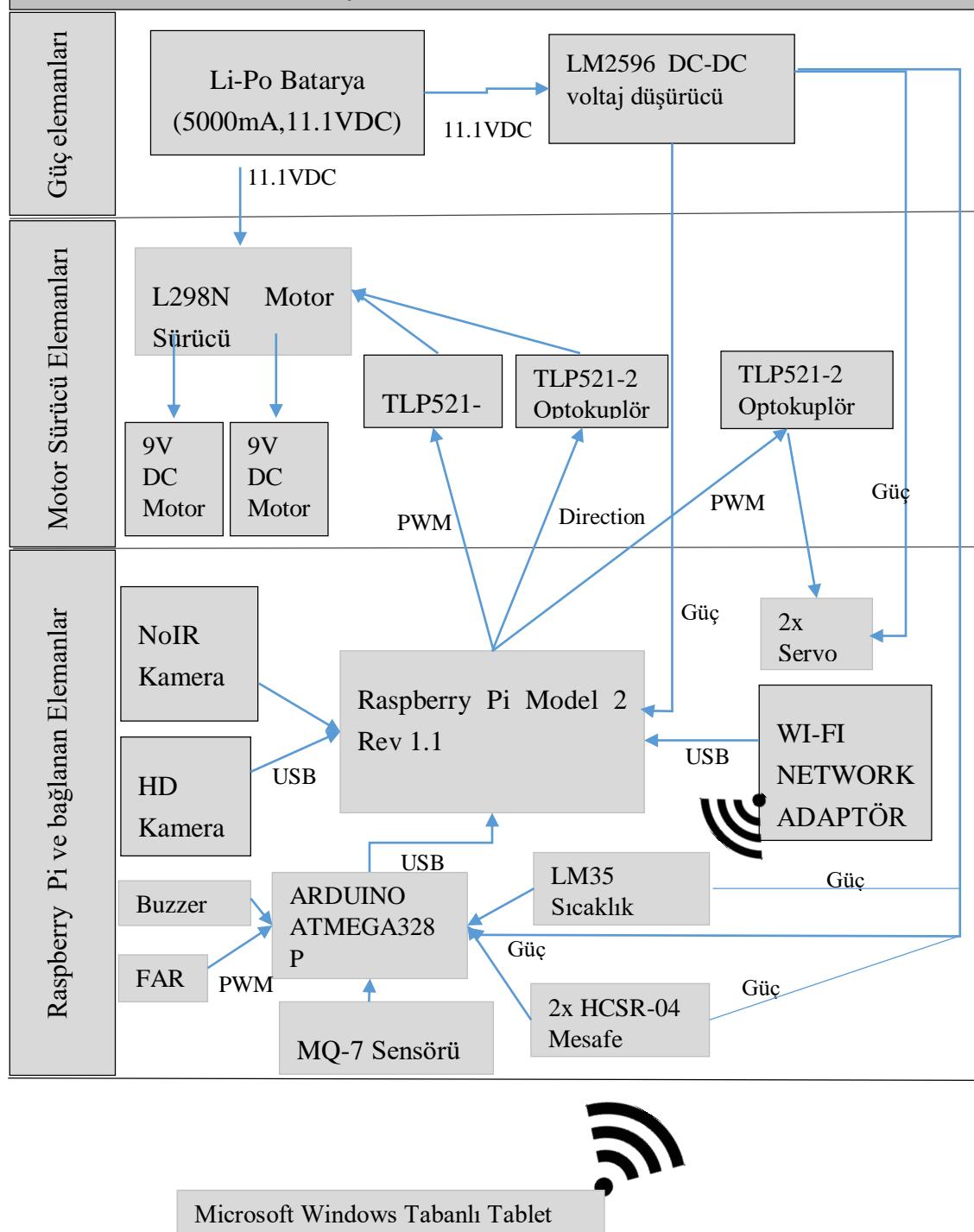
Keşif robotu Sistem aşağıdaki elemanlara sahiptir ve blok diyagramı Şekil 5.1’de gösterilmektedir. Bu diyagramda bağlantıları gösteren parçalar ve parçaların birbirleri ile olan ilişkileri gösterilmektedir.

5.2 Kontrolör Birimi

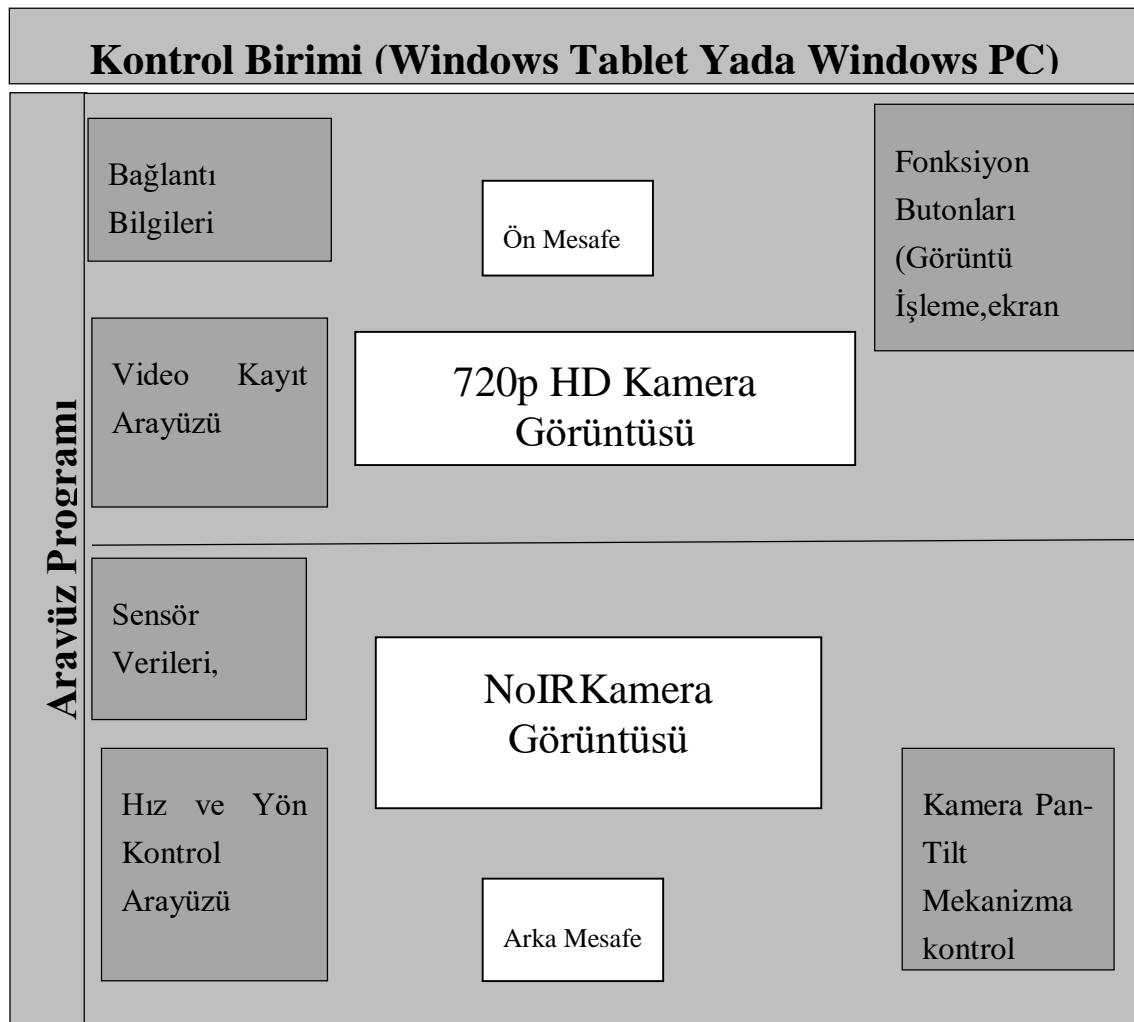
Bu uygulamada tasarlanan kontrolör Wi-Fi haberleşme ile kablosuz olarak keşif robotunu kontrol eden birimdir. Gezgin birimi olan keşif robotunu kontrol etmek üzere yeterli donanıma ve hız'a sahip Microsoft işletim sistemi bir tablet tercih edilmiştir. Bu sayede Windows işletim sistemi kullanan tüm bilgisayarlar tarafından rahatça kurulabilmekte ve C# tabanlı yazılım ile yazılan basitleştirilmiş arayüzü sayesinde kontrol edilebilmektedir. Kontrolör birimi olan tabletin blok diyagramı aşağıdaki şekilde gösterilmektedir.

Kontrolör birimi aşağıdaki parçalara sahiptir ve blok diyagramı Şekil 5.2’de gösterilmektedir. Bu diyagramda bağlantıları gösteren parçalar ve parçaların birbirleri ile olan ilişkileri gösterilmektedir.

TABLET KONTROLLÜ KEŞİF ROBOT



Şekil 5.1 Keşif robotu kısmının bileşenleri ve bağlantıları



Şekil 5.2 Kontrolör tablet PC kısmının bileşenleri ve bağlantıları

Şekil 5.1'de görülen robot ve Şekil 5.2'de görülen kontrolör birimleri birbirleri ile soket haberleşme üzerinden kablosuz olarak iletişimde olmaktadır. İlerleyen bölüm 5.3 ve bölüm 5.4'te bu birimlerin çalışmaları ayrıntılı olarak verilecektir.

5.3 Keşif Robotunun Çalışma Prensipleri

Yukarıda da belirtildiği üzere robot birimi şekillerden de görüleceği üzere 8 farklı bölümden oluşmaktadır.

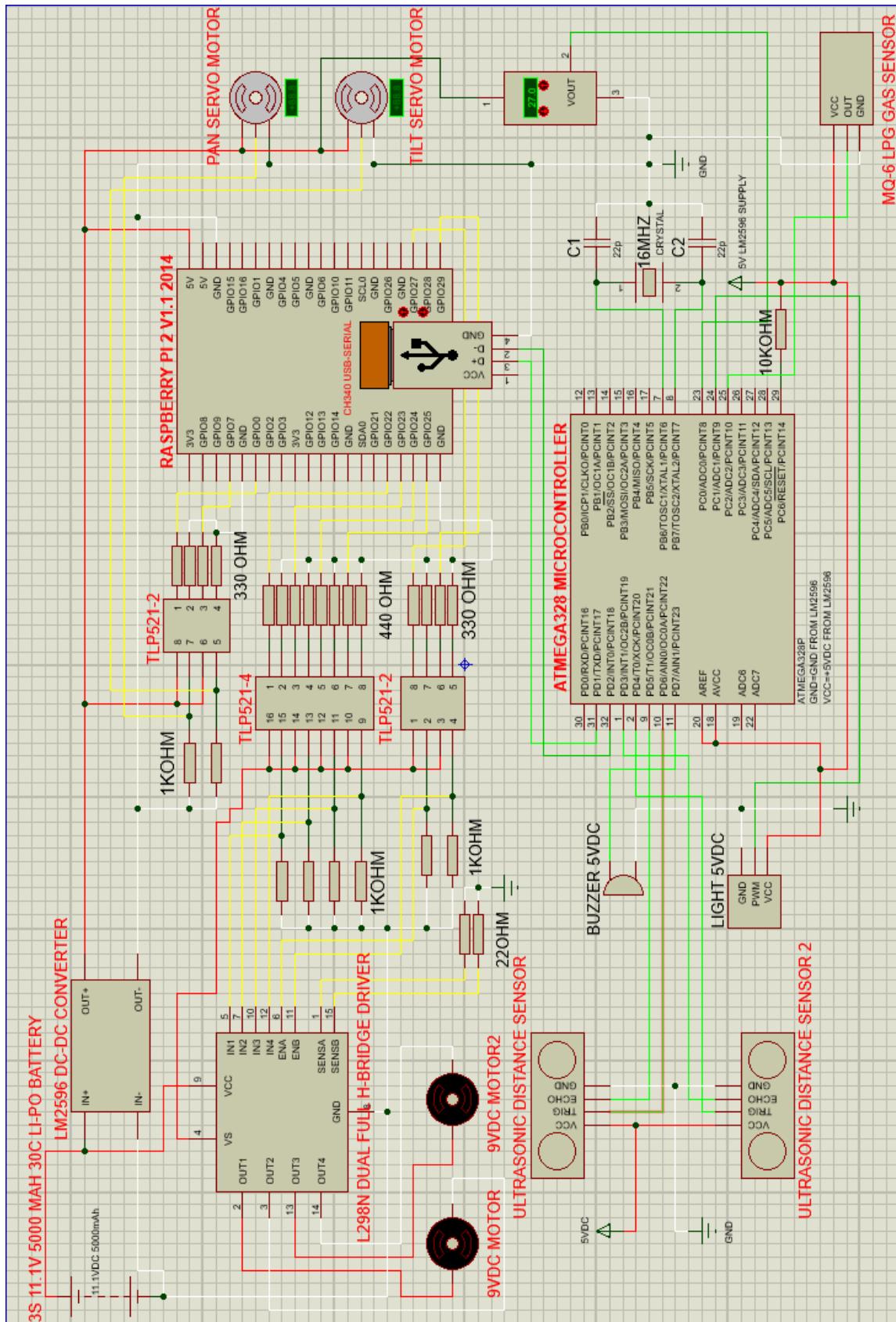
Bu uygulamada tasarlanan robot şu birimlerden oluşmaktadır.

- Batarya ve DC-DC çevirici
- Motorlar ve sürücüsü
- Optokoplör devreleri
- Raspberry Pi mikrobilgisayarı

- Kamera Servo Pan-Tilt mekanizması
- Kameralar
- Sensörler
- WiFi ağ adaptörü

Keşif robotunun ana beyni bu sistemde Raspberry Pi 2 minibilgisayarıdır. Raspberry Pi üzerinde kontrolör tablet ile kablosuz iletişim kurması için bir Wi-Fi ağ adaptörü bulundurmaktadır. Aynı Wi-Fi ağ adaptörü kontrolör tabletin üzerinde de bulunmaktadır. Raspberry Pi, USB üzerinden bağlı olduğu HD Kamera ve CSI portundan kendisine bağlı bulunan kameradan alınan görüntüyü jpg streamer aracılığı ile kablosuz tablete aktarım yapar. Aynı sırada Arduino'nun okuduğu sensör verilerini de kontrolör tablete yollamaktadır. Raspberry Pi ve Arduino birbirleriyle USB-TTL üzerinden UART haberleşme tekniği ile haberleşmektedir. Kontrolör tabletten gelen hız, yön, pan-tilt mekanizması bilgilerine göre motorlar için PWM sinyaller üreterek bu sinyali motor sürücü devresine ileter ve keşif robotunun hareketine sağlar.

Raspberry Pi'nin 2 tanesi PWM, 4 tanesi hareket yönünü kontrol etmek için 6 adet pini DC motorun dolayısı ile keşif robotunun hareketi için kullanılmıştır. Servo pan-tilt mekanizmasının kontrol edilebilmesi ve böylece kamera bakış açılarının değiştirilebilmesi için 2 adet Raspberry Pi GPIO pini PWM çıkış olarak kullanılmıştır. Keşif robotunun tüm devre çizimleri aşağıdaki şekil 5.3'de gösterilmektedir.



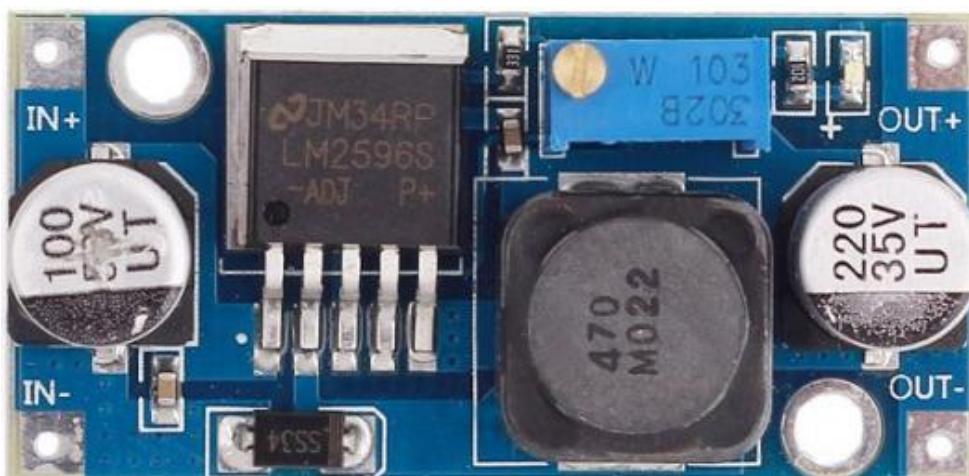
Şekil 5.3 Keşif robotunun tasarlanan tüm pin diyagramları

5.3.1 Sistemin Güç Kaynağı Olan Batarya ve DC-DC Çeviriciler

Şekil 5.1'de görüldüğü gibi keşif robot, Li-Po batarya üzerinden 12 V - 5000 mA ile beslenmektedir. Raspberry Pi çalışabilmek için 5V 700 mA gücü ihtiyaç duymaktadır. Diğer çevre birimlerinin de daha fazla amper çekerceği düşünüldüğünden keşif robotunda şekil 5.3.2'de görülen 11.VDC 5000 mA Li-Po batarya kullanılmıştır.[36]

Bu uygulamada kullanılan batarya 12VDC olduğundan ötürü ilgili DC-DC çeviriciler ile 5VDC gerilime düşürülmesi gerekmektedir. Çünkü Raspberry Pi 5VDC ve üzeri gerilimlerde kullanılamaz hale gelmektedir. Bu sebeple Raspberry Pi ile güç kaynağını araya herhangi bir eleman koymadan bağlamak, olumsuz durumlarda Raspberry Pi'nin hasar görmesine neden olabilir. Bu yüzden araya regülatör görevi gören bir LM2596 DC-DC voltaj düşürücü bağlanmıştır. Bu sayede 5VDC-30VDC arasında 3 Amper gücü olan bir batarya 5VDC'ye dönüştürülmüştür. Bu sayede Raspberry Pi oluşabilecek kaçak akımlara karşı da korumaya alınmıştır.

Şekil 5.3.1'de görülen kartlı devre Raspberry Pi'ye regülatör görevi görmesi için batarya ve Raspberry Pi arasında kullanılan kart devresidir[37]. Bu DC-DC step-down düşürücü üzerinde anahtarlamalı bir gerilim regülatörü entegresi olan LM2596 entegresi bulunmaktadır. Giriş gerilimi 5VDC-30VDC arasındadır. Bu giriş gerilimlerini referans alarak bu kartlı devrenin üzerinde bulunan ayarlı kondansatörün vidası çevrilerek 5VDC-30VDC arasında çıkış gerilimi elde edilebilir. Kart üzerinde 4 tane pin vardır. Bunlar sırası ile “IN+,IN-,OUT+,OUT-” dir. Bu kartta IN+ yazan giriş güç kaynağının pozitif voltaj girişi ve IN- negatif güç kaynağının voltaj girişidir. OUT+ ve OUT- ise Raspberry Pi'ye giden güç kablosunun kutuplarıdır. Bu sayede Raspberry Pi çalışması için gereken tüm gücün karşılanması olur.



Şekil 5.3.1 LM2596 Step down düşürücü elektronik kart

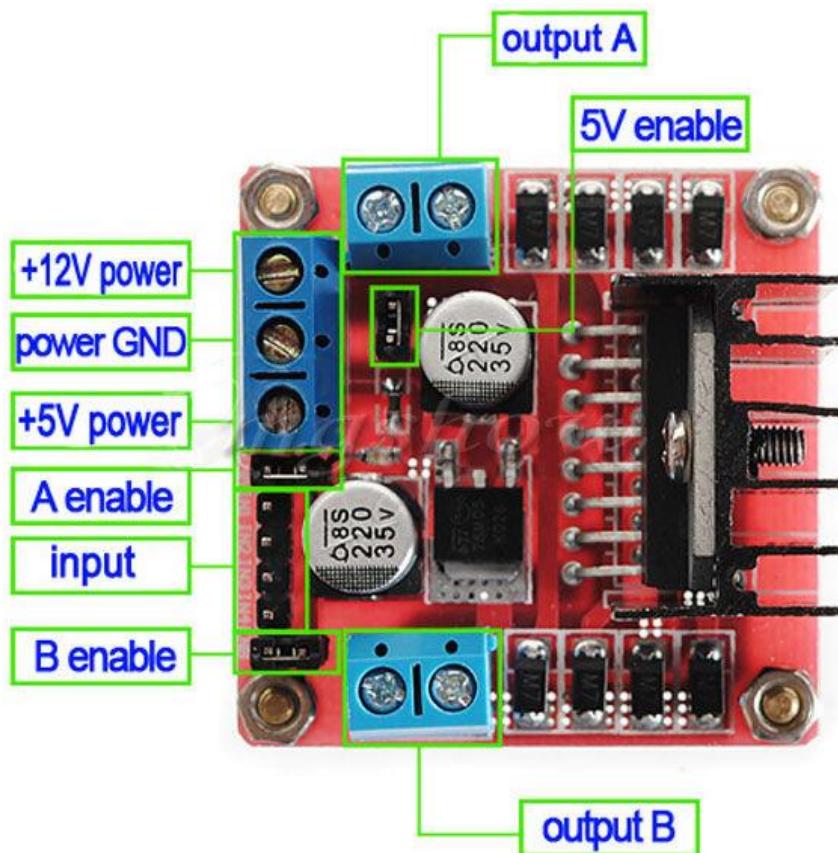


Şekil 5.3.2 11.1VDC 5000mAh 35C 3 Hücreli Li-Po pil

Burada bataryalarda güç tasarrufu sağlamak için şekilde görülen Raspberry Pi elemanlarının toprak uçları, N-kanallı bir mosfet transistore bağlanarak ve ayrıca bu mosfetin gate ucuna Raspberry Pi GPIO pinlerinden çekilebilecek bağlantılar vasıtası ile çalışması istenmeyen elemanların gücü kesilip, açılabilir. Bu sayede belirgin bir güç tasarrufu sağlanabilir.

5.3.2 Motor Sürücülerı ve Keşif Robotunda Bulunan Motorlar

Gezgin birim olan keşif robotuna hareketini sağlaması üzere şaseye entegre edilmiş iki adet 9V DC motor bulunmaktadır[38]. Bu uygulamada DC motorları sürebilmek için Şekil 5.3.4'te görülen L298N motor sürücü shieldi kullanılmıştır. Bu shield 6 pine sahiptir ve 2 kanallı motor sürücüsü olarak günümüzde kullanılmaktadır. Bu shield motorlara 5VDC ila 36VDC arasında kanal başına 2 Amper güç verebilmektedir. Bu shield içinde bulunan L298N entegresi içinde H- köprüsü yapısına sahiptir[39]. Bu sayede motorlara akan akımın yönü değiştirilerek yön ve hız kontrolü yapılmaktadır.

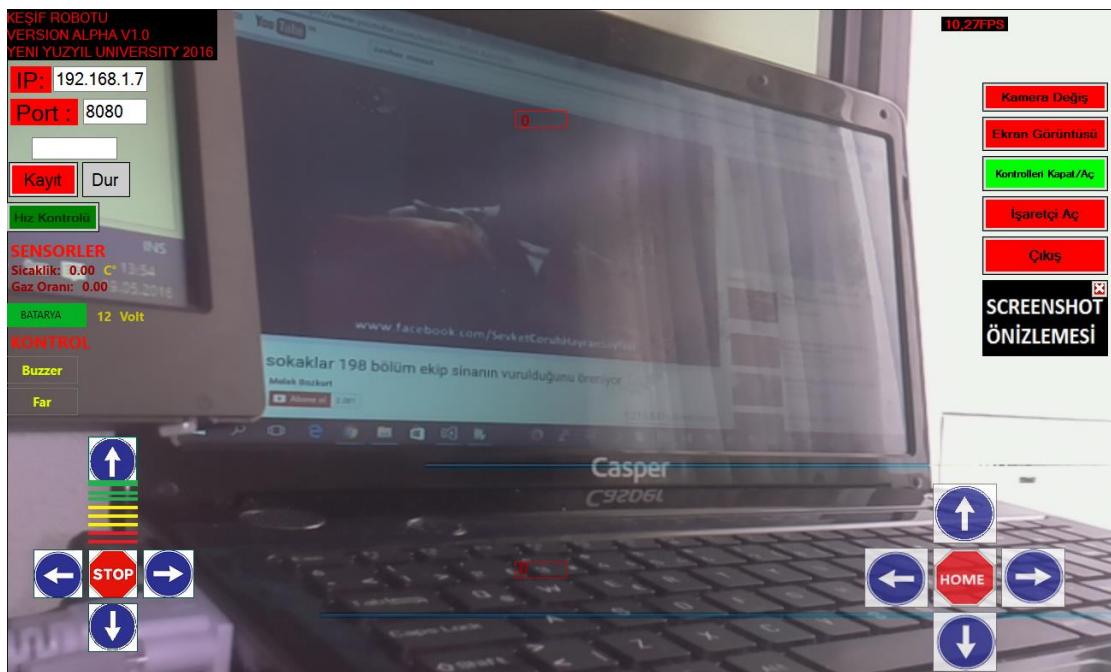


Şekil 5.3.3 L298N Motor sürücü shieldi

Keşif robottu standart 9VDC motorların oluşturduğu elektriksel gürültülerden ötürü kararlı çalışmamaktaydı. Bu yüzden şekil 5.3.5'te de görülebileceği üzere ekranada parazitler ve kablosuz iletişimde aksamalar meydana gelmiştir. Bu sorunu aşabilmek için Şekil 5.3.5'te görülen R260 9V DC titreşim motoru kullanılmıştır. Bu sayede motorlardan kaynaklanan elektriksel gürültü en az seviyeye indirilmiş ve motor torku arttırlılmıştır.



Şekil 5.3.4 R260 6V DC Motorlar



Şekil 5.3.5 Kontrolör arayüzde parazitlenmeler

Ayrıca keşif robotunda kameraların bakış açılarını değiştirebilmek için servo motor kontrollü bir pan-tilt mekanizması kullanılmıştır. Bu sayede kameraların bakış açıları yukarı-aşağı ve sağ-sol olmak üzere 180 derece değiştirilmektedir. Bu motorların kontrolü kontrol birimi olan tabletin arayüzünden alınan komutlara göre Raspberry Pi'nin üreteceği PWM sinyaller ile gerçekleşmektedir.

Bu servo motorlardan bahsedersek bu uygulamada kullanılan servo motor metal dişliye sahip olan Şekil 5.3.7'de görülen sg90 servo motorlardır[40]. Bu servo motorlar 180 derece dönüş yeteneğine sahiptirler.



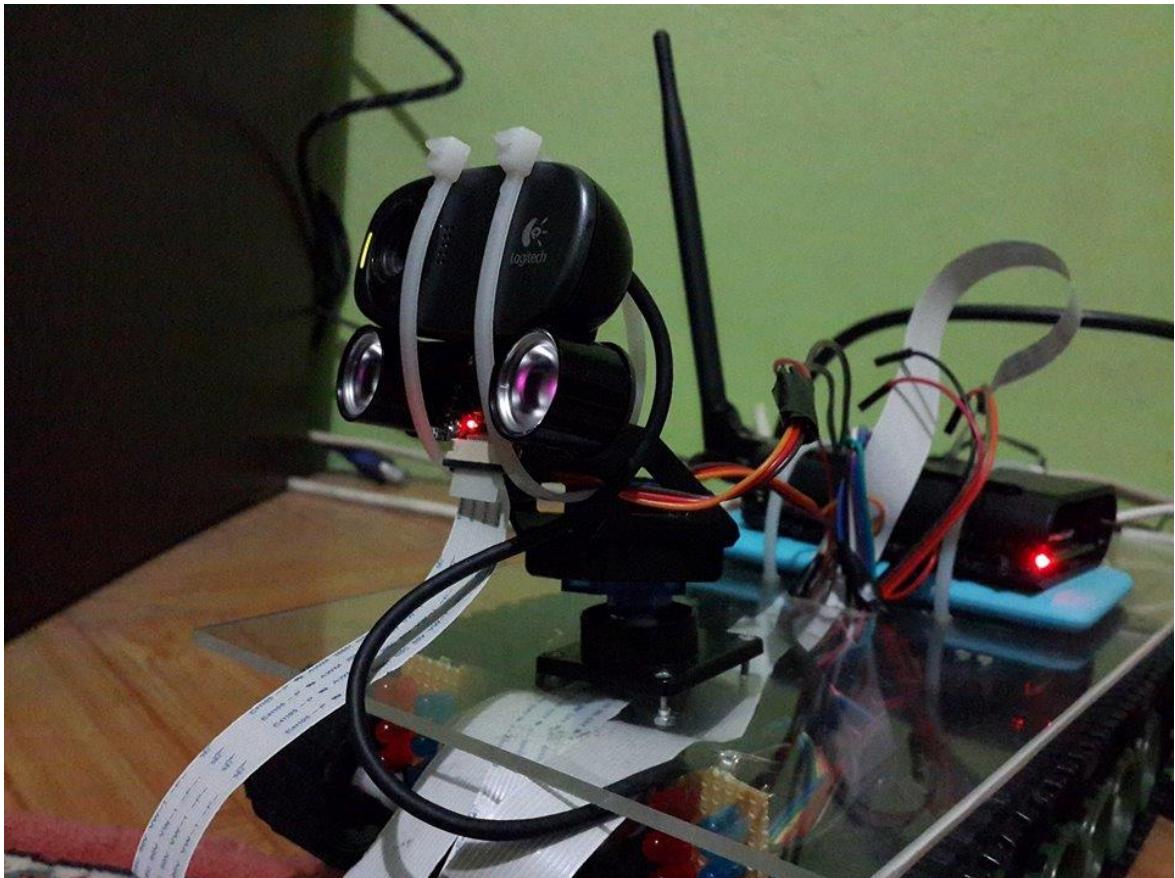
Şekil 5.3.6 Pan-Tilt Mekanizmasında kullanılan SG90 servo motor

Bu motorların kafa kısmı aşağıdaki Şekil 5.3.8'de verilen pan-tilt aparatına kameralarla birlikte monte edilmiş ve oluşturulan sistemlerle birlikte kameraların bakış açılarının yukarı-aşağı , sağa-sola 180 derece değiştirilmesi sağlanmıştır[41].



Şekil 5.3.7 Pan-Tilt Mekanizmasında kullanılan pan-tilt

Son oluşturulan sistemde bütün birleştirilen pan-tilt mekanizmasının son hali bir alttaki Şekil 5.3.8'da görülebilir.

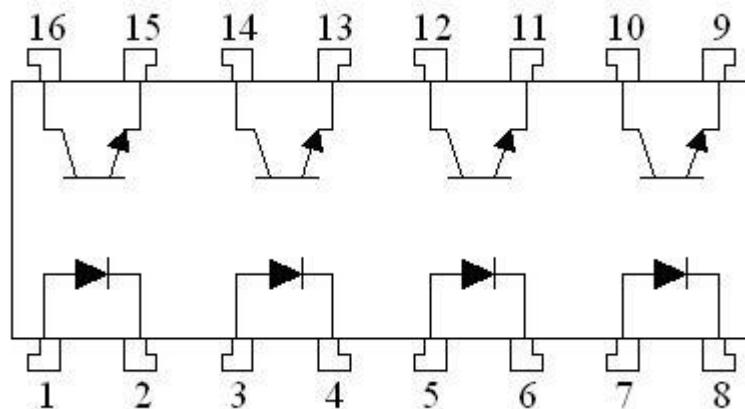


Şekil 5.3.8 Pan-Tilt Mekanizmasının keşif robotu üzerindeki son hali

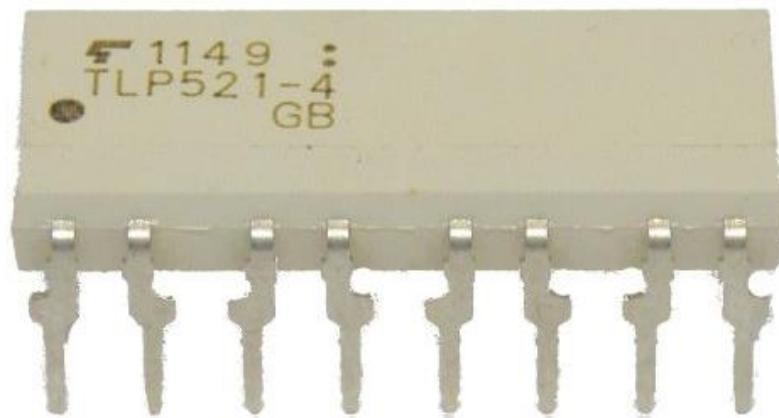
5.3.3 Optokuplör Devreleri

Raspberry Pi ile motor sürücü devresini kontrol edebilmek için direk pin bağlantısı yerine araya Şekil 5.1'de görüldüğü gibi optokuplörlü bir devre tasarlanarak motor sürücü devresi izole edilmiştir. Bunun nedeni ise L298N sürücüsünde meydana gelebilecek bir arızanın veya kısa devrenin Raspberry Pi'ye zarar vermesine engel olmaktr. Bunun için Şekil 5.3.10 ve Şekil 5.3.11'de görülen 2 adet TLP521-4 optokuplör entegresi kullanılmıştır. Ayrıca, bu optokuplörlor sayesinde Raspberry Pi, motor frekansından dolayı oluşabilecek gürültülerden ve gerilim dalgalanmalarından etkilenmeyecektir.

Bu optokuplörlor sayesinde sistemler arasında haberleşme Şekil 5.9'da da görülebileceği üzere kıızılötesi ledler üzerinden sağlanmaktadır ve sistemlerin ortak toprakları birbirinden ayrılarak yani izole edilerek bir tarafta oluşan kısa devrenin diğer haberleştiği devreye zarar vermesini önlemektir. Bu sayede L298N motor sürücü entegresinde meydana gelen gerilimler değişimlerinin ve kısa devrelerin Raspberry Pi'ye zarar vermesi engellenmiştir.



Şekil 5.3.9 TLP521-4 Entegresinin iç yapısı

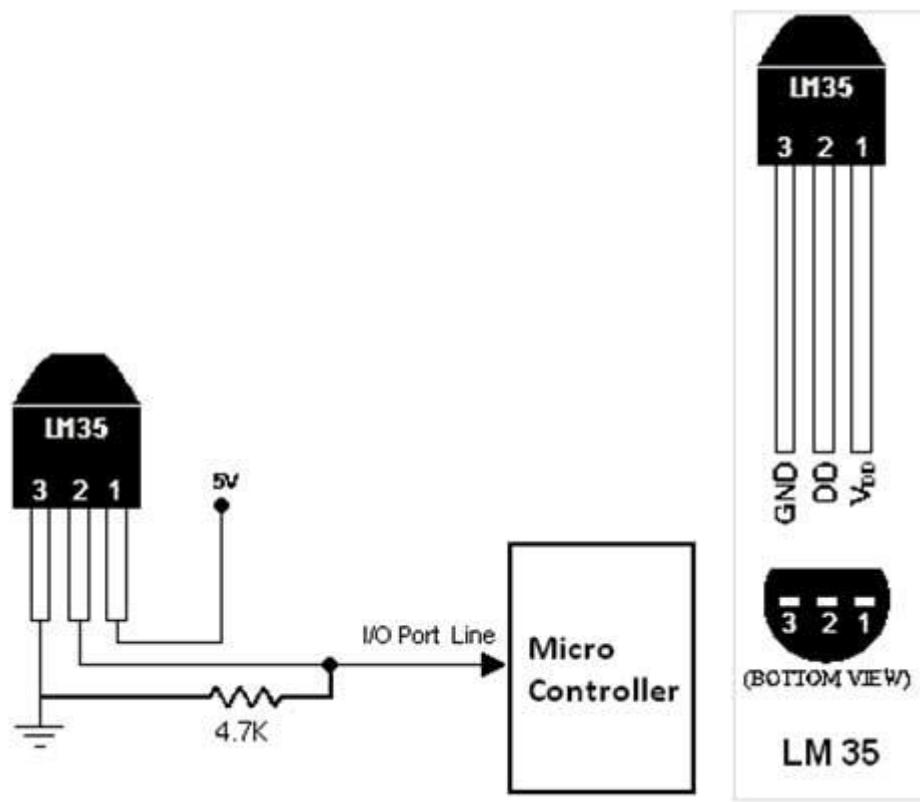


Şekil 5.3.10 TLP521-4 entegresinin dıştan görüntüsü

5.3.4 Sensörler

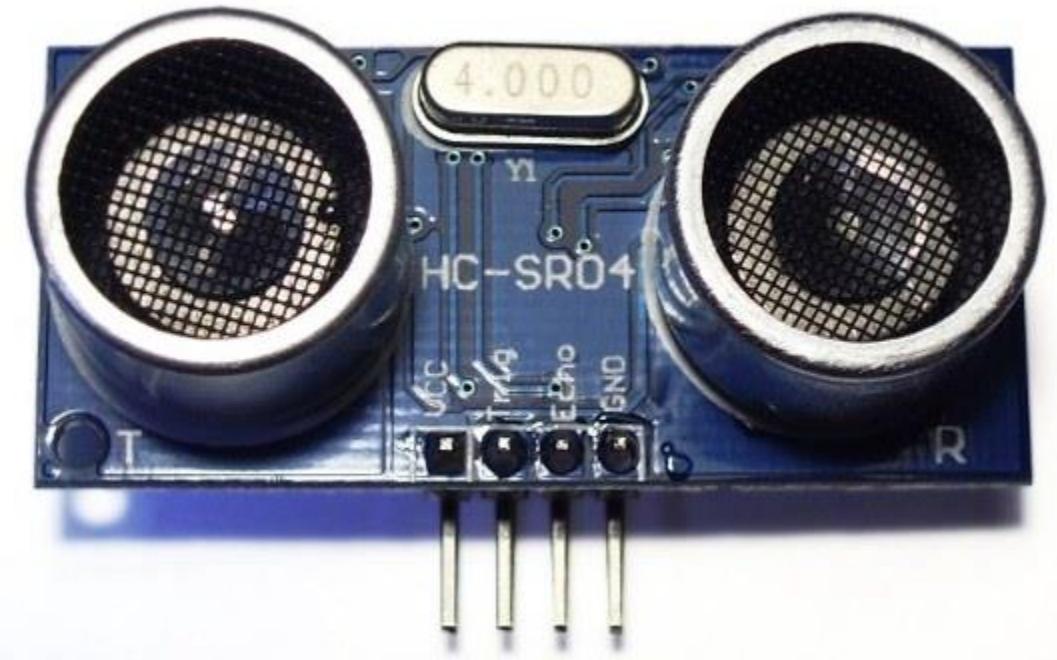
Keşif robotuna 4 tane sensör takılmıştır. Bu sensörler ortam sıcaklığını, ön veya arkadaki engelin mesafesini hesaplayan ve ortamdaki karbonmonoksit (CO) gazının saturasyonunu ölçen sensörlerdir. Bu sensörler sayesinde keşif robotunun gezdiği yerlerdeki ortam hakkında bilgi sahibi olunmaktadır. Keşif robotundan bu sensörleri Arduino mikrodenetleyicisi okuyup bu veriler UART üzerinden Raspberry Pi'ye göndermektedir.

Keşif robotunda ısıyı algılamak için aşağıdaki Şekil 5.3.12 de görülen LM35 ısı sensörü kullanılmıştır[42]. LM35 -150 ile +150 arasında ısı ölçen bir sensördür. 3 tane bacağı bulunmaktadır. 3VDC veya 30VDC ile beslenebilmektedir. Bu sensör her 1 Santigrad derecelik sıcaklık yükselmesinde sinyal bacağından 10mV'luk doğrusal bir artış gösterir. Arduino'nun ADC çevrimi sayesinde okunan gerilimin 10.0 bölünmesi ile sıcaklık değeri bulunmuş olur.



Şekil 5.3.11 LM35 sıcaklık sensörü

Keşif robotunun önündeki ve arkasında 2 adet Şekil 5.3.13'de görülen HC-SR04 ultrasonik mesafe sensörleri kullanılmıştır.[43] Bu sensörler sayesinde keşif robotunun önündeki veya arkasındaki engeller arasındaki mesafe kontrolör arayüzde gösterilmiştir. Bu sensör 4 tane bacağa sahiptir. Bu pinler sırası ile gnd, trig, echo, vcc pinleridir. Bu sensörlerdeki trig bacağını $10\mu\text{s}$ de $+5\text{V}$ gerilim uygulanır ve böylelikle 40 kHz 'de 8 devir ses dalgası oluşturulup dışarıya iletilir ve yansımıması beklenir. Yansıma algılandığı an Echo pini $+5\text{V}$ gerilime yükselir ve böylece formüller yardımı ile mesafe elde edilir.



Şekil 5.3.12 HCSR04 ultrasonik mesafe sensörü

Keşif robotunun gezdiği yerlerdeki karbonmonoksit gazının saturasyonunu ölçmek için Şekil 5.3.13'te ki MQ-7 gaz sensörü kullanılmıştır[44]. Bu sensör 3 adet çıkışa sahiptir. Orta bacağı olan sinyal bacağından ADC çevrim yapılarak gerekli formüller kullanılarak bu gazın saturasyonu elde edilmiştir.



Şekil 5.3.13 MQ-7 CO sensörü

Bu sensörlerin Arduino mikrodenetleyicisinde nasıl kodlanacağı ilgili eklerde verilmiştir.

5.3.5 Wifi Ağ Adaptörleri

Keşif robotu ile kontrolör tabletin kablosuz iletişim kurması ve kablosuz iletişim menzilini artırmak için hem kontrolör tablet hemde keşif robotunda aşağıdaki Şekil 5.3.14'te görülen IEEE 802.11b/g/n Wi-Fi Ağ adaptörleri kullanılmıştır[45]. Bu ağ adaptörleri sayesinde keşif robotu üzerinde kablosuz ağ kurulmuştur. Bu kablosuz ağ üzerinden kontrolör tablet, keşif robotu ile haberleşme kurmaktadır ve menzili açık alanda 1 kilometreye kadar çıkmaktadır.



Şekil 5.3.14 LB-link RTL8188Cus chipsetli Wi-Fi Ağ adaptörü

Bu Wi-Fi adaptörünün hem Raspberry Pi hemde Windows üzerinde otomatik olarak tanınabilmesi için Realtek RTL8188CUS chipseti olan modeli tercih edilmiştir.[45] Bu Wi-Fi adaptörünün fiyatı sadece 10\$ olup fiyat/performans oranı çok yüksektir[46].

5.3.5.1 Raspberry Pi Üzerinde Kablosuz Ağ Kurulumu

Keşif robottu üzerinde kablosuz bir ağın kurulumu Raspberry Pi'ye hostapd ve DHCP server kurularak gerçekleştirilmiştir. Bu işlemleri sırası ile vermek istersek öncelikle aşağıdaki paketlerin kurulması gerekmektedir.

```
sudo apt-get install bridge-utils hostapd
```

Ardından RTL8188CUS uyumlu hostapd driveri indirilir.

```
wget http://dl.dropbox.com/u/1663660/hostapd/hostapd.zip
```

İndirilen driverin, ilk olarak yüklenen driver ile değiştirilmesi gerekmektedir.

```
unzip hostapd.zip && \
sudo mv /usr/sbin/hostapd /usr/sbin/hostapd.original && \
sudo mv hostapd /usr/sbin/hostapd.edimax && \
sudo ln -sf /usr/sbin/hostapd.edimax /usr/sbin/hostapd && \
sudo chown root.root /usr/sbin/hostapd && \
sudo chmod 755 /usr/sbin/hostapd
```

Sonra ile /etc/network/interfaces ağ yapılandırması dosyasında değişiklik yapılmıştır.

```
auto lo

iface lo inet loopback

iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet manual
#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
#iface default inet dhcp

auto br0
iface br0 inet dhcp

bridge_ports eth0 wlan0
```

Bu aşamadan sonra kablosuz ağın ayarları yapılmıştır. Bu ayarlar yeni bir dosya oluşturularak yapılmıştır. Bu keşif robottu uygulamasında kanal ayarı, SSID ismi ve ağ şifresi gibi ayarlamalar yapılmıştır. Bu yüzden /etc/hostapd/hostapd.conf dosyası sudo nano komutu yardımı ile oluşturulmuştur.

```
interface=wlan0
driver=rtl871xdrv
bridge=br0
ssid=KESIFTANKBAGLANTINOKTASI
channel=1
wmm_enabled=0
wpa=1
wpa_passphrase=1234567890
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
auth_algs=1
macaddr_acl=0
```

Bu ayarlamalar yapıldıktan sonra bu ayarların çalışıp çalışmadığı bu komutlar ile test edilebilir. Eğer hata yoksa kablosuz ağ yayın yapmaya başlayacaktır.

```
sudo hostapd -dd /etc/hostapd/hostapd.conf
```

Bu aşamadan sonra artık Wi-Fi ağ adaptörü yukarıda oluşturulan dosyadaki bilgilerle yayın yapmaya başlayacaktır. Bu ayarların her açılışta çalışması için daemon servisine kayıt edilmesi gerekmektedir. Aşağıdaki komut sayesinde bu kablosuz ağ her sistem açılışında çalışabilir hale getirilir.

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

5.3.6 Raspberry Pi Mini Bilgisayarı

Raspberry Pi keşif robotunun ana kontrolörü olarak tasarlanmıştır. Tüm sistemi ayakta tutan ve tablet ile haberleşen kısımdır. Raspberry Pi, HD kamerasdan, NoIR gece görüş kamerasından ve Arduino'dan aldığı verileri bir sunucu gibi çalışarak kontrolör tablete Wi-Fi kablosuz haberleşme teknolojisi ile aktarmaktadır. Kontrolör tabletten gelen komutlara göre gerekli elemanları kontrol eder ve tanka hareketi sağlayacak PWM sinyalleri oluşturur. DC motorlar ve servo motorlar için gerekli PWM sinyaller sırası ile WiringPi ve ServoBlaster kütüphaneleri üzerinden sağlanmıştır. Tüm motorlar için

toplam 4 tane Raspberry Pi GPIO'su PWM çıkış olarak kullanılmıştır. Raspberry Pi üzerinde donanımsal olarak 2 adet GPIO pini mevcuttur. Bunlar motorun hareketi için yeterli olmamaktadır. Bu yüzden yazılımsal olarak PWM işaret üreten kütüphanelere ihtiyaç duyulmuştur. Bu sayede kullanılmayan pinler yazılımsal olarak PWM işaret üretebilmektedir. Bu üretilen PWM sinyaller Raspberry Pi CPU'sunun %0.5 ila %1 arasında kaynağını kullanmaktadır ve oldukça performanslı çalışmaktadır.

Servoblaster kütüphanesi servo motorları sürebilmek için 4 ve 7 numaralı Raspberry Pi GPIO pinine PWM üretmesi için ayarlanmıştır ve ilgili programlama C++ dili kullanılarak yapılmıştır.

WiringPi kütüphanesi DC motorları kontrol edebilmek için 29 ve 27 nolu Raspberry Pi pinine PWM üretmesi için ayarlanmıştır. 28, 29, 25, 24 Nolu WiringPi GPIO pinleri ise motora akan akımın yönünü ters çevirmek üzere programlanmıştır. Bu sayede motorların dönüş yönleri ayarlanabilmektedir.

Bu motorlarda ve motor sürücü kartlarındaki herhangi bir arızanın Raspberry Pi'ye zarar vermesini önlemek için Optokuplör devreleri tasarlanmıştır. Bu sayede motor sürücü ile Raspberry Pi birbirinden izole edilmiştir.

Raspberry Pi'ye güç verilmesinden itibaren Raspberry Pi Boot aşamasına geçer ve sırası ile kendi sisteminin çalışması için gereken dosyaları önyükler. Ardından sırası ile keşif robottu için yazılan sunucu programlar ve kamera görüntülerini aktaran programlar çalışırlar.

Raspberry Pi içerisinde kurulan C++ tabanlı sunucu programları kontrol birimi olan tabletten sürekli bir şekilde veri beklemektedir. Kontrolör üzerinden gelen verilere göre keşif robottu hareket eder ve istenilen bilgileri tablete geri gönderir. Örneğin motor hareketi için kontrolör üzerinden soket haberleşme için tabletten 2 tane veri alır ve bu veriler [1.Sol motorun PWM frekansı][2.Sağ motorun PWM frekansı] şeklindedir. Bu alınan veriye göre keşif robottu istenilen yönde ve hızda tablet üzerinden kontrol edilebilmektedir.

Raspberry Pi, üzerinde bulunan gece ve gündüz kamerasını çalıştırmak ve bu görüntüleri tablete aktarmak için öncelikle bu kameralardan alınan fotoğraf karelerini gerekli formata çevirmelidir. Çünkü kameralardan alınan görüntü ile tablete yollanan görüntü arasında bir zaman farklı bulunmaktadır. Bu zaman farkının azaltılması ve Raspberry CPU'sunun fazla zorlanmamışı için uygun bir format kullanılması gerekmektedir. Bu bitirme tezinde MJPEG formatı kullanılmıştır. Bu sayede alınan veriler gerçek zamana yakın olmakta ve Raspberry CPU'suna fazla bir yük bindirilmemiştir. Bu bitirme tezinde 2 kamera için en yüksek çözünürlüklerde en fazla %25 CPU kullanımı tespit edilmiştir.

Kameralardan alınan veriler Raspberry tarafından saklanılmaz ve bu sayede sistem yorulmaz. Kameralardan alınan 30 FPS'ye yakın veriler Wi-Fi üzerinden UDP portları üzerinden kontrolör tablete sürekli bir biçimde gerçek zamana yakın bir şekilde aktarılmaktadır.

Kameraların verdiği görüntünün formatı Raspberry CPU'sunun kaynak kullanımından önemli rol oynamaktadır. Bu bitirme tezinde kullanılan kameralar USB üzerinden bağlanılan Logitech HD Webcam C310 ve CSI kamera soketi üzerinden bağlanır Raspberry NoIR kamerasıdır[47][48]. Şekil 5.3.15'te Raspberry Pi üzerine takılı USB aygıtlarının listesi görülebilir.

```
root@raspberrypi:~# lsusb
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 148f:7601 Ralink Technology, Corp.
Bus 001 Device 005: ID 046d:081b Logitech, Inc. Webcam C310
root@raspberrypi:~#
```

Şekil 5.3.15 lsusb komutu ile Raspberry Pi'ye bağlı cihazların görünümü

Logitech firmasının ürettiği Şekil 5.3.16'da görülen Logitech C310 kamerası USB2.0 arayüzüne sahiptir ve Linux UVC driverleri tarafından tanınılmaktadır.[49] Tak çalıştır özelliği ve 30FPS kare hızında 1280*720 piksel görüntü yakalama özelliğine sahiptir. Bu ürünün görüntü yakalama özellikleri aşağıdaki Tablo 5.1 de verilmektedir. Bu bilgiler [49] numaralı referanstaki kaynaktan özetlenerek yazılmıştır.



Şekil 5.3.16 Logitech C310 5 MP 720p HD webcam

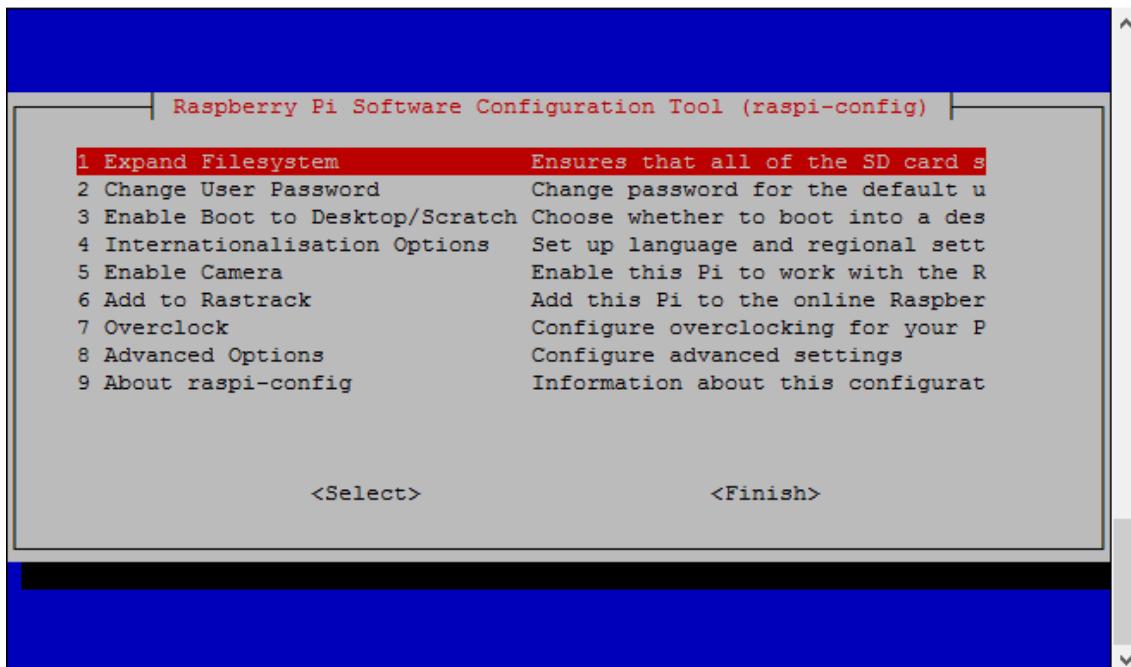
Tablo 5.1 Logitech C310 birkaç kamera parametreleri

Görüntü Yakalama (4:3 HD) modunda	Maximum 1280*720, 5MP, 3MP, 1.5MP
Görüntü Yakalama (4:3 SD) modunda	640*480, 360P
Film karesi oranı	Maksimum 1280*720p 30FPS

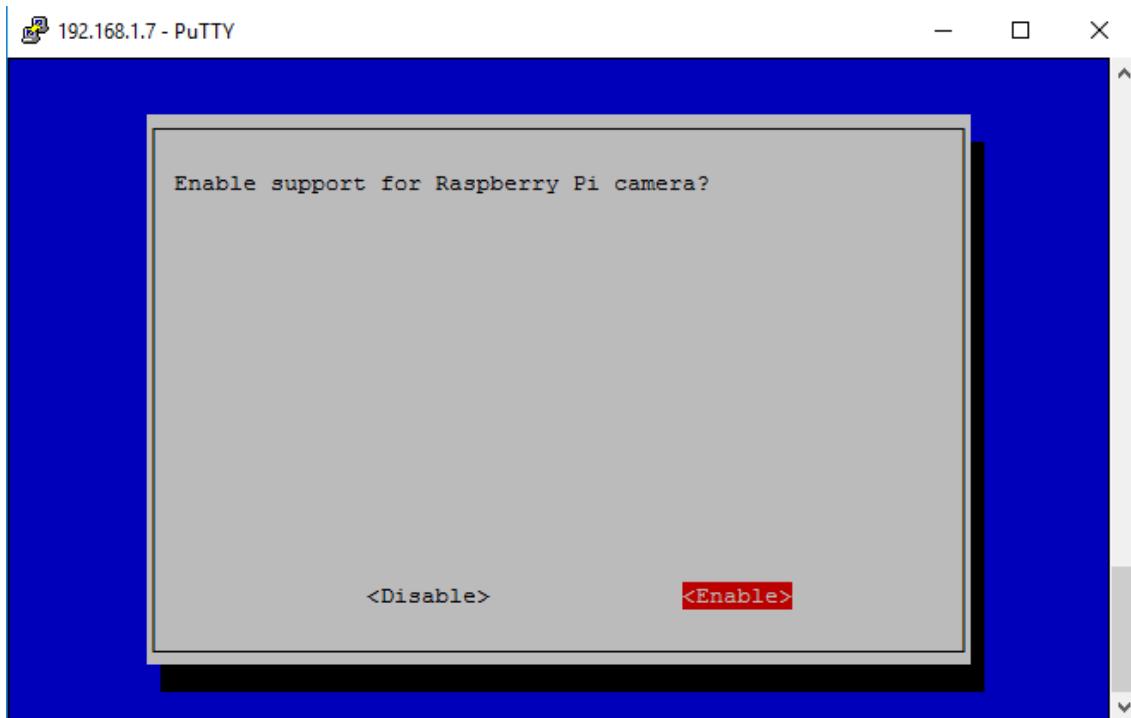
Gece görüş kamerası olarak Şekilde 5.3.17 de görülen Raspberry Pi NoIR kamerası kullanılmıştır. Bu kamera Raspbian işletim sistemi tarafından otomatik olarak tanınan bir kamera modülüdür. 1920*1080 piksele kadar görüntü alabilmektedir ve gece görüşü sağlayabilmek için yanına kızılötesi ledler takılmıştır.



Şekil 5.3.17 Raspberry NoIR kamerası



Şekil 5.3.18 sudo raspi-config komutunun ekran görüntüsü



Şekil 5.3.19 sudo raspi-config komutundan dsi kameranın aktif edilmesi

Bu NoIR kamerasına ait bilgiler [50] referanstaki kaynaktan özetlenerek yazılmıştır. Tablo 5.2'de bu özellikler görülebilir.

Tablo 5.2 Raspberry Pi birkaç kamera parametreleri

Görüntü Yakalama (4:3 HD) modunda	Maximum 1920*1080, 5MP, 3MP, 1.5MP
Görüntü Yakalama (4:3 SD) modunda	640*480, 360P
Film karesi oranı	Maksimum 1920*1080p 30FPS

Bu kameralar, YUV ve MJPEG formatlarını desteklemektedirler. Eğer görüntü alımı için YUV formatı seçilecek olursa bu alınan YUV formatının Raspberry Pi'de öncelikle depolanıp JPEG formatına çevrilerek alıcıya gönderilmesi gerekmektedir. Bu da Raspberry Pi CPU'suna yüzde %99 luk bir yük bindirir ve en fazla 15 FPS kare hızında veri gönderebilmektedir. Bu sistemin yavaşlamasına sebep olmaktadır. Bu nedenle bu bitirme tezinde MJPEG formatı kullanılmıştır. Bu format ile kamera ilk önce aldığı görüntüyü kendi üzerinde MJPEG formatına çevirir ve ardından Raspberry Pi'ye bu verir. Raspberry Pi'de bu formattaki görüntüyü anında UDP port üzerinden yollar ve üzerinde bir kodlanma gerçekleşmeyeceği için CPU'ya yük binmez. Bu sebeble MJPEG formatında alınan görüntü Raspberry Sistemini iyileştirir.

Raspberry Pi üzerinde çalışan Linux tabanlı işletim sisteminin MJPEG formatını desteklemesi için ilgili “MJPG-Streamer” kütüphanesi Raspberry Pi üzerine kurulmuştur. Linux işletim sistemi tarafından desteklenen kütüphaneler sorunsuzca kurulabilmektedir veya çıkarılabilmektedir.

MJPG, kısaca “MOTION JPEG” – “Motion Joint Photographic Experts Group-Hareketli Birleşik Fotoğraf Uzmanları Grubu” demektir. MJPEG alınan seri JPEG dosyalarının birbiri ardından gelecek şekilde akmasıdır. MJPEG her görüntü karesi, kameralardan gelen JPEG formatından kodlanmaktadır. MJPEG'in kullanım alanı çok genişir. Medya oynatıcılar, IP kameralar, Web kameraları, oyun konsolları ve medya oynatıcılarında sıkılıkla kullanılmıştır[51].

MJPEG kütüphanesi kameralardan alınan MJPEG formatındaki verileri alır ve IP tabanlı cihazlar tarafından TCPMP() oynatıcısını içeren hedef sistemde çalıştırır. Bu kütüphane donanımsal kapasitesi sınırlı olan cihazlar için kodlanmıştır.

“MJPG-Streamer” kütüphanesi Raspberry Pi'ye kurmak ilk önce github deposundan indirilmiştir[52]. Ardından ise bu kütüphanenin gereksinim duyduğu bazı paketler Raspberry Pi 'ye indirilmiştir. Bu paketler aşağıdaki komutlar aracılığı ile kurulmuştur.

```
$ sudo apt-get install libjpeg8-dev imagemagick libv4l-dev  
$ wget http://sourceforge.net/code-snapshots/svn/m/mj/mjpg-streamer/code/mjpg-  
streamer-code-182.zip  
$ unzip mjjpg-streamer-code-182.zip
```

Bu indirilen paketler mjpg streamer kütüphanesinin gerek duyduğu kütüphaneleri ve kaynak kodun derlenmesi için gerek duyulup yüklenmiştir. Bu aşamadan sonra mjpg-streamer kütüphanesi derlenmeye hazırır. Derleme işlemine başlamak için aşağıdaki komutlar kullanılır.

```
$ cd mjjpg-streamer-code-182/mjpg-streamer  
$ make mjpg_streamer input_file.so output_http.so  
$ sudo cp mjpg_streamer /usr/local/bin  
$ sudo cp output_http.so input_file.so /usr/local/lib/  
$ sudo cp -R www /usr/local/www
```

Bu aşamadan sonra test amaçlı olarak Raspberry Pi terminaline HD kamerayı ilgili porttan stream etmek için aşağıdaki komut yazılabilir.

```
$ sudo /usr/local/bin/mjpg_streamer -i "/usr/local/lib/input_uvc.so -d /dev/video0 -f 30  
-r 1280*720" -o "/usr/local/lib/output_http.so -p 8080 -w /usr/local/www" &
```

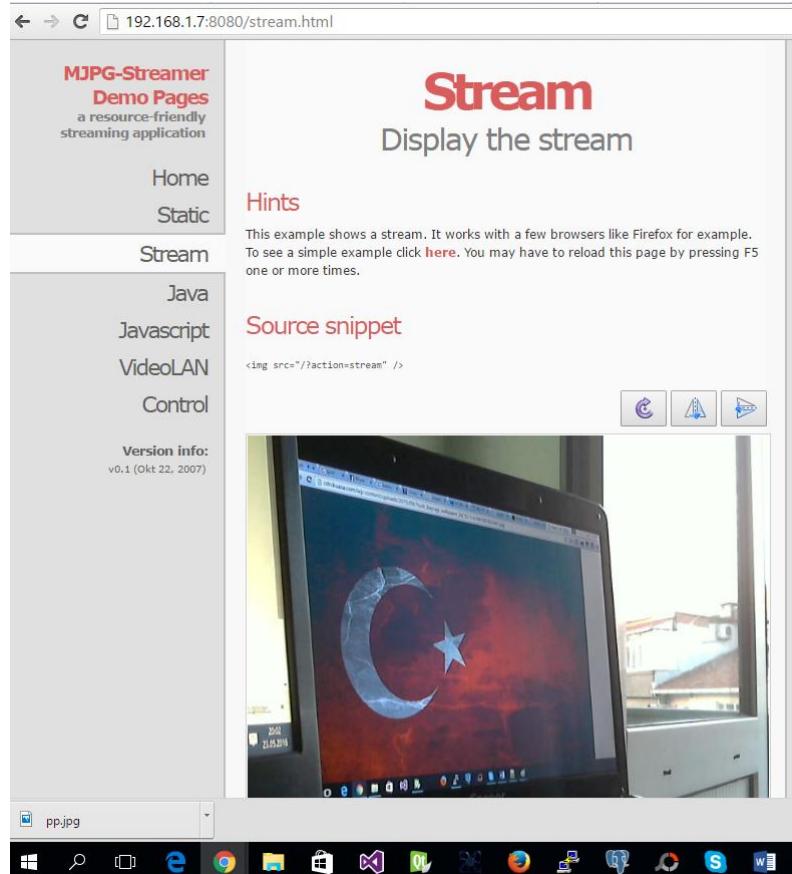
Üstte görülen kodla birlikte eğer hata yoksa webcam 30 FPS kare hızında 1280*720P HD yayın yapmaktadır ve bu yayın 8080 portundan www klasöründeki dizine stream edilir.

Bu aşamadan sonra Raspberry Pi'nin bağlı olduğu ağdaki herhangi bir bilgisayarda web tarayıcısına aşağıdaki adres girilerek görüntü elde edilebilir.

```
http://192.168.1.7:8080/stream.html
```

Burda adres olarak girilen "192.168.1.7" Raspberry Pi'nin sahip olduğu IP adresidir. Bu IP adres üzerinden kamera görüntüler adeta bir sunucu gibi yayın yaplığınımaktadır.

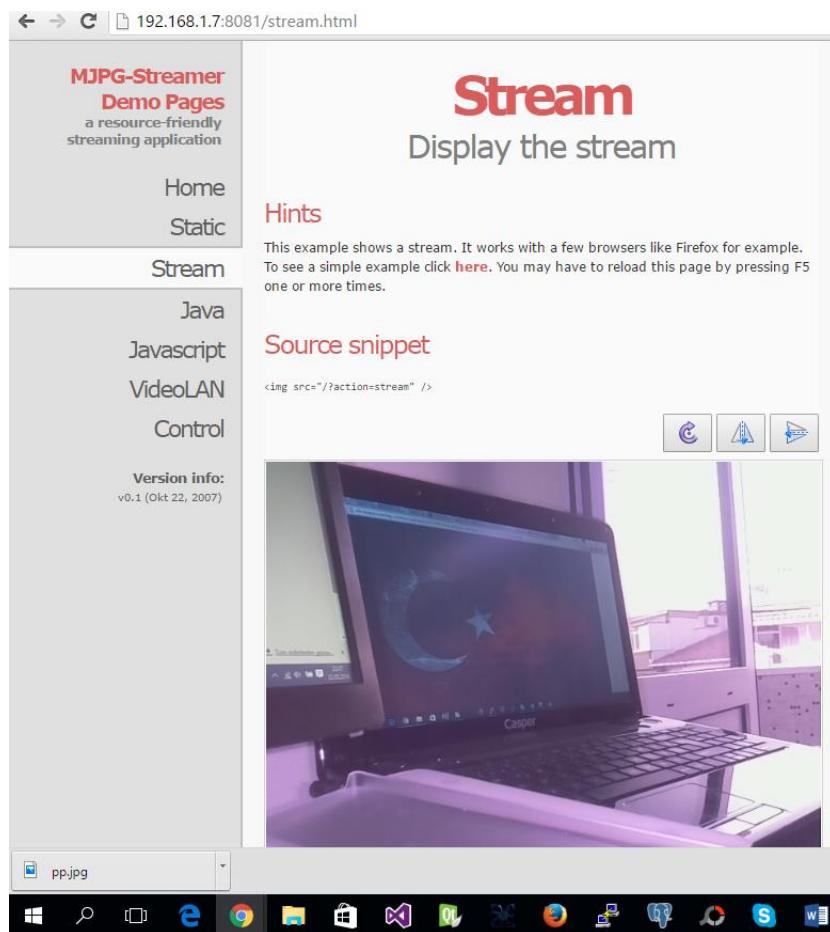
Şekil 5.3.10'da HD kamera için web yayını görüntülebilir



Şekil 5.3.20 HD kamera web yayınının aynı ağdaki başka bilgisayarda görüntülenmesi
Aynı şekilde NoIR kameralı porttan stream etmek için aşağıdaki komut kullanılır.

```
$ sudo /usr/local/bin/mjpg_streamer -i "/usr/local/lib/input_raspicam.so -fps 30" -o "/usr/local/lib/output_http.so -p 8081 -w /usr/local/www" &
```

Bir üstte görülen kodla birlikte NoIR Kamera 30 FPS kare hızında 800*600 çözünürlükte yayın yapmaktadır ve bu yayın 8081 portundan www klasöründeki dizine stream edilir.



Şekil 5.3.21 NoIR kamera web yayınının aynı ağdaki başka bilgisayarda görüntülenmesi

Bu komutlar ile Raspberry Pi, mjpeg akışını komutlarda belirtilen portlar olan 8090 ve 8091 portlarından stream etmeye başlar. Bu komutlar sayesinde HD kamera 1280*720p çözünürlük 30FPS kare hızında, NoIR kamera ise 800*600 çözünürlük 30FPS kare hızında yapılandırılmaktadır. Burada kullanılan “-d:ilgili webcam aygıtının adı”, “-r: çözünürlük oranı” ,”-f: film karesi oranı(framerate)” anlamlarına gelen ön tanımlayıcılardır. Bu komutlar ile ilgili kamera parametreleri ayarlanılır.

Aynı komutlarda bulunan “-p: port numarası”, ve “-w: web sağlayıcı dizinidir”. Bu bitirme tezinde 8090 ve 8091 portları çıkış olarak kullanılmıştır.

Raspberry Pi’de çalışan kameralar genel olarak 16.9 formatında görüntü vermektedir. Bu kameralardan alınan en büyük görüntü boyutu HD Kamera için 1280*720p ve NoIR kamera için 1920*1080p’dır. Raspberry Pi GPU’su nun fazla yorulmaması adına ideal çözünürlük değerleri bu kameralar için seçilmiştir.

Raspberry Pi içinde gömülü Linux çalıştırın bir mini bilgisayardır ve dış dünya ile bağlantı kurabilmek için Ethernet veya USB WLAN adaptörler kullanmaktadır. Raspberry Pi kablosuz ağ desteği ile gelmektedir. Bu sistem uzaktan kontrol edilebilmesi için Raspberry Pi USB portuna bir adet Realtek RTL8188CUS WiFi ağ adaptörü takılmıştır. Raspberry Pi bu ağ adaptörünü kendi kernelinde ilgili driveri mevcut olduğu için hiçbir işlem yapmaksızın tanıtmaktadır.

Keşif robotunda bulunan gömülü sistem kontrolör tabletten kablosuz olarak TCP/IP protokolü üzerinden gelen verilere göre istenen görevleri yerine getirmektedir. Kontrolör tablet pc'nin arayüzünde bulunan kamera pan-tilt sistemini kontrol eden formlar ve keşif robotunun hareket etmesi için gerekli tüm butonları içeren formlar bir TCP/IP iletişim protokolü içermektedir. Yani bu kontrol formları dediğimiz programlar arayüz açıldığı anda bir hedef sistem olan keşif robotunun sistemi ile TCP bağlantısı kurar.

Bu keşif robotunu kontrol etmek için gerekli bütün programlar ayrı ayrı oluşturulmuştur. Bunun nedeni ise aynı anda birçok TCP /IP bağlantısının keşif robotu Wi-Fi menzilinden uzaklaşıkça veri aktarımının zayıflaması ve de bu zayıflamanın zaten hali hazırda bulunan video aktarım protokolünü yavaşlatmasıdır. Bu yavaşlatma keşif robotundan gelen kamera görüntülerinin kare hızlarında (FPS) düşüş yaratmasından kaynaklanmaktadır. Keşif robotunun DC motorların oluşturduğu elektriksel gürültü nedeni ve hem keşif robotundan hem kontrolör tablette bulunan Wi-Fi adaptörünün menzilinin dışına çıkılması durumunda veri bozuk veya tam olarak aktarılamamaktadır. Bu sebeple TCP/IP protokolü yollanmak istenen veri karşı tarafta hatasız alındı mesajını bekleyene kadar bu gönderilen veriyi sürekli hedef sisteme yollar. Bu nedenle bütün programlar aynı arayüzde kullanılsa idi keşif robotu arayüzü giden veriyi hatasız alınana kadar sürekli yollayacak idi bu da ilgili programın bir sonraki kod bloğuna geçmesini önleyerek sistemde bir gecikme oluşturacaktır.

Bu yukarıdaki olayların önlenmesi için keşif robotu için gerekli tüm yazılımlar farklı farklı programlarla gibi derlenip oluşturulmuş ve yine bu programların arayüzde ilgili yerlerde gösterilmesi için C# programlama dilinde program başlangıç komutları kullanılmıştır.

Keşif robotunun kontrolöründeki arayüzü güzel bir görünümü sahip olması için Windows Form ve Windows Presentation Form gibi zengin içerikli formlar kullanılmıştır. Bu formlara transparant özelliği kazandırılarak saydam yapılmış ve ilgili komutlarla bu programların çalışacağı yerler tanımlanmıştır. Bu sayede video görüntüsünün üzerine bu formlar yerleştirilmiş ve güzel bir görünüm elde edilmiştir.

Keşif robotunda gömülü sistemde bulunan Raspberry Pi mini bilgisayarı üzerinde bu robotun uzaktan kontrol edilebilmesi için TCP/IP protokolünü kullanan threat tabanlı programlar yazılmıştır. Bu programlar sürekli arkaplanda TCP üzerinden gelen veriyi okumaktadır.

Bu programlardan ilki kontrolör tablet üzerinden gelen verileri okuyan ve DC motorların çalıştırılması için gerekli PWM sinyali motor sürücü devresi olan L298N için üreten network tabanlı bir programdır.

Kontrolör tabletten kablosuz olarak gelen yön ve hız bilgisine göre keşif robotunun hareketini sağlayan bu program, önce TCP üzerinden gelen veriyi bir byte dizisine çevirir ve bu byte dizisini sonrasında String veri türüne dönüştürür. Artık bu String veri türünde bu robotun hareket etmesi için gerekli olan PWM işaretin değerinin parçalanması gerekmektedir. Kontrolör tablet üzerinden gelen veri örnek verir isek [Sağ motorun PWM frekansı] [Sol motorun PWM frekansı] gibidir. Raspberry Pi tarafında böyle bir verinin okunabilmesi için öncelikle parçalanması gerekmektedir. Kontrolör tabletten örnek verir isek motorun düz hareketi için arayüzde ilgili butona basıldığında TCP üzerinden “\L255\R255\mi” komutu gönderilsin. Bu komutta sağ ve sol motorun PWM frekansı en son değer olan 255’tir ve yönü “mi” yani motor ileri komutudur. Böylece robotun düz ileri yönde son hızda gitmesi istenmektedir. Bu String türündeki veriyi parçalamak için öncelikle ‘\’ i C++ programlama dilinde substring olarak tanımlar isek yani bir nevi belirteç olarak tanıtır isek bu karakterden sonraki kelime dizileri ve gene bu karakter okunana kadar gelen dizileri aynı string üzerinde parçalar isek yeni Stringler elde etmiş oluruz. Bu şekilde gelen verilerden yerel değişkenlerimizi elde edebiliriz. Örnek kod satırını verir isek;

```
void threat::readyRead() //  
{  
  
    QByteArray client_verisi;  
    client_verisi=socket->readAll();  
    QString veri = client_verisi;  
    QStringList pieces = veri.split( "\\" );  
    Sag_motor_pwm = pieces.value( pieces.length() - 2 );  
    Sol_motor_pwm = pieces.value( pieces.length() - 1 );  
    yon = pieces.value( pieces.length() - 0 );
```

Burda threat tabanlı üzerinde ana programı etkilemeden sürekli olarak kontrolör tabletten veri bekleyen bir iş parçacığı oluşturulmuş ve TCP üzerinden uzaktan komut geldiğinden önce bu komutu bir byte dizisinde toplamış ve ardından bu veri bir String veri türüne çevrilmiştir. String veri türünde bu kelime dizisinde bulunan ‘\’ belirtecinin kaç tane olduğu öğrenilmiş ve bu belirteçlerin başladığı ve bittiği yerdeki tüm kelime dizileri farklı

bir String'e atılmış ve bu sayede motorun hareketi için gerekli tüm parametreler elde edilmiştir.

Elde edilen değişkenler tanımlandıktan sonra WiringPi kütüphanesine bu veriler gönderilmiş ve motor hareketi sağlanmıştır. Keşif robotunun öne doğru hareketi için örnek bir kod verir ısek;

```
void motorileri(int x,int y) {  
    softPwmWrite(29,x);  
    softPwmWrite(27,y);  
    digitalWrite(25,LOW );  
    digitalWrite(24,HIGH);  
    digitalWrite(23, LOW);  
    digitalWrite(22, HIGH);  
    delay(30);  
}
```

Yukardaki kodlardan da görüleceği üzere elde edilen parametre fonksiyona gönderilmiş ve 29 ve 28 numaralı WiringPi GPIO'larından PWM sinyal elde edilmiş ve 25, 24, 23, 22 nolu pinler lojik seviyeler yardımı ile yön ayarı yapılmıştır.

Kameraların bakış açılarını değiştirmek için pan-tilt sistemini kontrol etmesi gereken programlardan biri ise servokontrol programıdır. Aynı bu programda yukarıda belirtilen gibi kontrolör tabletten gelen veriye göre servo motorları adım adım hareket ettirecek şekilde C++ tabanında yazılmıştır. Servo motorların sürülmesi için gereken bu program gelen komutlara göre servo blaster kütüphanesine ilgili komutu işletim sistemi üzerinden kodlanacak şekilde yazılmış bir programdır. Örnek kod satırını verir ısek;

```
fp = fopen("/dev/servoblaster", "w");  
QString komut = "echo 1="+QString::number(y_ekseni)+"% > /dev/servoblaster";  
system(qPrintable(komut));  
QString komut2 = "echo 0="+QString::number(x_ekseni)+"% >  
/dev/servoblaster";  
system(qPrintable(komut2));
```

Örnekte de görüleceği gibi ilk önce servoblaster kütüphanesinin dosyası “fopen” ile yazma modunda açılmış ve TCP üzerinden gelen veriler komutlar system komutu kullanarak servoblaster komutunun yürütülmesi sağlanmış ve bu sayede servo motorlarının sürülmesi için gerekli tüm PWM sinyaller bu kütüphane ve yazılım sayesinde elde edilmiştir.

DC motor ve servo motorları kontrol eden tüm programlar yukarıdada belirtildiği gibi TCP bağlantısı açıldıktan itibaren harekete geçen iş parçacığı threat yapısına sahiptir.

```
void sunucu::incomingConnection(qint32 socketDescriptor)
{
    soket = new QTcpSocket;
    qDebug() << "Uzak bilgisayardan baglanti istegi geliyor";
    qDebug() << "Baglanti kuruluyor";

    qDebug() << "Bu cihaz icin bu = " << socketDescriptor << "id atandi ";

    //Yeni baglanan aygitlar icin idsine gore yeni yan parcacik yani threat olustur.
    threat *thread = new threat(socketDescriptor, this);
    //Baglanti kopumunda threati sonlandirmasi icin bir sinyal olusturalim.
    connect(thread, SIGNAL(finished()), thread, SLOT(deleteLater()));
    thread->start(); //Threadi baslat.ffg
}
```

Bu thread yapısı gelen TCP bağlantısı üzerinden istemciye bir id tanımlar ve bundan sonra haberleşmeyi bu istemci için o id üzerinden sağlatır. Böylece çoklu bağlantılarla izin verilmiş ve çoklu bağlantılar üzerinden gelen komutların karışması önlenmiş ve sistem stabilize edilmiştir.

```
void sunucu::sunucuyu_baslat()
{
    int port = 1234;
    qDebug() << "Program Baslatiliyor.";
    qDebug() << "TCP/IP Baglantisi icin soket acilmaya calisiliyor.";
    if(!this->listen(QHostAddress::Any, port)) //Egerki dinlenilecek port acilamadi iseç
    {
        qDebug() << "Dinlemek uzere ayarlanilan tcp portu = " << port << "acilamadi";
        qDebug() << "Portun kullanilmadigindan emin olun ve programi tekrar baslatin.";
    }
    else //port acildi ise
    {
        qDebug() << "Bu port TCP/IP baglantisi icin kullaniliyor = " << port ;
        qDebug() << "Port acildi ve Uzak cihazdan baglanti bekleniyor.";
    }
}
```

Bu yapı yukarıdaki görülen kodlardan da anlaşılabilcegi üzere TCP bağlantısı için bir port açmaya çalışır eğer ki açılamadı ise uyarı verir. Bu keşif robotu uygulamasında Raspbian işletim sisteminde kullanılmayan portlar seçilmiştir.

Keşif robotu üzerinde bulunan sensörleri okumak için Arduino mikrodenetleyicisi kullanılmıştır. Bu sensörler Arduino'nun sahip olduğu 10 bitlik ADC bloğu sayesinde ilk önce digital bir veriye çevrilmiştir ve ardından UART haberleşme üzerinde Raspberry Pi'ye gönderilmiştir.

Bu yazılan program sayesinde hedef sistemi analiz etmek için gerekli bütün sensörler keşif robotu prototipinin üzerine yerleştirilebilecektir. Bu uygulamada keşif robotunun ön ve arka tarafında birer HC-SR04 mesafe sensörü, ortam sıcaklığını ölçmek için LM-35 sıcaklık sensörü, ortamda karbonmonoksit gazının saturasyonunu ölçmek için bir tane MQ-7 gaz sensörü kullanılmıştır.

Bu ölçülen sensör datalarını kontrolör tabletin arayüzüne yollamak için öncelikle sensör verileri ADC yöntemi ve PulseIn yöntemleri ile Arduino tarafından okunmuş ve bu sensör verileri 115200 baudrate hızında Raspberry Pi'nin USB portuna UART haberleşme teknigi üzerinden bir String dizisi şeklinde yollandı. Raspberry Pi ise aldığı UART verisini depolamış ve kontrolör arayüzden gelen veri isteme bilgisine karşılık bu depoladığı sensör verilerini C# tabanında hazırladığı arayüz programına yollamıştır. Bu işlemleri aşağıdaki komutlar ile gösterilir.

```
void bataryaolc() {  
    analoggerilim2 = analogRead(A3); //A1'den değeri ölç  
    analoggerilim2 = (analoggerilim2/1023)*5000;//değeri mV'a dönüştür  
    vout = (analoggerilim2 * 5.0*2.0) / 1024.0;  
}
```

Yukarda 11.1 Voltluk bir Li-Po bataryanın gerilim bölücü yöntemi sayesinde ADC teknigi kullanılarak ölçülen gerilimin hesaplanması sağlayan kodlar görüntülenmektedir.

```

gaz();
onmesafe();
arkamesafe();
bataryaolc();
sicaklikolc();
delay(300);
veri += "-";
veri += gaz_degeri;
veri += "-";
veri += distance2;
veri += "-";
veri += distance;
veri += "-";
veri += vout*2;
veri += "-";
veri += sicaklik;
Serial.println(veri);
veri="";
Serial.flush();

```

Yukardaki kod bloğu 300 milisaniyede bir sensör fonksiyonlarının çağrılması ile ölçülen sensör verilerinin toplanarak ‘-‘ belirteci olan String dizisine çevrilip UART üzerinden Raspberry Pi haberleşme portuna gönderildiği bir kod bloğu gösterilmektedir.

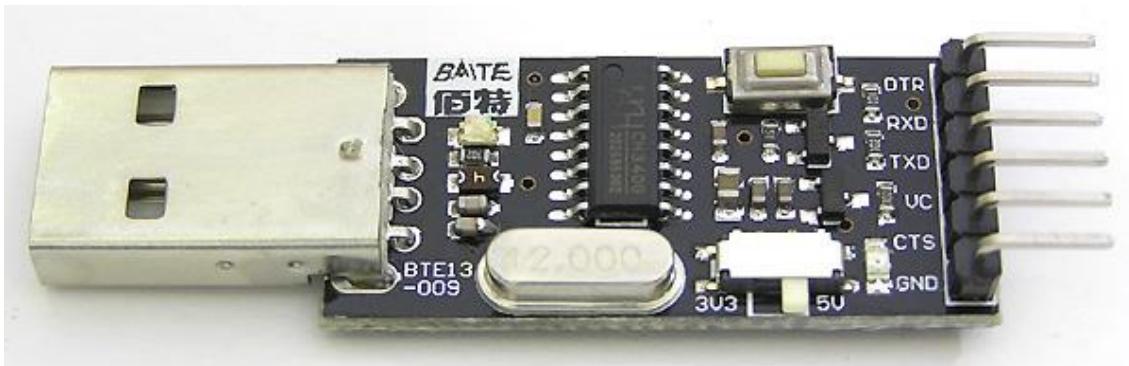
Raspberry Pi öncelikle Arduino ile haberleşebilmek için öncelikle ilgili COM portun kullanılmaması gerekmektedir. Raspberry Pi ile C++ ile ilgili com port şu kodlar ile açılabilmektedir.

```

#include "threat.h"
#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>
QSerialPort *serial;
QByteArray sensor_verisi;
threat::threat(qint32 ID, QObject *parent) :
    QThread(parent)
{
    serial = new QSerialPort(this);
    serial->setPortName("ttyAMA0");
    serial->setBaudRate(QSerialPort::Baud115200);
    serial->setDataBits(QSerialPort::Data8);
    serial->setParity(QSerialPort::NoParity);
    serial->setStopBits(QSerialPort::OneStop);
    serial->setFlowControl(QSerialPort::NoFlowControl);
    serial->open(QIODevice::ReadWrite);
    this->socketDescriptor = ID;
}

```

Yukardaki kodları açıklar ısek öncelikle QT'nin serialport kütüphaneleri koda dahil edilmiş ve ardından bir serial nesnesi oluşturulmuştur. Bu serial nesnesinin parent-child yapısı kurulmuş ve ardından serial portun açılması için gerekli tüm ayarlamalar yapılmıştır. Raspberry Pi ile Arduino Şekil 5.3.22 de görülebilen ve üzerinde ch340g serial-ttl çevirici bulunan bir modülle haberleşmektedir[53]. Bu modülün Raspberry Pi üzerinde hangi isimle atandığı aşağıdaki komut sayesinden görüntülenebilir. Bu komut sayesinde yukarıdaki C++ kodu hazırlanmış ve bu modülün sürekli bu isimle anılması sağlanmıştır.



Şekil 5.3.22 CH340G chipsetli USB-TTL dönüştürücü

Raspberry Pi üzerinde aşağıdaki Şekil 5.3.23'deki komut vasıtası ile COM portlar terminalde gösterilebilir ve böylece bu tarz USB-TTL çeviricilerin hangi ad ile Linux işletim sistemi tarafından tanımlandığı ilgili komut vasıtası ile görülebilir.

```
192.168.1.7 - PuTTY
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 15 18:02:19 2016 from 192.168.1.3
root@raspberrypi:~# ls dev/tty*
ls: cannot access dev/tty*: No such file or directory
root@raspberrypi:~# ls /dev/tty
/dev/tty
root@raspberrypi:~# ls /dev/tty*
/dev/tty  /dev/tty19  /dev/tty3   /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0  /dev/tty2  /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1  /dev/tty20 /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10 /dev/tty21 /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11 /dev/tty22 /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12 /dev/tty23 /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyAMA0
/dev/tty13 /dev/tty24 /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyprintk
/dev/tty14 /dev/tty25 /dev/tty36  /dev/tty47  /dev/tty58
/dev/tty15 /dev/tty26 /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16 /dev/tty27 /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17 /dev/tty28 /dev/tty39  /dev/tty5  /dev/tty60
/dev/tty18 /dev/tty29 /dev/tty4   /dev/tty50  /dev/tty61
root@raspberrypi:~#
```

Şekil 5.3.23 İletişim bağlantı noktalarının terminalde görüntülenmesi

Raspberry Pi de yazılan TCP/IP protokolü kullanan programdan birkaç komut verir isek; Yukardaki kod blokları ile ilgili readyRead() yani TCP ve serial den okunabilir sinyali geldiği zaman C++ programı ilgili kod bloklarına dallanacak şekilde ayarlanmıştır. Bu threat yapısı sayesinde aynı anda gelen verilerin karışmaması için gerekli önlemler alınmış olur.

```
connect(socket, SIGNAL(readyRead()), this, SLOT(readyRead()),
Qt::DirectConnection);
connect(serial,SIGNAL(readyRead()),this,SLOT(seriveri()));
```

Arduino veriyi UART üzerinden gönderdiği zaman C++ programında ilgili fonksiyon olan seriveri() fonksiyonuna dallanır ve UART verisi değişkene aktarılıp depolanır.

```
void threat::seriveri()
{
    sensor_verisi = serial->readAll();
    qDebug() << sensor_verisi;
}

void threat::readyRead() //Veriler okunmaya basladigin an calis
{
    QByteArray gelen_veri = socket->readAll();
    qDebug() << "Bu id tanimli pc'den örnekleme isteği verisi alindi
: " << socketDescriptor << " Alinan Veri : " << gelen_veri;
    socket->write(sensor_verisi);
}
```

Kontrolör tablet üzerinden gelen veri isteme komutu TCP üzerinden aktarıldığı zaman readyRead() fonksiyonunda TCP verisi okunur ve sensör verisi ilgili ID'ye sahip TCP istemcisine yollanır.

Keşif robottu ile kontrolör tablet arasında bağlantı kesildiği zaman robotun yerinde sabit durması için bir keşif robottu için bir ping uygulaması yazılmıştır. Böylece kontrolör tablet her 5 saniyede bir veri yollayacak ve keşif robottu bunu onaylayıp ACK mesajı yollayacaktır. Eğer ki keşif robottu ile kontrolör arasındaki kablosuz iletişim kopar ise robotun hareketini durdurması için bir C++ tabanlı timer uygulaması yazılmıştır. Buna göre 5 saniyede bir veri gelmesi gerekkirken gelmezse timer resetlenmeyecek ve program disconnect() fonksiyonuna dallanıp motorları susturacaktır. Bu uygulama için yazılan C++ tabanlı kodlar şu şekildedir.

```

void threat::run()
{
    wiringPiSetup () ;
    pinMode (25, OUTPUT) ;
    pinMode (24, OUTPUT) ;
    pinMode (23, OUTPUT) ;
    pinMode (22, OUTPUT) ;
    softPwmCreate(29,0.250);softPwmWrite(29,0);
    softPwmCreate(27,0.250);softPwmWrite(27,0);
    timer = new QTimer();
    timer->setInterval(5000);
    timer->start();
    qDebug() << "Threat Olusturuldu ve veriler okunmaya hazir.";
    socket = new QTcpSocket(); //cift taraflı haberleşme için bir adet soket acalim.

    //baglanan pcnin idsine gore doğru bir haberleşme için idleri aynı yapalım.
    if(!socket->setSocketDescriptor(this->socketDescriptor))
    {
        //Egerki socket acilamadi ise;
        qDebug() <<"Threat haberlesme soketi acilamadi.Threat sonlandirilacak." ;
        emit error(socket->error());
        return;

        connect(socket, SIGNAL(readyRead()), this, SLOT(readyRead()),
Qt::DirectConnection);
        connect(timer,SIGNAL(timeout()),this,SLOT(pingkontrol()));
        connect(socket, SIGNAL(disconnected()), this, SLOT(disconnected()));

        qDebug() << "Bu id icin baglanti kuruldu = " <<socketDescriptor;
        exec(); //bu threati sonsuz dongü yapıyoruz.
    }
}

```

Yukardaki kodlardan da görüleceği üzere öncelikle motor sürücü devresi için gerekli PWM sinyalleri üreten WiringPi kütüphanesi ayarları yapılmış, ardından timer nesnesi oluşturulmuş ve 5 saniyeye kurulmuştur. Eğer bu süre esnasında tekrardan resetlenmez ise disconnect() fonksiyonuna gitmesi sağlanmıştır.

```

void threat::readyRead() //Veriler okunmaya basladigin an calis
{
    QByteArray Data = socket->readAll();

    qDebug()<< Data;

    if(Data[0]=='p' && Data[1]=='k') {
        qDebug() <<"ping mesaji geldi";
        timer->start();      }
}

```

TCP/IP üzerinden veri geldiği zaman bu veri önce parçalanır ve gelen mesajın türüne bakılır. Eğer gelen ping mesajı ise timer resetlenir.

```
void threat::disconnected()
{
    softPwmWrite(29,0);
    softPwmWrite(27,0);

    digitalWrite(25, LOW);
    digitalWrite(24, LOW);
    digitalWrite(23, LOW);
    digitalWrite(22, LOW);
    qDebug() << socketDescriptor << " Baglanti Kesildi.";
    socket->deleteLater(); //Soketi temizle ve kapat.
    exit(0); //threati kapat.
}
```

Eğer ki TCP/IP üzerinden veri gelmez ise timer timeout olur ve bu yüzden disconnect() fonksiyonunda motoru durdurun kod çalışır ve böylece robotun hareketi sonlandırılmış olur. Bundan sonra ilgili TCP bağlantısı o ID'ye sahip olan TCP istemcisi için sonlandırılmış olur.

Keşif robotunun çalışması için gereken kütüphanelerin, programların ve ayarların başlangıçta çalışması için bu yöntemlerin hepsinin başlangıçta ilgili dosyaya kaydedilmesi gerekmektedir. Bu yöntemlerin hepsinin bir arada senkron çalışabilmesi kurulan bu sistem için önemlidir. Bu nedenle bu yöntemlerin çalışabilmesi için Raspberry Pi'de başlangıçta tetikleyici script olan “/etc/rc.local” dosyasının içinde değişiklikler ve eklemeler yapılmıştır. Bu script Raspberry Pi için başlangıçte çalışması istenen her türlü içinde barındıran bir dosyadır. Bu bitirme tezi için bu dosya aşağıdaki gibi düzenlenmiştir.

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
```

```

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

sudo /root/PiBits/ServoBlaster/user./servod

sudo /usr/local/bin/mjpg_streamer -i "/usr/local/lib/input_uvc.so -d /dev/video0 -f 30 -r 1280*720" -o "/usr/local/lib/output_http.so -p 8080 -w /usr/local/www" &
sudo /usr/local/bin/mjpg_streamer -i "/usr/local/lib/input_raspicam.so -fps 30" -o "/usr/local/lib/output_http.so -p 8081 -w /usr/local/www" &
sudo /root/MotorKontrolu-build-desktop-Qt_4_8_2__System__Release/motorkontrol &
sudo /root/raspberry servo-build-desktop-Qt_4_8_2__System__Release/servokontrol &
sudo /root/ping_kontrol-build-desktop-Qt_4_8_2__System__Release/pingkontrol &
sudo /root/arduino_tcp-build-desktop-Qt_4_8_2__System__Release/arduino_tcp &
exit 0

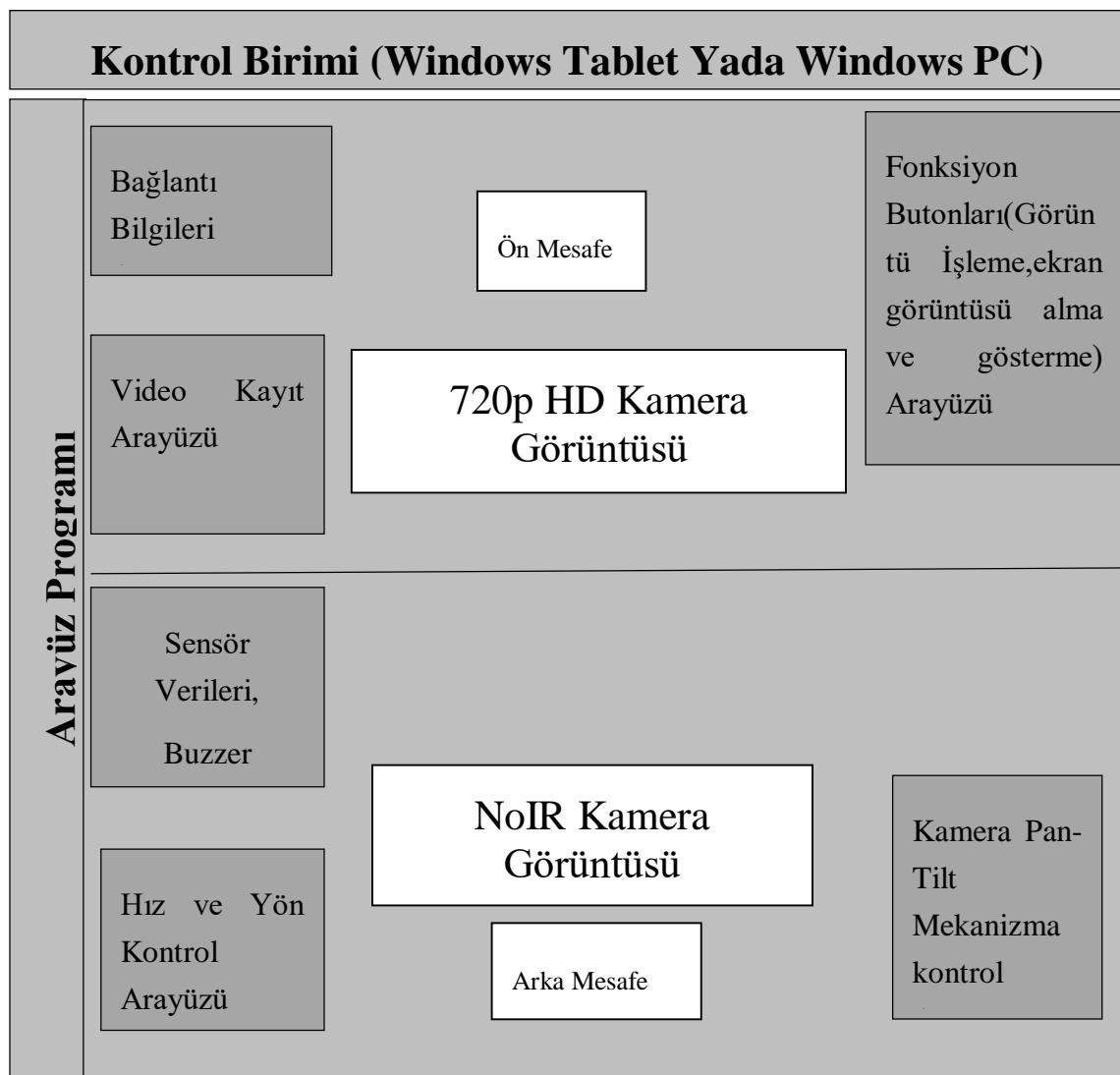
```

Bu script ile birlikte Raspberry Pi için gerekli yazılımsal PWM kütüphanesi etkinleştiriliyor, sunucu programları aktifleştiriliyor ve ardından kameraların başlatılması için gerekli komutlar eklenmiştir.

Böylelikle robot birimi artık harekete hazırır.

5.4 Windows Tabanlı Kontrolör Tablet Çalışma Prensibi

Gezgin birimi kontrol eden tablet bilgisayarımız gezgin birimden veri alıp işleyen ve komut gönderebilme özelliğini karşılayabilecek bir donanıma sahiptir. Kontrol bilgisayarı arayüzünde kullanılan programlar, zengin bir içeriğe sahip olması bakımından birbirile bağıntılı C# üzerinde Windows Form Application ve Windows Presentation Foundation (WPF) bileşenleri kullanılarak yazılan programlardır. Keşif robotun yönetmek isteyen kullanıcıya kullanımı oldukça kolay bir arayüz sağlamaktadır. Arayüz üzerinde iki farklı kamerasından alınan görüntüler, sensör verileri, hız, yön ve pan-tilt kontrol formları bulunmaktadır. Ayrıca görüntü işleme, ekran görüntüsü alma, video kayıt, batarya ve uzaklık sensörlerinden alınabilen transparant barlar bulunmaktadır. Bu tablet bilgisayar Windows 8.1 işletim sistemine sahiptir ve Şekil 2'de arayüzünün blok diyagramı gösterilmektedir.



Microsoft Windows Tabanlı Tablet veya PC

Şekil 5.4.1 Keşif robotu kısmının bileşenleri ve bağlantıları

Kontrolör birimi olan tablet pc kullanım kolaylığı ve her yerden erişebilmesi için kullanımı daha yaygın olan bir işletim sistemi için tasarlanmıştır. Bu sebeple bu tablet üzerinde çalışacak olan işletim sistemi olarak Windows işletim sistemi barındıran bir tablet seçilmiştir.

Kontrolör biriminde yapılması istenen görevleri madde ile verirsek:

- Raspberry Pi'nin yolladığı gece ve gündüz kameralarından alınan görüntüleri stream edip ilgili arayüzde göstermek,

- Keşif robotunun hareketi için gereken yön ve hız bilgisini Raspberry Pi'ye iletmek,
- Sensör verilerini ekranda ilgili arayüzde göstermek,
- Video kaydı ve screenshot alabilmek,
- Görüntü işleme yapması olarak tasarlanmıştır

Asağıdaki şekilde C# programlama dili kullanılarak yazılmış robot tankın alpha sürümünün arayüzü Şekil 5.4.2'de görüntülenebilir.



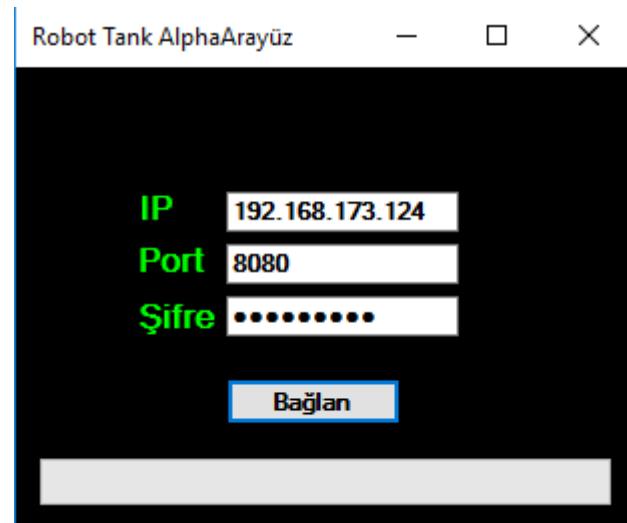
Şekil 5.4.2 Kontrolör birimi kontrol arayüzü

Yukardaki arayüzde de görüleceği üzere sol alt tarafta basit olarak keşif robotunu hareket ettirecek butonlar, sol alt tarafta kamera pan-tilt sistemini yönetmek için gerekli olan butonlar, sol üstte far ve sesli uyarı için transparant butonlar, yukarı sağ tarafta görüntü işleme için gerekli butonlar ve sol orta kısmında sensör verilerinin okunabileceği yer ve onun üstünde de ekran görüntüsü kaydetme butonları yer almaktadır. Orta kısmında yer alan transparant barlar ise mesafe sensörlerinden gelen verilerdir.

Keşif robotunun arayüzü C# programlama dili tabanlı programlardan oluşmaktadır. Daha önceden de bahsedildiği gibi arayüzü basit ve güzel bir görüntü sunmak için Windows Form ve WPF form yapıları kullanılmıştır.

Keşif robotu arayüzüne giriş yapmak için keşif robotunun sahip olduğu IP numarası, port numarası ve şifre girilerek giriş yapılır. Burdaki şifre harici parametreler bilinmekte ve her programda otomatik olarak yüklenirler. Çünkü keşif robotu sabit ip'ye sahiptir.

Bu programda bağlanan basıldığı andan itibaren keşif robotunun çalışması için gerekli tüm programlar sırası ile tam ekran açılır.



Şekil 5.4.3 Arayüz giriş ekranı

Bu arayüzdeki kodlardan bir kısmı paylaşır isek;

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.TextLength > 1 && textBox2.TextLength > 1)
    {
        varsayılan_ip = textBox1.Text;
        varsayılan_port = Int32.Parse(textBox2.Text);
    }
    ProcessStartInfo arayuzubaslat = new ProcessStartInfo();
    arayuzubaslat.FileName = @"C:\Robot\AlphaEkran\alphaversiyon.exe";
    string baglanti = string.Concat(varsayılan_ip, " ", varsayılan_port);
    arayuzubaslat.Arguments = baglanti;
```

```

        Process.Start(arayuzubaslat);
        Thread.Sleep(500);
        ProcessStartInfo motorkontrol = new ProcessStartInfo();
        motorkontrol.FileName = @"C:\Robot\TankMotor.exe";
        motorkontrol.Arguments = varsayılan_ip;
        Process.Start(motorkontrol);
        Thread.Sleep(500);
        ProcessStartInfo servokontrol = new ProcessStartInfo();
        servokontrol.FileName = @"C:\Robot\ServoMotor.exe";
        servokontrol.Arguments = varsayılan_ip;
        Process.Start(servokontrol);
        ProcessStartInfo sensor = new ProcessStartInfo();
        sensor.FileName = @"C:\Robot\Sensorler.exe";
        sensor.Arguments = varsayılan_ip;
        Process.Start(sensor);
        this.Close();
    }
}

```

Yukardaki C# programlama dili tabanlı arayüz giriş programı sayesinde öncelikle program arayüz programlarını girilen IP'yi argüman olarak programlara yollayarak bu programların açılmasını sağlar ve en son kendisini kapatır.

Bu keşif robottu üzerindeki arayüz programları ip bilgisini argümanlar sayesinde alır ve bu ip bilgisi ile keşif robotunun ipsi programlara tanıtılmış olur. Argüman yapısı ise aşağıdaki kod bloklarından daha iyi anlaşılabilir.

```

public MainWindow()
{
    InitializeComponent();
    string[] argument = Environment.GetCommandLineArgs();
    if (argument.Length > 1)
    {
        varsayılan_ip = argument[1];
    }
    try
    {
        arduino.Connect(varsayılan_ip, varsayılan_port);
    }
    catch (Exception a)
    {
        MessageBox.Show("Raspberry Bağlantısı Kurulamadı.");
    }
}

```

Örnek olarak yukarıdaki kod bloğunda program eğer ki argüman ile başlatılmış ise artık yeni IP bu argümandan gelen IP olur ve bu IP üzerinden keşif robottu ile TCP/IP protokolü haberleşmesi sağlanır.

Raspberry Pi üzerinden alınan mjpeg formatındaki yayın akışı AFORGE.net kütüphanesi sayesinde bir videosourceplayer ögesinin üzerinde görüntülenmektedir. Bu keşif robottu arayüzünde bu stream tam ekran görüntülenecek şekilde ayarlanmıştır. Bu mjpeg streamin alınması, bu streamin kayıt edilmesi ve görüntü işleme yapılması için aşağıdaki AFORGE.net kütüphanelerine ihtiyaç duyulmuştur. Yukardaki kodlar sayesinde arayüz programı için gerekli olan fonksiyonlar, kütüphane dosyaları import edilerek kullanılabilir hale getirilmiştir. Örnek bir mjpeg akışını arayüzde görüntülemek için aşağıdaki kodlar çağırılabilir.

```
using AForge;
using AForge.Video;
using AForge.Vision;
using AForge.Video.VFW;
using AForge.Imaging.Filters;
using AForge.Vision.Motion;
using AForge.Imaging;
```

```
private void hdcamera(IVideoSource link)
{
    hdcamera();
    videoSourcePlayer.VideoSource = new AsyncVideoSource(link);
    videoSourcePlayer.Start();
    statIndex = statReady = 0;
    timer.Start();
    videoSource = link;
}
```

```
string ip = textBox1.Text.ToString();
string port = textBox2.Text.ToString();
string protokol = "http://";
string slash = ":";
string erisim = "?action=stream";
string baglanti_linki = string.Concat(protokol, textBox1.Text, slash, port,
erisim);
MJPEGStream hdcamlinki = new MJPEGStream(baglanti_linki);
hdcamera(hdcamlinki);
timer.Start();
```

Bu sayede akış yapılan ip ilgili fonksiyonlara tanıtılmış ve videoSourceplayer'in bu streamı oynatması ayarlanmıştır.

Arayüz içerisinde diğer kamerasa geçmek için önce açılan streamin kapatılması gerekmektedir. Aşağıda gösterilen kodlarla bu akış sonlandırılabilir.

```
private void hdkamerakapa()
{
    videoSourcePlayer.SignalToStop();
    for (int i = 0; (i < 50) && (videoSourcePlayer.IsRunning); i++)
    {
        Thread.Sleep(100);
    }
    if (videoSourcePlayer.IsRunning)
        videoSourcePlayer.Stop();
    timer.Stop();
    if (hd_writer != null)
    {
        hd_writer.Dispose();
        hd_writer = null;
    }
}
```



Şekil 5.4.3 NoIR kamerasından alınan MJPEG stream akışından bir ekran görüntüsü



Şekil 5.4.4 NoIR kameradan alınan MJPEG stream akışından bir ekran görüntüsü

Bu ekran görüntülerinin alınabilmesi için aşağıdaki C# tabanlı kodlar yazılmıştır.

```
private void button3_Click(object sender, EventArgs e)
{
    pictureBox1.Visible = true;
    button7.Visible = true;
    if (hdkameradurumu == true)
    {
        button3.BackColor = Color.Green;
        Bitmap bitmap =
(Bitmap)videoSourcePlayer.GetCurrentVideoFrame().Clone();
        DateTime date = DateTime.Now;
        String fileName = String.Format(@"C:\Robot\ekrangoruntuleri\goruntu-
{0}-{1}-{2} {3}-{4}-{5}-{6}.jpg",
date.Year, date.Month, date.Day, date.Hour, date.Minute,
date.Second,date.Millisecond);
        bitmap.Save(fileName, ImageFormat.Jpeg);
        pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
        pictureBox1.Image = bitmap;
        pictureBox1.Visible = true;
    }
}
```

Yukardaki kodlardan da görülebileceği üzere öncelikle arayüz ekranında ekran görüntüsünü alma butonuna basıldığı an program bu fonksiyona dallanır ve öncelikle videosourceplayer üzerinden o anki karenin yani framenin bir kopyasını bitmap olarak alır. Sonra tarih ve zamanı date öğesine sahip bir yöntemden alır ve ilgili klasöre ekran görüntüsünün aldığı tarih ile kaydeder.

Eğer kontrolör tablet ekranın ekran görüntüsü alınmak isteniyorsa aşağıda belirtilen kodlar kullanılabilir ve bu sayede ekranda verilerde görüntülenebilir.



Şekil 5.4.5 HD kameradan alınan MJPEG stream akışından bir ekran görüntüsü

```

private void button1_Click(object sender, EventArgs e)
{
    Rectangle bounds = Screen.GetBounds(Point.Empty);
    using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height))
    {
        using (Graphics g = Graphics.FromImage(bitmap))
        {
            g.CopyFromScreen(Point.Empty, Point.Empty, bounds.Size);
        }
        DateTime date = DateTime.Now;
        String fileName = String.Format(@"C:\Robot\ekrangoruntuleri\goruntu-{0}-{1}-{2} {3}-{4}-{5}.jpg",
            date.Year, date.Month, date.Day, date.Hour, date.Minute,
            date.Second);
        bitmap.Save(fileName, ImageFormat.Jpeg);
    }
}

```

Yukardaki kodlardan bahseder isek öncelikle bir çerçeve oluşturulmuş ve ardından tüm ekranın görüntüsü bitmap olarak kopyalanmıştır. Kopyalanan bitmap hedef yola çekildiği tarih ve görüntü ile birlikte yazılmıştır. Aşağıdaki Şekil 5.4.6'da bu ekran görüntüsünün bir örneği görüntülenebilir.



Şekil 5.4.6 HD kameradan görüntü

Keşif robotunda mjpeg streamda HD kayıt yapılabilmesi ve tüm arayüz ekranının kayıtı yapılabilmesi için aşağıdaki kodlar yazılmıştır.

```
private void video_NewFrame( object sender, NewFrameEventArgs eventArgs )
{
    if ( kayittami )
    {
        yazici.WriteVideoFrame( eventArgs.Frame );
        this.lb_stopWatch.Invoke( new Action( () =>
        {
            this.lb_stopWatch.Text = kronometre.Elapsed.ToString();
        } ) );
    }
    else if ( duraklatmi==true )
    {
        kronometre.Stop();
    }
    else
    {
        kronometre.Reset();
        Thread.Sleep( 500 );
        ekrankaydedici.SignalToStop();
        Thread.Sleep( 500 );
        yazici.Close();
    }
}
```

Yukardaki kodlar ile tüm arayüz ekranın bir 30 fps hızında her yeni karenin video dosyasına yazılması ile elde edilen video dosyasının algoritmasının kodlarıdır. Bu sayede tüm ekranın video kaydı alınabilmektedir.

Eğer sadece jpeg akışını video olarak kaydedilmek istenirse aşağıdaki kodlar yazılabılır.

```
private void hdkamera_gelen_frame(object sender, ref Bitmap image)
{
    if (kayit_durumu == true && hdkameradurumu == true)
    {
        if (hd_writer == null)
        {
            // create file name
            DateTime date = DateTime.Now;
            String fileName = String.Format("{0}-{1}-{2} {3}-{4}-{5}-{6}.avi",
                date.Year, date.Month, date.Day, date.Hour,
                date.Minute, date.Second, date.Millisecond);

            try
            {
                // create AVI writer
                hd_writer = new AVIWriter("wmv3");
                // open AVI file

                hd_writer.FrameRate = 30;

                hd_writer.Open(fileName, 1280, 720);

            }
            catch (ApplicationException ex)
            {
                if (hd_writer != null)
                {
                    hd_writer.Dispose();
                    hd_writer = null;
                }
            }
        }
        //
        hd_writer.AddFrame(videoSourcePlayer.GetCurrentVideoFrame());
        hd_writer.AddFrame(image);
        this.textBox3.Invoke(new Action(() =>
        {
            this.textBox3.Text = kronometre.Elapsed.ToString();
        }));
    }
}
```

Yukardaki kodlar sayesinde jpeg streamdan gelen frameler ile yeni event oluşturulmuş ve bu eventlerden gelen bitmap türündeki görüntü kareleri video dosyasına yazılmıştır ve bu sayede. avi formatında 720p30FPS video elde edilmiştir.

Kontrolör arayüzde Raspberry Pi'nin TCP/IP protokolü üzerinden gönderdiği sensör datalarının parçalanıp ilgili WPF formlarda görüntülenmesi için aşağıdaki kodlar

yazılmıştır. Öncelikle bir threat oluşturulmuş ve bu threatın içerisinde TCP bağlantısı kurulmuştur. Bu threat TCP üzerinden gelecek verileri dinleyen bir listener(dinleyici) olarak tasarlanmıştır. Örnek kodlar aşağıda verilmektedir.

```
public MainWindow()
{
    InitializeComponent();

    dispatcherTimer.Tick += dispatcherTimer_Tick;
    dispatcherTimer.Interval = TimeSpan.FromMilliseconds(300); ;
    dispatcherTimer.Start();

    string[] argument = Environment.GetCommandLineArgs();
    if (argument.Length > 1)
    {
        varsayılan_ip = argument[1];

    }

    try
    {
        arduino.Connect(varsayılan_ip, varsayılan_port);

    }
    catch (Exception a)
    {
        MessageBox.Show("Raspberry Bağlantısı Kurulamadı.");
    }
    t = new Thread(new ThreadStart(arduino_oku));// threadi arduino oku
fonksiyonuna baglıyoruz
    t.Start(); // bağlantı sağlandıktan sonra thread başlıdı

}
```

```

void arduino_oku()
{
    NetworkStream st = arduino.GetStream();
    while (true)
    {
        if (st.DataAvailable)
        {
            byte[] data = new byte[100];
            using (MemoryStream ms = new MemoryStream())
            {
                int numBytesRead;
                while ((numBytesRead = st.Read(data, 0, data.Length)) > 0)
                {
                    ms.Write(data, 0, numBytesRead);
                    break;
                }
                str = Encoding.ASCII.GetString(ms.ToArray(), 0, (int)ms.Length);
                string[] degerler = str.Split('-');
                try
                {
                    sicaklik_label.Dispatcher.BeginInvoke((Action)(() =>
sicaklik_label.Content = degerler[5].ToString()));
                    bataryalabel.Dispatcher.BeginInvoke((Action)(() =>
bataryalabel.Content = degerler[4].ToString()));
                    label1.Dispatcher.BeginInvoke((Action)(() => label1.Content =
degerler[2].ToString()));
                    label2.Dispatcher.BeginInvoke((Action)(() => label2.Content =
degerler[3].ToString()));
                    gaz_label1.Dispatcher.BeginInvoke((Action)(() =>
gaz_label1.Content = degerler[1].ToString()));

                    on_mesafe = int.Parse(degerler[2]);
                    arka_mesafe = int.Parse(degerler[3]);
                    if (on_mesafe > 100)
                    {
                        label1.Dispatcher.BeginInvoke((Action)(() => label1.Content =
"+100"));
                    }

                    if (arka_mesafe > 100)
                    {
                        label2.Dispatcher.BeginInvoke((Action)(() => label2.Content =
"+100"));
                    }
                }
            }
        }
    }
}

```

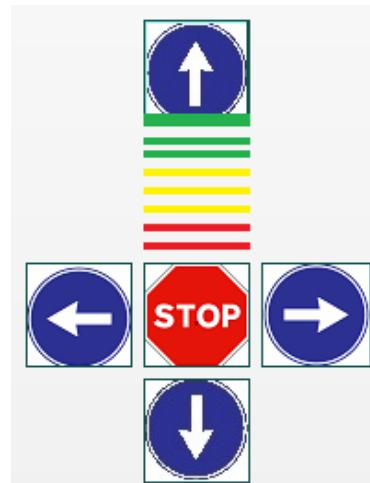
Yukardaki kodlarda TCP de gelen veriler bir diziye atandıktan sonra bu dizi memory stream'a kaydedilmiştir. Ardından ise bu veri String'e dönüştürülmüş ve önbölirteç '-' olan karakter ile parçalanıp gerekli veriler Dispatcher. BeginInvoke() methodu ile threat üzerinden GUI güncellemesi yapılmıştır. Bu sayede aşağıdaki şekilde görülen sensör verileri elde edilmiştir.



Şekil 5.4.7 Sensör arayüzü formu



Şekil 5.4.8 Mesafe arayüzü formu



Şekil 5.4.9 Motor kontrol formu

Hız kontrolü ve keşif robotunun hareketi için Windows Form ve WPF te tasarlanmış farklı farklı uygulamalar yazılmıştır. Bu programlar keşif robotu ile ilgili bölümde anlatıldığı gibi keşif robotuna kablosuz olarak komut gönderir ve bu komutlar sayesinde keşif robotu hareketi sağlanmış olur. Bu program video görüntüsünün üstüne yerleştirilmiştir ve de saydam yapılmıştır. Bu sayede güzel bir görünüm elde edilmiştir.

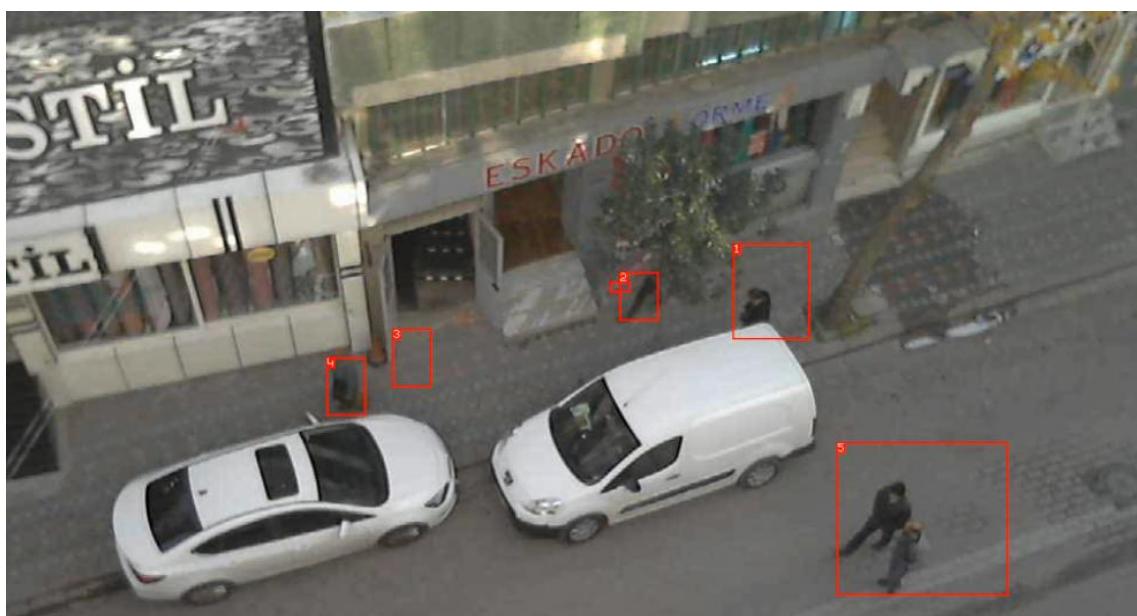
Keşif robotu arayüz programında görüntü işleme için çeşitli uygulamalar yazılmıştır. Görüntü işleme için yine AFORGE.net kütüphanesi kullanılmıştır. Bu uygulamalarda genellikle jpeg stream üzerinden gelen veriler kare kare işlenip gerekli görüntü işleme işlemleri yapılmıştır.



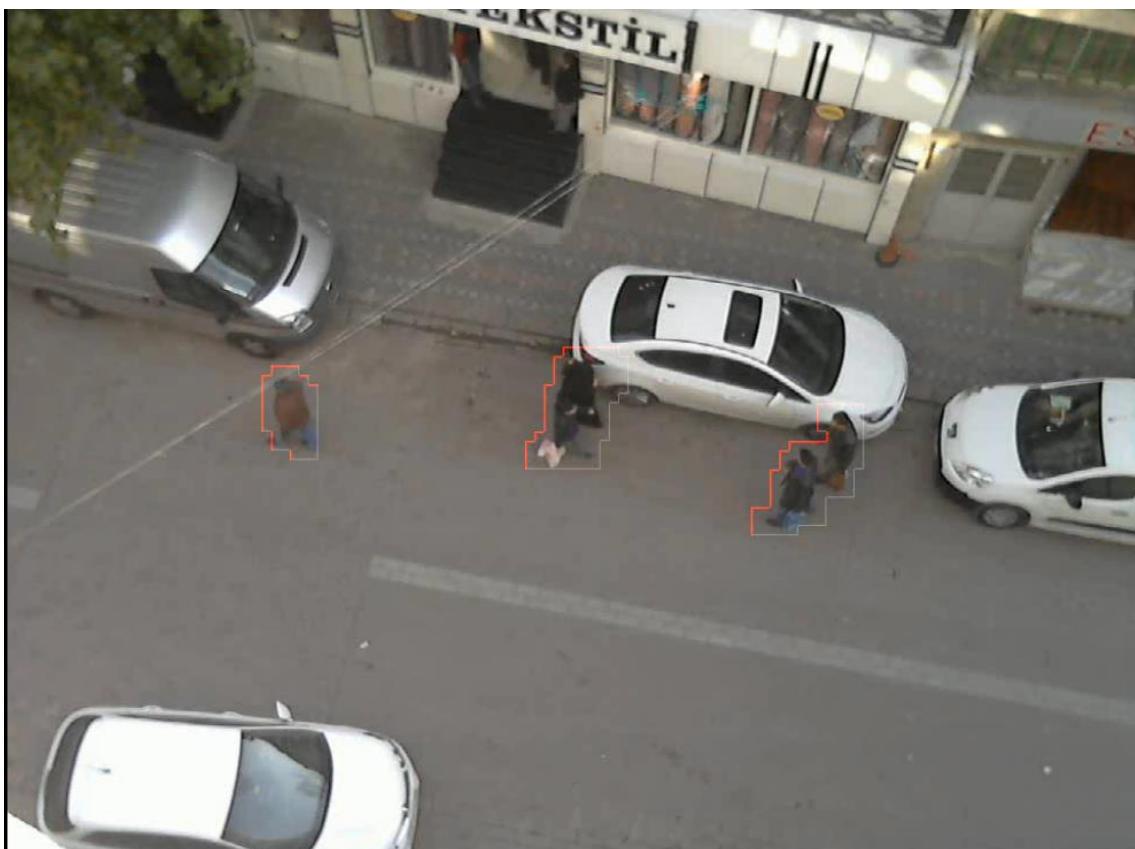
Şekil 5.4.10 Görüntü işleme kütüphanelerinden bir örnek



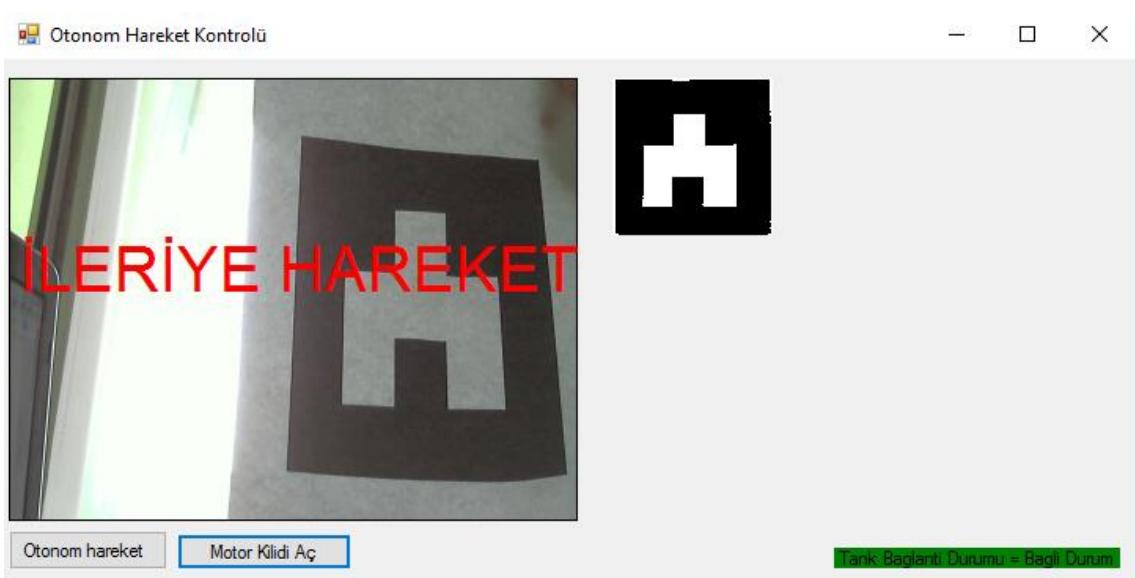
Şekil 5.4.11 İki kare farkı ile hareketli nesne takibi



Şekil 5.4.12 İki kare farkı ile hareketli nesne sayımı



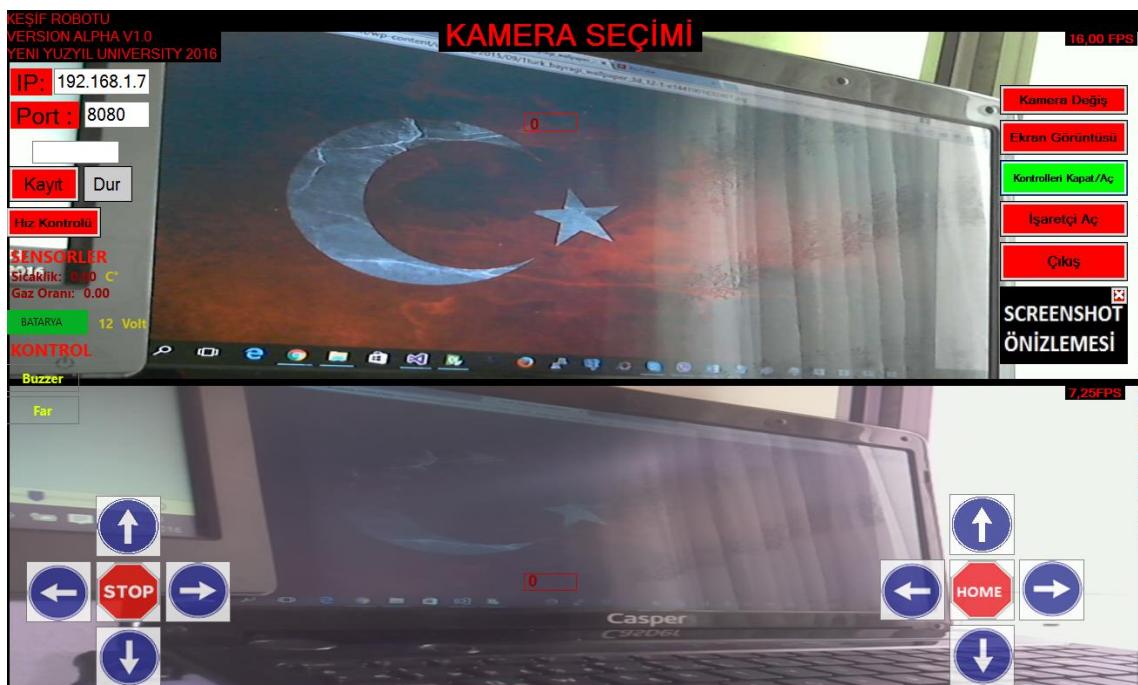
Şekil 5.4.13 İki kare farkı ile iyileştirilmiş hareketli nesne takibi



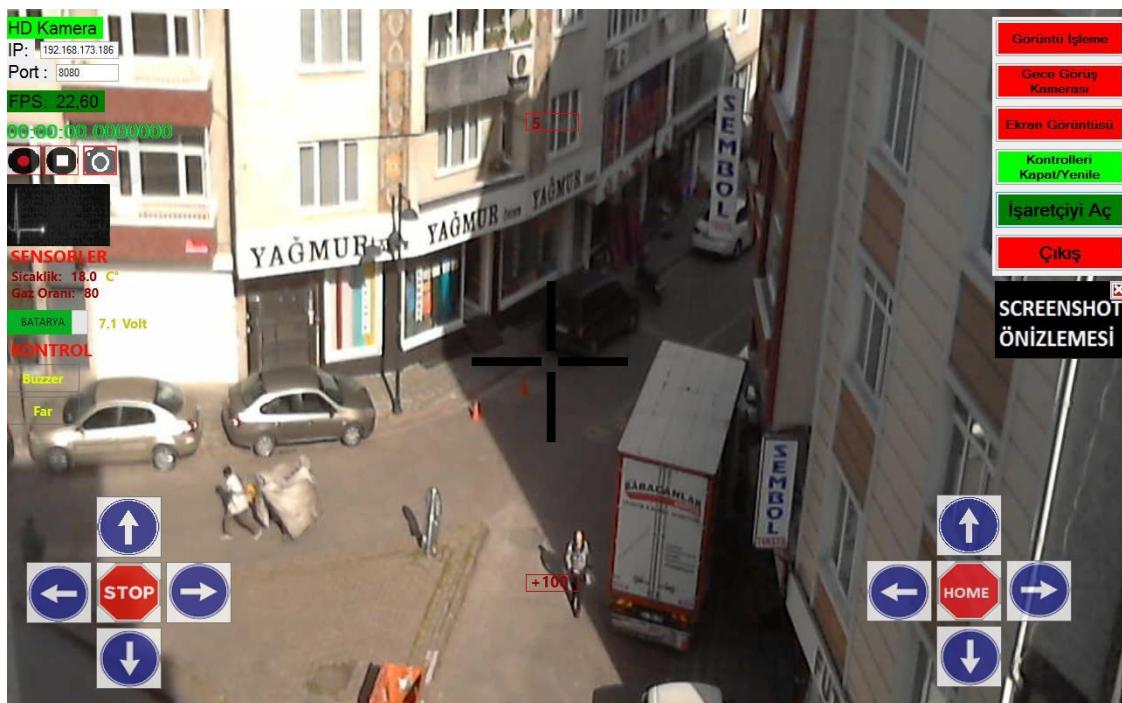
Şekil 5.4.14 Karakter algılama ile ileri otonom kontrol arayüzü



Şekil 5.4.15 Karakter algılama ile sola otonom kontrol arayüzü



Şekil 5.4.16 İki kamera görüntüsü aynı anda



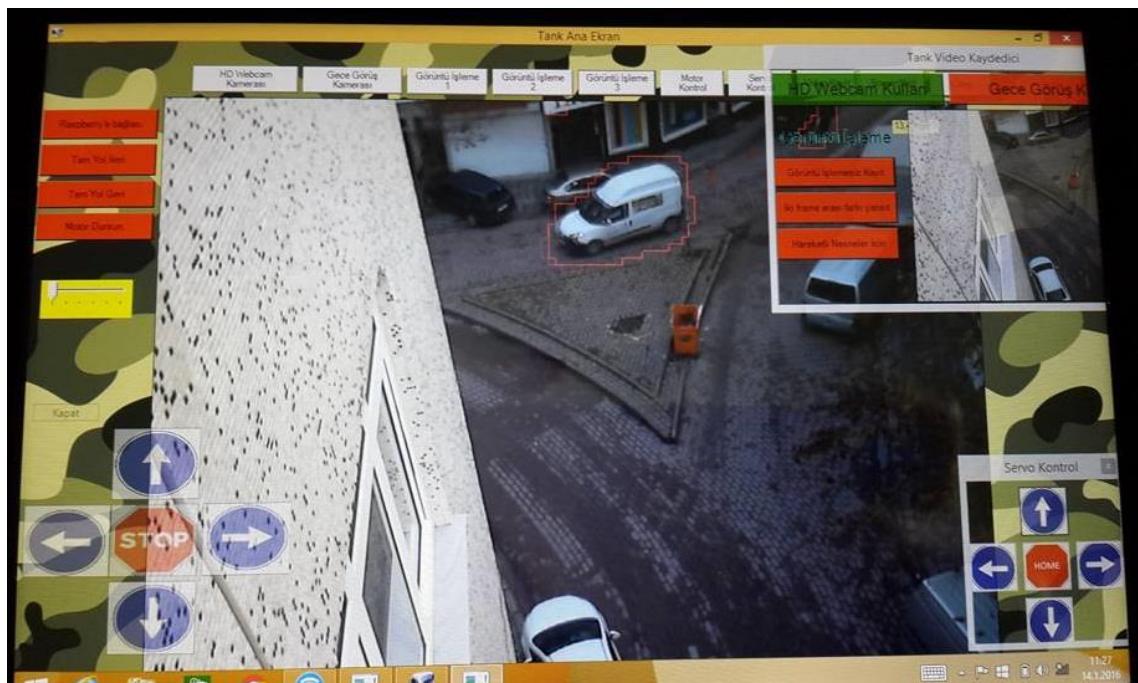
Şekil 5.4.17 Kuşbakısı kontrol arayüzü HD kamera ekranı



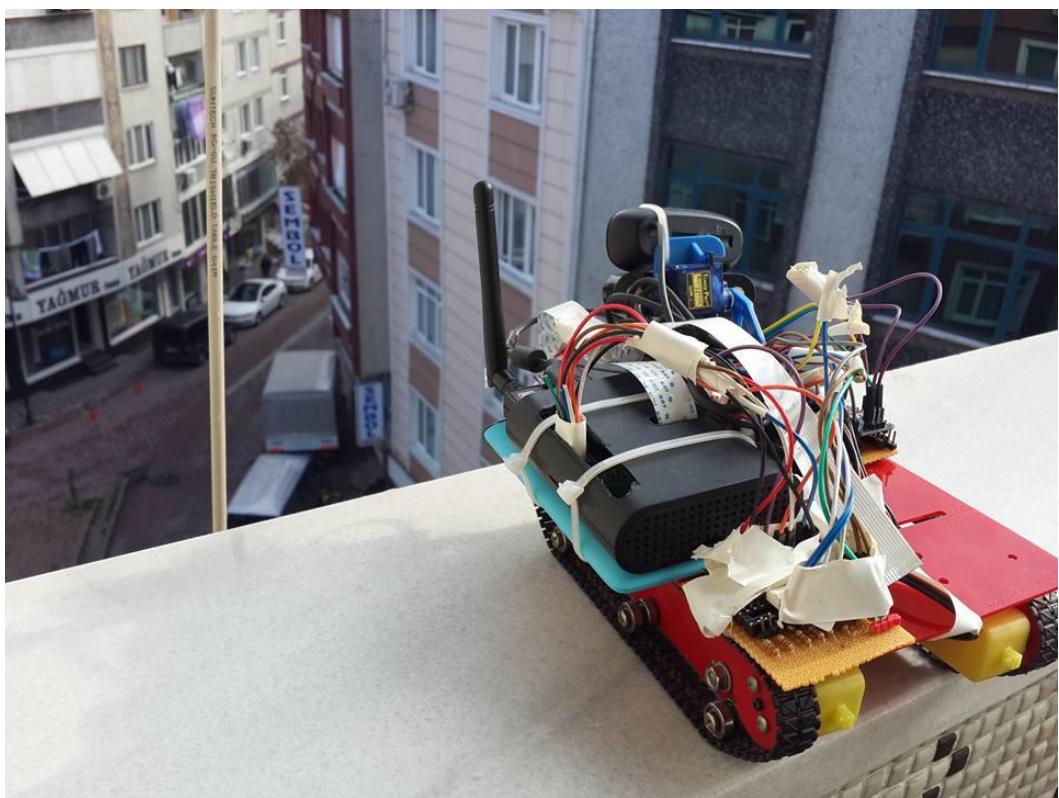
Şekil 5.4.18 Kuşbakısı kontrol arayüzü NoIR kamera ekranı



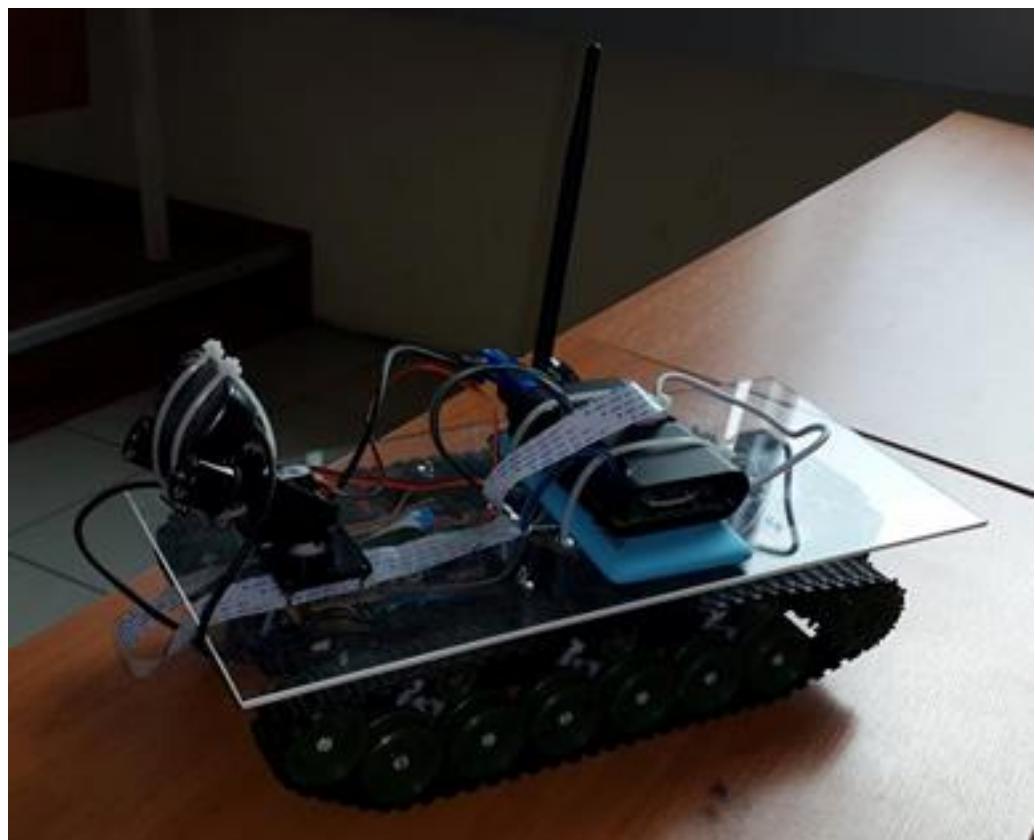
Şekil 5.4.19 Hareket halinde kontrol arayüzü



Şekil 5.4.20 İlk görüntülü işlemeli arayüz



Şekil 5.4.20 İlk prototip keşif robottu



Şekil 5.4.21 Son prototip keşif robottu

Bu bitirme tezinin bu uygulamasında kullanılan tüm C++ ve C# tabanlı yazılımlar ilerleyen bölümler olan ekler kısmında verilmiştir.

BÖLÜM 6

KABLOSUZ MULTİ PLATFORM HASTA İZLEME SİSTEMİ

Bu bitirme tezinin ikinci uygulamasında ise hastane odalarına konabilecek bir sistem tasarlanmıştır. Bu tasarımda hastaya ait bütün biyomedikal verilerin uzak bir sistemde depolanması, hastaya ait bu verilerin sınır değerlerini aştığında ilgili personeli uyarabilicek ve hastaya ait verilerin uzak bir bilgisayar, tablet veya telefon üzerinden sağlık personeli tarafından görülebilmesini sağlayacak multiplatform programlar yazılmıştır. Bu programlar sayesinde sağlık personeli hastalarındaki bilgilere uzaktan erişebilicek, hastalara ait notlar okunabilicek ve sisteme yüklenen röntgen gibi veriler görüntülenebilecektir. Bu uygulamada ise bunları yapılabilecek tüm programlar ve elektronik devreler kurulmuştur.

Kablosuz hasta takip sistemimi için 3 ayrı birim oluşturulmuştur. Bunlar sırası ile biyomedikal sensörlerin bağlı olduğu hasta üzerindeki elektronik sistem kısmı, sunucu modülü kısmı ve istemci modülü kısmıdır.

Bu bitirme tezinin ilerleyen kısımlarında, EKG, Pulse oksimetre, sistemimizde kullanılan elektronik gömülü sistem, sunucu ve monitör birimleri anlatılacaktır.

6.1 Elektrokardiyografi

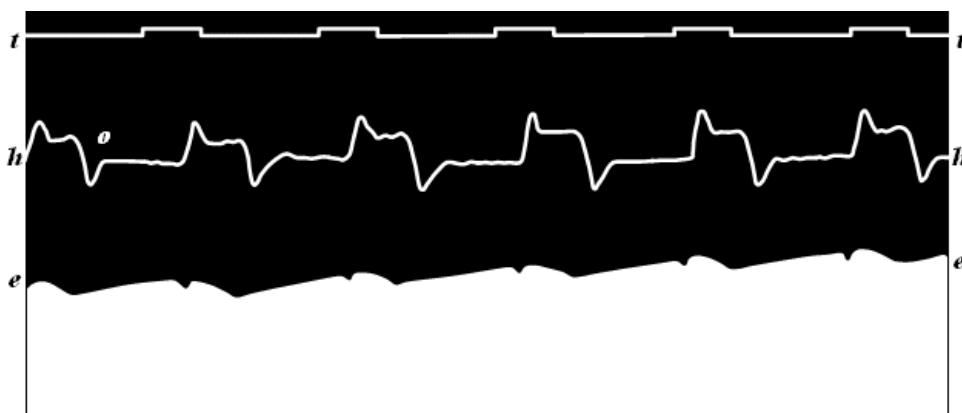
Elektrokardiyografi (EKG), sağlık sektöründe kalp ile ilgili problemleri tespit etmede kullanılan bir cihazdır[60]. EKG, kalbin elektriksel aktivitesini ve bu elektriksel değişiklikleri kaydetmeye yönelik bir yöntemdir. Bu elektriksel aktivite kollara, bacaklara ve göğüste kalbe yakın noktalara yerleştirilen elektrotlar vasıtası ile ölçülmektedir ve bu elektrotlar EKG cihazına kablo ile bağlıdır.

6.1.2. Tarihsel Gelişimi

EKG cihazları, hastaların kalbindeki elektriksel sinyalleri incelemek için geliştirilmiştir. EKG cihazından alınan bu sinyaller vasıtası ile hastanın kalbindeki problemler açığa çıkartılabilmektedir. EKG'nin tarihine bakıldığında, 17. ve 18. yüzyıllarda elektrik bulunmuş ve hayvanlar üzerinde bu elektriğin etkileri araştırılmıştır. 19. yüzyıla gelindiğinde ise insan kalbinde de ufak voltajlarda gerilim tespit eden cihaz yapılmıştır. İlk kez doğru bir şekilde EKG sinyallerinin alınması ise 20. yüzyılda kaydedilmiştir.

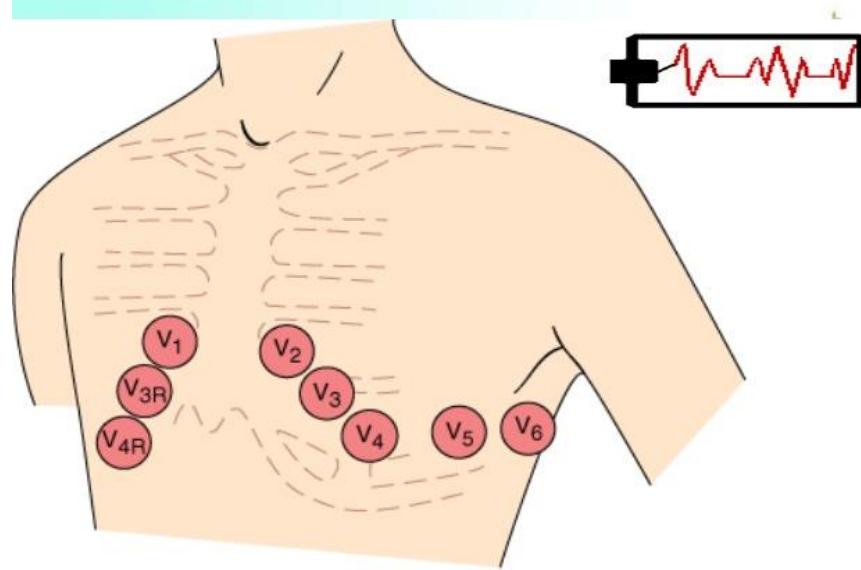
EKG cihazlarının tarihsel gelişimi şu şekildedir:

- Floransa'da fizik profesörü olan Leopoldo Nobilli Astatil Galvanometre'yi icat etmiştir. İcat edilen bu galvanometer ile 1827 yılında kurbağanın vücutundan akan elektrik akımını tespit edilmiştir[61].
- 1838 yılına gelindiğinde, Pisa üniversitesi fizik profesörü Carlo matteucci ve öğrencisi her bir kalp atımıyla eş zamanlı olarak bir elektrik akımının oluştuğunu göstermişlerdir.
- 1856 yılında, Fransız fizikçi Gabriel Lippmann Kılcal damar Elektrometre'yi icat etmiştir. Bu icadını 1876 yılında Marley, kurbağalar üzerinde deneyen kurbağanın kalbindeki elektriksel aktiviteyi kaydetmeyi başarmıştır.
- 1878 yılında ise İngiliz fizyologlar John burder Sanderson ve Frederic Page kalbin elektrik akımını aynı yöntem ile ölçerek bunun iki fazdan olduğunu açığa çıkartmışlardır ve bu fazlara QRS ve T ismini vermişlerdir.
- 1887 yılında, İngiliz fizyolog Augutus D.Walker insanın ilk elektrokardiyografisini yayınlamıştır.



Şekil 6.1 İlk insan kardiyografisi

- 1928 yılında, Ernsthine ve Levine EKG sinyalinin yükseltilmesinde vakum tüpünü kullanmışlardır. Bu araştırmalar sonucunda Frank Sanborn'un şirketi EKG cihazının 23 kilo ve 6V ile çalışan portatif sürümünü geliştirmiştir.
- 1938 yılında, Amerikan kalp vakfı ve İngiliz kardiyoloji vakfı kalbin elektriksel aktivitesini ölçen elektrotların göğüste bağlantı yerlerinin standartını belirtmişlerdir. (V1, V2, V3, V4, V5, V6). Aşağıdaki Şekilde örnek bağlantı yerleri gösterilebilir.



Source: Fauci AS, Kasper DL, Braunwald E, Hauser SL, Longo DL, Jameson JL, Loscalzo J: *Harrison's Principles of Internal Medicine*, 17th Edition: <http://www.accessmedicine.com>

Copyright © The McGraw-Hill Companies, Inc. All rights reserved.

6

Şekil 6.2 Standart göğüs derivasyon noktaları

- 1948 yılında, isveçli mühendisler Rune Elmqvist fizyolojik sinyalleri yazdırma için ilk kez mürekkepli bir yazıcı geliştirmiştir. Bu yazıcı 1950 yılında EKG sinyallerinin yazımında ilk kez kullanılmıştır.
- 1999 yılında, Tekşaslı araştırmacılar, 12 kanallı EKG'nin kablosuz bağlantı üzerinden bilgisayarla aktarılabilceğini ve kardiyologların bu verileri eksiksiz bir şekilde görüntüleyip yorumlayabileceğini göstermişlerdir.

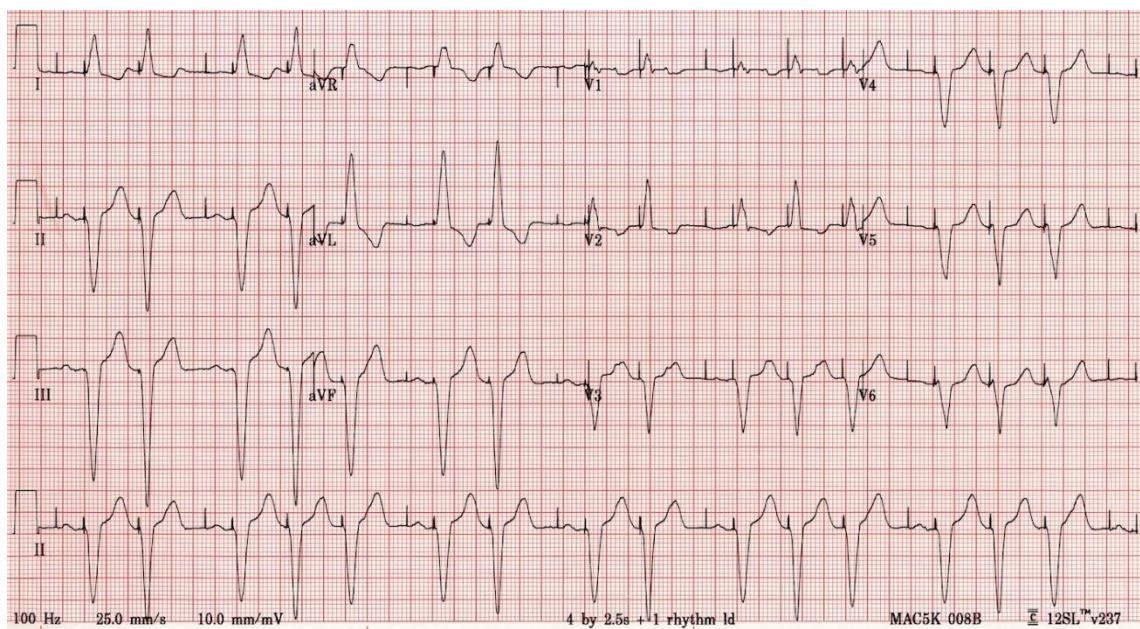


Şekil 6.3 EKG cihazı

6.1.3 Elektrokardiyogram İşareti

Vücutta durgun halde bulunan hücrelerin dış yüzeylerinin herhangi bir kısmı uyarılınca, o noktadan itibaren hücrenin içi de dışı da negatifleşir. Buna depolarizasyon denir[62]. Bu durumda hücrenin bir tarafı negatifleştiği için pozitif kısımdan buraya doğru bir akım akacaktır. Bu akıma depolarizasyon akımı denilmektedir. Bu kutuplaşma durumunun geri dönüşümüne repolarizasyon denir. Bu durumda akan akıma da repolarizasyon akımı denir.

Kalpte bu şekilde oluşan depolarizasyon ve repolarizasyon akımları vücuduma yayılır. Yayılan bu akımlar EKG cihazına bağlı elektrotlar ile bunları algılamak ve kaydetmek mümkündür. Kaydedilen bu polarizasyonlara Elektrokardiyogram denir [63]. Şekil 2.2'de örnek bir elektrokardiyogram işaretini görülmektedir.



Şekil 6.4 Örnek EKG grafiği

Şekil 2.2' de gösterilen harfler değişik bölgeleri temsil etmektedir. Atriumların kasılması P dalgası olarak temsil edilir. His demetimin iletimi P-Q aralığında, ventriküllerin depolarize olması QRS aralığında, ventrikül hücrelerinin yavaş bir şekilde repolarize olması ST aralığında olur. Örneğin sağlıklı bir kimsede kalp vuruş hızı dakikada 75 ise P, PR, QRS sırası ile 0,1 ms, 0,13 ms, 0,08 ms'dır [64].

6.2 Pulse Oksimetresi

Pulse oksimetre, hastanın kanında mevcut oksijeni ve beraberinde nabız hızını ölçen küçük bir cihazdır[65]. Pulse Oksimetre sürekli ve anlık arteriyal hemoglobin yoğunluğunu gösterir. Kanda ölçülen bu oksijen oranı sağlık personelinin teşhislerinde fayda sağlamaktadır. Dolayısıyla bu verilerin kolayca, hızlı ve kesin olarak elde edilmesi gereklidir. Bu verileri elde etmenin iki genel yöntemi vardır: Bunlardan birincisi, zor, acı veren ve gerekli netlikte sonuç vermeyen, klasik kan alma yöntemidir. Bu bildiğiniz gibi uygun iğnenin, doğru damara, doğru yerde batırılarak kan alınmasına dayanır. Aksi halde doğacak olumsuzlukları hepimiz biliriz. İkinci yöntem ise bir oksimetre kullanmaktır.

Pulse oksimetreler çok çeşitli durumlarda kullanılabilir ama anestezi sırasında oksijenin ve nabız atışının gösteriminde kullanılması en önemli özelliklerindendir. Bunlar iyileşme safhalarında da oldukça kullanılır. Oksijen saturasyonu %95'in üstünde olmalıdır. Uzun süre yoğun bakımda kalan hastalarda, solunum hastalıkları olan hastalarda veya doğuştan kalp hastası olanlarda değerler daha düşüktür ve temelde yatan hastaların hastalık şiddetini yansıtır.

Sonuç olarak Oksimetrenin anestezide ve cerrahi müdahalelerde hasta görüntülemedeki vazgeçilmezliği yıllarca sürecekdir. Bu kadar karmaşık bir teknolojinin böyle küçük bir aletle kullanılabilmesi büyük avantaj sağlar.



Şekil 6.5 Pulse oksimetre cihazı

6.2.1 Pulse Oksimetrenin Çalışma Prensibi

Arteriyal kandaki O₂ satürasyonunu noninvaziv olarak, oksimetri ve pletismografi prensiplerinin kombinasyonu ile ölçümüdür. 1930'lu yıllarda bu yana bilinen bu yöntem, 1970'li yılların sonlarına doğru cihazların geliştirilmesi ile günümüzde anestezi ve yoğun bakımda vazgeçilmez nesneler haline gelmiştir. Bir ışık kaynağı ve ışık dedektöründen oluşan sensörün arasına parmak ucu, kulak memesi gibi iyi perfüze olan dokuların yerleştirilmesi ile ölçüm yapılabilir. Kapiller dolaşımında arteriyoller pulsasyon olması dolayısıyla bu yönteme "pulse oksimetri" adı verilmiştir. Oksimetredede temel kural, oksijene ve redükte hemoglobinin ayırt edilmesidir. Bu ayrılmız kırmızı ve kızılıotesi işinlerin absorbсиyon oranının bir mikroprosesör yardımıyla analiz edilmesi sonucunda noninvaziv olarak pulse eden arteriyel oksijen satürasyonu (SpO₂) ölçülebilir. Pulse etmeyen venöz kan ve dokulardan, arteriyel pulsasyonun ayırt edilebilmesi için pletismografi yöntemi kullanılır[66].

Ölçüm; pletismografi ve oksimetri denilen iki prensibin kombinasyonu ile sağlanır. Bir puls oksimetresi vücudun bir bölgesine (genellikle bir parmağa bununla birlikte kalın yapay tırnakların ya da derin koyu renkte tırnak ojesinin olması durumunda ayak parmağına ve kulak memesine) gönderilen ışık dalgasını kullanır. ışık dalgası kanın oksijenli olup olmadığını saptamak için onun rengini kullanır. Kırmızı kan hücrelerinin yeteri kadar oksijenli olup olmamasına bağlı olarak kan rengimiz değişir.

İki dalga boyundaki (650nm ve 805nm) probe'dan bir ışık kaynağı çıkar. ışık, hemoglobin tarafından doymuş ya da doymamış olmasına bağlı olarak farklı miktarlarda kısmen soğurulur. İşlemci iki dalga boyundaki soğurmayı hesaplarken oksijenlenmiş hemoglobinin oranını da hesaplayabilir. Oksimetre nabız akışına bağımlıdır ve akış özelliğinin grafiğini gösterir. Akışın yavaş olduğu yerde oksimetre çalışmamayabilir. Bol oksijenli kan genellikle parlak kırmızı renktedir ve oksimetreden gönderilen ışığın çoğunu soğurur. Az oksijenli kan daha koyu kırmızı-mor renktedir ve ışık dalgasını soğurmaz. Oksimetrenin bir diğer etkileyici yönü ise venous kan seviyeleri ve atardamar kan seviyeleri arasında ayrılmış yapabilme yeteneğidir. Bu muhteşem teknolojinin mekaniği onun titreyen venous kanın ve atardamarla taşınan kanın sabit akışı arasındaki ayırmayı yapmasını sağlar. Pulse oksimetre kullanılarak elde edilen oksijen seviyesi ölçümleri, daha girişken kan alımı ile elde edilen ölçümlere son derece yakındır.

6.2.2 Pulse Oksimetrenin Kullanımını Etkileyen Nedenler

- çevre ışınının etkisi
- titreme,atım hızı
- düşük saturasyon değeri
- ritim,kas,kardiyak değişimler
- Anormal hemoglobinler
- hareket
- aşırı killi ve üzeri boyanmış deri

6.3 Hasta Üzerindeki Kısım

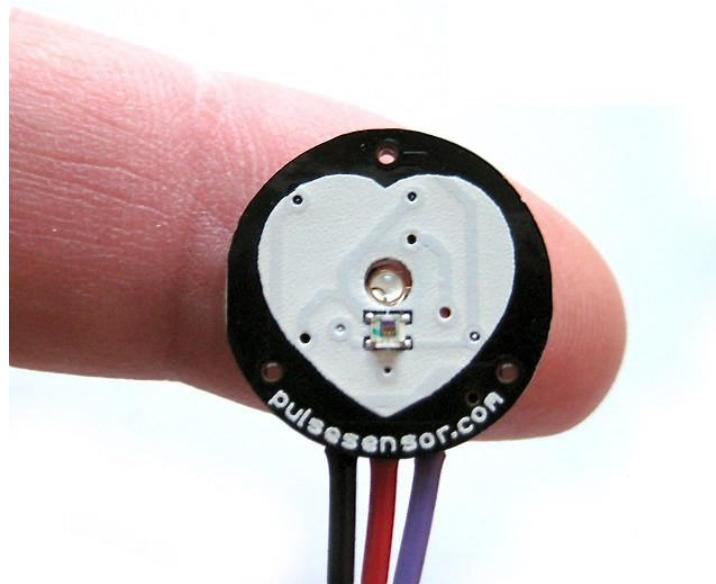
Biyomedikal sensörlerle bağlı bulunan Arduino Nano işlemcisi ve ESP8266'dan oluşan gömülü sistem, bu sensörlerin verdiği analog ve dijital sinyalleri okuyarak istenilen süre aralığına göre bu verileri sunucuya göndermek üzere programlanmıştır ve hasta için gerekli olan alt ve üst limitler aşıldığında ilgili birimi uyarabilmektedir. Bu tezde gömülü sistem, her 5 saniyede bir hasta verilerini ve odaya yerlestirdiğimiz sensör verilerini sunucuya gönderir ve bu sunucuda veriler veri tabanında ilgili tablolara kaydedilir. Veri aktarım süresi 200 ms'ye kadar düşürülebilmektedir. Böylece hastanın anlık verisi platform farkı gözetmeksizin hem hasta yakını hem de doktor tarafından takip edilebilmektedir.

Arduino Nano bu gömülü sistemin beynidir. Hastaya bağlı sensörleri ve ortama ait sensörleri okuyup göndermek için gömülü sistemde Arduino ATmega328P mikrodenetleyicisi bulunmaktadır.

6.3.1 Hasta Üzerindeki Sensörler

Bu uygulamada hasta üzerine Şekil bir tane pulse oksimetre sensörü ve bir adet şekilde görülen metal ds18b20 sıcaklık sensörü kullanılmıştır[67][68]. Bu sensörlerden hariç olarak bir adet yapay EKG işaretini Arduino üzerinde oluşturmuştur. Arduino ayrıca hasta odasının nemini, sıcaklığını ve aydınlığını ölçmek için aşağıdaki şekillerde gösterilen sensörlere de sahiptir. Bütün bu sensörler kablosuz sistemi yapabilmek için deneme amaçlı kullanılmıştır.

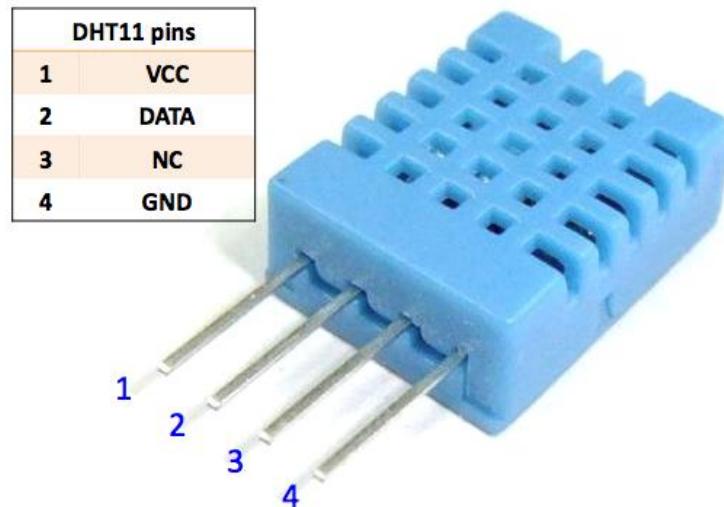
Arduino Nano bu gömülü sistemin beynidir. Hastaya bağlı sensörleri ve ortama ait sensörleri okuyup göndermek için gömülü sistemde Arduino ATmega328P mikrodenetleyicisi bulunmaktadır.



Şekil 6.6 Pulse Amped Sensor



Şekil 6.7 ds18b20 sıcaklık sensörü



Şekil 6.8 DHT11 ısı ve nem sensörü [69]

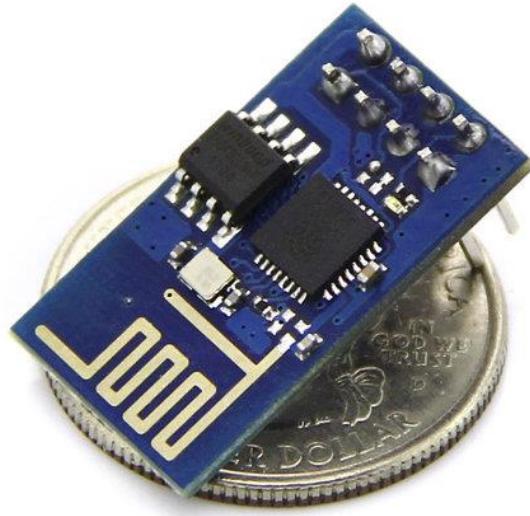


Şekil 6.9 LDR ışık sensorü

6.3.1 Hasta Üzerindeki Gömülü Sistem

Arduino mikrodenetleyicisi hastaya takılı olan ve hastanın odasındaki sıcaklığı, nemi ve oksijeni ölçen sensörleri okuyan kısımdır. Bu sensörler analog sinyallerdir ve bu sinyalleri Arduino Nano'da bulunan ATmega328p işlemcisinin 10bit ADC bloğu sayesinde okunabilmektedir. Bu okunan verilerin hastane sunucusuna yollandanabilmesi için öncelikle Arduino'nun ESP8266 Wi-Fi modül aracılığı ile hastane sistemine bağlanması gerekmektedir[70].

Şekilde görülen ESP8266 modülü nesnelerin interneti için üretilmiş içinde TCP/IP protokolü barındıran ARM mikroişlemcili bir 2.4Ghz Wi-Fi modüldür. Bu modül sayesinde UART destekli Arduino veya başka bir mikrodenetleyici verilerini internete açabilmektedir. Bu sayede sensör verilerinin uzak bir sistemde depolanıp görüntülenebilmesi mümkün hale getirilmiştir.



Şekil 6.10 ESP Wi-Fi modül

ESP8266 modülü üzerindeki ARM mikroişlemci modülü programlanabilirdir ve amaca yönelik olarak programlanabilmektedir. Bu bitirme tezi uygulamasında AT komut satırı kullanılarak ESP8266 modüle UART üzerinden komutlar yollandı ve gerekli hastane sunucusuna bağlanarak gerekli sensör verilerini aktarılması sağlanmıştır.

ESP8266 modüle ait özellikler tablosu aşağıdaki Tablo 6.1 de görülebilir.

Tablo 6.1 ESP8266 teknik özellikler

Entegre 32 bit RISC mimarili düşük güçlü mikroişlemci
Entegre TCP / IP protokol yığını, Quality of Service Yönetimi
Wi-Fi Direct (P2P), SoftAP, 802.11 b / g / n Desteği
SDIO 2.0, SPI, UART, STBC, 1 × 1 MIMO, 2 × 1 MIMO
802.11b modunda, +19.5dBm çıkış gücü

Entegre WEP, TKIP, AES, ve WAPI şifreleme bileşenleri
Bekleme durumunda güç tüketimi <1.0mW

Yukardaki tablodan da görülebileceği üzere ESP8266 içerisinde 32 bitlik RISC mimarisine sahip ARM mikroişlemci barındıran düşük güçlü bir modüldür. Bu modülü güçlü kılan içinde bulundurduğu TCP/IP stacktir. Bu sayede aynı bir modem gibi çalışarak verileri internete açabilir ve uzak istemcilerle bağlantı kurar. Bu Wi-Fi modül üzerinde pcb ye döşenmiş anten bulunmaktadır ve 2.4 Ghz hızında iken QoS 802.11 b/g/n modlarını desteklemektedir. Ayrıca bu modülünde kendine ait GPIO pinleri, ADC bloğu, PWM ,UART ve SPI gibi özellikleri vardır.

Bu Wi-Fi modül bu uygulamada AT komut takımı ile çalışmaktadır ve aşağıdaki Tablo 6.2 de örnek AT komutları ve açıklamaları görülebilir.

Tablo 6.2 AT komut takımı

AT KOMUTU	İLGİLİ KOMUTUN AÇIKLAMASI
AT+RST	Modülü resetlemeye yarar.
AT+CWMODE= <Mode>	Modülü Client, Access Point veya both moduna almayı sağlar. 1=Client, 2=AP, 3= Client&AP
AT+CWLAP	Etraftaki Kablosuz ağların listesini görmeyi sağlar.
AT+CWJAP= <SSID>,<PWD>	Modülü belirtilen SSID'e ve şifreli ağa bağlatır.
AT+CIFSR	Modülün Access Point'den ip almasını sağlar.
AT+CIPSTART=<TYPE>,<URL>,<PORT>	Modülün UDP veya TCP bağlantı kurmasını sağlar.
AT+CIPSEND=<LEN>	Modülün bağlanılan servere veri yollamasını sağlar.

AT+CIPCLOSE	UDP veya TCP bağlantıyı kapatır.
-------------	----------------------------------

Arduino yapay olarak oluşturulan EKG işaretini ve Arduino pulse oksimetre sensöründen aldığı verileri 2 ms de bir örnekler ve örneklediği dataları ESP modüle UART üzerinden yollar. Bu örneklenen dataların UART teknigi üzerinden gönderilmesi için öncelikle Arduino'nun hastane sunucusuna bağlanması ve bu verilerin bufferlanması gerekmektedir. Arduino'nun hastane sunucusuna bağlanması için aşağıdaki C++ tabanlı kodlar yazılıp UART üzerinden ESP modüle gönderilmiştir.

```
void setup() {  
    pinMode(blinkPin,OUTPUT);  
    Serial.begin(115200); //UART 115200 baudrate/saniye hızında  
    interruptSetup();  
    sendCommand("AT+RST\r\n", 2000, DEBUG); // modüle reset at  
    sendCommand("AT+CWMODE=1\r\n", 1000, DEBUG); // modulu ap moduna al  
    sendCommand("AT+CWJAP=\"HastaneSunucu\",\"1234567890.\\"\r\n",3000,  
    DEBUG);  
    delay(5000); //Baglanana kadar 5 sn bekle  
    sendCommand("AT+CIFSR\r\n", 1000, DEBUG); //IP adresi al  
    sendCommand("AT+CIPMUX=1\r\n", 1000, DEBUG); //çoklu bağlantı modu
```

Yukarda gözlenen fonksiyon Arduino'nun setup() fonksiyonudur ve genellikle bu fonksiyonda Atmega328p mikroişlemcisinin ana ayarları yapılır. Bu fonksiyon sadece Arduino resetlenince ve ilk kez güç verildiğinde çalışmaktadır. Bu fonksiyonda Arduino'nun hastane sunucusuna bağlanabilmesi için öncelikle ESP modülün resetlenmesi gerekmektedir ve “AT+RST” komutu ile modüle reset attırılmıştır.

ESP8266 hem client modunda hem AP modunda hemde bu iki mod birlikte çalışabilmektedir. Bu uygulamada hastane kablosuz ağına bağlanabilmesi için “AT+CWMODE=1” komutu kullanılmıştır. Ardından ise bağlanılmak istenen kablosuz ağın adı ve şifresi UART üzerinden ESP modüle gönderilmiş ve bağlanana kadar

Arduino'nun 5 saniye beklenmesi sağlanmıştır. Bu süre sonunda aşağıdaki komutlarla kablosuz ağdan alınan IP adresinin bilgisi ESP modülden alınmıştır.

```
long int time = millis();

while( (time+1000) > millis())  {

    while(Serial.available())  {

        char c = Serial.read(); // read the next character.

        ipadresi+=c;      }      }

ilkindex = ipadresi.indexOf('CIFSR:STAIP,');

ikincindex = ipadresi.indexOf('\r\n+CIFSR:STAMAC+', ilkindex +1);

String ilkdeger = ipadresi.substring(0, ilkindex);

String ikincideger = ipadresi.substring(ilkindex +1, ikincindex);

ipadresi= ikincideger;
```

Kablosuz ağdan ip adresi elde edildikten sonra Arduino'nun kendisine atanın hasta bilgilerini EEPROM'una kaydetmesi için öncelikle hastane sunucuna bir bağlantı isteği ile birlikte kendi id'sini yollamıştır. Bu istek karşısında ise kendisine 5 saniye içinde hasta bilgileri gönderilmesi için sunucu programları programlanmıştır. Bu sayede ip adresi alındıktan 5 saniye sonra Arduino kişi bilgileri için sunucudan yanıt beklemektedir. Bu işlem için aşağıdaki kodlar yazılmıştır.

```

String cmd="AT+CIPSTART=3";
cmd += "\",\" + TARGET_TYPE + "\",\"" + TARGET_ADDR + "\";";
cmd += ",1235";
Serial.println(cmd);
delay(3000);
String komut;
komut+="+";
komut+=rfid;
String cmdSEND_length = "AT+CIPSEND=";
cmdSEND_length += TARGET_ID1 + "," + rfid.length() + "\r\n";
Serial.println(cmdSEND_length);
Serial.println(cmd);
Serial.println("waiting >");
if(!Serial.available()){
    if(Serial.find(">")){
        Serial.println("> received");
        Serial.println( rfid );
        Serial.println( rfid );
    }
    else {
        Serial.println("> algılanmadı, hata var!");
    }
    long int time2 = millis();
    while( (time2+5000) > millis())
    {
        while(Serial.available())
        {
            char c = Serial.read();
            otaverisi+=c;
        }
    }
    Serial.println("Gelen Veri = "+otaverisi);
    int commaIndex2 = otaverisi.indexOf(':');
    int secondCommaIndex2 =otaverisi.indexOf('\r\n'+CIFSR:STAMAC+',',
commaIndex2+1);
    String secondValue2 = otaverisi.substring(commaIndex2+1, secondCommaIndex2);
    kimlik=secondValue2 ;
    Serial.println("Gelen Veri Sonu");
    interruptSetup();
}

```

Kablosuz ağdan kişi bilgileri alındıktan sonra aşağıdaki kodlar sayesinde Arduino'nun pulse oksimetre ve yapay ekg sinyalini örneklemesi için gereken 2 ms sürenin timer kesmeleri ile oluşturulması sağlanmıştır.

```

void interruptSetup() {
    TCCR2A = 0x02;
    TCCR2B = 0x06;
    OCR2A = 0X7C;
    TIMSK2 = 0x02;
    sei();      }

```

Kesme fonksiyonunda ayarlarda tamamlandıktan sonra Arduino hastane sunucuna bağlanır ve kendisine atanın hastanın bilgilerini alıp EEPROM hafızasına kaydeder.

Bundan sonra ise Arduino sensör verilerini 2 milisaniyede bir örneklemeye başlar. Arduino sensör datalarını dizi halinde önbelleğe alır ve veri gönderme zamanı geldiğinde bu veriyi UART üzerinden haberleştiği ESP8266 ile sunucuya gönderir. Ayrıca LCD ekran bağlantısı ile bu verileri de bu ekranda göstermektedir.

Sensör verileri aşağıda yazılan komutlar aracılığı ile bufferlanmıştır.

```

sensor += "*";           //substring parçalama belirteci
sensor += deneme;        //yapay ekg sinyali bufferi
sensor += "*";           //substring parçalama belirteci
sensor += ipadresi;      //modülün ip adresi
sensor += "*";           //substring parçalama belirteci
sensor += "0000000000";   //tc kimlik nosu
sensor += "*";           //substring parçalama belirteci
sensor += kimlik;         //kimlik bilgileri,hasta tanisi, yaș
sensor += "*";           //substring parçalama belirteci
sensor += BPM;            //ortalama kalp atimi
sensor += "*";           //substring parçalama belirteci
sensor += oksijen;        //pulse oksimetre bufferi
sensor += "*";           //substring parçalama belirteci
sensor += tansiyon;       //ortalama tansiyon
sensor += "*";           //substring parçalama belirteci
sensor += hastaderece;   //ortalama hasta sicakligi
sensor += "*";           //substring parçalama belirteci
sensor += sicaklik;       //ortalama oda hava sicakligi
sensor += "*";           //substring parçalama belirteci
sensor += nem;             //ortalama oda nemi
sensor += "*";           //substring parçalama belirteci
sensor += odaaydinlik;    //ortalama oda aydinligi
sensor += "*";           //substring parçalama belirteci
sensor += ekgverisi;      //substring parçalama belirteci

```

```

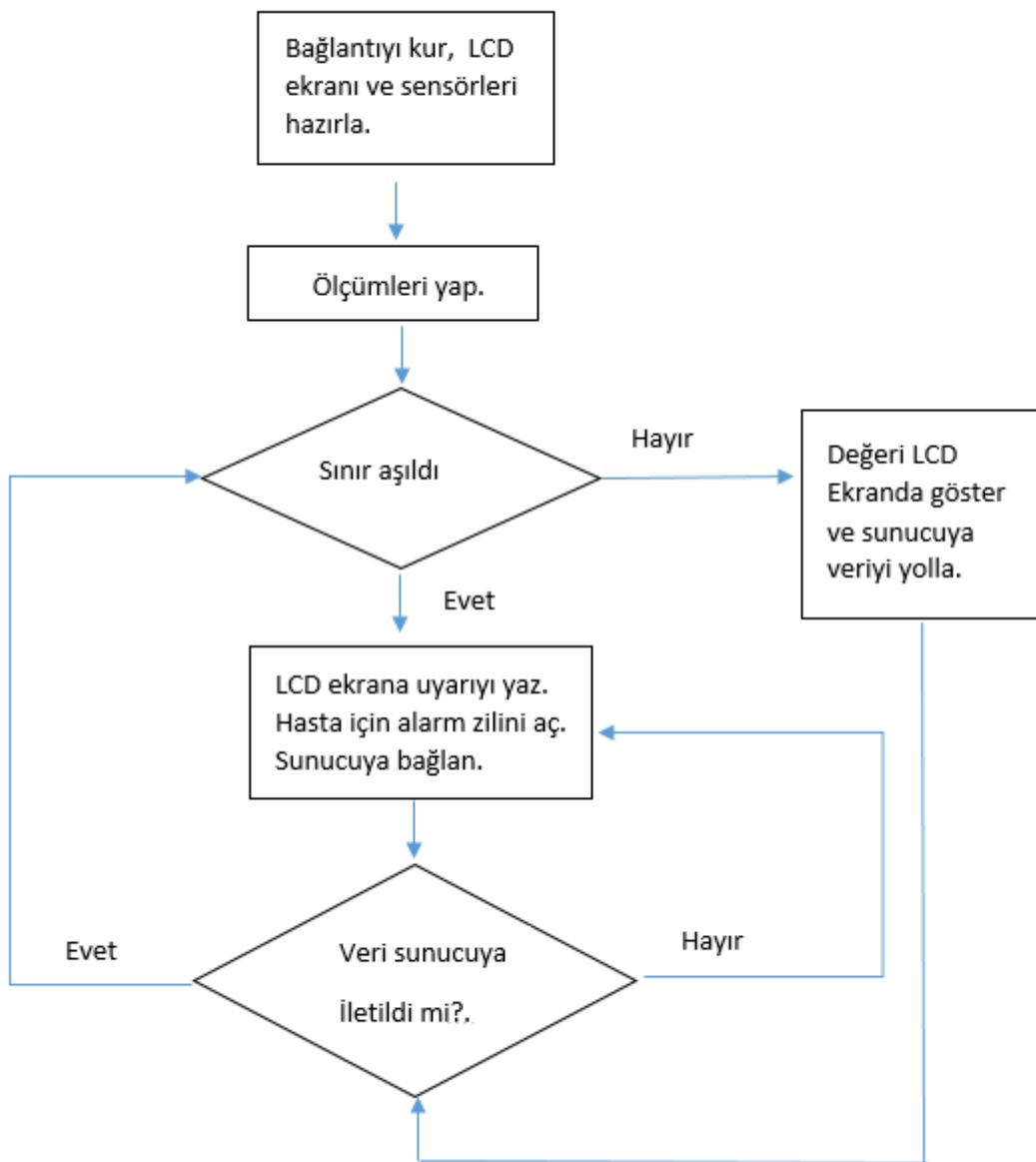
String cmd="AT+CIPSTART=" + TARGET_ID;
cmd += ",\" + TARGET_TYPE + "\",\"" + TARGET_ADDR + "\"";
cmd += "," + TARGET_PORT;
Serial.println(cmd);
oksijen=analogRead(2);
oksijen=map(oksijen,0,1023,0,100);
delay(10);
String cmdSEND_length = "AT+CIPSEND=";
cmdSEND_length += TARGET_ID + "," + sensor.length() +"\r\n";
Serial.print(cmdSEND_length);
Serial.println(cmdSEND_length);
Serial.println(cmd);
Serial.println("waiting >");
if(!Serial3.available());
    if(Serial3.find(">")){
        Serial.println( sensor );
        Serial.println( sensor );
        sensor="";
    }
else {
    Serial.println("> algılanmadı, hata var");
}
deneme="";
}

```

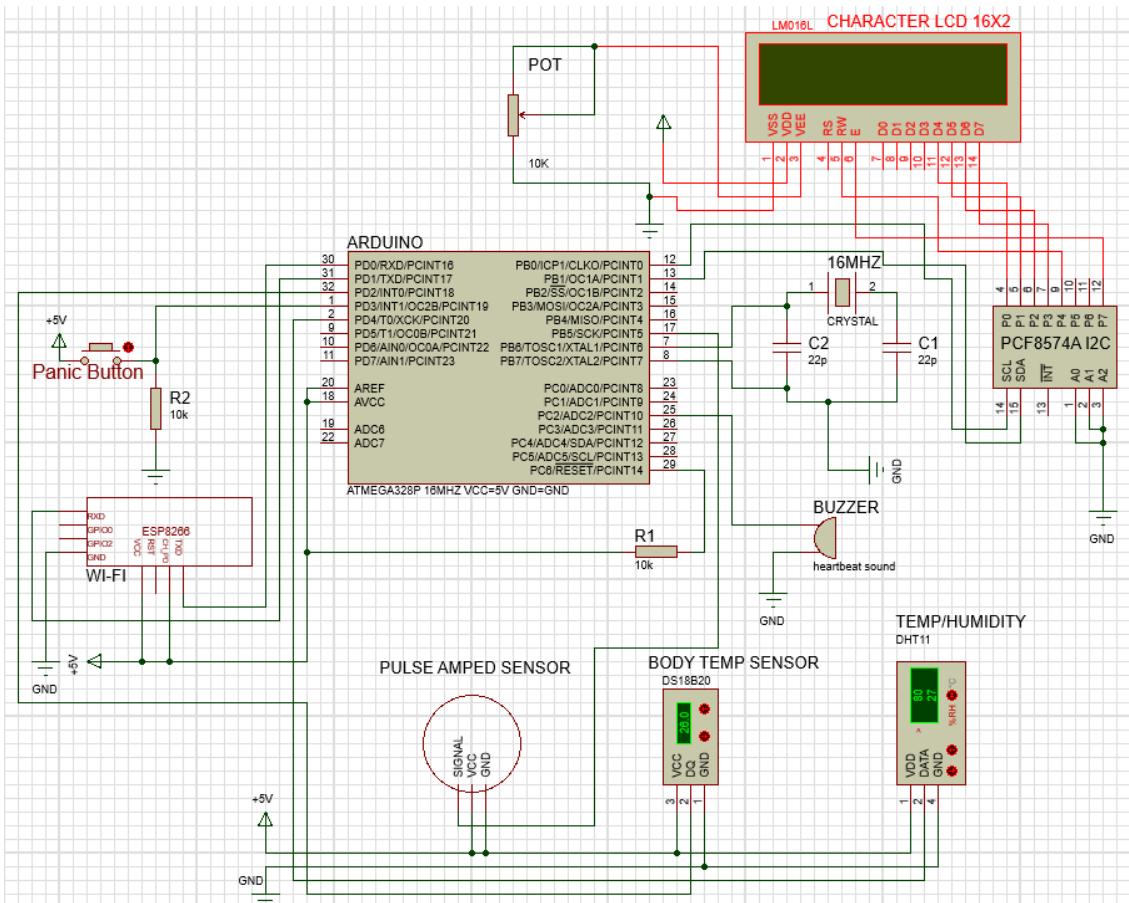
Yukarda yazılan kodlar sayesinde Arduino'nun örneklediği veriler birer önbölirteç vasıtası ile bir String dizisine bufferlenmiş ve bu bufferlanan veriler timeout süresi sunucu tarafından belirlenen bir süre sonunda UART haberleşme tekniği üzerinden saniyede 115200 baudrate hızında ESP modüle gönderilmiştir.

UART üzerinden komutları alan ESP modül bu sensör verilerini ilgili sunucu adresine yollamıştır.

Bu gömülü sistem aşağıdaki algoritma göre çalışmaktadır ve bazı önemli örnek kodlar yukarıda verilmiştir. Ayrıca bu gömülü sistem için tasarlanan pin bağlantıları da şekil de görülebilir.



Şekil 6.11 Elektronik sistem çalışma prensibi

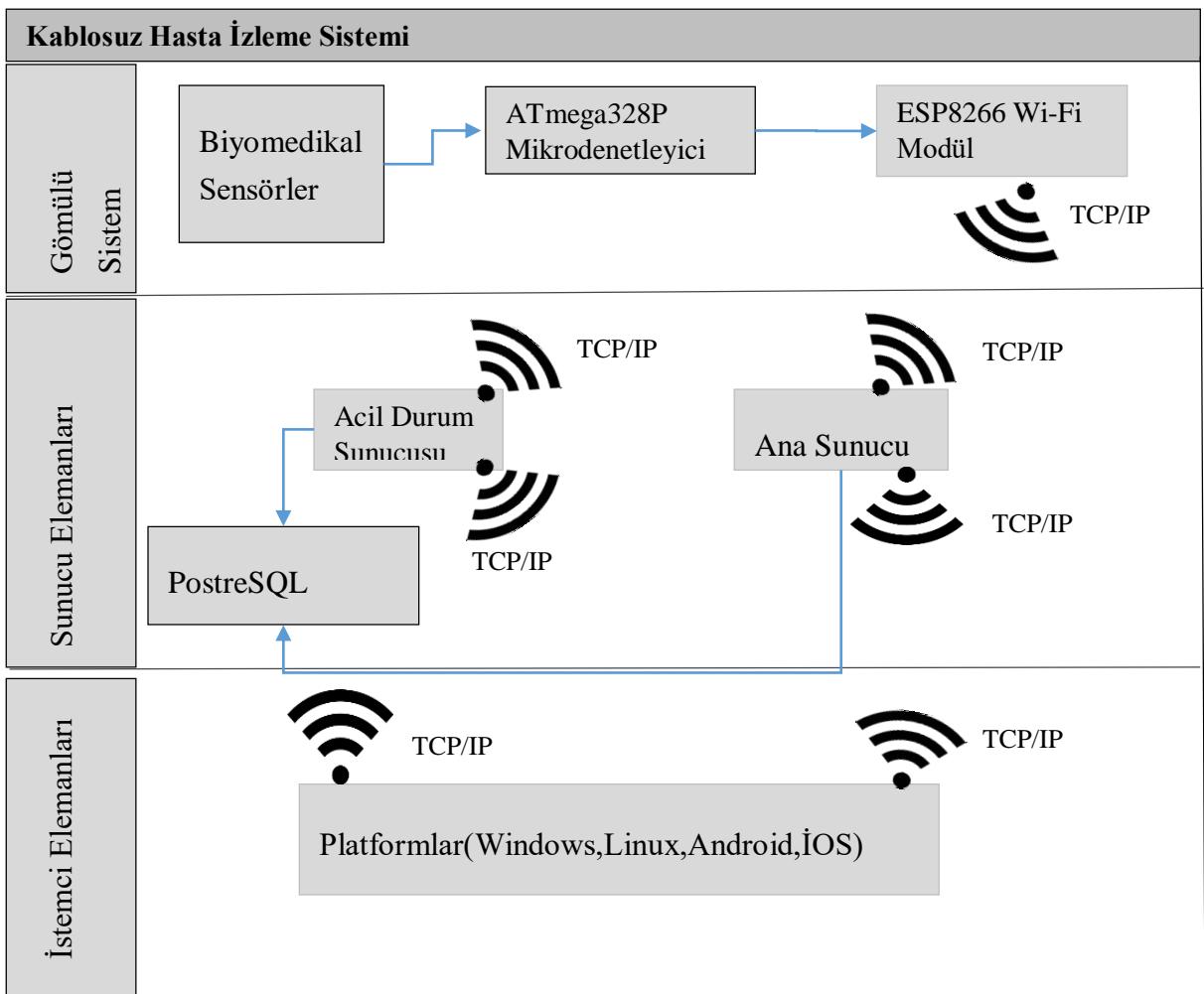


Şekil 6.11 Elektronik sistem pin bağlantıları

6.4 Sunucu Birimi

Hasta izleme sisteminizde ara katman görevini gören sunucu modülü istemciler ve hastaya bağlı gömülü sistem ile haberleşmeyi sağlamaktadır. Bu sunucu C++ tabanlı sunucu yazılımı, gömülü sistemde veri gönderme işlevini üstlenen ESP modülü ile haberleşmektedir. Acil durumda hastalar için, sisteme bağlı Online hastalar için ve tüm hastalar için gerekli sensör ağrı trafiği veritabanı işlemleri aracılığı ile bu C++ tabanlı sunucu programında yapılmaktadır. Sensörlerden gelen veriler veri tabanına bu sunucu modül üzerinden kaydedilmektedir. Aynı anda birden çok hastanın verisi ilgili veri tabanına kaydedilmektedir. ESP modüllerin göndermiş olduğu veriler bu programda yorumlanarak alınan ESP ID'sine göre ilgili tablolara kaydedilmektedir.

Aşağıda Şekilde tüm sistemin blok diyagramı verilmiştir. Bu şekilde birimler arasındaki bağlantılar görülebilir.



Şekil 6.12 Sunucu birimi blok diyagramı

Veri tabanı olarak bu uygulamada hızlı ve kararlı olmasından ötürü PostgreSQL tabanlı veri tabanı hizmeti kullanılmıştır. PostgreSQL veritabanı sunucusu network tabanlı bir sunucudur ve istemci uygulamaların bağlantı noktasıdır. Yani istemci uygulamalar hastalar için gerekli verileri bu veritabanına kaydedilen tablolardan almaktadır. Ana sunucu programı ise sensor ağındaki tüm ESP modüllerinin sensör verilerini ettiği programdır.

Bu sunucu sisteminin nasıl işlediği yazılımları ile birlikte açıklar isek, öncelikle sunucu programına TCP/IP kütüphaneleri eklenmiştir ve aşağıdaki kodlar sayesinde bir TCP sunucusu yaratılmıştır. Bu sunucuya bağlanan istemciler için yeni bir threat yani iş parçacığı oluşturularak diğer istemcilerin verilerinin birbiriyile karışmaması sağlanmıştır.

```

void sunucu::sunucuyu_baslat()
{
    int port = 1234;
    qDebug() << "////////////////KABLOSUZ HASTA TAKIP////////////////";
    qDebug() << "////////////////MURAT DEMIRTAS////////////////";
    qDebug() << "Program Baslatiliyor.";
    qDebug() << "TCP/IP Baglantisi icin soket acilmaya calisiliyor.";
    if(!this->listen(QHostAddress::Any, port)) //Egerki dinlenilecek port acilamadi
iseç
    {
        qDebug() << "Dinlemek uzere ayarlanilan tcp portu = " << port << "acilamadi";
        qDebug() << "Portun kullanilmadiginden emin olun ve programi tekrar
baslatin.";
    }
    else //port acildi ise
    {
        qDebug() << "Bu port TCP/IP baglantisi icin kullaniliyor = " << port ;
        qDebug() << "Port acildi ve Uzak cihazdan baglanti bekleniyor.";
    }
}

//Egerki yeni baglanti gelir ise bu baglantiya bir id atayacak fonksiyon ve threati
baslatacak bir fonk.oluşturalim.
void sunucu::incomingConnection(qint32 socketDescriptor)
{
    soket = new QTcpSocket;
    soket->setSocketDescriptor(socketDescriptor);

    QString ipadres = soket->peerAddress().toString();
    qDebug() << "baglanti adresi = " + ipadres;
    qDebug() << "Uzak bilgisayardan baglanti istegi geliyor";
    qDebug() << "Baglanti kuruluyor";

    qDebug() << "Bu cihaz icin bu = " << socketDescriptor << "id atandi ";

    //Yeni baglanan aygitlar icin idsine göre yeni yan parcacik yani threat olustur.
    threat *thread = new threat(socketDescriptor, this);
    //Baglanti kopumunda threati sonlandirmasi icin bir sinyal oluşturalim.
    connect(thread, SIGNAL(finished()), thread, SLOT(deleteLater()));
    connect(this,SIGNAL(ip(QString)),thread,SLOT(ipal(QString)));
    thread->start(); //Threadi baslat
}

```

```

threat::threat(qint32 ID, QObject *parent) :
    QThread(parent)
{
    this->socketDescriptor = ID;
    QSqlDatabase db;
    db = (QSqlDatabase::addDatabase("QPSQL"));
    db.setDatabaseName("onlinehastalar");
    db.setHostName("127.0.0.1");
    db.setUserName("hasta");
    db.setPassword("hasta");
    db.setPort(5555);
    if (!db.open()) {
        qDebug() << "Veritanina baglanilamadi. Nedeni :";
        qDebug() << db.lastError().databaseText();
    }
    else {
        qDebug() << "Veritabanina Baglanildi. ";
    }
}

```

Yukardaki kod ile online hastalar için açılan sunucu programının veritabanına bağlanabilmesi için veritabanı parametrelerinin QSqlDatabase nesnesi olan db ye girilmesi gösterilmektedir. Bu sayede threat programı veritabanına bağlanabilicektir. Diğer durumda bağlantı hatasının neden gerçekleşmediğini ekranda gösterecektir.

TCP/IP üzerinden bağlantı kuran ESP modülün veri gönderdiği an programın ilgili fonksiyona gitmesi için sinyal-yuva yapısı kullanılmıştır. Bu sayede veri geldiği an program kesmeye gidicektir.

```

connect(socket, SIGNAL(readyRead()), this, SLOT(readyRead()),
Qt::DirectConnection);

connect(socket, SIGNAL(disconnected()), this,
SLOT(disconnected()));

```

Bu aşamadan sonra ESP sensör verilerini yolladıktan sonra ilk önce readyRead() fonksiyonunda bu verilerin bir karakter dizisine ardından String dizisine dönüştürülüp parçalanması gerekmektedir. Çünkü birçok sensör değişkeni bir arada bufferlanmış şekilde gelmektedir.

```

void threat::readyRead() //Veriler okunmaya basladigin an calis
{
    QByteArray client_verisi;
    client_verisi=socket->readAll();
    qDebug() << client_verisi;
    qDebug() << "Sunucu/is parcacigi : Bu IP:"+client_ip+" Ad:"+ad+"
    Soyad:"+soyad+" Veri aliniyor";
}

```

Veriler bir karakter dizisine çevrilir ve kimden yani hangi sensörden veri geliyorsa deneme amaçlı ekrana yazdırılır. Gelen verinin ilk karakterine bakılır eğer sensör verisi parçalama işaretti ise işlemlere geçilir.

```

if(client_verisi[0]=='*') {
    hasta_verisi=client_verisi;
    gun = QDate::currentDate().toString();
    zaman = QTime::currentTime().toString();
    QString deneme=client_verisi;
    QStringList pieces = deneme.split( "*" );
    oksimetre=pieces.value( pieces.length() - 12);
    client_ip=pieces.value( pieces.length() - 11);
    tckimlikno=pieces.value( pieces.length() - 10);
    ota=pieces.value( pieces.length() - 9);
    nabiz =pieces.value( pieces.length() - 8);
    tansiyon=pieces.value( pieces.length() - 7);
    oksijen=pieces.value( pieces.length() - 6);
    hastasicaklik=pieces.value( pieces.length() - 5);
    odasicaklik=pieces.value( pieces.length() - 4);
    odanem=pieces.value( pieces.length() - 3);
    odaaydinlik=pieces.value((pieces.length()-2));
    ekgverisi=pieces.value( pieces.length() - 1);

    QStringList pieces2 = ota.split( "!" );
    ad=pieces2.value( pieces2.length() - 6);
    soyad=pieces2.value( pieces2.length() -5);
    yas=pieces2.value( pieces2.length() - 4);
    tanı=pieces2.value((pieces2.length()-3));
    lokasyon=pieces2.value( pieces2.length() - 2);

    QStringList pieces3 = client_ip.split( "\\" );
    client_ip=pieces3.value( pieces3.length() - 2);

    deneme="";
    debug(); //sensör verileri ekranda görüntülenmek istenirse
}

```

İlgili parçalama işaretini içeren veriler ilgili değişkenlere parçalanır ve böylece hastanın verilerine göre ilgili veritabanına sensör verilerini yollama aşamasına geçilir.

```

//////////////////VERITABANINA YOLLAMA KISMI/////////////////////////////
hasta = ad+soyad;
QString komut3;

komut += "INSERT INTO ";
komut += hasta;
komut +=

"(ip,gun,zaman,tckimlikno,ad,soyad,yas,tanı,lokasyon,nabiz,tansiyon,oksijen,ha
stasicaklik,odasicaklik,odanem,ekgverisi,oksimetre,odaaydinlik) VALUES (" ;
QString degerler=
"""+client_ip+"','"+""+gun+"','"+""+zaman+"','"+""+tckimlikno+"','"+""+ad+"'",
"+""+soyad+"','"+""+yas+"','"+""+tanı+"','"+""+lokasyon+"',";
QString degerler2=
"""+nabiz+"','"+""+tansiyon+"','"+""+oksijen+"','"+""+hastasicaklik+"','"+""+o
dasicaklik+"','"+""+odanem+"','"+""+ekgverisi+"','"+""+oksimetre+"','"+""+oda
ydinlik+"');"

```

Yukardaki kodlardan anlaşılabileceği üzere veritabanına gidecek veriler önce soruya bindirilir ve ardından ilgili işlemi gerçekleştirmek query işlemi kullanılır.

```
komut3=komut+degerler+degerler2;
qDebug() << "oluşturulan komut = "+komut3;
bool ok2 = query.exec( komut3);
if(!ok2){
    qDebug() << query.lastError().databaseText();
    qDebug() << "Sunucu/is parcacigi : Veriler Veritabanina aktarilirken
Hata Olustu:";
    // do something
}
else
    qDebug() << "Sunucu/is parcacigi : Veriler Veritabanina Basari ile
aktarildi";

komut="";
```

Bu aşamadan hastanın tc kimlik nosu ve adına göre ilgili sensör verileri tablolara aktarılır. Eğer veritabanında ilgili kişinin tabloları mevcut değil ise yeni bir tablo oluşturulur. Bu tablo oluşturma işlemi C++ dilinde aşağıdaki komutlar ile gerçekleştirilir.

```
qDebug() << "Sunucu/is parcacigi : Yeni Bir Veritabani Acilmaya
hazirlanıyor";
QSqlQuery query;    QString komut,tablo;      tablo=hastaadi;
komut += "CREATE TABLE IF NOT EXISTS ";
komut += tablo;
komut += "(no serial primary key, ";
komut += "ip VARCHAR(30), ";
komut += "gun VARCHAR(30), ";
komut += "zaman VARCHAR(30), ";
komut += "tckimlikno VARCHAR(30), ";
komut += "ad VARCHAR(30), ";
komut += "soyad VARCHAR(30), ";
komut += "yas VARCHAR(30), ";
komut += "tani VARCHAR(30), ";
komut += "lokasyon VARCHAR(30), ";
komut += "nabiz VARCHAR(30), ";
komut += "oksijen VARCHAR(30), ";
komut += "tansiyon VARCHAR(30), ";
komut += "hastasicaklik VARCHAR(30), ";
komut += "odasicaklik VARCHAR(30), ";
komut += "odenem VARCHAR(30), ";
komut += "ekgverisi VARCHAR(150)), ";
komut += "odaaydinlik VARCHAR(30))";

bool ok = query.exec( komut);
if(!ok) {
    qDebug() << "Hata!! Tablo olusturulamadi.";
    qDebug() << query.lastError().databaseText();
}
else {
    qDebug() << "Tablo olusturuıldı";
}}
```

Bu uygulamada yazılan sunucu programının kaynak kodları ilgili eklerde bulunabilir.

6.5 Monitör Birimi

Bu kablosuz hasta takip uygulamasından hastalara ait bilgilerin ve sensör verilerinin incelenmesi için bütün hem masaüstü hem de akıllı telefonlarda çalışabilen kullanıcı arayüzü, basit ve anlaşılır ve C++ tabanlı uygulama yazılmıştır. Bu uygulamanın hem PC hem tabletler hemde akıllı telefonlarda çalışabilmesi için çeşitli modifikasyonlar yapılmıştır. Bu uygulamada doktor hastanede veya evde yatan ve sisteme bağlanmış hastaların veya önceki hastaların listesini görebilmektedir. Doktor istediği hastayı bu listeden seçerek hastanın sisteme girişinin başlama tarihinden o anki zamana kadar olan tüm sonuçları görebilip yorumda bulunabilmektedir. Eğer hasta sisteme bağlı ise anlık sonuçları da görüntüleyebilmektedir. Bu uygulamada ayrıca başka birimlerce hastanın veri tabanına yüklenen tahlil sonuçları da örneğin(röntgen,emg) görüntülenemektedir. Bütün platformlar için aynı kod yazılmıştır. Bu kodların temel çalışma prensibleri şu şekildedir.

Öncelikle bu monitör uygulamalarında acil, online ve tüm hastalar için ayrı yarı belirtilen butonlara basıldığı zaman ilgili veritabanını açılması gerekmektedir. Bu nedenle öncelikle veritabanını acmamız gerekmektedir. Aşağıda yazılan C++ tabanlı kodlar ile veritabanı acılır.

```
QSqlDatabase db = QSqlDatabase::addDatabase("QPSQL");
db.setDatabaseName("onlinehastalar");
db.setHostName("127.0.0.1");
db.setUserName("hasta");
db.setPassword("hasta");
db.setPort(5555);
if (!db.open()) {
    qDebug() << "Databaseye Erisirken Hata Olustu";
}
else {
    qDebug() << "Online Hasta Veritabanina Baglanildi.";
```

Ardından ise görsel arayüzler için resource dosyasından gerekli resimler ve kalp sesi programda tanıtılır.

```
QPixmap pixmap(":/new/prefix1/resources/logo.png");
ui->label->setPixmap(pixmap);
ui->label->setMask(pixmap.mask());
ui->label->setScaledContents(true);
ui->label->show();
QPixmap pixmap5(":/new/prefix1/resources/call.png");
ui->label_23->setPixmap(pixmap5);
ui->label_23->setMask(pixmap5.mask());
ui->label_23->setScaledContents(true);
ui->label_23->show();
```

```

QPixmap pixmap3(":/new/prefix1/resources/derece.png");
ui->label_13->setPixmap(pixmap3);
ui->label_13->setMask(pixmap3.mask());
ui->label_13->setScaledContents(true);
ui->label_13->show();
QPixmap pixmap6(":/new/prefix1/resources/oksijen.png");
ui->label_22->setPixmap(pixmap6);
ui->label_22->setMask(pixmap6.mask());
ui->label_22->setScaledContents(true);
ui->label_22->show();
player = new QMediaPlayer;
player->setMedia(QUrl("qrc:/new/prefix1/resources/ses.wav"));
player->setVolume(100);

```



Şekil 6.13 Monitör birimi arayüzü

Ardından ise seçilen veritanabına göre hastaların isimleri listede gözükmür. Bunun için aşağıdaki kodlar yazılmıştır.

```

query = new QSqlQuery(); liste = new QSqlQueryModel();
QString listekomutu = "SELECT tablename FROM pg_tables WHERE schemaname
= 'public';";
bool ok = query->exec( listekomutu );
if(!ok) { qDebug() << query->lastError().databaseText(); }
else qDebug() << "sorgu basarili";
liste->setQuery(*query);
ui->listView->setModel(liste);

```

Bu kodlar aracılığı ile acil, online veya tüm hastalar butonundan birine basıldığında o veritabanında bulunan hastaların isimleri ve soyisimleri liste görünümünde gözükür.



Sekil 6.14 Hasta listesi formu

Ardından listede hangi hastanın seçildiğini tanımlamak için aşağıdaki yapı kullanılmıştır.

```
void anaekran::on_pushButton_22_clicked() {
    query = new QSqlQuery();
    QModelIndexList list = ui->listView->selectionModel()->selectedIndexes();
    QStringList slist;
    foreach(const QModelIndex &index, list){
        slist.append( index.data(Qt::DisplayRole ).toString());    }
```

Ardından ise seçilen hastanın grafiklerini görüntüle butonuna basıldığı zaman hastaya ait fotoğraf, tanı bilgileri, kimlik bilgileri, tüm geçmiş zamandaki sensör verileri ve ortalama nabız, SpO2, derece gibi veriler ekrana yansır. Bu işlemleri yapan kısım aşağıdaki kod yapısıdır. Öncelikle fotoğraf ve hasta tanı bilgileri ekranda görüntüleyecek olan kod yapısı çalışır.

```
QString tckimlikno,ad,soyad,tani,lokasyon,yas; int nb; QString database_sec="";
database_sec += "SELECT tckimlikno,ad,soyad,yas,tani,lokasyon,no FROM ";
database_sec +=kimlik;
database_sec += " ORDER BY no;";
if (query->exec(database_sec)) {
    qDebug() << "Veri cekme Basarili";
    if (query->isActive()) { if (query->isSelect()){
        if (query->last()){
            tckimlikno = query->value(0).toString();
            ad = query->value(1).toString();
            soyad = query->value(2).toString();
            yas = query->value(3).toString();
            tani = query->value(4).toString();
            lokasyon = query->value(5).toString();    }
        }
    } }
```

```

if( !query->exec( "SELECT imagedata from imgTable" ) )
    qDebug() << "Resim alinirken hata olustu:\n" << query->lastError();
query->first();
QByteArray outByteArray = query->value( 0 ).toByteArray();
QPixmap outPixmap = QPipixmap();
outPixmap.loadFromData( outByteArray );
ui->label->setPixmap(outPixmap);
ui->label->show();

```

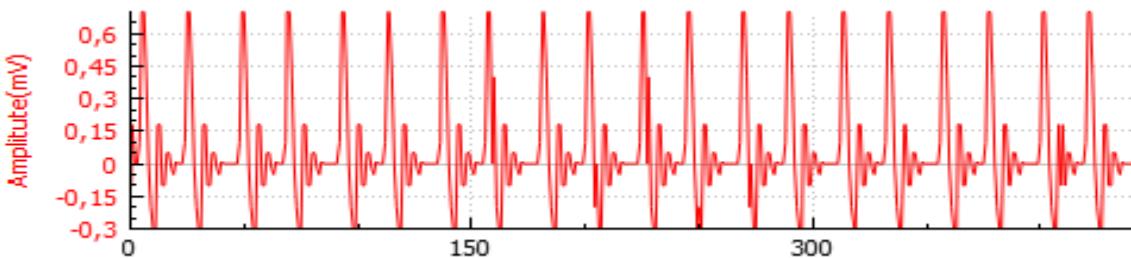
Bu aşamadan sonra ise sensör verileri grafik gösterim ögesi olan qcustomplot üzerinde görüntülenecek olan kodlar çalışmaktadır.

```

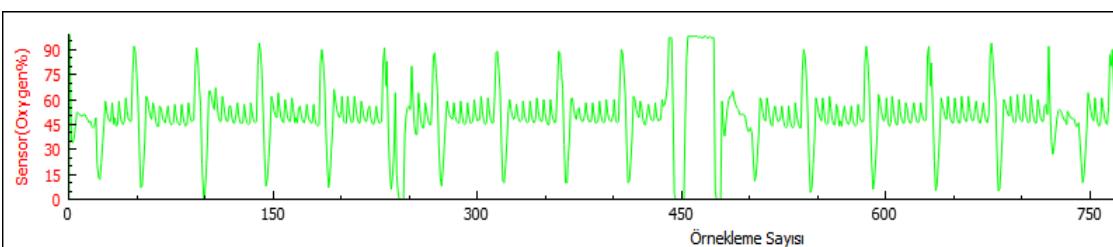
void anaekran::grafikciz() {
int olcum_sikligi= ui->horizontalSlider->value();
QVector<double> k(10100),n(101000),y(100000);
for(int i =0;i<1000;i++) {
    y[i]=i;
}
query = new QSqlQuery();
QVector<double> nabiz(100000),
oksijen(100000),hastasicaklik(100000),sira(100000);
QVector<double> odasicaklik(100000), odanem(100000);
QByteArray ba,po;
int nb;
QString database_sec="";
database_sec += "SELECT nabiz,oksijen,hastasicaklik,ekgverisi,oksimetre,no
FROM ";
database_sec +=kimlik;
qDebug() << "Veri cekiliyor..";
if (query->exec(database_sec))
{
    qDebug() << "Veri cekme Basarili";
    nb = query->size();
    for( int i = 0; query->next(); i++ )
    {
nabiz[i] = query->value(0).toDouble();
oksijen[i] = query->value(1).toDouble();
hastasicaklik[i] = query->value(2).toDouble();
ba += query->value(3).toByteArray();
po += query->value(4).toByteArray();
sira[i] = query->value(5).toDouble();
    }
}
else
{
    qDebug() << query->lastError();
}
QString deneme = QString(ba);
ba.clear();
QStringList myStringList = deneme.split("!");
for(int index =0;index < myStringList.length();index++)
{
    k[index] = myStringList.at(index).toFloat();
}
QString deneme2 = QString(po);
po.clear();

```

Yukarıdaki kodlar sayesinde veritabanından çekilen sensörlerin bufferlanmış verileri teker teker çözümlenerek ilgili grafiklerde görüntülenebilmesi sağlanmıştır. Örnek olarak Arduino'da oluşturulan yapay bir ekg sinyalinin qcustomplot ögesi üzerinde çizilmiş hali ve SPO2 sensörünün sinyalinin örnek görüntüsü sırası ile şekil 5 ve şekil 6 da görüntülenebilir.



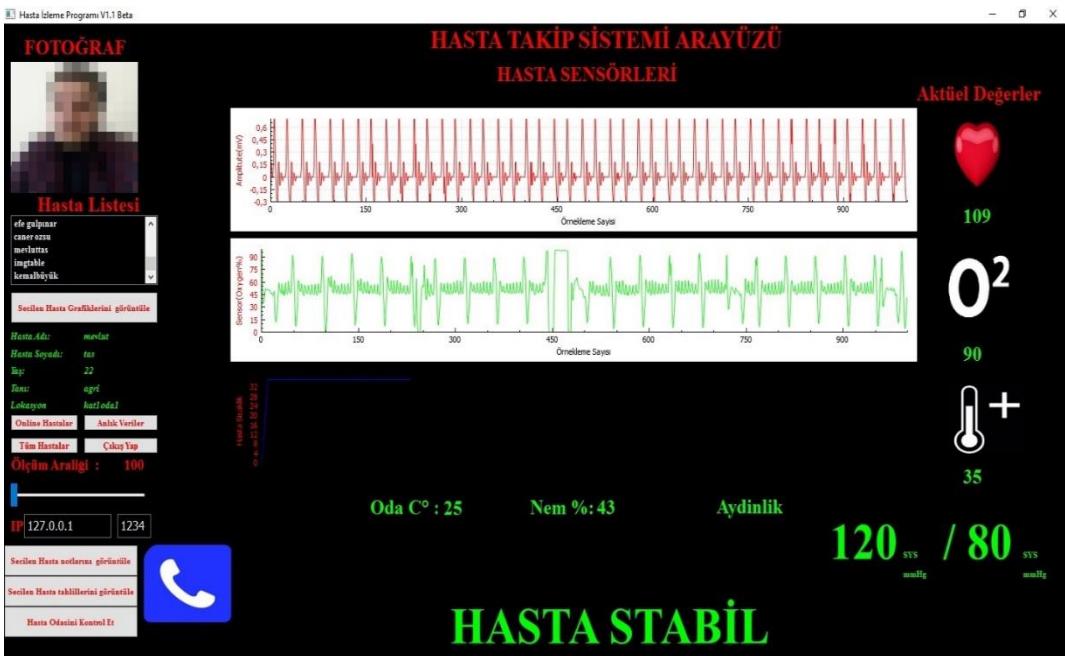
Şekil 6.15 Monitör arayüzünden alınan EKG grafiği örneği



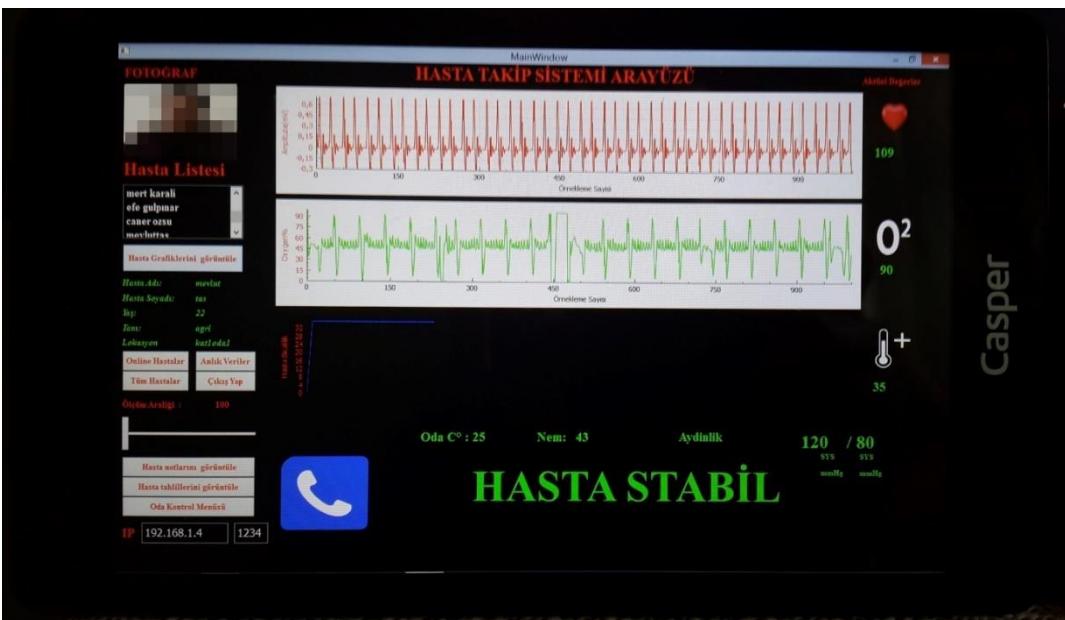
Şekil 6.16 Monitör arayüzünden alınan SpO2 grafiği örneği

Tüm bu monitör programlarının başlıca özellikleri şu şekilde ayarlanmıştır.

- Kullanıcı adı ve şifre yapılarak ilgili istemci programa giriş yapılabilmesi,
- Hastaya ait tüm bilgileri görebilme,
- Hastanın odasında bulunan ESP modüle veri tabanından çekilen bu modülün IP adresi kullanılarak gerçekleştirilen TCP bağlantısı ile odadaki lamba, klima, sesli uyarıyi açabilme ya da kapatabilme, Hastanın odasına sağlık personeli çağrıma butonları,
- Sistemde aktif durumda olan hastalardan geçmiş veya anlık verileri grafiksel olarak görüntüleyebilme ve bu sistemi daha önce kullanmış olan hastaların verilerini okuyabilme,
- Hasta üzerinden alınan verilerin grafikte daha iyi incelenmesi için grafik üzerinde zoom ve ileriye dönük kaydırma gibi işlemlerin yapılabilmesi,
- Hasta üzerine yerleştirilen EKG sensörleri ile sürekli olarak EKG verisi alabilme,
- Sisteme yüklenen hasta grafiklerini görebilme(EMG, röntgen vb.),
- Hastaya ait not yazma veya geçmiş notları görüntüleyebilmesidir.



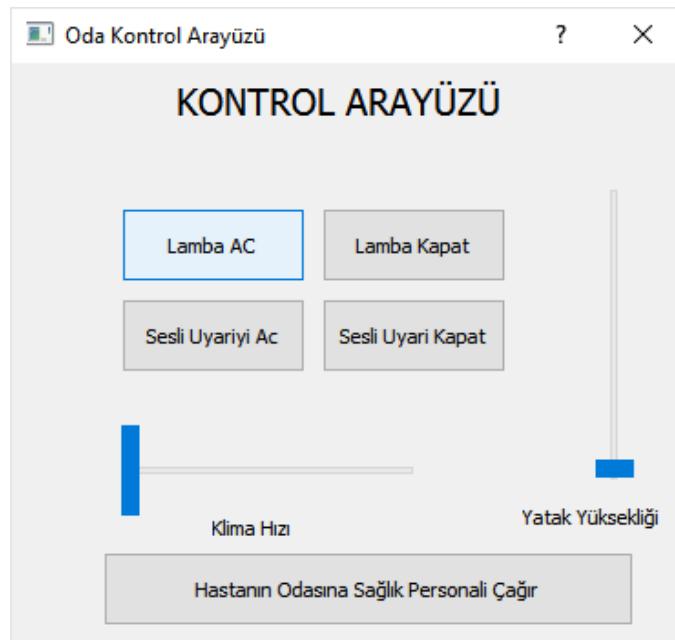
Şekil 6.17 Aktarım halinde monitör arayüzü



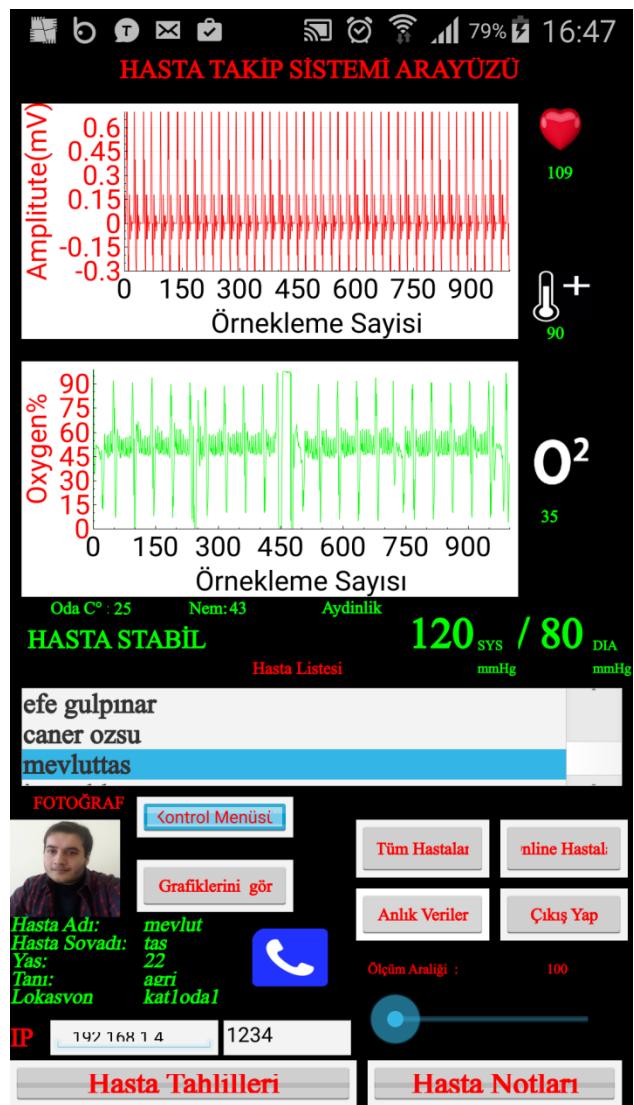
Şekil 6.18 Aktarım halinde monitör arayüzü



Şekil 6.19 Aktarım halinde multiplatform arayüzler



Şekil 6.20 Oda kontrol arayüzü



Şekil 6.21 Mobil cihazlar için tasarlanılan arayüz

SONUÇ VE ÖNERİLER

Bu bitirme tezinin ilk uygulaması olan keşif robotu çalışmasında insanların can güvenliğinin riskli olduğu şartlarda kullanılmak üzere kablosuz olarak uzaktan tablet bilgisayar vasıtasyyla kontrol edilebilen keşif robotu prototipi tasarlanıp gerçekleştirılmıştır. Keşif robotunun üzeri, sensörler ve farklı özellikte iki kamera ile donatılmış ve kablosuz haberleşme teknolojisi ile Windows işletim sistemine sahip olan tablet veya bilgisayardan kontrol edilebilmektedir. Bu tasarım kapsamında gerçekleştirilen keşif robotunun temel özellikleri; C++ ve C# tabanlı olması, HD kalitede gece-gündüz video ve fotoğraf çektebilmesi, üzerindeki sensörler ile hedef ortamı analiz etmesi, geliştirilebilir olması, üzerinde isteğe bağlı görüntü işleyebilmesi ve gerçek zamanlı çalışabilmesidir.

Bu keşif robotu daha da geliştirilerek tırmanan veya hem uçan hemde gezen versiyonları yapılabilir. Bu sayede insanın girmesinin riskli olabileceği yerlerde kullanılabilir.

Askeri ve istihbarat gerektiren yerlerde, daha sessiz motorlar kullanılarak hedef ortamda ses ve görüntü kaydı sayesinde veri toplayabilir.

Otonom kontrol gerektiren fabrikalarda görüntü işleme yaparak, gerekli olan işlemleri yapabilir ve işyükünü azaltabilir. Örnek olarak sektörler arasında yük taşıyan bir keşif robotu gerçekleştirilebilir.

Bu bitirme tezinde ikinci uygulamasında kablosuz hasta izleme projesinin anlatımına yer verilmiştir. Yapılan bu çalışma hastaların yaşam kalitesini arttırmak, sürekli takip altında tutulmalarını ve sağlık personelinin erken tanı ve müdahale yapabilmesini sağlamak için tasarlanmış ve gerçeklenmiştir. Bu proje sayesinde hasta sadece hastanede değil, sisteme bağlanabildiği her yerde doktorun gözetimi altında olmaktadır. Ayrıca gömülü sistem olumsuz durumlarda hastanedeki sağlık çalışanlarını alarma geçirecek şekilde programlanmıştır. Bu sayede hasta kendi sağlık durumunu üzerinde bu uygulamalar sayesinde kontrol edebilecektir. Örneğin kalp ritminin bozulduğunu düşünen bir panik atak hastası gün boyunca kendi kalp ritmini kaydedip sonuçlarını hem kendisi hem de doktoru görebilecektir. Bu sayede uyku apnesi, ritim bozukluğu gibi önemli hastalıklar ölçüm sonuçlarına göre tespit edilip gerekli erken önlemler alınabilecektir. Proje 3G modemler vasıtasy ile ilgili donanımı içeren Linux gömülü sistem bağlanarak (Raspberry Pi Zero vb.) daha portatif ve taşınabilir hale getirilebilir. Bu sayede internet erişiminin

olduğu her yerde hasta birey gözetim altında olur. Bu sistemde hastaların verisi veri tabanında kalıcı olarak tutularak bu hastanın ilerleyen zamanlarda geçireceği rahatsızlıklar için referans olarak kullanılabilecektir.

Bu sistem Arduino tabanlı E-Health biyomedikal sensor kalkanı içinde uygulabilir. Bu geliştirme kitinde bulunan biyomedikal sensörlerden (EKG, EMG, Pulse Oksimetre, glukometre vb.) alınan verilerde yine bu sistemle iletilebilir.

Bu sistem ambulanslara 'da uygulanarak acil durumdaki hastanın bütün verileri sunucuya gönderilerek acil serviste bulunan sağlık görevlerinin hasta hakkında bilgi edinmesi sağlanabilir ve hasta henüz acil servise gelmeden gerekli hazırlıklar yapılmış olunur. Ayrıca bu sayede hastanın gerekli ihtiyaçları gidilen hastanenin veri tabanında sorgulaması yapıldığında bu ihtiyaçlar karşılanamıyor ise farklı bir hastaneyeye yönlendirme yapılabilir. Örneğin hastayı ilgili uzman bir ekip barındıran üniversite hastanesine yönlendirmek gibi çözümler üretilebilir. Bu sayede hastaya vakit kazandırılarak olumsuz durumların önüne geçilebilir.

Her iki bitirme projesi uygulamasının makaleleri yazılmış olup gerekli yerlerde yayınlanması için uğraşılmaktadır. Bu çalışmaların bu konularda çalışma yapmak isteyenlere yardımcı olacağı düşünülmektedir.

KAYNAKLAR

- [1] Erişti,E.,”Kablosuz görüntü aktarımı ile yer belirleme amaçlı bir mobil robot uygulaması”,Yüksek Lisans Tezi,İstanbul Teknik Üniversitesi,İstanbul,2009
- [2] Yiğiter,E.,”Mobil keşif robotu tasarımı”,Yüksek Lisans Tezi,Yıldız Teknik Üniversitesi,İstanbul,2010
- [3] Önal,Ö.,”Seri porttan kablosuz ağ ile haberleşebilen kameralı araç kontrolü”, Yüksek Lisans Tezi, Kahramanmaraş Sütçü İmam Üniversitesi,Kahramanmaraş,2011
- [4] Fazel-Rezai, R.: “ A Low- Cost Biomedical Signal Transceiver based on a Bluetooth Wireless System”, Proceedings of the 29th Annual International Conference of the IEEE EMBS Cite Internationale, Lyon, France, August 23-26, (2007)
- [5] K. Kaya, “Kablosuz EKG sistem tasarımı”, Yüksek Lisans tezi, Erciyes Üniversitesi, Kayseri, Türkiye, Ekim 2010.
- [6] S. Can, “EKG işaretlerinin cep telefonu ile iletilmesi”, Gazi Üniversitesi, Ankara, Türkiye, Eylül 2010.
- [7] Cosmanescu, A.; Miller, B.; Magno, T.; Ahmed, A.; Krementic, I.: Design and Implementation of a Wireless (Bluetooth) Four Channel Bio- Instrumentation Amplifier and Digital Data Acquisition Device with User-Selectable Gain, Frequency, and Driven Reference, Proceedings of the 28th IEEE, EMBS Annual International Conference Newyork City, USA, Aug 30- Sept3, (2006)
- [8] Raspberry Pi , https://tr.wikipedia.org/wiki/Raspberry_Pi
- [9] Raspberry Pi 2 , <https://www.raspberrypi.org/products/raspberry-pi-2-model-b/>
- [10] 7 Operating Systems You Can Run With Raspberry Pi,
<http://www.makeuseof.com/tag/7-operating-systems-you-can-run-with-raspberry-pi/>
- [11] Raspberry Pi Guide - Quick Start Guide for Raspberry Pi,
<https://www.raspberrypi.org/help/quick-start-guide/>

- [12] Class 10 micro SD kart , <http://www.amazon.com/SanDisk-Ultra-MicroSDHC-Memory-Adapter/dp/B007XZM6VG>
- [13] Dördüncü genişletilmiş dosya sistemi , <https://tr.wikipedia.org/wiki/Ext4>
- [14] SD kart formatlama aracı , https://www.socard.org/downloads/formatter_4/
- [15] SD kart formatlama aracı MAC versiyonu,
https://www.socard.org/downloads/formatter_4/eula_mac/index.html
- [16] Raspbian işletim sistemi , <https://www.raspberrypi.org/downloads/raspbian/>
- [17] Raspbian Win32 disk imager, <http://www.raspberry-projects.com/pi/pi-operating-systems/win32diskimager>
- [18] Putty programı, <http://www.putty.org/>
- [19] Xming x Server programı, <https://sourceforge.net/projects/xming/>
- [20] Broadcom BCM2836 mikroişlemci,
<https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2836/>
- [21] Raspberry Pi GPIO, <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/>
- [22] BCM2836 Peripherals,
https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2836/QA7_rev3.4.pdf
- [23] UART hakkında bilgi,
https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter
- [24] SPI protokolü hakkında bilgi, <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>
- [25] Arduino , <https://www.arduino.cc/>
- [26] Arduino Nano, <http://tr.aliexpress.com/item/Free-Shipping-Nano-3-0-Controller-Board-Compatible-with-Arduino-Nano-CH340-USB-Driver/2021663360.html?spm=2114.49010308.4.15.S2AiqI>
- [27] Arduino IDE indirme sayfası, <https://www.arduino.cc/en/Main/Software>
- [28] Atmega328 pin diagramı, <http://thewindycitylab.com/little-p-makes-lot-difference/>
- [29] ADC ile ilgili temel bilgi, https://en.wikipedia.org/wiki/Analog-to-digital_converter
- [30] Arduino ADC birimi , <https://www.arduino.cc/en/Reference/AnalogRead>

- [31] PWM tekniği hakkında genel bilgi, https://en.wikipedia.org/wiki/Pulse-width_modulation
- [32] BCM2836 PWM diagramı, <https://cdn-shop.adafruit.com/product-files/2885/BCM2835Datasheet.pdf>
- [33] WiringPi kütüphanesi, <http://wiringpi.com/>
- [34] Servoblaster kütüphanesi, <http://sliceipi.com/setting-up-servoblaster-on-raspberry-pi/>
- [35] Raspberry Pi için super kullanıcı ayarı ,
<https://www.raspberrypi.org/documentation/linux/usage/root.md>
- [36], 3 hücreli 5000 mA li-po pil, <http://www.ebay.com/bhp/3s-lipo-5000mah>
- [37] LM2596 Step down converter <http://www.ti.com/lit/ds/symlink/lm2596.pdf>
- [38] 6V R260 DC motor, <http://www.amazon.com/2PCS-3V-6V-Motors-Remote-Control/dp/B00FHA9Q4E>
- [39] L298N DC motor sürücü,
https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf
- [40] SG90 servo motor , <http://www.micropik.com/PDF/SG90Servo.pdf>
- [41] Pan-Tilt Aparatı,
<http://g01.a.alicdn.com/kf/HTB18xFXIXXXXXb5XFXXq6xXFXXXf/1-5-kg-1set-Nylon-FPV-Pan-font-b-tilt-b-font-Camera-font-b-Mount.jpg>
- [42] LM35 ıstı sensörü, <http://www.ti.com/lit/ds/symlink/lm35.pdf>
- [43] HCSR04 mesafe sensörü, <http://www.micropik.com/PDF/HCSR04.pdf>
- [44] MQ-7 karbon monoksit sensörü,
<https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf>
- [45] LB-link RTL8188cyus Wi-Fi adaptörü, <https://www.adafruit.com/product/1030>
- [46] LB-Link aliexpress linki , <http://tr.aliexpress.com/item/Raspberry-Pi-2-WIFI-USB-of-5DB-Antenna-raspberry-pi-WIFI-dongle-MCU-RTL8188CUS/32506610290.html?spm=2114.49010508.4.8.llAvcY>
- [47] Logitech C310 kamera, <http://www.logitech.com/tr-tr/product/hd-webcam-c310>
- [48] Raspberry Pi NoIR kamera , <https://www.raspberrypi.org/products/pi-noir-camera/>
- [49] Linux UVC driver listesi, http://www.linux-drivers.org/usb_webcams.html

- [50] Raspberry Pi NoIR kamera, <https://www.raspberrypi.org/blog/new-8-megapixel-camera-board-sale-25/>
- [51] MJPG video formatı https://en.wikipedia.org/wiki/Motion_JPEG
- [52] Mjpg-streamer github linki, <https://github.com/jacksonliam/mjpg-streamer>
- [53] CH340 TTL çevirici, <https://www.olimex.com/Products/Breadboarding/BB-CH340T/resources/CH340DS1.PDF>
- [60] Elektrokardiyografi hakkında bilgi, <https://tr.wikipedia.org/wiki/Elektrokardiyografi>
- [61] Galvonometre hakkında bilgi, <https://en.wikipedia.org/wiki/Galvanometer>
- [62] Depolarizasyon, [http://www3.istanbul.edu.tr/itf/fizyoloji/Ogrenci_Isleri/Ders_Notlari/Kardiyopulmoner_ve_kan_fizyolojisi_\(3_yy\)/Prof_Dr_Muge_Devrim_Ucok/Normal_EKG.ppt](http://www3.istanbul.edu.tr/itf/fizyoloji/Ogrenci_Isleri/Ders_Notlari/Kardiyopulmoner_ve_kan_fizyolojisi_(3_yy)/Prof_Dr_Muge_Devrim_Ucok/Normal_EKG.ppt)
- [63] Elektrokardiyogram, <http://www.saglikvakti.com/elektrokardiyogram-ekg/>
- [64] QRS süre aralığı , <http://www.acilci.net/qrs-kompleksi/>
- [65] Pulse oximeter , http://www.hopkinsmedicine.org/healthlibrary/test_procedures/pulmonary/oximetry_92_p07754
- [66] Pletismografi nedir?,<http://tr.healthline.com/health/pletismografi>
- [67] Pulse amped sensor, <http://pulsesensor.com/products/pulse-sensor-amped>
- [68] DS18b20 sensörü, <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>
- [69] DHT 11 ıslı ve sıcaklık sensörü, <http://www.micropik.com/PDF/dht11.pdf>
- [71] ESP8266 Wi-Fi modülü, <http://www.esp8266.com/>
- [72] QT çalışma alanı, [https://en.wikipedia.org/wiki/Qt_\(software\)#cite_note-4](https://en.wikipedia.org/wiki/Qt_(software)#cite_note-4)
- [73] Qt open source , <https://www.qt.io/download-open-source/>

EK A

QT ÇALIŞMA ALANI

QT, bir multiplatform grafiksel kullanıcı arayüzü geliştirme araç takımıdır. Genellikle görsel arayüzlü yani kullanıcı etkileşimli programlar yazmak için kullanılsada günümüzde QT'nin içерdiği sınıf ve bileşenleri sayesinde birçok alanda kullanılmaktadır. QT, en çok KDE masaüstü ortamında, Opera ağ tarayıcısında ve Skype anlık mesajlaşma programlarıyla ün salmıştır.[72]

Nokia'nın, Qt'nin orijinal geliştiricisi olan Norveçli firma Trolltech'i satın almasıyla oluşan Qt Programlama Çatısı birimi tarafından geliştirilmektedir. Dikkatleri ilk kez KDE masaüstü ortamının bu araç takımını kullanması ile çekmiştir. İlk sürümleri özgür olmayan bir lisansla dağıtılmaktadır. Linux gibi hızla büyüyen bir özgür işletim sisteminin en popüler masaüstü ortamının özgür olmayan bir araç takımını kullanması elbette hoş karşılanmamıştır. Tepki olarak GTK+ kullanarak GNOME masaüstü geliştirilmeye başlanmıştır. Bununla birlikte Harmony denen ve Qt'ye benzeyen bir araç takımı da geliştirilmeye başlanmıştır.

Trolltech, bu tepkilere karşı Qt'yi QPL denen, GPL benzeri bir lisansla yayınlamıştır. Bunu da beğenmeyen özgür yazılım takipçileri sonunda Trolltech'e Qt'yi GPL olarak dağıtmasından başka çare bırakmamıştır. Eğer olur da Trolltech'in başına bir şey gelirse diye, hem Qt hem de KDE kütüphanelerinin gelişimini devir alacak KDE Free Qt Foundation kurulmuştur.

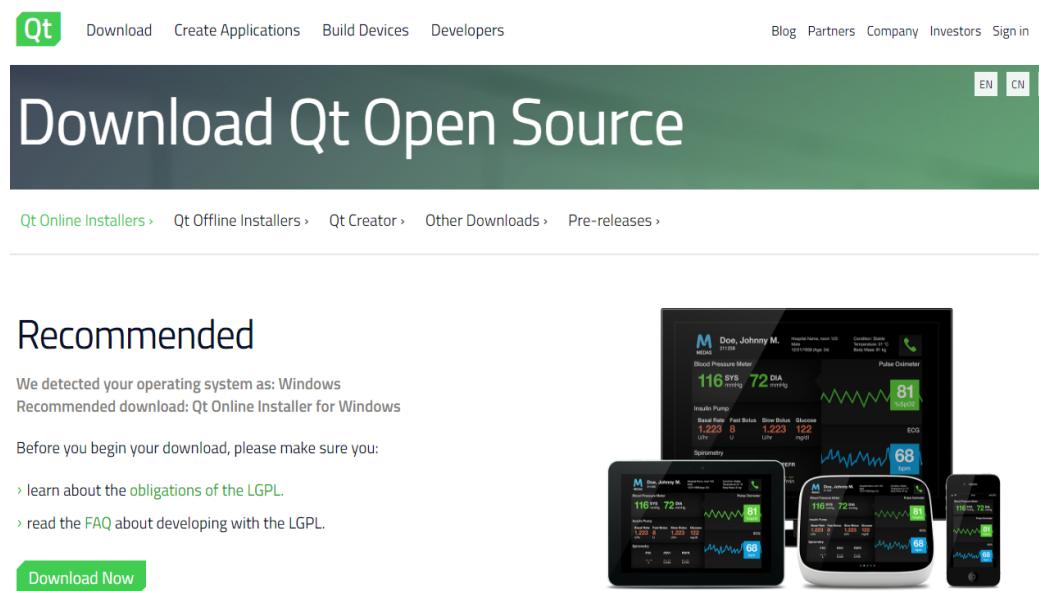
Qt, dördüncü sürümüne kadar Windows platformu üzerinde GPL olarak dağıtılmıyordu. Dördüncü sürümle birlikte Windows için de GPL lisanslı dağıtılmaya başlandı. Bu sürüm ile de GUI olmadan, konsol uygulamaları geliştirilebilir hale gelmiştir.

Qt'nin GPL sürümü ile geliştirilen uygulamalar mutlaka GPL olarak lisanslanmalıdır. Farklı lisanslama yapmak için Qt'nin ücretli sürümleri kullanılabilir.

Nokia tarafından satın alınan Trolltech firması QT'yi 4.5 sürümünden itibaren LGPL lisansı altında dağıtmaya başlamıştır, böylelikle Qt, ticari yazılımlarda para ödenmeden kullanılabilmektedir.

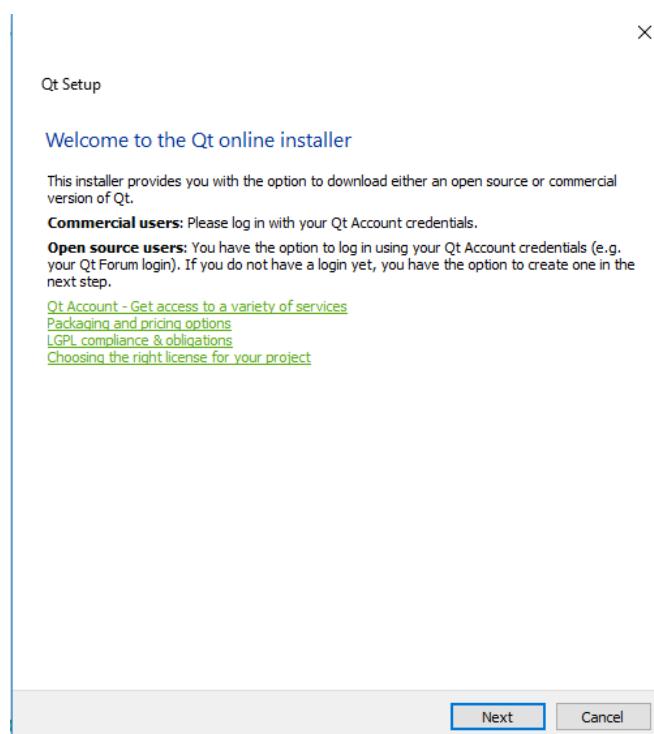
A.1 QT Creator IDE'sinin Windows Sistemlere Kurulumu ve Örnek Program

1. Öncelikle QT web sayfasında Qt'nin Open source versiyonu indirilmelidir[73].



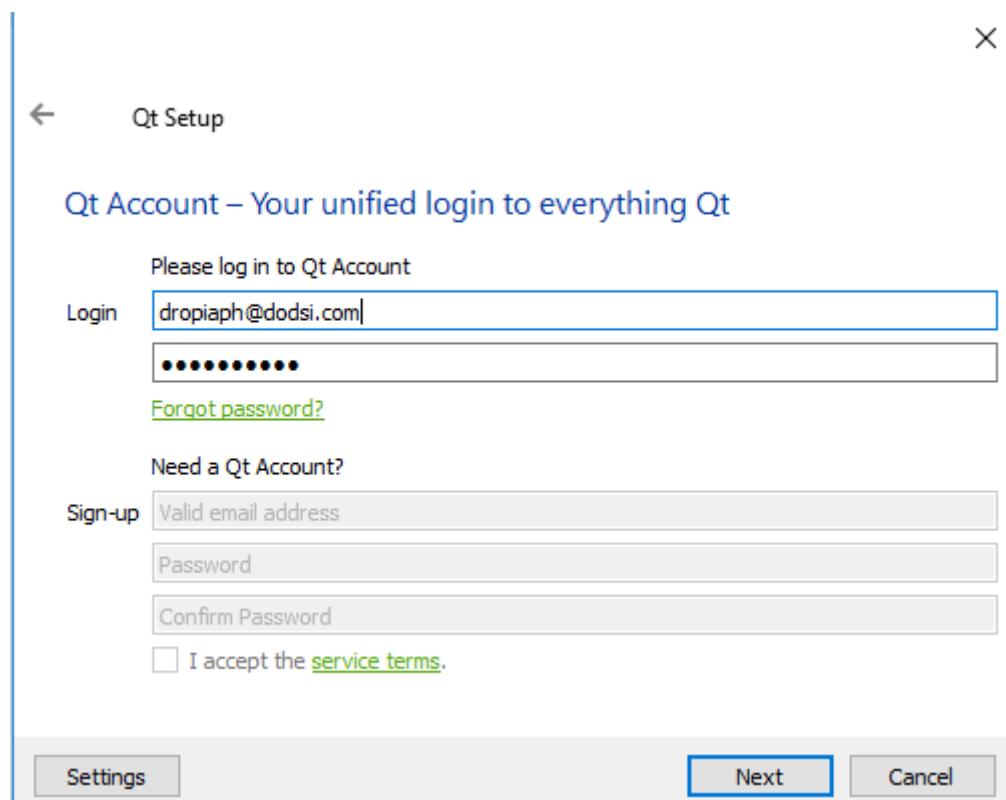
Şekil A.1 QT web sayfası

2. Ardından ise QT web indirici programı açılır.



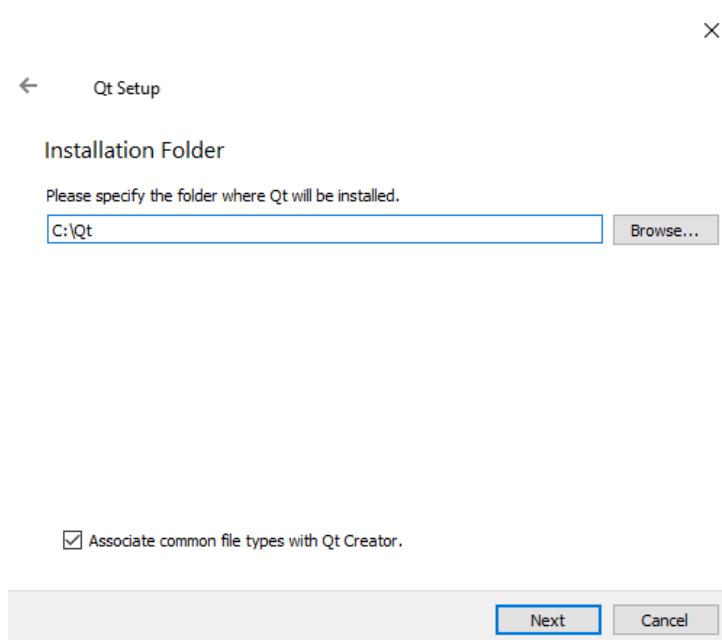
Şekil A.2 QT Creator online web yükleyicisi ekranı

3. Qt sitesinden veya programdan alınabilen üyelik ile programa giriş yapılır.



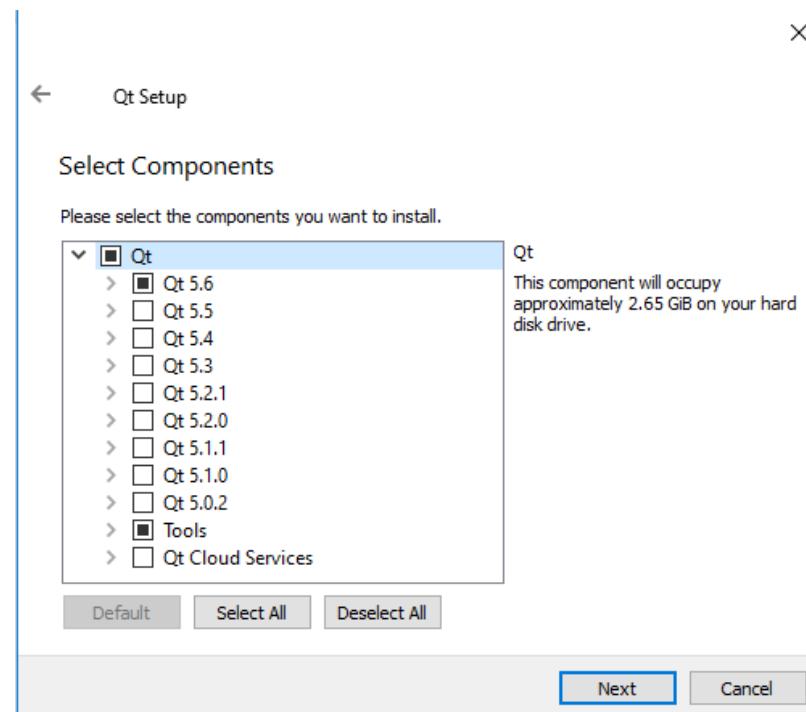
Şekil A.3 QT web yükleyici kullanıcı giriş ekranı

4. Bu aşamadan sonra QT kendi deposundan ilgili dosyaları çekmeye başlayacaktır ve size nereye indirmesi gerektiği sorucaktır.



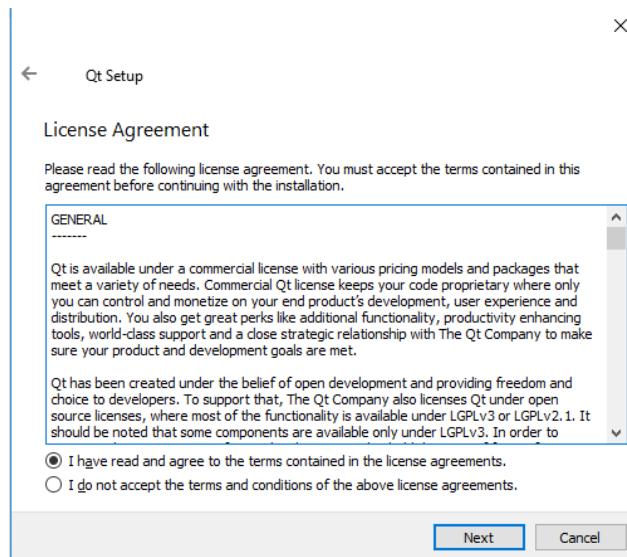
Şekil A.4 QT Creator yükleme dizini seçim ekranı

5. Bu aşamadan sonra istenilen QT versiyonu ve sınıfları aşağıdaki listelene kısımlardan indirilip seçilebilir.



Şekil A.5 QT versiyonu seçim ekranı

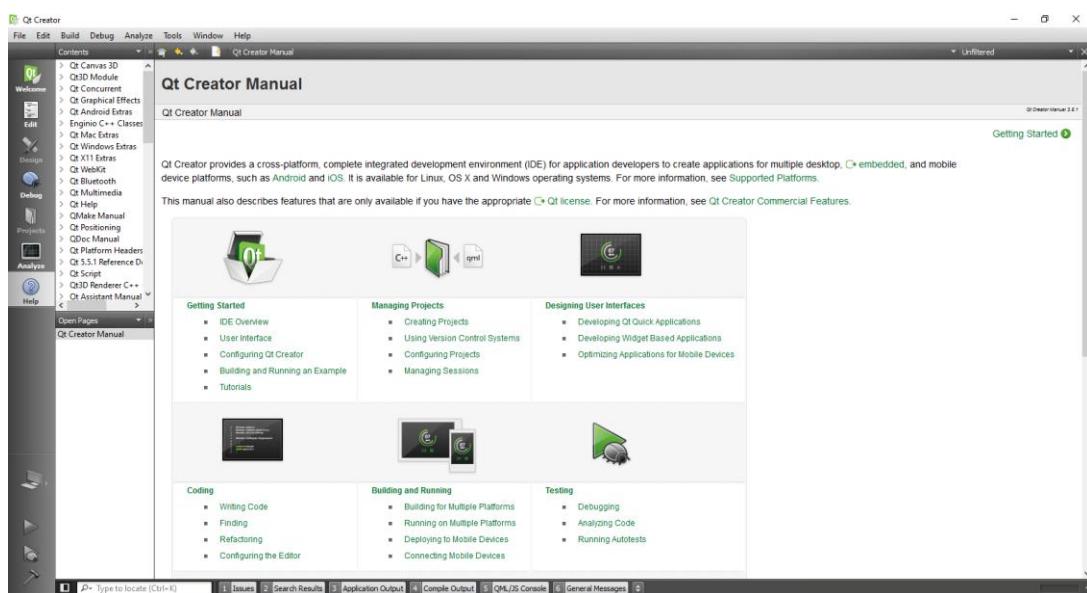
6. Bu aşamadan sonra sözleşmeyi kabul etmeniz gerekmektedir. Kabul ettikten sonra indirme başlayacaktır.



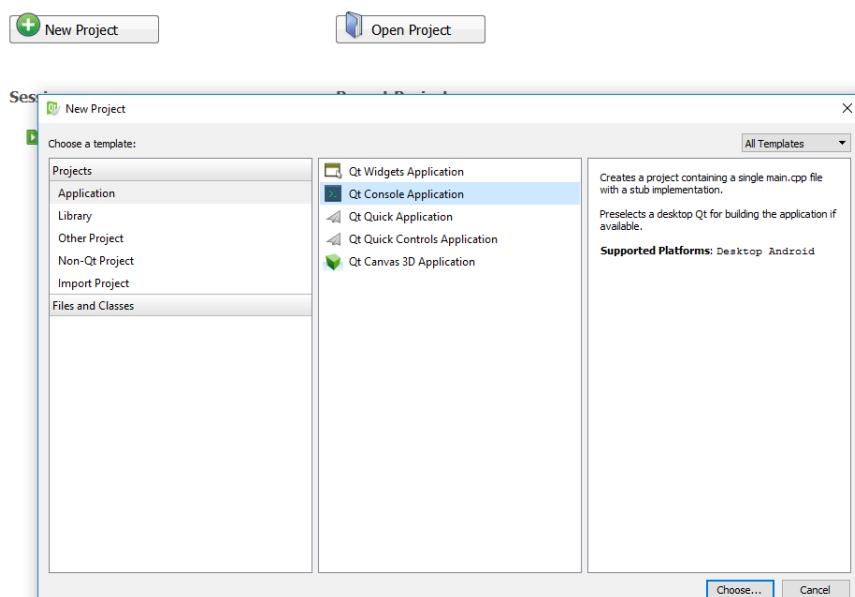
Şekil A.6 Lisans koşullarını kabul ekranı

A.2 QT Creator IDE'si Üzerinde Örnek Program

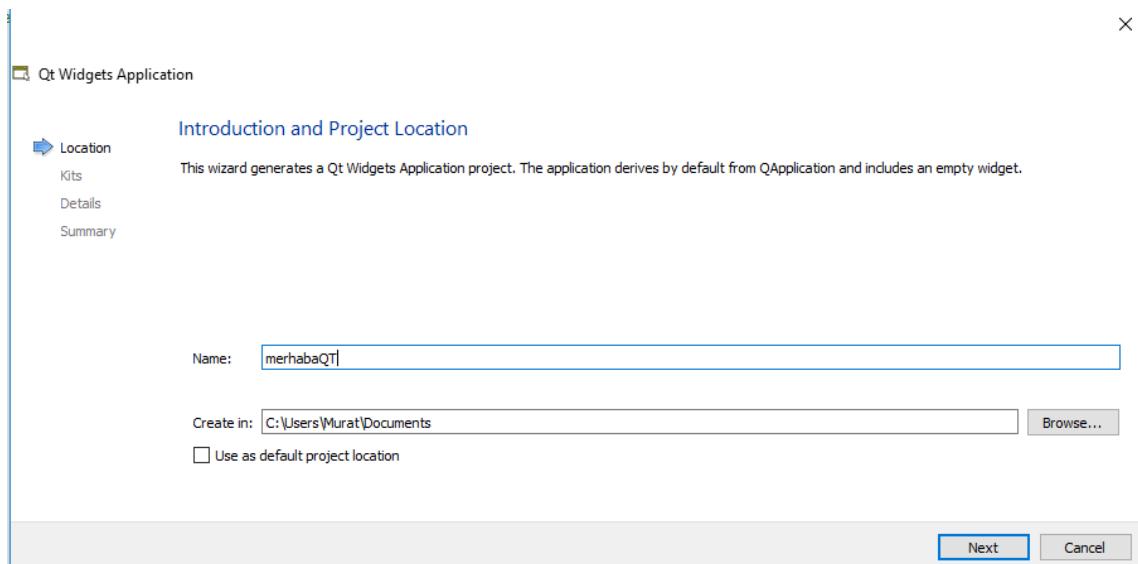
QT Creator indirildikten sonra masaüstündeki kısayolu bulunur ve program açılır. Örnek bir uygulama yazmak istersek File>New File or Project tıklanır ve kodlamak istenen uygulama türü listeden seçilir. Ardından projeye isim verilir ve hedef platforma göre derleyici seçilir. Ondan sonra Run butonuna basılır ve örnek program çalıştırılabilir. Tüm bu anlatılanlar aşağıdaki şekillerde görülmektedir.



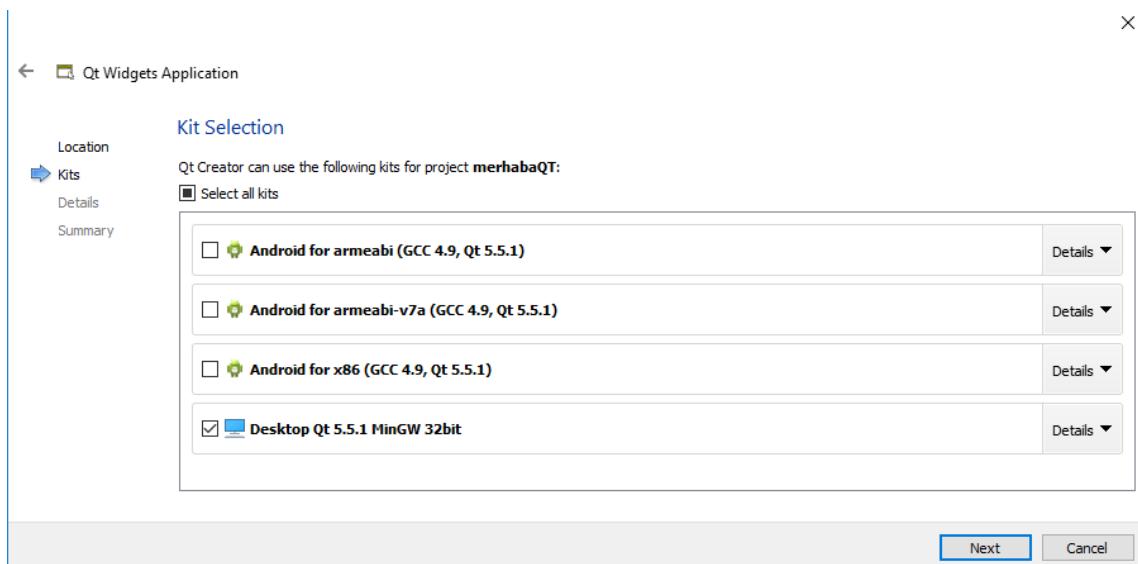
Sekil A.7 QT Creator IDE yardım arayüzü



Şekil A.8 Proje türü seçim ekranı



Şekil A.9 Proje ismi seçme ekranı



Şekil A.10 Proje için derleyici seçim ekranı

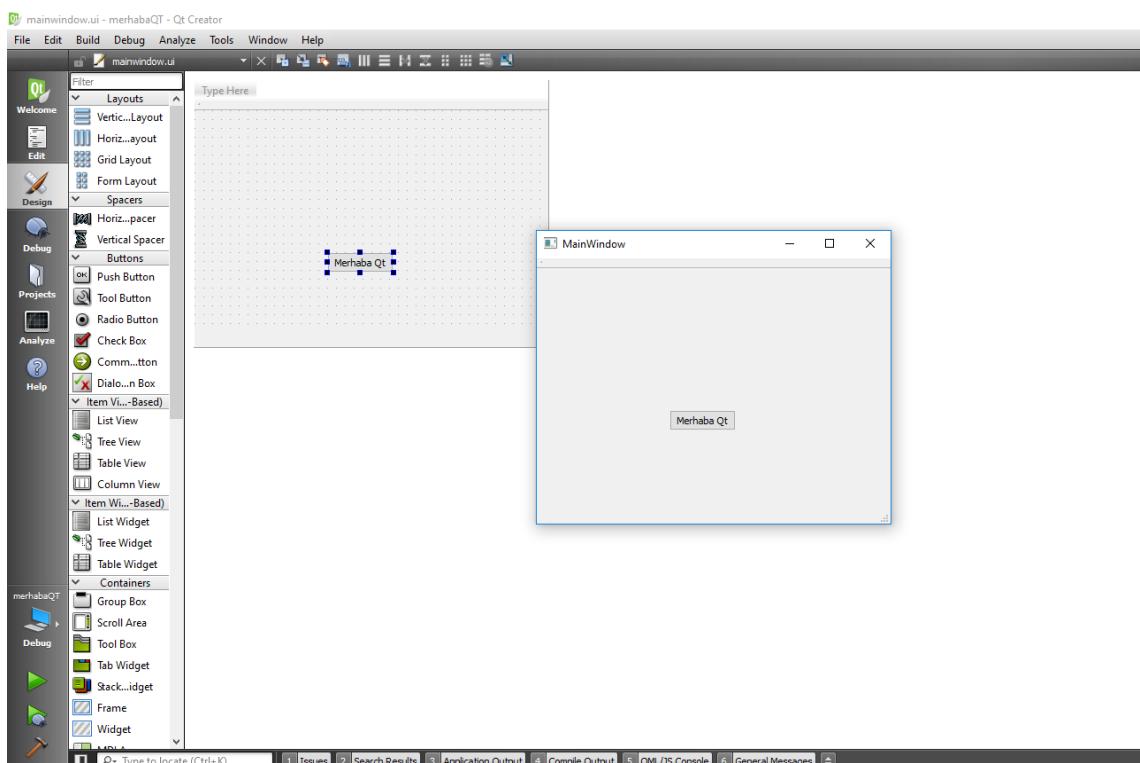
The screenshot shows the Qt Creator interface with the project 'merhabaQT' open. The left sidebar has 'Edit' selected. The main area displays the code for mainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

Şekil A.11 Proje için derleyici seçenek ekranı



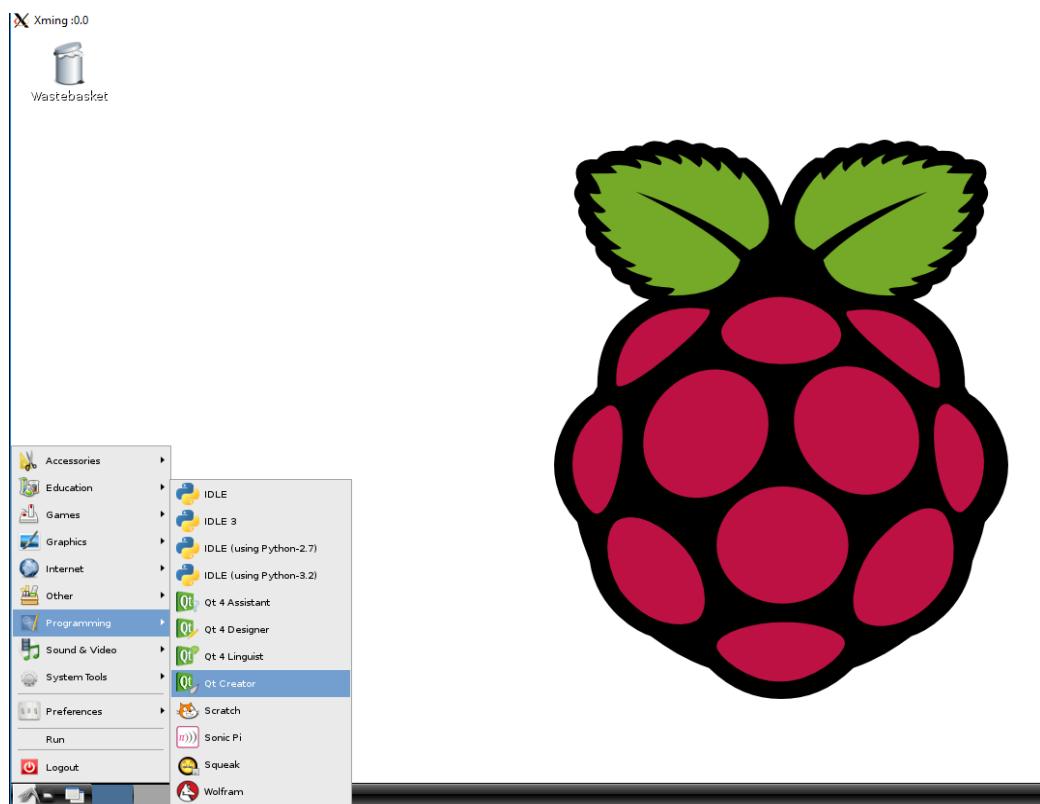
Şekil A.12 İlk örnek uygulamanın çalıştırılması

A.3 QT Creator IDE'sinin Raspbian Sisteme Kurulumu ve Örnek Program

1. Öncelikle QT çalışma alanı ilgili alandan apt-get komutu ile indirilmelidir. Ardından QT Creator IDE'si indirilir ve bu linux ortamı için gerekli derleyici araç takımları kurulur.

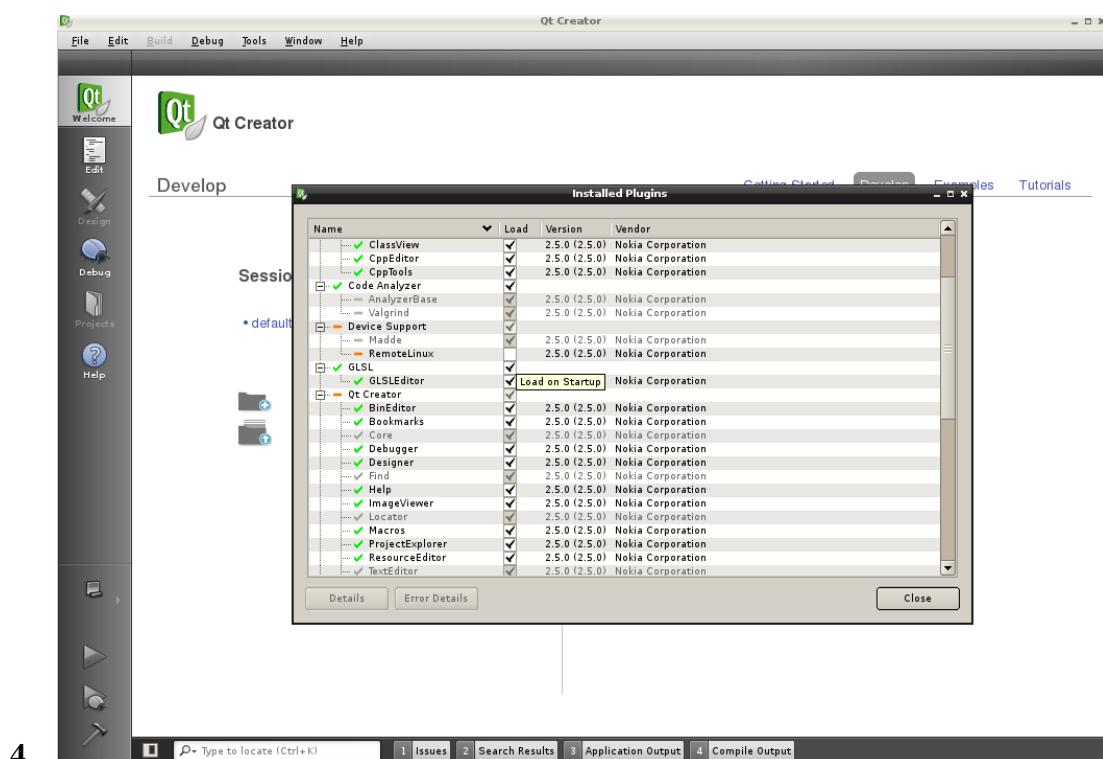
```
sudo apt-get install qt4-dev-tools  
sudo apt-get install qtcreator  
sudo apt-get install gcc  
sudo apt-get install xterm  
sudo apt-get install git-core  
sudo apt-get install subversion
```

2. Sonuç olarak QT Creator versiyonu olan 2.5 sürümü, Qt 4.8.1 32 bit kütüphaneleri kullanılabilir hale hazırlır.



Şekil A.13 QT Creatorun bulunması

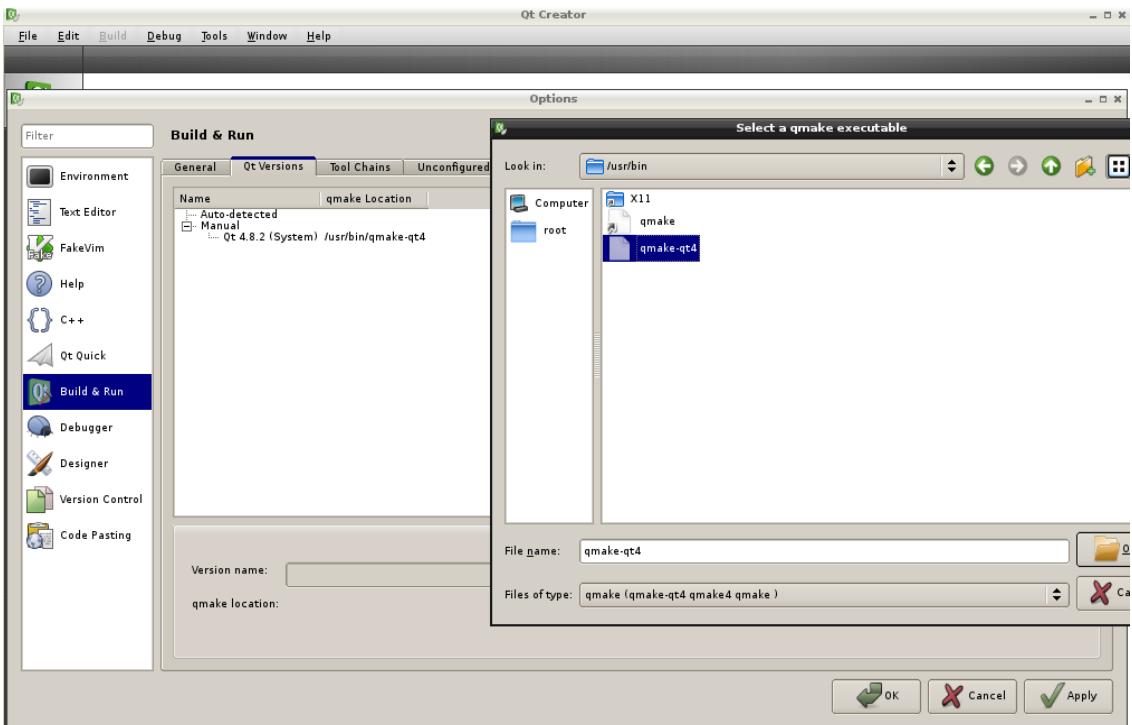
3. Ardından QT Creator açılır. QT ‘yi yapılandırmak için öncelikle Help>About plugins kısmından açılacak olan pencerede “Remote Linux” iptal edilir. Ardından Qmake versiyonu ve derleyicileri seçmek için Tools>options>build seçenekine gidilir. Burda öncelikle qmake versiyonu için build sekmesindeki “Qt versiyons” tıklanır ve “/usr/bin” yolundaki qt4-make dosyası seçilir. Ardından gömülü linux C++ derleyicisi için Toolchains sekmesine tıklanır. Toolchains te yeni bir gcc seçilir ve “/usr/bin” klasöründe “/usr/bin/arm-linux-gnueabihf-gcc-4.6” derleyicisi tanıtılır. Debugger kısmı ise “/usr/bin/gdb” olarak tanıtılar ve Qt Creator yeniden başlatılır. Artık Qt çalışmaya hazır. Aşağıdaki şeillerde bu anlatılanlar gösterilmektedir.



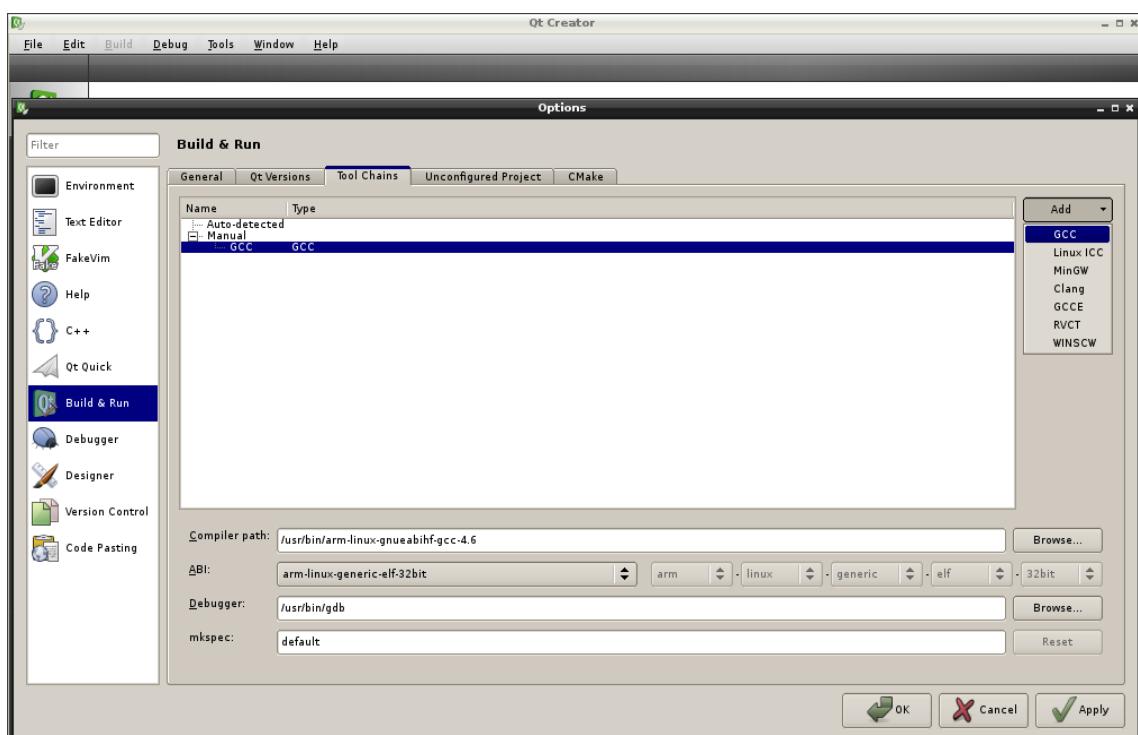
4.

Şekil A.14 QT Creator Remote Linux özelliğinin devre dışı bırakılması

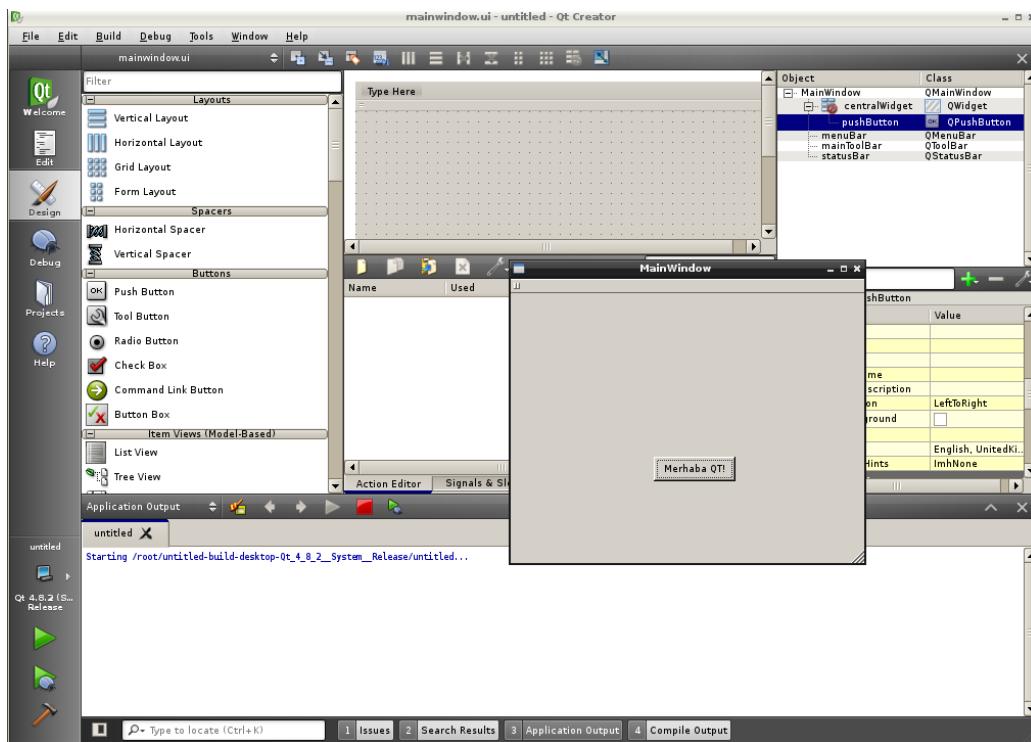
5. QT Creator yeniden başlatılır.



Sekil A.15 Qt qmake versiyonunu seçme ekranı.



Sekil A.16 Qt gcc derleyici seçim ekranı



Şekil A.17 Örnek uygulamanın çalışması

EK-B

SOKET HABERLEŞME

TCP-IP protokollerı, bilgisayarlar arası haberleşmeyi sağlayan protokollerdir. Bir bilgisayar başka bir bilgisayara göndermek istediği verileri paketler halinde gönderir. Bu veri alışverişi IP paketleri aracılığı ile yapılmaktadır. Bu paketlerin içinde X'in ve Y'nin adresleri bulunmaktadır. Örneğin X bilgisayarının y bilgisayarına veri göndermesini ele alalım. X bilgisayarının gönderdiği pakete IP-Paketleri, X'in adresine IP-Adresi denmektedir. Gönderilen bu paketler Y'nin adreslerini bulundurduğu için bilgisayar ağları arasında kullanılan Router (yönlendirici)'ler ile hedef bilgisayara bu paketler ulaştırılır.

Soketler, aynı anda veya farklı sunucular üzerine haberleşmeyi sağlayan bir haberleşme yöntemidir. Soyut olarak bir haberleşme ucu olarak tanımlanabilir. Pratikte soketler bir dosyaya benzetilebilir. Soket haberleşme üzerinden veri okumak ile dosya okumak arasında hiçbir fark yoktur. Hemen hemen tüm internet tabanlı programlar soket haberleşme kullanır. Örnek olarak web sunucuları, ftp ve telnet sunucuları verilebilir. Soket haberleşmede kullanılan bazı terimler şöyle tanımlanabilir

İstemci(Client): Hizmet kullanmak isteyen ve soket haberleşme kullanan programlara denir.

Sunucu(Server): Hizmeti sunan soket programıdır.

Port: Bir bilgisayarda birden fazla soket haberleşme için farklı farklı Port adı altında bir numara vardır. Soket bağlantıları birbirinden ayırt etmek ve sunucuda clienti uygun programa ulaşımak için her soket programınınında bir PORT denilen numarası vardır. Örneğin Telnet protokolü 23 nolu soketi kullanır. 1-1024 arasındaki portlar yalnızca kök dizin tarafından kullanılabilir ve normal kullanıcılar bu portta işlem yapamamaktır.

IP Numarası: TCP/IP protokolü hostları, sadece kendisini ait olan bir IP numarası ile tanımlamaktadır. Bu ip numarası sayesinde bilgisayarlar birbiriyle haberleşirler. Bir client yani istemci program önce hedef bilgisayarın önce IP adresini sonra ise port adresini kullanarak haberleşmek istediği sunucu program ile temas geçer. IPV4 standartına sahip IP'ler 192.168.1.5 gibi [0-255]. [0-255]. [0-255]. [0-255]. Formatına sahiptirler. IPV& standartı 128 bit adres kullanır.

Sarmalama: Her katman kendisine gelen pakete bir başlık bilgisi ekler ve bir sonraki protokole aktarır. Buna sarmalama denir. Karşı tarafta paket açılırken yine her katman kendisi ile ilgili başlığı açar.

B.1 TCP/IP Protokol Yapısı

TCP/IP, "Transaction Control Protocol and Internet Protocol" için bir kısaltmadır. TCP/IP protokolü DARPA (Defense Advanced Research Agency) projesi olarak 1970'lerde Amerika'daki bazı devlet kurumlarını birbirine bağlamak amacıyla tasarlanmıştır ve geliştirilmiştir. Daha sonra ise UNIX' e eklenerek uygulama alanı genişletilmiştir.

TCP/IP 4 katmanlı bir modelden oluşmaktadır.

1. Uygulama Katmanı
2. Taşıma Katmanı
3. İnternet katmanı
4. Ağ Erişim Katmanı

Bu katmanlar aşağıda sırası ile açıklanacaktır.

B.1.1 Uygulama Katmanı

Uygulama katmanı: farklı sunucular üzerine çalışan süreç ve programlar arasında iletişimini sağlar. TCP/IP protokolünün en üstünde yer almaktadır. Bu katman Taşıma katmanın sağladığı UDP ve TCP protokollerini kullanarak veri aktarımı yapabilirler. Örnek olarak verirsek telnet, ftp, smtp, http birer uygulama katmanı protokolleridir.

B.1.2 Taşıma Katmanı

Noktadan noktaya veri akışı sağlamaktadır. Bu katman transparant bir şekilde verinin hosttan hosta aktarılmasını sağlar. Verilerin akış kontrolünü ve hataları düzeltmeyi sağlamaktadır. Bu katman sayesinde veri transferi bittiğinden emin olunur. Taşıma katmanı bağlantı yönelimli bağlantı sağlamaktadır. Ağ katmanı ise bunu sağlamaz. Ayrıca ağ katmanı, ulaşan paketlerin gönderildiği sırada ulaştığında garanti etmez. Taşıma katmanı ise bu işlemi her paketi numaralandırarak basitçe çözmektedir. Hataoluğu durumda paketi yeniden ister. UDP protokülünde yeniden istenmez. Böylece oluşabilecek veri hatalarının önüne geçilmiş olur.

Taşıma Kontrol Protokolü (Transmission Control Protocol) (TCP): TCP, IP ortamında üç soketler arasında güvenli haberleşme imkanı sunan bağlantı yönelimli (connection oriented) bir protokoldür. RFC 793'te tanımlanmıştır ve uygulama katmanın altında bulunmaktadır. Aynı zamanda süreçler arasında haberleşme protokülüdür. İki süreç

arasında sanal bir devre oluşturur. Telnet, TCP protokolü kullanarak haberleşen popüler uygulamalardan birisidir.

TCP, verilerin aktarılması sırasında zarar görmüş, sırası bozulmuş veya kaybolan veriyi kurtarabilir. Aktardığı her sekizlik için bir sıra numarası tutar ve alıcı noktadan ACK (Acknowledge) alındı-anlaşıldı mesajını almak için bekler. Eğer ACK mesajı belli bir süre içinde(timeout) gelmez ise veri yeniden aktarılır. Alıcı taraf birden fazla ACK mesajı almış ise TCP bunlarida düzeltebilmektir. Her bir segmente bir kontrol toplamı (checksum) ekleyerek alıcı tarafın aldığı verinin doğru olup olmadığını denetler.

TCP, veri akışlarının birbirlerinden ayırtmak için port tanımlayıcı kullanır. TCP her client için birbirinden bağımsız port tanımlayıcı sunmaktadır. Bu yüzden çoklu bağlantılar için birden fazla soket tanımlayıcı olabilir. Bu nedenle soket oluşturulurken internet adresi de kullanılır. Oluşturulan bir soket pek çok dış soketler ile bağlantı yapabilirler ve iki yönlü veri taşımada kullanabilir. Bu bağlantılar tamamen üç soketten üç sokete doğru oluşur. TCP segmentleri, IP tarafından paketlenip gönderilirler.

Kullanıcı Datagram Protokolü (User Datagram Protocol) (UDP): RFC 768'de tanımlanmıştır. Bu protokol uygulamalar için en az veri yükü ile haberleşme olanağı sağlar. Dağıtım ve güvenliği temin etmez. Giden verinin kontrol toplamı değerini tutar ama paket bozulmuş ise yeniden paketi dağıtmaz. TCP'ye nazaran daha yavaştır. DNS ve FTP bu protokü kullanan popüler uygulamalardandır.

TCP ve UDP arasındaki farkları sıralar ısek:

1. Akış soketleri sırayla gönderir, datagram soketleri sıralı göndermeyebilir. Çünkü TCP paketleri sıralı göndermeyi garanti eder. UDP garanti etmez. TCP paketlerin başlık bilgisinde sıra numarası vardır fakat UDP'te yoktur. TCP her zaman sıradaki paketi ister. Örneğin 4 numaralı paket yerine 5 gelirse bunu karşı tarafa bildirip isteyebilir ve bu paketleri bir sıraya koyabilmektedir.

2. Akış soketleri güvenilirdir, UDP güvenilir değildir. Çünkü TCP bildirmek yerine denetim yapmaktadır. Yani bir veri paket gönderildiği zaman, karşı taraf veriyi aldığıni bildirmeden paketi göndermiş sayılmaz ve sürekli tekrar gönderirir. Ayrıca paketin doğru gidip gitmediğini anlamak için başlık bilgisini (checksum) tutar. UDP'de checksum tutar ancak bu bilgi yanlışsa aynı paketi istemez.

3. Akış soketleri, işlem bitinceye kadar kesintisiz bir bağlantı kurar, datagram soketler ise bağlantı kurmaz. Sadece veri gönderileceği zaman bağlantı kurar ve gerekli işlemler bitince bağlantıyı koparır.

UDP Kullanılmasının en önemli nedeni çok az bir protokol yükü kullanmasıdır. Video aktarımı gibi gerçek zamanlı veri akışı gerektiren uygulamalar için TCP fazla yük getirir ve veriler gerçek zamanlı iletilmez. Çünkü TCP kullanan bir veri karşıya $6x32+ \text{veri boyu}$ kadar bir paket olarak gider. Bu yüzden her paket fazladan 192 bit başlık bilgisi taşır. UDP paketler ise 64 bitlik başlık bilgisine sahiptir. Bu da bu protokoller arasında hız farkı oluşturan bir özelliktir. Bu yüzden multicast uygulamalarında Datagram soketler kullanılır. Video ve ses aktarımlarında genellikle az bir verinin bozulması ses ve görüntüyü bozmaz. Bu nedenle paket kontrolüne gerek kalmamaktadır. Eğer iyi bir fiziksel bağlantınız varsa veriler iyi iletilir ve hata oranı düşer. Bu nedenle TCP'nin yaptığı denetim bu bağlantıda fazladan bir yük oluşturacaktır.

UDP paket güvenliğini denetlemese de bunu yazılımcı yapabilir. Örneğin TCP giden verinin iletilğini anlamak için alıcı taraftan ACK mesajı bekler. Ama UDP'de alınan veriler için ACK mesajı gönderilmez. Fakat yazılımcı bunu gönderdiği her pakete yanıt isteyecek şekilde programlarsa bu sorunu aşmış olur.

B.1.3 Internet Katmanı

Yönlendiriciler (router) ile birbirine bağlanmış olan ağlar boyunca aktarılacak olan verinin kaynaktan hedefe doğru yönlendirilmesini sağlar.

Internet katmanı adresleme, paketleme ve yönlendirme fonksiyonlarını yerine getirir. IP, ARP, ICMP ve IGMP protokolleri bu katmana ait çekirdek protokollerdir. Bu katmanda yer alan bazı terimlerin açıklamaları aşağıdadır.

Internet Protokülü(Internet Protocol)(IP) : Adres bilgilerini ve paket yönlendirme için bazı bilgileri içerir RFC 791'de tanımlanmış olup en önemli internet protokolüdür. İki görevi bulunmaktadır.

- 1.Ağlar arasında bağlantısız datagram dağıtımını yapmak
- 2.Freamantasyon ve veri katmanına yardımcı olarak maksimum taşıma birimi(MTU) değerleri ile datagramları yeniden oluşturmaktır.

Versiyon: Kullanılan internet başlığının versiyonunu göstermektedir.

Servis Tipi(Type of Service):İstenilen hizmet kalitesi ile ilgili soyut parametler sunmaktadır. Örneğin trafiğin bir kısmına öncelik verebilir.

Toplam Uzunluk():Veri paketinin başlığında bulunan toplam datagram boyutunu gösterir. Toplam 16 bittir, buradaki değerler byte(bit) olarak gösterilir. Bu yüzden IP paketi en fazla 64K boyutunda olabilir.

Tespit(Identification): Gönderen uç tarafından yazılır. Datagram parçalarını biraraya getirmeye yardımcı olur.

Bayraklar(Flags): Paketin parçalanileceğini veya parçalanamayacağını gösterir.

Parça Denkleştirme (Fargment Offset): Bu paketin datagram içerisinde nereye ait olduğunu tanımlar.

Yaşam Süresi (Time to Live): Sürekli azalan tam sayıdır. Paketin hangi noktada yok edileceğini belirtir. Paketin sonsuza dek ağıda kalmasına engel olur.

Protokol: IP'nin işi bittikten sonra paketi hangi üst protokol alacağını gösterir.

Başlık Kontrol Bilgisi (Header Checksum): IP başlığının bozulmadığını emin olmak için tutulan değerdir.

Kaynak Adresi (Source Adress): Gönderen noktayı gösterir.

Varış Adresi (Destination Adress): Alıcı noktayı gösterir.

Opsiyonlar(Options): IP, güvenlik gibi değişik seçenekleri destekler.

Veri (Data): Üst katmana verilecek veriyi tutar.

B.1.4 Ağ Erişim Katmanı:

Uç sistem ile alt ağ arasındaki lojik arabirimle ilişkin katmandır. Bu katman TCP/IP paketlerini fizikal sel ağa bırakmak ve aynı zamanda fiziksel ağdan gelen paketleri almakla görevlidir.

KEŞİF ROBOTU KAYNAK KODLARI

Bu bölümde keşif robotunda yapımında kullanılan C++ ve C# tabanlı kodlara yer verilicektir. Bütünlüğün bozulmaması adına sadece ana threat yapılarının kodları verilecektir ve bu bölümde header, .pro, designer gibi dosyalar verilmeyecektir.

C.1 Raspberry Pi Üzerinde Çalışan Kodlar

Bu kısımda Raspberry Pi üzerinde QT sınıfları kullanılarak hazırlanmış threat yapıları kodları verilicektir.

C.1.1 Motor Kontrol Programı

Keşif robotunun hareketini sağlayan joystick için gerekli olmayan TCP/IP tabanlı kodlardır.

```
#include "threat.h"
#include <wiringPi.h>
#include <softPwm.h>
int hiz_degeri=250;
threat::threat(qint32 ID, QObject *parent) :
    QThread(parent) {
    this->socketDescriptor = ID;
}
void threat::run() {
    wiringPiSetup () ;
    pinMode (25, OUTPUT) ;
    pinMode (24, OUTPUT) ;
    pinMode (23, OUTPUT) ;
    pinMode (22, OUTPUT) ;
    softPwmCreate(29,0,250);softPwmWrite(29,0);
    softPwmCreate(27,0,250);softPwmWrite(27,0);
    timer = new QTimer();
    timer->setInterval(5000);
    timer->start();
    qDebug() << "Threat Olusturuldu ve veriler okunmaya hazir." ;
    socket = new QTcpSocket(); //cift tarafli haberlesme icin bir adet soket acalim.
    //baglanan pcnin id sine gore doğru bir haberleşme icin idleri aynı yapalim.
    if(!socket->setSocketDescriptor(this->socketDescriptor))
    {
        //Egerki socket acilanmadı ise;
        qDebug() <<"Threat haberlesme soketi acilanmadı.Threat sonlandirlacak." ;
        emit error(socket->error());
        return;
    }
}
```

```

//sinyallere göre fonksiyonları ayarlıyoruz.
    connect(socket, SIGNAL(readyRead()), this, SLOT(readyRead()),
Qt::DirectConnection);
    connect(timer,SIGNAL(timeout()),this,SLOT(pingkontrol()));
    connect(socket, SIGNAL(disconnected()), this, SLOT(disconnected()));
    qDebug() << "Bu id için bağlantı kuruldu = " <<socketDescriptor;
    exec(); //bu threati sonsuz dongü yapıyoruz.
}

void motorileri(int x) {
    softPwmWrite(29,x);
    softPwmWrite(27,x);
    digitalWrite(25,LOW );
    digitalWrite(24,HIGH);
    digitalWrite(23, LOW);
    digitalWrite(22, HIGH);
    delay(30);
}

void tamyolileri(int x) {
    softPwmWrite(29,x);
    softPwmWrite(27,x);

    digitalWrite(25,LOW );
    digitalWrite(24,HIGH);
    digitalWrite(23, LOW);
    digitalWrite(22, HIGH);delay(30);
}

void tamyolgeri(int x) {
    softPwmWrite(29,x);
    softPwmWrite(27,x);

    digitalWrite(25,HIGH );
    digitalWrite(24,LOW);
    digitalWrite(23, HIGH);
    digitalWrite(22, LOW);
    delay(30);
}

void motorgeri(int x) {
    softPwmWrite(29,x);
    softPwmWrite(27,x);

    digitalWrite(25,HIGH );
    digitalWrite(24,LOW);
    digitalWrite(23, HIGH);
    digitalWrite(22, LOW);
    delay(30);
}

```

```

void sagadon() {
    softPwmWrite(29,200);
    softPwmWrite(27,200);
    digitalWrite(25,HIGH );
    digitalWrite(24,LOW);
    digitalWrite(23, LOW);
    digitalWrite(22, HIGH);
    delay(30);
}
void soladon() {
    softPwmWrite(29,200);
    softPwmWrite(27,200);

    digitalWrite(25,LOW );
    digitalWrite(24,HIGH);
    digitalWrite(23, HIGH);
    digitalWrite(22,LOW);
    delay(30);
}
void motordur() {
    softPwmWrite(29,0);
    softPwmWrite(27,0);

    digitalWrite(25, LOW);
    digitalWrite(24, LOW);
    digitalWrite(23, LOW);
    digitalWrite(22, LOW);
}
void threat::pingkontrol() {          //pingi kontrol edicek fonksiyonumuz.

qDebug() << "Ping yaniti alinamadiginden Oturu motorlar susturuluyor!!";
softPwmWrite(29,0);
softPwmWrite(27,0);
digitalWrite(25, LOW);
digitalWrite(24, LOW);
digitalWrite(23, LOW);
digitalWrite(22, LOW);
}

void threat::readyRead() //Veriler okunmaya basladigin an calis
{
    QByteArray Data = socket->readAll();

    if(Data[0]=='p' && Data[1]=='k') {

```

```

timer->start();
}
if( Data[0]=='h' && Data[1]=='g') {
    QString guncelleme(Data);

    hiz_degeri=guncelleme.right(3).toInt();

}
if( Data[0]=='h' && Data[1]=='l') {
    QString guncelleme(Data);
    hiz_degeri=guncelleme.right(2).toInt();
}
if( Data[0]=='m' && Data[1]=='i') {
motorileri(hiz_degeri);
}
else if(Data[0]=='t' && Data[1]=='i') {
tamyolileri(hiz_degeri);
}
else if(Data[0]=='t' && Data[1]=='g') {
tamyolgeri(hiz_degeri);
}
else if(Data[0]=='m' && Data[1]=='g') {
motorgeri(hiz_degeri);
}
else if(Data[0]=='m' && Data[1]=='l') {
soladon();
}
else if(Data[0]=='m' && Data[1]=='r') {
sagadon();
}
else if(Data[0]=='m' && Data[1]=='s') {
motordur();
}
}

void threat::disconnected()
{
    softPwmWrite(29,0);
    softPwmWrite(27,0);

    digitalWrite(25, LOW);
    digitalWrite(24, LOW);
    digitalWrite(23, LOW);
    digitalWrite(22, LOW);
    qDebug() << socketDescriptor << " Baglanti Kesildi.";
    socket->deleteLater(); //Soketi temizle ve kapat.
    exit(0);           //threati kapat.
}

```

C.1.2 Servo Motor Kontrol Programı

Bu kısımda TCP/IP üzerinden gelen komutlarla pan-tilt mekanizmasını kontrol ederek kameraların bakış açılarını değiştirebilen bir C++ tabanlı kodlar verilmiştir.

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include "caprazuygulama.h"
#include <string>
#include <stdlib.h>
#include <QtGlobal>
using namespace std;

int x_ekseni=50;
int y_ekseni=50;

MyThread::MyThread(qint32 ID, QObject *parent) :
    QThread(parent)
{
    wiringPiSetup();

this->socketDescriptor = ID;
}

void MyThread::run()
{
    fp = fopen("/dev/servoblaster", "w");
    QString komut = "echo 1="+QString::number(y_ekseni)+"% >
/dev/servoblaster";
    system(qPrintable(komut));
    QString komut2 = "echo 0="+QString::number(x_ekseni)+"% >
/dev/servoblaster";
    system(qPrintable(komut2));

    if (fp == NULL) {
        printf("Error opening file\n");
        exit(0);
    }
    // thread starts here
    qDebug() << " Thread started";

    socket = new QTcpSocket();
```

```

if(!socket->setSocketDescriptor(this->socketDescriptor))

{
    emit error(socket->error());
    return;
}

connect(socket,      SIGNAL(readyRead()),      this,      SLOT(readyRead()),
Qt::DirectConnection);

connect(socket, SIGNAL(disconnected()), this, SLOT(disconnected()));

connect(socket,SIGNAL(readyRead()),this,SLOT(denemee()));

qDebug() << socketDescriptor << " Client connected";

exec();

}

void MyThread::readyRead()

{

}

void MyThread::disconnected()

{

    qDebug() << socketDescriptor << " Disconnected";

    socket->deleteLater();

    exit(0);

}

void MyThread::deneme()

{
}

```

```

void MyThread::denemee() {

    QByteArray Data = socket->readAll();

    qDebug()<< Data;

    if( Data[0]=='y' && Data[1]=='b') {

        if(y_ekseni>=90) {

            y_ekseni=85;

            QString komut = "echo 1="+QString::number(y_ekseni)+"% >
/dev/servoblaster";

            system(qPrintable(komut));

        }

        y_ekseni = y_ekseni+2;

        QString komut = "echo 1="+QString::number(y_ekseni)+"% >
/dev/servoblaster";

        system(qPrintable(komut));

    }

    if( Data[0]=='a' && Data[1]=='b') {

        if(y_ekseni<=0) {

            y_ekseni=5;

            QString komut = "echo 1="+QString::number(y_ekseni)+"% >
/dev/servoblaster";

            system(qPrintable(komut));

        }

        y_ekseni = y_ekseni-2;

        QString komut = "echo 1="+QString::number(y_ekseni)+"% >
/dev/servoblaster";

        system(qPrintable(komut));

    }

}

```

```

if( Data[0]=='l' && Data[1]=='b') {

    if(x_ekseni>=90) {

        x_ekseni=85;

        QString komut = "echo 0="+QString::number(x_ekseni)+"% >
/dev/servoblaster";

        system(qPrintable(komut));

    }

    x_ekseni = x_ekseni+2;

    QString komut = "echo 0="+QString::number(x_ekseni)+"% >
/dev/servoblaster";

    system(qPrintable(komut));

}

if( Data[0]=='s' && Data[1]=='b') {

    if(x_ekseni<=0) {

        x_ekseni=5;

        QString komut = "echo 0="+QString::number(x_ekseni)+"% >
/dev/servoblaster";

        system(qPrintable(komut));

    }

    x_ekseni = x_ekseni-2;

    QString komut = "echo 0="+QString::number(x_ekseni)+"% >
/dev/servoblaster";

    system(qPrintable(komut));

}

if( Data[0]=='h' && Data[1]=='m') {

    x_ekseni=50;

    y_ekseni=50;

    QString komut = "echo 0="+QString::number(x_ekseni)+"% >/dev/servoblaster";

    system(qPrintable(komut));
}

```

```

QString komut2 = "echo 1="+QString::number(y_ekseni)+"% > /dev/servoblaster";
system(qPrintable(komut2));
}
}

```

C.1.3 Sensör Programı

Keşif robotunda Arduino ile UART haberleşme üzerinden aldığı veriyi TCP/IP üzerinden keşif robotuna aktaran kodların kaynak kodu aşağıda verilmiştir.

```

#include "threat.h"

#include <QtSerialPort/QSerialPort>
#include <QtSerialPort/QSerialPortInfo>
#include <QString>

QByteArray sensor_verisi,veri;QByteArray sicaklik;QByteArray on_mesafe;
QByteArray arka_mesafe;QByteArray komutlar[4];
QByteArray aydinlik,batarya,gaz_degeri;QString deneme;

threat::threat(qint32 ID, QObject *parent) :
    QThread(parent)
{
    serial = new QSerialPort(this);

    socket = new QTcpSocket(this); //cift tarafli haberlesme icin bir adet soket acalim.

    serial->setPortName("ttyUSB0");
    serial->setBaudRate(QSerialPort::Baud115200);
    serial->setDataBits(QSerialPort::Data8);
    serial->setParity(QSerialPort::NoParity);
    serial->setStopBits(QSerialPort::OneStop);
    serial->setFlowControl(QSerialPort::NoFlowControl);
    serial->open(QIODevice::ReadWrite);

    this->socketDescriptor = ID; }

```

```

void threat::run()
{
    qDebug() << "Threat Olusturuldu ve veriler okunmaya hazir./";

    komutlar[0]="h";
    komutlar[1]="b";
    komutlar[2]="k";
    komutlar[3]="o";

    //baglanan pcnin idsine gore doğru bir haberleşme icin idleri aynı yapalım.

    if(!socket->setSocketDescriptor(this->socketDescriptor))

    {
        //Egerki socket acilamadi ise;

        qDebug() <<"Threat haberlesme soketi acilamadi.Threat sonlandirlacak./";

        emit error(socket->error());

        return;
    }

    connect(socket,      SIGNAL(readyRead()),      this,      SLOT(readyRead()),
Qt::DirectConnection);

connect(serial,SIGNAL(readyRead()),this,SLOT(seriveri()));

    connect(socket, SIGNAL(disconnected()), this, SLOT(disconnected()));

    qDebug() << "Bu id icin baglanti kuruldu = " <<socketDescriptor;

    exec(); //bu threati sonsuz dongü yapıyoruz.

}

void threat::seriveri()
{
    veri=serial->readAll();

    qDebug() << veri;

}

```

```
void threat::readyRead() //Veriler okunmaya basladiğin an calis
{
    QByteArray gelen_veri = socket->readAll();

    if(gelen_veri[0]=='B' && gelen_veri[1]=='A') {
        switch(gelen_veri[2]) {
            case 'h' :
                if(serial->isWritable()) {
                    serial->write(komutlar[0]);
                    serial->flush();
                }
                break;
            case 'b' :
                if(serial->isWritable()) {
                    serial->write(komutlar[1]);
                    serial->flush();
                }
                break;
            case 'k' :
                if(serial->isWritable()) {
                    serial->write(komutlar[2]);
                    serial->flush();
                }
                break;
            case 'o' :
                if(serial->isWritable()) {
                    serial->write(komutlar[3]);
                    serial->flush();
                }
        }
    }
}
```

```
break;

default :
    if(serial->isWritable()) {
        serial->write(komutlar[2]);
        serial->flush();
        serial->write(komutlar[3]);
        serial->flush();
    }
    break;
}

}

socket->write(veri);
socket->flush();
}

void threat::disconnected()
{
    qDebug() << socketDescriptor << " Baglanti Kesildi.";
    socket->deleteLater(); //Soketi temizle ve kapat.
    serial->close();
    exit(0);           //threati kapat.
}
```

C.1.4 Sensör Programı

Keşif robottu ile kontrolör tablet arasındaki bağlantıyı kontrol eden ping yazılımıdır.,

```
#include "threat.h"

threat::threat(qint32 ID, QObject *parent) :
    QThread(parent)
{
    this->socketDescriptor = ID;
}

void threat::run()
{
    qDebug() << "Threat Olusturuldu ve veriler okunmaya hazir./";

    socket = new QTcpSocket();      //cift tarafli haberlesme icin bir adet soket
acalim.

    //baglanan penin idsine gore doğru bir haberleşme icin idleri aynı yapalım.

    if(!socket->setSocketDescriptor(this->socketDescriptor))
    {
        //Egerki socket acilamadi ise;
        qDebug() <<"Threat haberlesme soketi acilamadi.Threat sonlandirlacak.";
        emit error(socket->error());
        return;
    }

    connect(socket, SIGNAL(readyRead()), this, SLOT(readyRead()),
Qt::DirectConnection);

    connect(socket, SIGNAL(disconnected()), this, SLOT(disconnected()));

    qDebug() << "Bu id icin baglanti kuruldu = " <<socketDescriptor;
    exec();   //bu threati sonsuz dongü yapıyoruz.

}
```

```

void threat::readyRead() //Veriler okunmaya basladigin an calis
{
    QByteArray gelen_veri = socket->readAll();

    qDebug() << "Bu id tanimli pc'den veri alindi : " << socketDescriptor << "
Alinan Veri : " << gelen_veri;

    if(gelen_veri[0]=='p' && gelen_veri[1]=='k') {
        qDebug() <<"ping mesaji geldi";
        socket->write("ptt");
    }
}

void threat::disconnected()
{
    qDebug() << socketDescriptor << " Baglanti Kesildi.";
    socket->deleteLater(); //Soketi temizle ve kapat.
    exit(0);
}

```

C.2 Kontrolör Tablet Üzerinde Çalışan Kodlar

Kontrolör tabletin keşif robotunu uzaktan komuta edebilmesini sağlayan kodlar C# üzerinde Windows Form ve WPF formları kullanılarak yazılmıştır. Bu kısımlarda designer dosyaları verilmeyecektir. Sadece ana çalışan kodlar verilecektir.

C.2.1 Motor Kontrol Yazılımı (Windows WPF)

Bu WPF yapısı sayesinde yazılan yazılım, transparant yapılmış ve ekranda sol altta çalışacak şekilde kodlar yazılmıştır. Bu program keşif robottu ile bağlantı kurar ve keşif robottu hareket ettirir.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Net.Sockets;
namespace WpfApplication3
{
    public partial class MainWindow : Window
    {
        System.Net.Sockets.TcpClient raspberry = new
        System.Net.Sockets.TcpClient();
        System.Windows.Threading.DispatcherTimer dispatcherTimer = new
        System.Windows.Threading.DispatcherTimer();
        string varsayilan_ip = "192.168.173.124";
        Int32 varsayilan_port = 1234;
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

```
dispatcherTimer.Tick += new EventHandler(dispatcherTimer_Tick);

dispatcherTimer.Interval = TimeSpan.FromSeconds(2);

dispatcherTimer.Start();

BitmapImage bitimg = new BitmapImage();

bitimg.BeginInit();

bitimg.UriSource = new Uri(@"C:\Robot\resimler\yukari.jpg",
UriKind.RelativeOrAbsolute);

bitimg.EndInit();

BitmapImage bitimg2 = new BitmapImage();

bitimg2.BeginInit();

bitimg2.UriSource = new Uri(@"C:\Robot\resimler\asagi.png",
UriKind.RelativeOrAbsolute);

bitimg2.EndInit();

BitmapImage bitimg3 = new BitmapImage();

bitimg3.BeginInit();

bitimg3.UriSource = new Uri(@"C:\Robot\resimler\sol.png",
UriKind.RelativeOrAbsolute);

bitimg3.EndInit();

BitmapImage bitimg4 = new BitmapImage();

bitimg4.BeginInit();

bitimg4.UriSource = new Uri(@"C:\Robot\resimler\sag.png",
UriKind.RelativeOrAbsolute);

bitimg4.EndInit();

BitmapImage bitimg5 = new BitmapImage();

bitimg5.BeginInit();

bitimg5.UriSource = new Uri(@"C:\Robot\resimler\dur.png",
UriKind.RelativeOrAbsolute);

bitimg5.EndInit();
```

```
Image img = new Image();
    img.Stretch = Stretch.Fill;
    img.Source = bitimg;

    Image img2 = new Image();
    img2.Stretch = Stretch.Fill;
    img2.Source = bitimg2;

    Image img3 = new Image();
    img3.Stretch = Stretch.Fill;
    img3.Source = bitimg3;

    Image img4 = new Image();
    img4.Stretch = Stretch.Fill;
    img4.Source = bitimg4;

    Image img5 = new Image();
    img5.Stretch = Stretch.Fill;
    img5.Source = bitimg5;

    Button.Content = img;
    Button2.Content = img2;
    Button3.Content = img3;
    Button4.Content = img4;
    Button5.Content = img5;

    Button.Background = new ImageBrush(bitimg);
    Button2.Background = new ImageBrush(bitimg2);
    Button3.Background = new ImageBrush(bitimg3);
    Button4.Background = new ImageBrush(bitimg4);
    Button5.Background = new ImageBrush(bitimg5);

    string[] argument = Environment.GetCommandLineArgs();

    if (argument.Length > 1)      {
        varsayilan_ip = argument[1];
    }
```

```

try      {
    raspberry.Connect(varsayilan_ip, varsayilan_port);
}

catch (Exception a)      {
    MessageBox.Show("Raspberry Bağlantısı Kurulamadı.");
}

}

void raspberry_veri_gonder(string veri)      {
    try      {
        NetworkStream serverStream = raspberry.GetStream(); //server stream
adında iletişim ağrı kurduk ve arduino adındaki client ile ilişkilendirdik

        byte[] gonderi = System.Text.Encoding.ASCII.GetBytes(veri); //byte
olarak bir gonderi oluşturduk ve encoding tipini ascii belirledik verilerimizi bu
stream üzerinden yollayacağız

        serverStream.Write(gonderi, 0, gonderi.Length); //outStream ile
göndereceğimiz veriyi tcp üzerinden arduinoya gönderiyoruz.

        serverStream.Flush(); //veriyi gönderdikten sonra tcp hattımızı
temizliyoruz.

    }
}

catch      {
    MessageBox.Show("Mesaj Gonderilirken/Alinirken Hata Olustu");
}

}

private void dispatcherTimer_Tick(object sender, EventArgs e)
{
    raspberry_veri_gonder("pk");
}

```

```

private void Button_TouchDown(object sender, TouchEventArgs e)
{
    raspberry_veri_gonder("mi");
}

private void Button_TouchUp(object sender, TouchEventArgs e)
{
    raspberry_veri_gonder("ms");
}

private void Button3_TouchDown(object sender, TouchEventArgs e)
{
    raspberry_veri_gonder("ml");
}

private void Button3_TouchUp(object sender, TouchEventArgs e)
{
    raspberry_veri_gonder("ms");
}

private void Button2_TouchDown(object sender, TouchEventArgs e)
{
    raspberry_veri_gonder("mg");
}

private void Button2_TouchUp(object sender, TouchEventArgs e)
{
    raspberry_veri_gonder("ms");
}

private void Button4_TouchDown(object sender, TouchEventArgs e)
{
    raspberry_veri_gonder("mr");
}

private void Button5_TouchDown(object sender, TouchEventArgs e)
{
    raspberry_veri_gonder("ms");
}

private void Button6_TouchDown(object sender, TouchEventArgs e)
{
    this.Close();
}

private void Button4_TouchUp(object sender, TouchEventArgs e)
{
    raspberry_veri_gonder("ms");
}
}
}

```

C.2.2 Servo Kontrol Yazılımı (Windows WPF)

Bu WPF yazılım ile TCP/IP protokolü kullanılarak keşif robottuna komut gönderilir ve pant-tilt mekanizması çalıştırılır.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Net.Sockets;
namespace WpfApplication3
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        System.Net.Sockets.TcpClient raspberry = new
System.Net.Sockets.TcpClient();

        string varsayilan_ip = "192.168.1.11";
        Int32 varsayilan_port = 1235;

        public MainWindow()
        {
            InitializeComponent();
            BitmapImage bitimg = new BitmapImage();
            bitimg.BeginInit();
            bitimg.UriSource = new
Uri(@"C:\Robot\resimler\asagi.png",
UriKind.RelativeOrAbsolute);
            bitimg.EndInit();

            BitmapImage bitimg2 = new BitmapImage();
            bitimg2.BeginInit();
            bitimg2.UriSource = new
Uri(@"C:\Robot\resimler\yukari.jpg",
UriKind.RelativeOrAbsolute);
            bitimg2.EndInit();
            BitmapImage bitimg3 = new BitmapImage();
            bitimg3.BeginInit();
            bitimg3.UriSource = new
Uri(@"C:\Robot\resimler\sag.png",
UriKind.RelativeOrAbsolute);
            bitimg3.EndInit();
        }
    }
}

```

```

BitmapImage bitimg4 = new BitmapImage();
bitimg4.BeginInit();
bitimg4.UriSource = new
Uri(@"C:\Robot\resimler\sol.png",
UriKind.RelativeOrAbsolute);
bitimg4.EndInit();

BitmapImage bitimg5 = new BitmapImage();
bitimg5.BeginInit();
bitimg5.UriSource = new
Uri(@"C:\Robot\resimler\dur2.png",
UriKind.RelativeOrAbsolute);
bitimg5.EndInit();

Image img = new Image();
img.Stretch = Stretch.Fill;
img.Source = bitimg;
Image img2 = new Image();
img2.Stretch = Stretch.Fill;
img2.Source = bitimg2;
Image img3 = new Image();
img3.Stretch = Stretch.Fill;
img3.Source = bitimg3;
Image img4 = new Image();
img4.Stretch = Stretch.Fill;
img4.Source = bitimg4;
Image img5 = new Image();
img5.Stretch = Stretch.Fill;
img5.Source = bitimg5;

Button.Content = img;
// Set Button.Content
Button2.Content = img2;
Button3.Content = img3;
Button4.Content = img4;
Button5.Content = img5;
// Set Button.Background
Button.Background = new ImageBrush(bitimg);
Button2.Background = new
ImageBrush(bitimg2);
Button3.Background = new
ImageBrush(bitimg3);
Button4.Background = new
ImageBrush(bitimg4);
Button5.Background = new
ImageBrush(bitimg5);
string[] argument =
Environment.GetCommandLineArgs();

```

```

if (argument.Length > 1)
{
    varsayilan_ip = argument[1];
}
try
{
    raspberry.Connect(varsayilan_ip,
varsayilan_port);
}
catch (Exception a)
{
    MessageBox.Show("Raspberry Bağlantısı
Kurulamadı.");
}

void raspberry_veri_gonder(string veri)
{
    NetworkStream serverStream =
raspberry.GetStream(); //server stream adında iletişim
ağrı kurduk ve arduino adındaki client ile
ilişkilendirdik
    byte[] gonderi =
System.Text.Encoding.ASCII.GetBytes(veri); //byte olarak
bir gönderi oluşturduk ve encoding tipini ascii
belirledik verilerimizi bu stream üzerinden
yollayacağız
    serverStream.Write(gonderi, 0,
gonderi.Length); //outStream ile göndereceğimiz veriyi
tcp üzerinden arduinoya gönderiyoruz.
    serverStream.Flush(); //veriyi gönderdikten
sonra tcp hattımızı temizliyoruz.
}

private void Button_TouchDown(object sender,
TouchEventArgs e)
{
    raspberry_veri_gonder("yb");
}

private void Button3_TouchDown(object sender,
TouchEventArgs e)
{
    raspberry_veri_gonder("sb");
}
private void Button2_TouchDown(object sender,
TouchEventArgs e)
{
    raspberry_veri_gonder("ab");
}

```

```
private void Button6_TouchDown(object sender,
TouchEventArgs e)
{
    this.Close();
}
private void Button_MouseDown(object sender,
MouseButtonEventArgs e)
{
    raspberry_veri_gonder("yb");
}
private void Button_MouseUp(object sender,
MouseButtonEventArgs e)
{
}
private void Button2_MouseDown(object sender,
MouseButtonEventArgs e)
{
    raspberry_veri_gonder("ab");
}
private void Button2_MouseUp(object sender,
MouseButtonEventArgs e)
{
}
private void Button_Click(object sender,
RoutedEventArgs e)
{
    raspberry_veri_gonder("yb");
}
private void Button2_Click(object sender,
RoutedEventArgs e)
{
    raspberry_veri_gonder("ab");
}
private void Button3_Click(object sender,
RoutedEventArgs e)
{
    raspberry_veri_gonder("sb");
}
private void Button4_Click(object sender,
RoutedEventArgs e)
{
    raspberry_veri_gonder("lb");
}
private void Button5_Click(object sender,
RoutedEventArgs e)
{
    raspberry_veri_gonder("hm");
}
}
```

C.2.3 Sensör Kontrol Yazılımı (Windows WPF)

Aşağıdaki kodlar ile sensör barları yaratılmış ve TCP/IP üzerinden gelen sensör verilerinin uygun yere yerleştirilmesi sağlanmıştır.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO;
using System.Globalization;
using System.Net.Sockets;
using System.Threading;//Eklediğimiz .net için socket
kütüphanemiz
namespace Sensorler
{
    public partial class MainWindow : Window
    {
        string str, str2; string aydinlik,gaz_degeri,
batarya; float batarya_voltaj,sicaklik_degeri;
        string varsayılan_ip = "192.168.1.7";

        private string sicaklik = "";
        private int on_mesafe = 0;
        private int arka_mesafe = 0;
        private int sayac, sayac2 = 0; Int32 x;
        Int32 varsayılan_port = 1236;
        Thread t;// yeni thread surekli olarak
arduinodan veri kontrolü
        System.Net.Sockets.TcpClient arduino = new
System.Net.Sockets.TcpClient(); //arduinom adında yeni
bir tcp client tanımladık
        String gelen_veri;
        System.Windows.Threading.DispatcherTimer
dispatcherTimer = new
System.Windows.Threading.DispatcherTimer();
```

```

public MainWindow()
{
    InitializeComponent();
    dispatcherTimer.Tick += dispatcherTimer_Tick;
    dispatcherTimer.Interval =
TimeSpan.FromMilliseconds(300); ;
    dispatcherTimer.Start();
    string[] argument =
Environment.GetCommandLineArgs();
    if (argument.Length > 1)
    {
        varsayilan_ip = argument[1];
    }
    try
    {
        arduino.Connect(varsayilan_ip,
varsayilan_port);

    }
    catch (Exception a)
    {
        MessageBox.Show("Raspberry Bağlantısı
Kurulamadı.");
    }
    t = new Thread(new ThreadStart(arduino_oku));// threadi
arduino oku fonksiyonuna bagliyoruz
    t.Start(); // baglanti saglandiktan sonra
thread basladi
}
void arduino_veri_gonder(string veri)
{
    try
    {
NetworkStream serverStream = arduino.GetStream
byte[] gonderi =
System.Text.Encoding.ASCII.GetBytes(veri); //byte olarak
bir gönderi oluşturduk ve encoding tipini ascii
belirledik verilerimizi bu stream üzerinden yollayacağız
    serverStream.Write(gonderi, 0,
gonderi.Length); //outStream ile göndereceğimiz veriyi tcp
üzerinden arduinoya gönderiyoruz.
    serverStream.Flush(); //veriyi
gönderdikten sonra tcp hattımızı temizliyoruz.
    }
    catch
    {
        MessageBox.Show("Mesaj
Gonderilirken/Alinirken Hata Olustu");
    }
}

```

```

void arduino_oku()
{
    NetworkStream st = arduino.GetStream();
    while (true)
    {

        if (st.DataAvailable)
        {

            byte[] data = new byte[100];

            using (MemoryStream ms = new
MemoryStream())
            {

                int numBytesRead;
                while ((numBytesRead =
st.Read(data, 0, data.Length)) > 0)
                {
                    ms.Write(data, 0,
numBytesRead);
                    break;
                }

                str =
Encoding.ASCII.GetString(ms.ToArray(), 0,
(int)ms.Length);

                string[] degerler = str.Split('-');

                try
                {

sicaklik_label.Dispatcher.BeginInvoke((Action)() =>
sicaklik_label.Content = degerler[5].ToString()));

bataryalabel.Dispatcher.BeginInvoke((Action)() =>
bataryalabel.Content = degerler[4].ToString()));

label1.Dispatcher.BeginInvoke((Action)() =>
label1.Content = degerler[2].ToString()));

label2.Dispatcher.BeginInvoke((Action)() =>
label2.Content = degerler[3].ToString()));

gaz_label1.Dispatcher.BeginInvoke((Action)() =>
gaz_label1.Content = degerler[1].ToString()));

```

```

private void dispatcherTimer_Tick(object sender,
EventArgs e)
{
    arduino_veri_gonder("hello");
    Thread.Sleep(3000);
}

private void far_TouchDown(object sender,
TouchEventArgs e)
{
    sayac++;
    int sayac_degeri3 = sayac % 2;

    if (sayac_degeri3 == 1)
        arduino_veri_gonder("BAh");
    else
        arduino_veri_gonder("BAo");
}

private void buzzer_TouchDown(object sender,
TouchEventArgs e)
{
    sayac2++;
    int sayac_degeri3 = sayac2 % 2;

    if (sayac_degeri3 == 1)
        arduino_veri_gonder("BAb");
    else
        arduino_veri_gonder("BAk");
}

private void b_Click(object sender,
RoutedEventArgs e)
{
    sayac++;
    int sayac_degeri3 = sayac % 2;

    if (sayac_degeri3 == 1)
        arduino_veri_gonder("BAh");
    else
        arduino_veri_gonder("BAo");
}
}

```

C.2.4 Ana arayüz programı Yazılımı (Windows FORM)

Bu kısımda gerekli AFORGE.net kütüphaneleri ile iki kamera için MJPEG streamlarının alınmasını sağlayan kodlar verilmiştir.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;
using System.Drawing.Imaging;
using AForge.Video;
using System.IO;
using System.Net.Sockets;
using System.Runtime.InteropServices;
namespace RobotArayuz
{
    public partial class Form1 : Form
    {
        [DllImport("user32.dll", SetLastError = true)]
        internal static extern bool MoveWindow(IntPtr hWnd, int
X, int Y, int nWidth, int nHeight, bool bRepaint);

        private const int statLength = 15;
        // current statistics index
        private int statIndex = 0;
        // ready statistics values
        private int statReady = 0;
        // statistics array
        private int sayac = 0;
        private int sayac2,sayac3 = 0;
        String gelen_veri;
        private int[] statCount = new int[statLength];
        private IVideoSource videoSource = null;
        Process[] prg, prg2, prg3, prg4, prg5,prg6;
        string varsayılan_ip="192.168.1.10";
        Int32 varsayılan_port=8080;
        Int32 varsayılan_port2 = 8081;
```

```

public Form1()
{
    string[] argument =
Environment.GetCommandLineArgs();
    if (argument.Length > 1)
    {
        varsayilan_ip = argument[1];
        varsayilan_port =
Int32.Parse(argument[2]);
    }

    InitializeComponent();
    // this.TopMost = true;
    this.FormBorderStyle = FormBorderStyle.None;
    this.WindowState =
FormWindowState.Maximized;
    IVideoSource videoSource =
videoSourcePlayer1.VideoSource;

    pictureBox1.Image =
Image.FromFile(@"C:\Robot\resimler\onizleme.jpg");
    pictureBox1.SizeMode =
PictureBoxSizeMode.StretchImage;
    textBox1.Text = varsayilan_ip;
    textBox2.Text = varsayilan_port.ToString();

    pictureBox2.Visible = false;
pictureBox3.Visible = false; pictureBox4.Visible =
false; pictureBox5.Visible = false;

}

private void OpenVideoSource(IVideoSource
source)
{
    CloseVideoSource();
    videoSourcePlayer1.VideoSource = new
AsyncVideoSource(source);
    videoSourcePlayer1.Start();
    statIndex = statReady = 0;
    timer1.Start();
    videoSource = source;
}

```

```
private void CloseVideoSource()      {
    videoSourcePlayer1.SignalToStop();
    for (int i = 0; (i < 50) && (videoSourcePlayer1.IsRunning); i++)
    {
        Thread.Sleep(100);
    }
    if (videoSourcePlayer1.IsRunning)
        videoSourcePlayer1.Stop();
    timer1.Stop();
}
private void Form1_Load(object sender, EventArgs e)
{
    videoSourcePlayer1.SignalToStop();
    CloseVideoSource();
    string ip = textBox1.ToString();
    string port = "8080";
    string protokol = "http://";
    string slash = ":";
    string erisim = "/?action=stream";
    string baglanti_linki = string.Concat(protokol, textBox1.Text, slash, port, erisim);
    MJPEGStream mjpegSource = new MJPEGStream(baglanti_linki);
    OpenVideoSource(mjpegSource);
}
```

```

private void button1_Click(object sender, EventArgs e)
{
    prg = Process.GetProcessesByName("TankMotor");
    if (prg.Length > 0)
        prg[0].Kill();
    prg2 = Process.GetProcessesByName("ServoMotor");
    if (prg2.Length > 0)
        prg2[0].Kill();
    prg3 = Process.GetProcessesByName("Sensorler");
    if (prg3.Length > 0)
        prg3[0].Kill();
    prg5 = Process.GetProcessesByName("ScreenRecord");
    if (prg5.Length > 0)
        prg5[0].Kill();
    prg6 = Process.GetProcessesByName("Pingkontrol");
    if (prg6.Length > 0)
        prg6[0].Kill();
    CloseVideoSource();
    this.Close();
}

private void button6_Click(object sender, EventArgs e)
{
    sayac = sayac + 1;
    int sayac_degeri = sayac % 2;
    if (sayac_degeri == 1)
    {
        prg = Process.GetProcessesByName("TankMotor");
        if (prg.Length > 0)
            prg[0].Kill();
        prg2 = Process.GetProcessesByName("ServoMotor");
        if (prg2.Length > 0)
            prg2[0].Kill();
        prg3 = Process.GetProcessesByName("Sensorler");
        if (prg3.Length > 0)
            prg3[0].Kill();
        prg5 = Process.GetProcessesByName("ScreenRecord");
        if (prg5.Length > 0)
            prg5[0].Kill();
        prg6 = Process.GetProcessesByName("Pingkontrol");
        if (prg6.Length > 0)
            prg6[0].Kill();
    }
}

```

```

else if (sayac_degeri == 0)
{
System.Diagnostics.Process.Start(@"C:\Robot\EkranKayit\ScreenRecord.exe");

ProcessStartInfo motorkontrol = new ProcessStartInfo();
motorkontrol.FileName = @"C:\Robot\TankMotor.exe";
motorkontrol.Arguments = varsayılan_ip;
Process.Start(motorkontrol);
Thread.Sleep(500);
ProcessStartInfo servokontrol = new ProcessStartInfo();
servokontrol.FileName = @"C:\Robot\ServoMotor.exe";
servokontrol.Arguments = varsayılan_ip;
Process.Start(servokontrol);
ProcessStartInfo pingkontrol = new ProcessStartInfo();
pingkontrol.FileName = @"C:\Robot\Pingkontrol.exe";
pingkontrol.Arguments = varsayılan_ip;
Process.Start(pingkontrol);
Thread.Sleep(500);
ProcessStartInfo sensorler = new ProcessStartInfo();
sensorler.FileName = @"C:\Robot\Sensorler.exe";
sensorler.Arguments = varsayılan_ip;
Process.Start(sensorler);
}

}

private void button3_Click(object sender, EventArgs e)
{
button3.BackColor = Color.Green;
Bitmap bitmap =
(Bitmap)videoSourcePlayer1.GetCurrentVideoFrame().Clone();

DateTime date = DateTime.Now;
String fileName =
String.Format(@"C:\Robot\ekrangoRuntuleri\goruntu-{0}-{1}-
{2} {3}-{4}-{5}.jpg",
date.Year, date.Month, date.Day, date.Hour, date.Minute,
date.Second);
bitmap.Save(fileName, ImageFormat.Jpeg);
pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
pictureBox1.Image = bitmap;
pictureBox1.Visible = true;
button7.Visible = true;
}
private void button7_Click(object sender, EventArgs e)
{
pictureBox1.Visible = false;
button7.Visible = false;
}

```

```
private void pictureBox1_Click(object sender, EventArgs e)
{
    Process.Start("explorer.exe", @"C:\Robot\ekrangoruntuleri");
}

private void button8_Click(object sender, EventArgs e)
{
    string ip = textBox1.ToString();
    string port = textBox2.ToString();
    string protokol = "http://";
    string slash = ":";
    string erisim = "?action=stream";
    string baglanti_linki = string.Concat(protokol, textBox1.Text, slash,
    textBox2.Text, erisim);
    MJPEGStream mjpegSource = new MJPEGStream(baglanti_linki);
    OpenVideoSource(mjpegSource);
}

private void button4_Click(object sender, EventArgs e)
{
}

private void timer2_Tick(object sender, EventArgs e)
{
    label8.Text = DateTime.Now.ToString();
    button3.BackColor = Color.Red;
}

private void button2_Click(object sender, EventArgs e)
{
    sayac3++;
    int sayac_degeri3 = sayac3 % 2;
```

```

private void button4_Click_1(object sender, EventArgs e)
{
    sayac2++;
    int sayac_degeri2 = sayac2 % 2;
    if (sayac_degeri2 == 1)
    {
        button4.BackColor = Color.Green;
        pictureBox2.Visible = true; pictureBox3.Visible = true;
        pictureBox4.Visible = true; pictureBox5.Visible = true;
    }
    else
    {
        button4.BackColor = Color.Red;
        pictureBox2.Visible = false; pictureBox3.Visible = false;
        pictureBox4.Visible = false; pictureBox5.Visible = false;
    }
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    videoSourcePlayer1.SignalToStop();
    CloseVideoSource();
    prg = Process.GetProcessesByName("TankMotor");
    if (prg.Length > 0)
        prg[0].Kill();
    prg2 = Process.GetProcessesByName("ServoMotor");
    if (prg2.Length > 0)
        prg2[0].Kill();
    prg3 = Process.GetProcessesByName("Sensorler");
    if (prg3.Length > 0)
        prg3[0].Kill();
}

```

```

prg5 = Process.GetProcessesByName("ScreenRecord");
    if (prg5.Length > 0)
        prg5[0].Kill();
prg6 = Process.GetProcessesByName("Pingkontrol");
    if (prg6.Length > 0)
        prg6[0].Kill();
}

private void button5_Click(object sender, EventArgs e)
{
}
}
}
}

```

C.2.5 Ping Kontrol Yazılımı (Windows WPF)

Aşağıda verilen, C# WinForm tabanlı yazılım ile keşif robottu ile bağlantı koptuğunda kullanıcıyı uyarın ping yazılımının kodlarıdır.

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net.Sockets;
namespace Pingkontrol
{
    public partial class Form1 : Form
    {
        Thread t;
        System.Net.Sockets.TcpClient arduinom = new
        System.Net.Sockets.TcpClient(); //arduinom adında yeni
        bir tcp client tanımladık
        String gelen_veri;
        String ping_mesaji = "pk";
        bool baglanti_durumu = false;
        string varsayılan_ip = "192.168.1.10";
        Int32 varsayılan_port = 1237;

```

```

public Form1()          {
    InitializeComponent();
    this.TransparencyKey = Color.Turquoise;
    this.BackColor = Color.Turquoise;
    this.Top = 200;
    this.Left = 0;           }
private void Form1_Load(object sender, EventArgs e)
{
    pictureBox1.Image =
Image.FromFile(@"c:/Robot/resimler/baglantiyok.jpg");
    pictureBox1.SizeMode =
PictureBoxSizeMode.StretchImage;
    string[] argument =
Environment.GetCommandLineArgs();
    if (argument.Length > 1)
    {
        varsayilan_ip = argument[1];
    }
    try
    {
arduinom.Connect(varsayilan_ip, varsayilan_port);// 23
portu üzerinde arduino ip si ile baglanmayı deniyoruz
        MessageBox.Show("Robot ile Baglantı
Saglandı!"); // baglantı saglanırsa ekrana baglantı
saglandı bilgisi basılıyor
        pictureBox1.Image =
Image.FromFile(@"c:/Robot/resimler/ping.gif");
        pictureBox1.SizeMode =
PictureBoxSizeMode.StretchImage;
        baglanti_durumu = true;
        t = new Thread(new
ThreadStart(raspberry_oku)); // threadi arduino oku
fonksiyonuna baglıyoruz
        t.Start(); // baglantı saglandıktan sonra
thread basladı
        timer2.Start();
    }
    catch (Exception a) // baglantı kurulamadı ise
    {
        MessageBox.Show("baglantı yok tekrar
baglanmayı deneyiniz"); //baglantı yok tekrar baglan
uyarısı ekrana bastırılıyor
        pictureBox1.Image =
Image.FromFile(@"c:/Robot/resimler/baglantiyok.jpg");
        pictureBox1.SizeMode =
PictureBoxSizeMode.StretchImage;
        baglanti_durumu = false;
    }
}

```

```

void raspberry_veri_gonder(string veri)
{
    try
    {
        NetworkStream serverStream =
arduinom.GetStream(); //server stream adında iletişim ağı
kurduk ve arduino adındaki client ile ilişkilendirdik
        byte[] gonderi =
System.Text.Encoding.ASCII.GetBytes(veri); //byte olarak
bir gönderi oluşturduk ve encoding tipini ascii belirledik
verilerimizi bu stream üzerinden yollayacağız
        serverStream.Write(gonderi, 0,
gonderi.Length); //outStream ile göndereceğimiz veriyi tcp
üzerinden arduinoya gönderiyoruz.
        serverStream.Flush(); //veriyi gönderdikten
sonra tcp hattımızı temizliyoruz.
    }
    catch (Exception a) // bağlantı kurulamadı ise
    {
        pictureBox1.Image =
Image.FromFile(@"c:/Robot/resimler/baglantikoptu.jpg");
        pictureBox1.SizeMode =
PictureBoxSizeMode.StretchImage;
        baglanti_durumu = false;
    }

}
void raspberry_oku()
{
    NetworkStream st = arduinom.GetStream();
    byte[] myReadBuffer = new byte[10];
    while (true)
    {
        if (st.DataAvailable)
        {
            st.BeginRead(myReadBuffer, 0,
myReadBuffer.Length,
null,
null);
        }
        gelen_veri =
System.Text.Encoding.UTF8.GetString(myReadBuffer);
        timer3.Start();
        Thread.Sleep(100);
    }

}

```

```

private void labell_Click(object sender, EventArgs e)
{
}

private void timer2_Tick(object sender, EventArgs e)
{
    raspberry_veri_gonder(ping_mesaji);
}

private void timer3_Tick(object sender, EventArgs e)
{
    pictureBox1.Image =
Image.FromFile(@"c:/Robot/resimler/baglantikoptu.jpg");
    pictureBox1.SizeMode =
PictureBoxSizeMode.StretchImage;
    baglanti_durumu = false;
    arduinom.Client.Disconnect(true);
    timer3.Stop();
    timer2.Stop();
    t.Abort();
}

private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
{
    t.Abort();
    Environment.Exit(0);
}

}

```

C.3 Arduino Yazılımı

Keşif robotunda sensör okuma işlevini gören kodlar aşağıda verilmiştir.

```

float sicaklik; //Analog değeri dönüştüreceğimiz sıcaklık
float analoggerilim,analoggerilim2; //Ölçeceğimiz analog
değer
String inputString = ""; // a string to hold
incoming data
boolean stringComplete = false; // whether the string is
complete

#define echoPin 4 // Echo Pin
#define trigPin 3 // Trigger Pin
String veri;
#define echoPin2 6 // Echo Pin
#define trigPin2 5 // Trigger Pin
int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration,duration2, distance,distance2; // Duration
used to calculate distance
int gaz_degeri,batarya=0;
float vout;
void setup () {
pinMode(trigPin, OUTPUT);
pinMode(8, OUTPUT);
pinMode(7, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
inputString.reserve(200);
Serial.begin(115200); //Seri haberleşme,Sıcaklığını ekranda
görücez
}

void onmesafe() {
digitalWrite(trigPin2, LOW);
delayMicroseconds(2);
digitalWrite(trigPin2, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin2, LOW);
duration2 = pulseIn(echoPin2, HIGH);
distance2 = duration2/58.2;
delay(50);
}

```

```

void sicaklikolc() {
    analoggerilim = analogRead(A0); //A1'den değeri ölç
    analoggerilim = (analoggerilim/1023)*5000;//değeri mV'a
    dönüştür
    sicaklik = analoggerilim /10,0; // mV'u sıcaklığı
    dönüştür

    delay(50);
}

void gaz() {
    gaz_degeri = analogRead(A5); //A1'den değeri ölç
    delay(50);
}

void bataryaolc() {
    analoggerilim2 = analogRead(A3); //A1'den değeri ölç
    analoggerilim2 = (analoggerilim2/1023)*5000;//değeri mV'a
    dönüştür
    vout = (analoggerilim2 * 5.0) / 1024.0;
}

void loop () {

    if (stringComplete) {
        if(String(inputString)== "BAh")
            digitalWrite(8,HIGH);
        else if (String(inputString)== "BAo")
            digitalWrite(8,LOW);

        if(String(inputString)== "BAb\n")
            digitalWrite(7,HIGH);
        else if (String(inputString)== "BAk\n")
            digitalWrite(7,LOW);

        // clear the string:
        inputString = "";
        stringComplete = false;
    }
    gaz();
    onmesafe();
    arkamesafe();
    bataryaolc();
    sicaklikolc();
    delay(300);
}

```

```
veri += "-";
veri += gaz_degeri;
veri += "-";
veri += distance2;
veri += "-";
veri += distance;
veri += "-";
veri += vout*2;
veri += "-";
veri += sicaklik;
Serial.println(veri);
veri="";
Serial.flush();
}

void serialEvent() {
    while (Serial.available()) {
        // get the new byte:
        char inChar = (char)Serial.read();
        // add it to the inputString:
        inputString += inChar;
        // if the incoming character is a newline, set a flag
        // so the main loop can do something about it:
        if (inChar == '\n') {
            stringComplete = true;
        }
    }
}
```