**T.C.**

**MARMARA ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**MEKATRONİK MÜHENDİSLİĞİ**

# BİTİRME PROJESİ ARA RAPORU

# SES KONTROLÜ İLE SAĞLANAN AKILLI EV OTOMASYONU DİZAYNI

# 04.04.2018

# Doç. Dr. Mustafa Caner AKÜNER

# Mehmet Can YÜNEY – 170214013

# İÇİNDEKİLER

[İÇİNDEKİLER i](#_Toc511346498)

[ÖZET ii](#_Toc511346499)

[KISALTMALAR iii](#_Toc511346500)

[ŞEKİL LİSTESİ iv](#_Toc511346501)

[TABLO LİSTESİ v](#_Toc511346502)

[1. BOLÜM: GİRİŞ VE TEZİN AMACI 1](#_Toc511346503)

[1.1. ÇALIŞMANIN ENDÜSTİRİYEL ÖNEMİ 1](#_Toc511346504)

[1.2. ÇALIŞMANIN AMACI 2](#_Toc511346505)

[1.3. LİTERATÜR ÇALIŞMASI 3](#_Toc511346506)

[2. BÖLÜM: MATERYAL VE YÖNTEM 7](#_Toc511346507)

[2.1. TEZDE KULLANILACAK ARAŞTIRMA ARAÇLARI 7](#_Toc511346508)

[2.1.1. RASPBERRY Pİ 7](#_Toc511346509)

[2.1.1.1. RASPBERRY Pİ’IN TEKNİK ÖZELLİKLERİ 8](#_Toc511346510)

[2.1.1.2. Raspbian 9](#_Toc511346511)

[2.1.2. STM32F411E DİSCOVERY 9](#_Toc511346512)

[2.1.2.1. STM32F411E DİSCOVERY KİT’İN TEKNİK ÖZELLİKLERİ 9](#_Toc511346513)

[2.1.3. MODBUS HABERLEŞME PROTOKOLÜ 10](#_Toc511346514)

[2.1.3.1. MODBUS RTU 10](#_Toc511346515)

[2.1.3.2. RS485 11](#_Toc511346516)

[2.1.4. SM2028EC-TOUCH KLİMA KONTROL ÜNİTESİ 12](#_Toc511346517)

[2.1.4.1. SM2028EC-TOUCH KLİMA KONTROL ÜNİTESİ TEKNİK ÖZELLİKLERİ 12](#_Toc511346518)

[2.1.5. RÖLE 13](#_Toc511346519)

[2.1.6. SERVO MOTOR İLE PERDE SİSTEMİNİN KONTROLÜ 14](#_Toc511346520)

[KAYNAKLAR 15](#_Toc511346521)

# ÖZET

Yapılan bu çalışmada, Raspberry Pi ile alınan ses verileri işlenerek ev otomasyonu içerisinde bulunan cihazların ve sistemlerin kullanımının kolaylaştırılması amaçlanmıştır. Çalışmada klima kontrol sistemi için Modbus protokolü ile çalışan ve Ges Teknik tarafından üretilen SM2028EC-Touch kullanılırken, perde sistemi için servo motor ve ışıklandırma için röle kullanılmıştır. Çalışma ses verilerinin ev otomasyonuna komut olarak aktarılması için gerekli işlemleri yaparak, bu sistemlerin kontrolünü sağlar. Böylelikle ev otomasyonu içerisinde bulunan sistemler ses ile kontrol edilmiş olur. Yapılan kontrol sonucunda kullanıcıya yine ses yolu ile bilgilendirilir.

**Anahtar Kelimeler:** Ses Kontrolü, TTS(text to speech), STT(speech to text), Ev Otomasyonu, Modbus

# KISALTMALAR

HMI : Human Interface

AI : Artificial Intelligence

IoT : Internet of Things

TTS : Text To Speech

STT : Speech To Text

API : Application Programming Interface

GPIO : Generic Purpose Input Output

UART : Universal Asynchronous Receiver – Transmitter

RTU : Remote Terminal Unit

PWM : Pulse – Width Modulation

ms : Milisaniye

DC : Duty Cycle

# ŞEKİL LİSTESİ

Şekil 1.2 – Speech To Text Format

Şekil 1.3 – Öğrenme Mekanizması

Şekil 1.4 – Text To Speech Format

Şekil 1.5 – Klima Kontrol Sistemi

Şekil 2.1 – Sistemin Genel Planı

Şekil 2.2 – Raspberry Pi

Şekil 2.3 – Raspberry Pi Model B

Şekil 2.4 – STM32F411E Discovery Kit

Şekil 2.5 – Modbus Protokolü

Şekil 2.6 – Modbus Paket Gönderim Şekli

Şekil 2.7 – RS485 Mantığı

Şekil 2.8 – SM2028EC – Touch

Şekil 2.9 – Röle

Şekil 2.10 – Servo PWM

# TABLO LİSTESİ

Tablo 2.1 – Modbus kodları

Tablo 2.2 – Parametreler

# BOLÜM: GİRİŞ VE TEZİN AMACI

# ÇALIŞMANIN ENDÜSTİRİYEL ÖNEMİ

Gelişen teknoloji ile birlikte insanların günlük yaşamlarında gerçekleştirdiği bazı eylemler artık bilgisayarlar ve otomatik sistemler tarafından uygulanabilmektedir. Bu sistemler, insanların günlük işlerini daha kolay hale getirmek amacıyla hizmet verirken, insanın en uygun ve en rahat koşullarda yaşamasını sağlamak için geliştirilmektedirler.

Ev otomasyonu olarak adlandırabileceğimiz klima kontrolü, perde ve ışıklandırma kontrolü gibi sistemler artık bilgisayarların karar algoritmalarına bırakılmaya başladı. Gerek sensörler tarafından algılanan fiziksel büyüklükler sayesinde, gerek insanların bu sistemlere dışarıdan verdiği komutlar ile bu tür faaliyetler otomatikleştirildi. Bununla beraber, 21. Yüzyılın başlarında ortaya atılan akıllı ev projeleri hızla şekillenmeye başladı. Artık bu sistemler hem otomatik zamanlı veya dış etkenlere bağlı olarak insanların konforuna yönelik kararlar verebiliyor, hem de insanlar evlerinde değilken bile bazı sistemleri kontrol edebiliyorlar.

Ses kontrolü ve kontrol sonrası ortaya çıkan sonuçların, bilgisayar tarafından insanlara bildirilmek üzere sese çevrilmesi işleminin baz alındığı sistemler sadece ev ortamındaki kullanımıyla sınırlı kalmamış, birçok alanda da gündelik yaşamımıza girmeye başlamıştır. Örneğin, Google’ın sağlamış olduğu sistemlerin yanı sıra Apple firmasının ortaya çıkardığı SIRI projesi hem görme engelli bireylerin hem de herhangi bir engeli bulunmayan kullanıcıların ilgisini çekmeyi başarmış ve sistemin son derece doğru çalışması, insanlarda bu teknolojiye olan güveni artırmıştır. Sonuç olarak, eski teknolojide gerek kodla, gerek tuşla manual şekilde yapılması gereken işlemler artık ses yardımı ile kolayca yapılabilmektedir. Bunlara ek olarak, her ne kadar bu projenin amaçları arasında olmasa da, bahsedilen sistemler aynı zamanda endüstriyel alana da entegre edilebilir; ses aracılığıyla kontrol edilen SCADA’lar ile fabrikalardaki sistemler de insanların kolaylıkla kontrol edebileceği hale getirilebilir.

# ÇALIŞMANIN AMACI

Teknolojinin öncelikli amacı insan hayatını kolaylaştırmaktır. Kontrol sistemlerinde meydana getirilen çoğu yenilik ve geliştirmeler de günlük yaşamı pratikleştirmek üzerinedir. Bu projenin amacı ise, kullanıcıların ev ortamında gerçekleştirdiği bazı eylemleri ses aracılığı ile kontrol edebilmesidir. Yapılacak proje ile hem ev içerisinde gerçekleştirilen faaliyetlerin daha pratik hale getirilmesi hem de görme engelli bireylerin günlük yaşamlarının kolaylaştırılması ön görülmüştür.

Yapılacak sistemin amaçlarından biri, ev içindeki klima kontrolünün ses aracılığı ile sağlanmasıdır. Bir süredir özellikle PLC teknolojilerinin gelişimi ile birlikte, ev ve alışveriş merkezleri gibi iç ortamlarda bulunan klima sistemleri otomatikleştirilmeye başlamıştır. Bu sistemler, tüm evin ya da belirli bir odanın sıcaklık ve nem parametrelerini kişilerin en rahat olduğu değerlere göre ayarlayabilmektedir. Bu projede bahsedilen kontrol sistemi, ses aracılığı ile ara işlemciye gönderilen komutlar üzerine kurulu olacak ve bu komutlar sayesinde klima sistemlerindeki oda sıcaklığı, fan hızı gibi önemli parametrelerin kontrolü kolaylıkla sağlanabilecektir.

Sistemin bir diğer amacı ise ev içerisinde yer alan ışıklandırma, perde konumlandırmaları gibi mekanizmaların ses ile kontrol edilebilmesidir. Proje içerisinde kullanılacak sistem ne yapması gerektiğini kullanıcının söylediklerinden yani verdiği komutlardan anlayacaktır. Bu sistemin en önemli avantajlarından biri komutların kullanıcıya göre değişkenlik gösterecek olmasıdır. Diğer bir deyişle, sistemin aktive edildiği komutlar sisteme tanımlanmış kalıp cümleler halinde bulunmayacaktır. Sistem, kullanıcının verdiği komutu kelime kelime analiz edip bir sonuç çıkaracak ve bu sonuç doğrultusunda istenen faaliyeti gerçekleştirebilecektir. Herhangi bir sonuç elde edilemediği durumlarda ise sistem kullanıcıdan doğrulama isteyecektir. Böylelikle, sistemin ilk seferde algılayamadığı komutlar kullanıcı tarafından belirginleştirildikten sonra işlemci tarafından hafızaya atılarak kaydı tutulabilecektir. Bu sayede de sistem tekrar aynı komutu duyduğunda hangi işlemi yerine getirmesi gerektiğini anlayacak ve gerçekleştirebilecektir.

Sistem aynı zamanda kullanıcı ile etkileşimli olarak çalışacaktır. Başka bir ifadeyle, sistemin gerçekleştirdiği faaliyetler, kullanıcının isteğine bağlı olarak HMI’da bildirim şeklinde gözükebilecek ve bunun yanında görme engelli bireylerin de bu faaliyetlerden haberdar olabilmesi için ses mesajı şeklinde de kullanıcıya bildirilebilecektir.

Sonuç olarak sistemin genel amacı günlük yaşamda ev içerisinde yapılması gereken işlerin ses kontrolü ile daha pratik hale getirilmesi ve gerçekleştirilen bu işlemlerin sesli ve görüntülü mesaj yardımı ile kullanıcıya bildirilmesi şeklindedir.

# LİTERATÜR ÇALIŞMASI

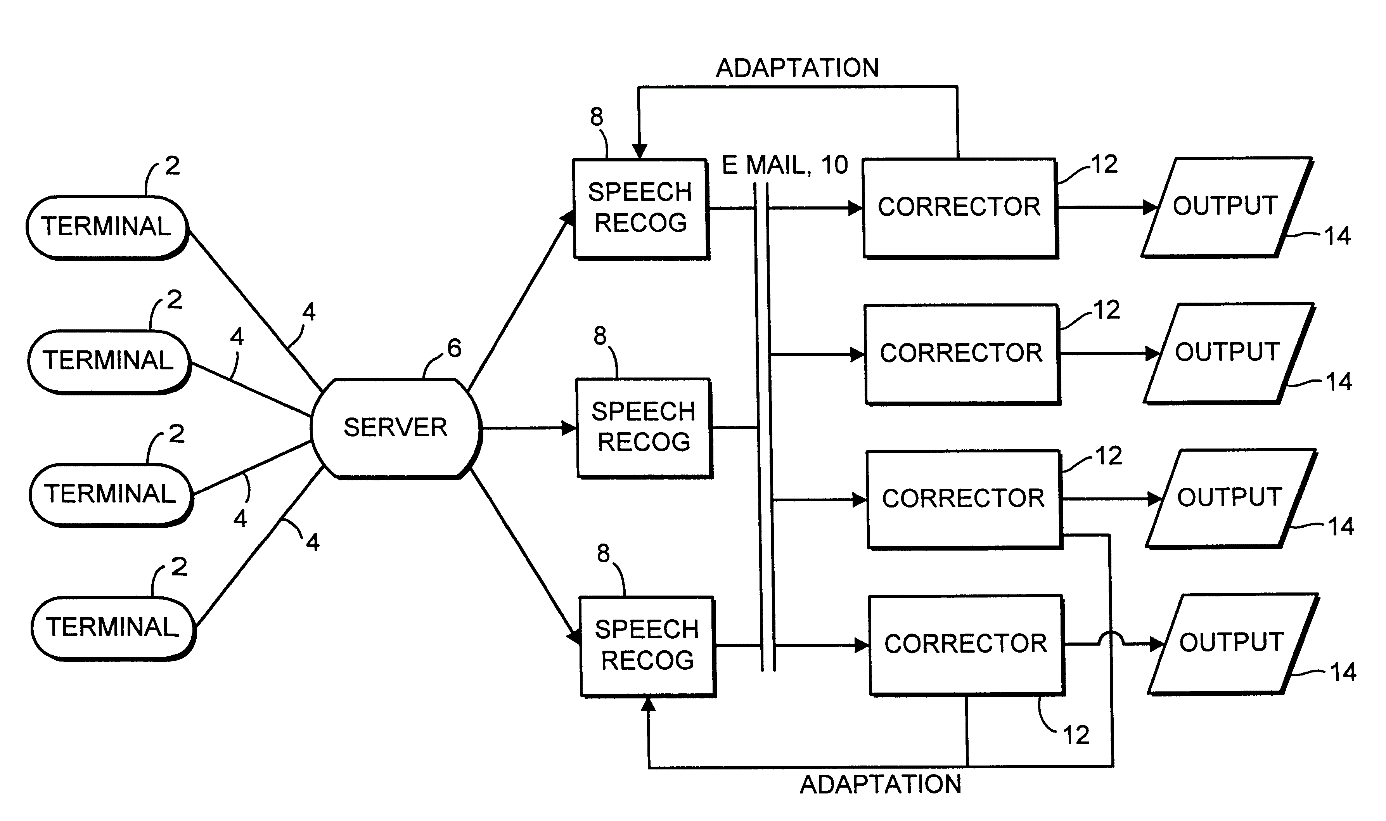
Ev otomasyonu veya daha üst sistemleri olarak tanımlayabileceğimiz akıllı evler, teknolojinin gelişmesi ile daha çok ilgi görmeye başladılar. İnsanlara sağladıkları rahatlık sayesinde akıllı ev projelerine her geçen gün birçok sistem eklendi. Kişilerin klima, aydınlatma gibi; kendi istekleri doğrultusunda değiştirebileceği, optimumu deneyip yanılarak bulabileceği pek çok sistem, bilgisayar algoritmaları ve temel fizik kuralları sayesinde insan faktörü gerektirmeyecek ve daima arzu edilen rahatlığı sağlayacak şekilde ayarlandı.

(Sriskanthan, 2002) makalesinde bu sistemlerin, herkesin uzaktan kolayca erişim sağlayabileceği hale getirildiği belirtilmiştir. Çalışmada cihaz odaklı kontrole değinilmemiş, kullanılmakta olan ev otomasyon protokolü üzerine çalışmalardan bahsedilmiştir. Bluetooth üzerinden alınan veriler ile bu tür sistemlerin kontrolü üzerine bilgilendirme yapılmıştır. Bahsedilen kontrol mikro denetleyici aracılığıyla yapılmıştır. Özellikle oda ısısını düzenleme konusuna değinilmiş ve bu konuda bluetooth üzerinden alınan bilgiler ve Max1669 yardımı ile gerekli fan kontrolü sağlanmıştır.[1]

(Piyare,2011) makalesinde ise bahsedilen bluetooth haberleşmesinin, modern yaşama tamamen adapte olmuş akıllı telefonların yardımı ile kontrol edilebileceğine değinilmiştir. Çalışma incelendiğinde, bir Arduino denetleyicisi aracılığıyla akıllı telefondan alınan verilerin gerekli sisteme nasıl yönlendirileceği anlaşılabilmektedir. Sistem, öncelikli olarak telefonun bluetooth yoluyla bağlanması ile başlatılmıştır. Sisteme bağlanan telefon gerekli komutları Android ortamı için yazılmış GUI sayesinde sisteme iletmektedir. Yapılan testlerde sistemin bina içerisinde 50 m’lik bir aralıkta işlem gördüğü gözlenmiştir.[2][3]

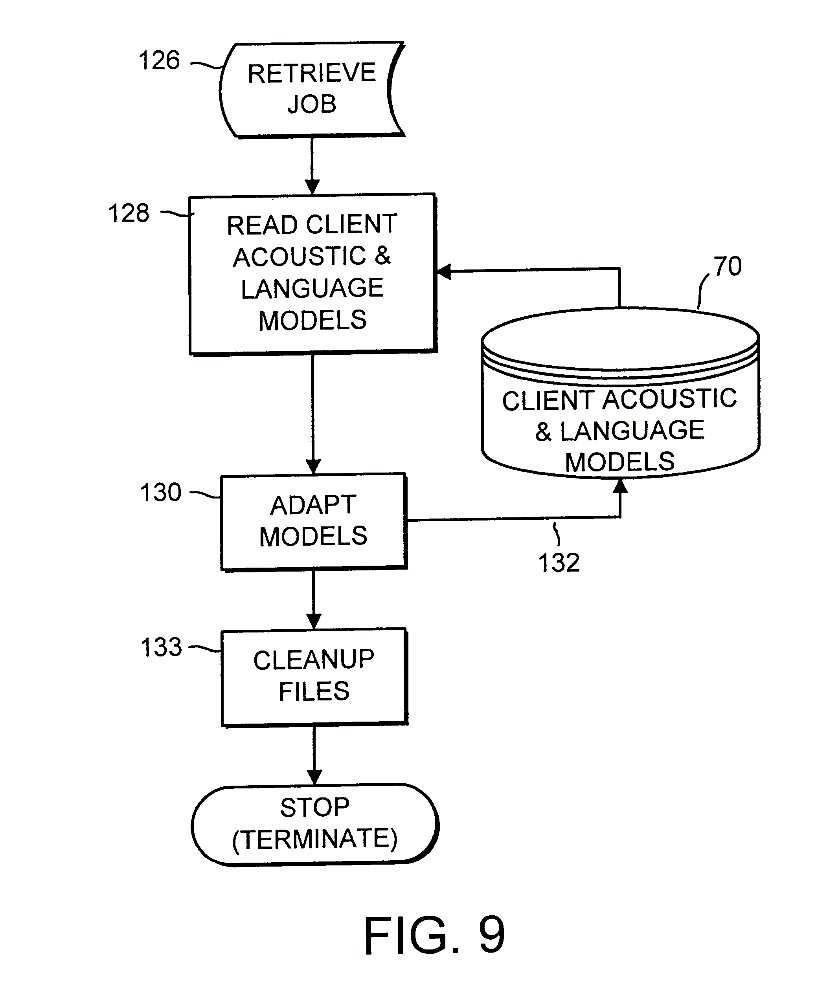
Endüstri 4.0 ve IOT (internet of things) kavramlarının ortaya atılmasıyla beraber ev otomasyonunda da bu konseptler ön plana çıkarılmaya başlamıştır. Örneğin, (Gomez, 2010) makalesinde bir ev için gerekli tüm sistemlerin artık herhangi bir fiziksel etki gerektirmeden, herhangi bir temel bilgiye sahip olmadan ve kablosuz olarak kontrol edilebileceğine değinilmiştir. Bunun için ise WHANs, Z-Wave, Wavenis ve ZigBee protokollerinin kullanıldığı belirtilmiştir. Makalede belirtilen sistem tüm evi kontrol ettiği için güvenlik ile ilgili sistemlerde Insteon protokolünün kullanıldığı belirtilmiştir. Kullanılan protokoller belirli katmanlara ayrılıp, her katman için bir router atanmıştır. Bu katmanlardan alınan veriler, internet üzerinden belirli bir sunucuya aktarılarak gerekli algoritmalar kablo bağlantısı olmadan çalıştırılmıştır. Yapılan testlerde ise gelecek çalışmalar için WHANs protokolü önerilmiştir.[4]

Yapılacak projede, ev otomasyonu bünyesinde bulunan sistemlerin ve haberleşme protokollerinin kullanıcı tarafından ses ile kontrolü sağlanacaktır. Bunun için Google’ın sunduğu Speech API kullanılacaktır. US6173259B1 numaralı patentte, kullanılacak olan STT özelliğinin formatı belirtilmiştir. (Şekil 1.2)[5] [6]



Şekil 1.2 – Speech To Text Formatı (Konuşmadan metne)

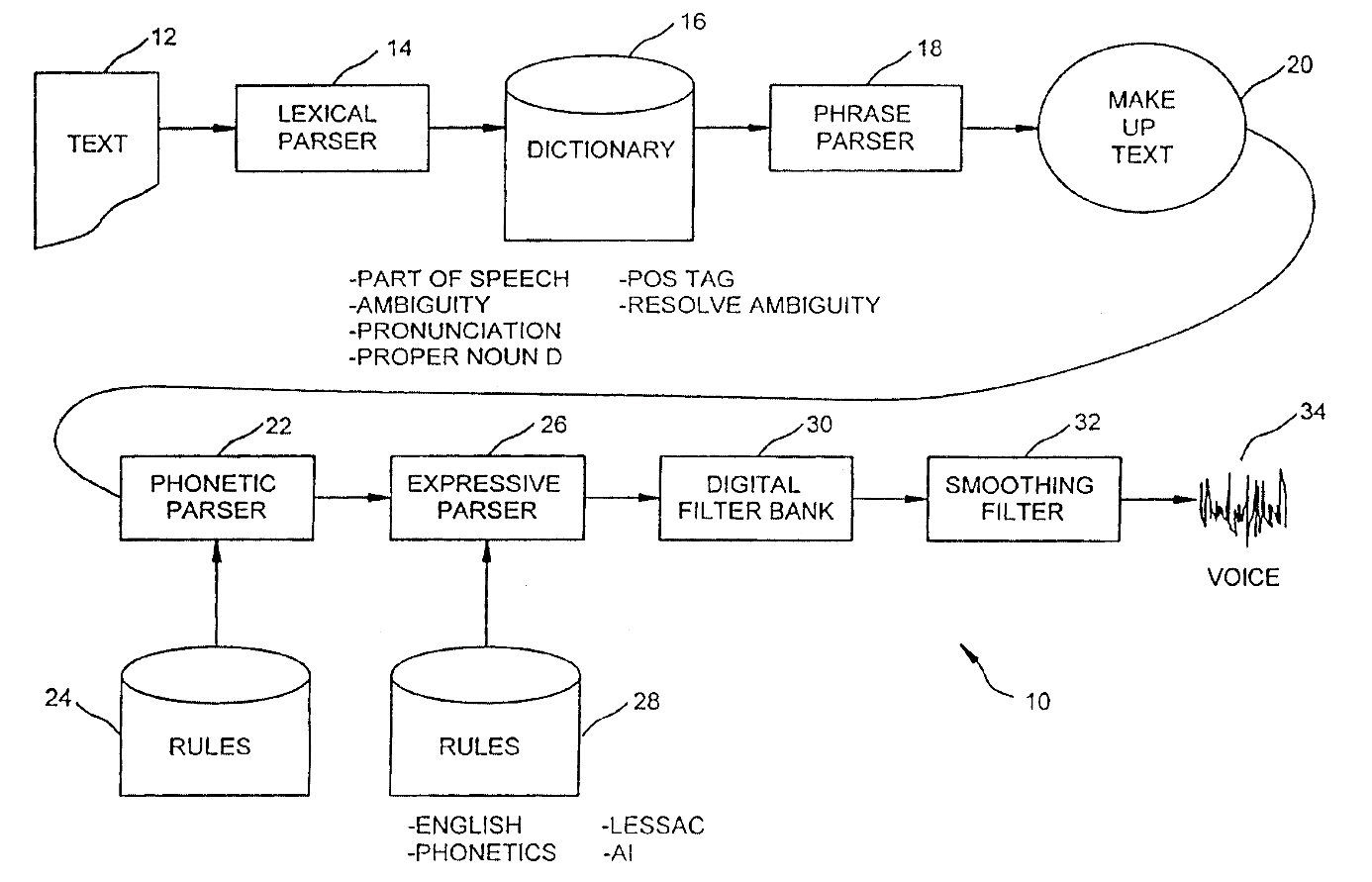
Verilen şekilde de görülebileceği gibi, alınan konuşma dosyaları öncelikle bir sunucuya yollanır. Sunucu, gerekli algoritmayı kendi bünyesinde çalıştırdıktan sonra bir sonuç üretir. Fakat sonuç her zaman doğru olmayabilir. Bunun üzerine gerekli algoritma, çıkan sonuç doğrulanıncaya kadar yeniden çalışır. Başarılı sonuç elde edildiğinde ise kullanıcıya gerekli metin (text) bildirimi yollanır.

Başarılı çıktı elde edilebilmesi için sistemin kullanıcı aksanını ve dilini öğrenmesi gerekmektedir. Bahsedilen patentte bu durum yanda görülen model (Şekil 1.3) ile açıklanmıştır. 

AI (yapay zeka) özelliği olan öğrenme şekli ses sistemlerinde de ön plana çıkmaktadır. Kullanılan sistemlerin doğruluğunu ve her kişiye aynı çıktıyı verebilmesini sağlayan sistem aslında bu öğrenme özelliğidir. Her kullanıcıdan alınan örnekleri toplayarak bunu bir adaptasyon modeli haline getirir.

Şekil 1.3 – Öğrence Mekanızması

Aynı işlemden TTS için US6865533B2 numaralı patentte de bahsedilmiştir. Gerekli format ise Şekil 1.4’de verilmiştir.[7]

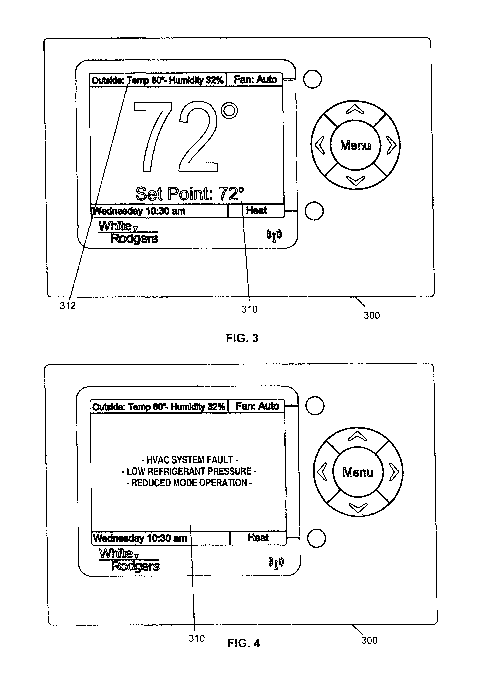


Şekil 1.4 – Text To Speech Format (Metinden konuşmaya)

Patentte verilen şekilde de görüldüğü üzere, sistem öncelikli olarak kullanıcıdan bir text (metin) istemektedir. TTS protokolü ilk olarak bu metni parçalarına ayırır ve bunu sözlük olarak adlandırılan tanımlama kısmına yollar. Tanımlama kısmı, parçalarına ayrılan metni ses haline getirmeye başlar. Tüm ses öğeleri bir araya getirilip anlamlı bir cümle haline gelmeleri için filtrelerden geçirilir. Ara filtrelerde istenilen dile göre formatlar yer almaktadır. AI olarak geçen bu kısım, sistemin öğrendiği dilin verilerine göre metni gramer kurallarına uygun hale getirir. Çıkan sesin kulağa daha doğal gelmesi için düzeltme filtresi uygulanır ve sonrasında kullanıcıya gerekli bildirim yollanır.

Projede kullanılacak Google Speech API, bu işlemi kendi sunucularında gerçekleştirir. Tüm dünyadan topladığı konuşma verilerini bu işlem için kullanır ve çıkardığı sonuçların doğruluğunu belirli aralıklarla kontrol eder. Bu yüzden dünya üzerinde çok gelişmiş bir kullanım alanına sahiptirler.

Bilgisayarların gelişip insan ihtiyaçlarına gerekli çözümleri sağlayabildiği görüldüğünde çoğu sistem gibi klima sistemleri de otomatikleştirilmeye başladı. Ayrıca, sadece ev içerisinde değil, insanın bulunduğu her ortamda bu sistemlerin varlığında zamanla orantılı bir artış görülmektedir. Bahsi geçen sistemler insanların en rahat koşullarda yaşamasını sağlayacak sıcaklık değerlerini ayarlamaktadır. US8689572B2 patentinde bu sistemler ele alınmıştır. Şekil 1.5’de gösterilen kontrol aparatı ortama konularak bu işlem gerçekleştirilmiştir.[8]

Bu aparat, ortama yerleştirilen fan, klima, fancoil gibi sistemlerin kontrolünü sağlar ve insanın refahını sağlamak için gerekli ortamı ayarlar.

Bu projede kullanılacak olan ve GES Teknik tarafından sağlanan fancoil kontrol aparatı bu sistemlere örnek olarak verilebilir. Bahsedilen sistem, bir evin veya bir iş ortamının tüm havalandırma ihtiyaçlarını karşılayabilmektedir.

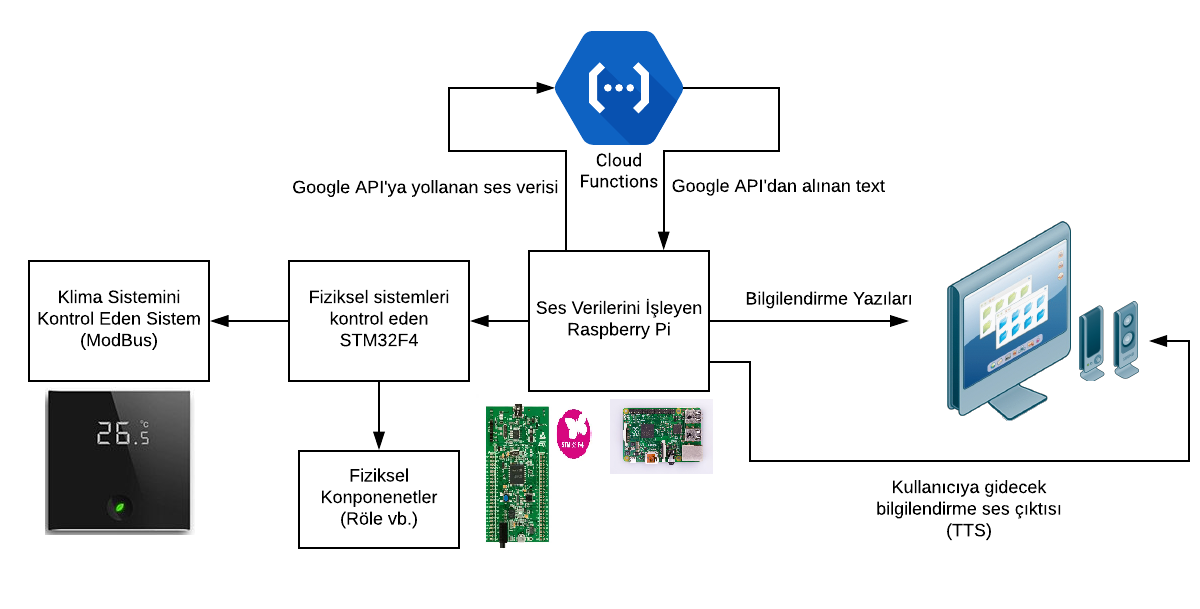
Şekil 1.5 – Klima Kontrol Sistemi

# BÖLÜM: MATERYAL VE YÖNTEM

Bu bölümde ses kontrolü ile ev otomasyonu için kullanılacak materyaller ve teknikler anlatılmıştır ve bölüm, tezde kullanılacak araştırma araçları ve tezde kullanılacak programlama yöntemleri olarak ikiye ayrılmıştır. Araçlar bölümünde tezin prototiplemesi süresince kullanılacak olan yazılım ve malzemeler tanıtılmıştır. Yöntemler başlığı altında ise projede yer alan programlama metotları ve kodlar hakkında bilgiler verilmiştir.

# TEZDE KULLANILMIŞ ARAŞTIRMA ARAÇLARI

Şekil 2.1’de kullanılan araçlar hakkında basit ve genel bir plan verilmiştir.



Şekil 2.1 – Sistemin Genel Planı

# RASPBERRY Pİ

Tezde kullanılacak Raspberry Pi kullanıcıdan ses verisinin alınması, alınan verinin işlenmesi gibi işlemlerin tümünden sorumlu olacaktır. Alınan ses Google Speech API sayesinde metne çevrilecek ve metin haline getirilen veri, yine Raspberry Pi tarafından anlamlandırılacaktır.

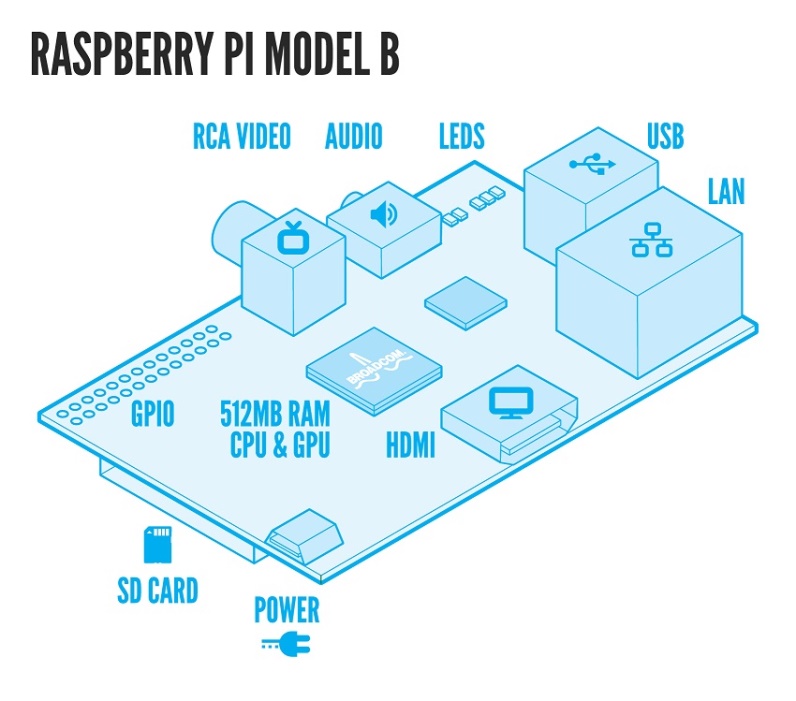
Şekil 2.2 – Raspberry Pi

Raspberry Pi, İngiltere’de bulunan Raspberry Pi Vakfı tarafından desteklenen; öğrenci, amatör ve hobi amaçlı kullanıcılara sunulan kredi kartı büyüklüğünde, tek bir board’dan oluşan bir mini bilgisayardır. 2009 yılından bu yana Raspberry Pi Foundation tarafından geliştirilmektedir. İlk satışı 29 Şubat 2012’de başlamıştır. A ve B olmak üzere iki modeli piyasaya sürülmüştür. 20 Nisan 2012 tarihinden itibaren Raspberry Pi’ın A ve B modelleri açık kaynak olarak kullanıma sunulmuştur.

# RASPBERRY Pİ’IN TEKNİK ÖZELLİKLERİ

Tez boyunca kullanılacak olan Raspberry Pi 3 B modelinin özellikleri verilmiştir:

* Quad Core 1.2 GHz Broadcom BCM2837 64 bit CPU
* 1 GB RAM
* BCM43438 kablosuz LAN ve Bluetooth Low Energy (BLE) on-board modülü
* 40 adet GPIO
* 4 adet USB 2 portu
* 4 Kutuplu streo çıkış ve kompozit video portu
* Tam boy HDMI
* CSI kamera portu, Raspberry Pi kamerasına uygun
* DSI görüntü portu, Raspberry Pi touchscreen’ine uygun
* Micro SD portu, işletim sistemi ve depolama için
* 45 gram ağırlığında
* Model B olduğu için 3.5W güç tüketmektedir
* Çalışma gerilimi +5V DC’dir.

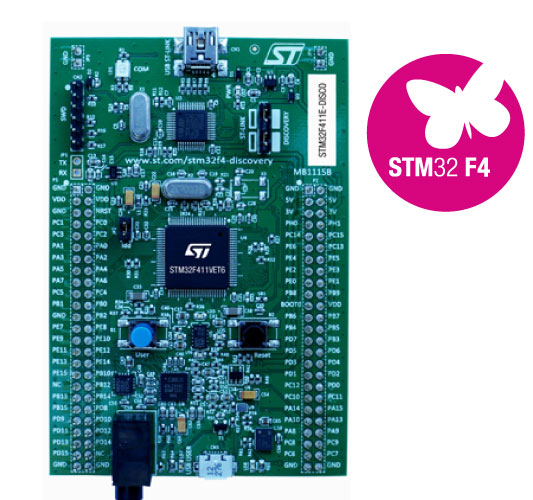


Şekil 2.3 – Raspberry Pi Model B

# Raspbian

Raspbian, Raspberry Pi Foundation’ın resmi olarak desteklediği açık kaynak kodlu bir Linux dağıtımıdır. Rapsbian, bünyesindeki programlama, eğitim ve genel kullanım için kurulan birçok program ile beraber gelmektedir. Bu programlara Python, Scratch, Sonic Pi, Java ve Mathematica örnek verilebilir.

# STM32F411E DİSCOVERY

ST firmasının ürettiği hem başlangıç seviyesi için hem de profesyonel amaçlar için uygun bir board’dur. ARM Cortex-M tabanlı bir işlemciye sahiptir. Tez içerisinde kullanılacak STM32F411E board’u gerekli fiziksel sistemin (Röle – Modbus protokolü) kontrolünü üstlenecektir.

STM32F4 programlaması için IAR IDE’si ve ST firmasının çıkardığı STM32Cube programı kullanılmıştır.

Şekil 2.4 – STM32F411E Discovery Kit

1. Raspberry Pi üzerinden gelen sinyal klima sistemini kontrol etmek için gönderildi ise STM32F4 bu mesaj için gerekli ModBus kodunu üretir ve klima kontrol ünitesine yollar.
2. Raspberry Pi üzerinden gelen sinyal ışıklandırma-perde için geldi ise, STM32F4 bunu gerekli pin’e aktarır.

# STM32F411E DİSCOVERY KİT’İN TEKNİK ÖZELLİKLERİ

* STM32F411VET6 (512 KB hafıza, 128KB RAM, LQFP100 paketi) mikro denetleyicisini içerir,
* Hem programlama hem de debug özelliğinden faydalanabilmek için board üzerinde ST-LINK/V2
* Kit’in çalışması için gerekli güç: USB hattından veya harici 5V
* L3GD20: 3-eksen dijital çıkış veren gyroscope (ST MEMS)
* LSM303DLHC: 3D lineer ivme sensörü ve 3D dijital manyetometre
* MP45DT02: STMEMS ses sensörü
* CS43L22, ses DAC, birleşik D hoparlör sürücüsü ile
* 8 adet fonksiyon LED’i
* 2 adet push-button(user-reset)
* USB OTG ile micro-AB connector
* Hızlı bir bağlantı ve programlama için LQFP100 I/O uzatma başlıkları
* ST’nin kendi sitesinde indirilebilir kütüphaneler ve örnekler. (ST’nin HAL kütüphanesi)

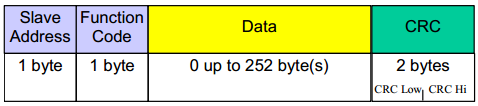
# MODBUS HABERLEŞME PROTOKOLÜ

STM32F411E’nin kontrol edeceği en önemli elementlerden biridir. Tez içerisinde klima kontrol biriminin çalıştırılıp, kontrol edilmesi için gereklidir.

Modbus, 1979 yılında Modicon tarafından geliştirilmiştir. Hala endüstriyel alanda otomasyon cihazlarının haberleşmesi için kullanılmaya devam etmektedir. Protokol gereği bir adet master cihaza 247 adet slave cihaz takılabilmektedir. Modbus’ın kullanımının tercih edilmesinin sebebi gayet basit ve anlaşılır olmasının yanı sıra ücretsiz bir lisans altında kullanıcılara sunulmasındandır.

Modbus genel anlamda haberleşme katmanı ve fiziksel katman olarak ikiye ayrılır. Haberleşme katmanı Modbus ASCII, RTU ve TCP/IP’dir. Fiziksel katman olarak ise UART tabanlı RS232/RS485, USB/CAN gibi değişik birimler ile TCP/IP için Ethernet örnek verilebilir. Tez içerisinde haberleşme katmanı olarak Modbus RTU, fiziksel katman olarak ise RS485 kullanılacaktır.

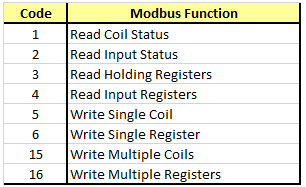
# MODBUS RTU



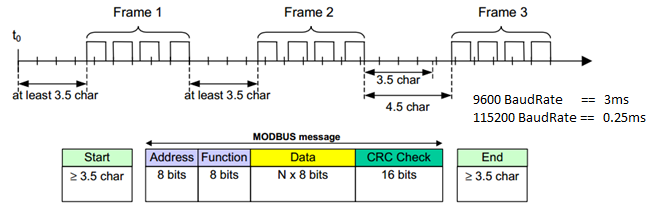
Şekil 2.5 – Modbus Protokolü

Modbus RTU gerekli veriyi Şekil 2.5’de verildiği şekilde yollar. İlk byte iletişim kurulacak cihazın adresini belirtmektedir. Slave cihaz sayısı en fazla 247 olabildiğinden bu slave adress’e 0-247 arasında bir değer yazılır. Function code olarak gösterilen kısma Modbus için belirtilmiş ve standartlaşmış fonksiyon kodları yazılmalıdır. (Tablo 2.1)[9]

Tablo 2.1 – Modbus kodları



Gerekli kod seçilip, slave adress belirlendikten sonra CRC16 hesaplanmaktadır. Hesaplanan CRC16, gerekli yere yerleştirildikten sonra oluşturulan paket artık geçerli Modbus cihazına gönderilebilir. Bu paket gönderimleri arasında 3.5 karakterlik boşluklar olmak zorundadır, fakat bu boşluk baudrate’e göre değişmektedir. (Şekil 2.6)



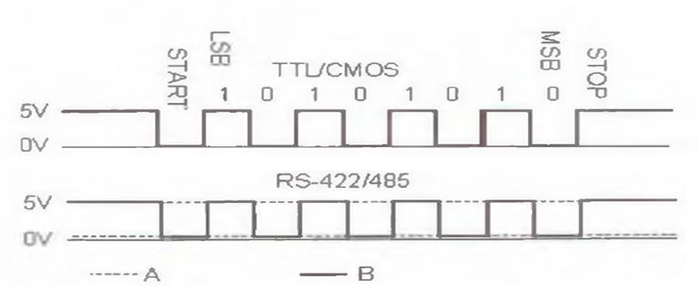
Şekil 2.6 – Modbus Paket Gönderim Şekli

# RS485

RS-485 (Balanced Data Transmission), uzun mesafelerdeki veri iletişiminin sağlanması için hazırlanmış araçlardan biridir.

Veriler RS-485’de A ve B adı verilen (+ ve - olabilir) iki kablo üzerinden iletilir. RS-485 birbirine tam zıt iki sinyal kullanır. Yani A sıfır iken B birdir, iletilen verinin durumu bu iki sinyalin aritmetik farkından tespit edilir. (Şekil 2.7)

RS-485 standardı 32 adet Alıcı-Verici çiftinin aynı anda sisteme bağlanmasına izin verir. Ancak düşük hızlarda bu sayı 64’e hatta 128’e kadar çıkabilir. Fakat sistemin kararlı çalışması garanti edilemez. Bu bağlantı 2 kablolu olabildiği gibi 4 kablolu da olabilir. 2 kablolu bağlantıda master ile slave’ler arası bağlantı aynı anda tek yönlü (half dublex) olarak oluşturulur. Yani aynı anda sadece bir tarafa veri aktarılabilir. Slave’ler kendi arasında haberleşme sağlayabilir fakat bu haberleşmeyi hatta düzensizlik olmaması için yine master kontrol eder. 4 kablolu bağlantı ise aynı anda iki yönlü (full dublex) olarak yapılır. Bu bağlantıda slave’ler kendi araslarında veri aktarımı yapamaz. Bu bağlantıda iki veya daha fazla slave bir anda veri göndermek istediğinde ortaya çıkan düzensizlik master tarafından kontrol edilir.



Şekil 2.7 – RS485 Mantığı

# SM2028EC-TOUCH KLİMA KONTROL ÜNİTESİ



Smallart markası altında, programlanması GES Teknik tarafından yapılan ve yine GES Teknik tarafından satılan bir klima kontrol ünitesidir. Dokunmatik ekranı aracılığı ile bir mekanın (ev, iş yeri, oda vb.) havalandırma ile alakalı birçok özelliği kontrol edilebilmektedir. Modbus üzerinden haberleştiği için bu tez içerisinde klima kontrol ünitesi olarak kullanılması kararlaştırılmıştır. (Şekil 2.8)[10]

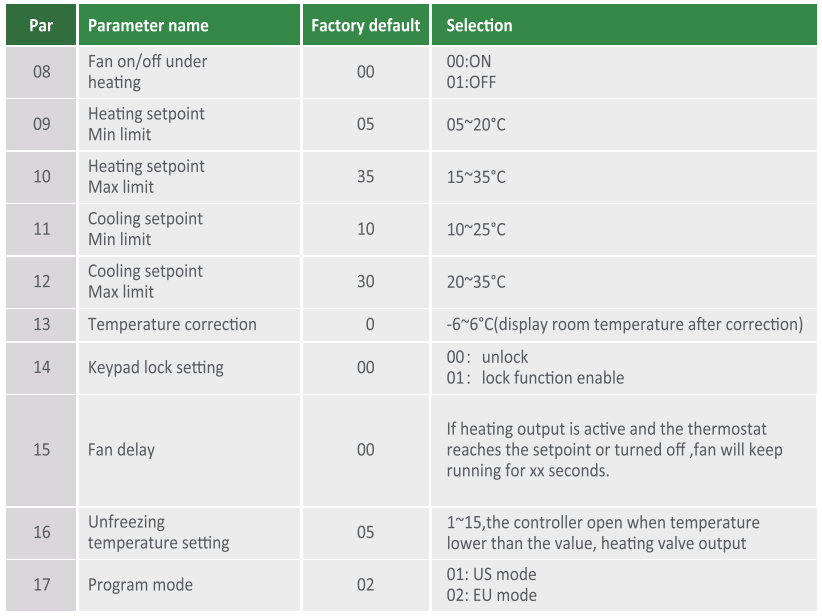
Şekil 2.8

# SM2028EC-TOUCH KLİMA KONTROL ÜNİTESİ TEKNİK ÖZELLİKLERİ

* Dokunmatik geniş dijital LCD ekran
* NTC ısı sensörü
* Tuş kilidi ile istenmeyen kullanıcılara erişim engellenebilir
* Heat/Cool veya Auto çalışma modu seçimi
* Manuel/Auto 3 kademe fan hızı kontrolü ve EC fan kontrol için modeller
* Fan için sürekli veya termostatik çalışma seçimi
* Düşük sıcaklık koruması
* Haftalık programlama
* Tek tuş ile ekonomi modu
* Isıtma çıkışı için iki farklı çalışma seçeneği
* Ayarlanabilir ölü band aralığı (Otomatik mod için geçerlidir.)
* Ayarlanabilir vana çıkışları

Tablo 2.2’de kullanılan fonksiyonların önemlileri verilmiştir:

Tablo 2.2 – Parametreler



SM2028EC’ye gönderilen Modbus paketi, STM32F4 tarafından Raspberry Pi’ın ürettiği ses verisine göre düzenlenir. Örneğin 8. Parametreyi yönetmek için Raspberry Pi üzerinden “Fan” ve “Kapat” veya “Aç” kelimelerini içeren bir cümle ses yolu ile alınmalıdır. Alınan bu veri işlendikten sonra STM32F4’e yollanarak, gerekli Modbus paketi oluşturulur. Oluşturma işlemi tamamlanıp SM2028EC’ye yollandıktan sonra SMT32F4 bir cevap beklemektedir. Gelen cevap sayesinde işlemin tamamlandığı anlaşılmaktadır. Gerekli işlem tamamlandığında ise Raspberry Pi kullanıcıyı sesli olarak bilgilendirir. Sesli bilgilendirmenin yanı sıra hazırlanan HMI ekranı sayesinde aynı bildirim ekranda da görülecektir. (Geçerli programlama metodu yöntemler kısmında daha ayrıntılı anlatılacaktır.)

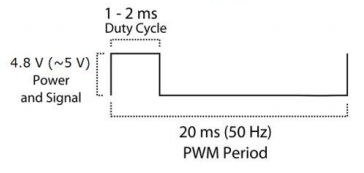
# RÖLE

 Röle, bu tez içerisinde ışıklandırma sistemini kontrol etmek için kullanılacak ve STM32F4 tarafından kontrol edilecektir.

Şekil 2.9 – Röle

# SERVO MOTOR İLE PERDE SİSTEMİNİN KONTROLÜ

Perde sistemi ise bir adet servo-motor ile kontrol edilecektir. Servo motora STM32F4 tarafından PWM yollanacaktır. Servo motor olarak SG90 micro servo kullanılacak ve konum kontrolü 20 ms (50Hz) periyot süreli olarak ayarlanan bir PWM sayesinde yapılacaktır. (Şekil 2.10)[11]



Şekil 2.10 – Servo PWM

# TEZDE KULLANILMIŞ ARAŞTIRMA ARAÇLARI

# Bu bölüm 2 ana başlık altına ele alınmıştır ve alınan iki başlıkta kullanılan tüm algoritmalar anlatılmıştır. Bunların ilki sistemin haberleşme ve kontrol sistemini yöneten STM32F4’dür. İkincisi ise sistemin ses kontrolünü ve gerekli diğer özellikleri denetleyen Raspberry Pi’dır. Algoritma tasarımındaki amaçlar ve neden böyle bir yola başvurulduğuna değinilmiştir.

# STM32F4 İLE KULLANILAN ALGORİTMALARI

# STM32F4 sistemde Raspberry Pi üzerinden gelen verilen işlenerek gerekli fonksiyonlar için gerekli işlemler yapılır. Fonksiyon tablosu Tablo 2.3’de verilmiştir. Bu tablo aynı zamanda Raspberry Pi içerisinde bir dosya içerisine yazılmıştır. Amacı ve yazılış şekli Raspberry Pi kısmında açıklanmıştır.

# Tablo 2.3 – Raspberry Pi Tanıtılmış Fonksiyonlar

|  |  |
| --- | --- |
| Fonksiyon Kodu | Açıklama |
| 0 | Kullanıcı ayarları |
| 1 | STM32F4 sinyalleri |
| 2 | İnternette aratma |
| 3 | Hava durumu |
| 4 | Tarih ve Saat sorgulama |
| 5 | Sistemi Kapatma |

# STM32F4’de ayarlanan tüm sistemin görüntüsü Şekil 2.11’de verilmiştir.

# 

# Şekil 2.11 – STM32F4 için açılan pinlerin gösterimi (STM32Cube Programı)

# MODBUS ALGORİTMASI

# Modbus haberleşme protokolünün ne olduğu bir önceki bölümde anlatılmıştı. Modbus cihazları hem veri okumak için hem de veri yazmak için belirli bir plana sahiptir. (Şekil 2.6) Modbus haberleşme protokolü klima kontrol cihazı için yazılmıştır.

# Verinin oluşması için gereken CRC verisine bir sonraki bölümde değinilecektir. CRC işleminden önce istenilen işleme göre veri oluşturulmalıdır. Bu veri modbus cihazının adresi, modbus operasyonu (Tablo 2.1), gerekli veri ve CRC’den oluşmaktadır.

# Bu işlem için createModBusData fonksiyonu yazılmıştır. Yazılan fonksiyon integer değişken tipine sahip olan verileri modbus verisinin kurallarına uygun şekilde düzenler. Düzenleme integer olan değişkenlerin, özellikle yollanacak verinin, iki ayrı 8 bit’e ayrılmasını içerir.

# STM32F4 bu veriyi seri haberleşme portu olan UART üzerinden yollar. STM32F4 üzerinde bulunan UART, verileri RX - TX olmak üzere iki pin ile yönetir. RX veri alma pini iken TX veri yollama pinidir. Fakat projede kullanılan haberleşme hattı RS485 veri yolunu kullanmaktadır. Bu yüzden Şekil 2.12’de verilen TTL to RS485 dönüştürücü kullanılmıştır. Gerekli bağlantı diyagramı ise 2.13’de verilmiştir.

# ttl to rs485 ile ilgili gÃ¶rsel sonucu Şekil 2.12 – TTL – RS485 dönüştürücü

# STM32F4 içerisinde bulunan UART2, modbus cihazının haberleşmesi için kullanılmaktadır. Gerekli ayarlar STM32Cube içerisinde yapılmıştır ve gerekli ayar kodları STM32Cube tarafından üretilmiştir. Gerekli UART haberleşmesi için baudrate (bit/saniye) 9600, yollanan verinin genişliği 8 bit, durma biti 1 ve parity None olarak ayarlanmıştır. (Şekil – 2.14)

# https://documents.lucidchart.com/documents/0e88638f-a06b-4a62-970e-da5d298cb38e/pages/0_0?a=344&x=359&y=188&w=902&h=325&store=1&accept=image%2F*&auth=LCA%200bc4fa373e96361043a76eea73f9268c9440bf40-ts%3D1528058081

# Şekil 2.13 – TTL-RS485 Bağlantı Diyagramı

# 

# Şekil 2.14 – UART2 Ayarları (STM32Cube)

# Modbus haberleşmesinde STM32F4 master olarak programlanmıştır. Yani veri yollama ve veri alma işlemlerini STM32F4 yönetir.

# Modbus kısmında da anlatıldığı gibi her bir veri arasında en azından 3.5 karakterlik boşluklar bulunmaktadır. 3.5 karakterlik boşluk aslında 3.5 bit’in gönderilme süresidir.(1 karakter -> 1 bit) 9600 baudrate için bu bekleme süresi yaklaşık olarak 3 ms’dir. ST firmasının sağlamış olduğu HAL kütüphanesi içerisindeki HAL\_Delay() fonksiyonu ile bu işlem gerçekleştirilir.

# Modbus üzerinden veri gönderme kodu aşağıda verilmiştir.

# void writeSingleRegister(UART\_HandleTypeDef \*huart,uint8\_t\* modbusData, uint8\_t\* responseData){

# HAL\_GPIO\_TogglePin(Modbus\_Control\_GPIO\_Port,Modbus\_Control\_Pin);

# HAL\_UART\_Transmit(huart, (uint8\_t\*)modbusData, 8, 1000);

# HAL\_Delay(3);

# HAL\_GPIO\_TogglePin(Modbus\_Control\_GPIO\_Port,Modbus\_Control\_Pin);

# HAL\_UART\_Receive\_IT(huart, (uint8\_t\*)responseData, 8);

# HAL\_Delay(3);

# }

# Görüldüğü üzere ST firması tarafından hazırlanan HAL Kütüphanesi kullanılmaktadır. Gerekli veri modbusData ile fonksiyona verilir ve haberleşme sağlanan cihazdan responseData üzerine yazılacak bir veri beklenir. Karşılık gelen veri işlemin yapılıp yapılmadığının bir göstergesidir. Eğer işlem başarılı bir şekilde tamamlandıysa modbus haberleşmesi sağlayan cihaz, master tarafından yollanan verinin aynısını geri döndürür.

# void readSingleRegister(UART\_HandleTypeDef \*huart, uint8\_t\* modbusData, uint8\_t\* responseData){

# HAL\_GPIO\_TogglePin(Modbus\_Control\_GPIO\_Port,Modbus\_Control\_Pin);

# HAL\_UART\_Transmit(huart, (uint8\_t\*)modbusData, 8, 1000);

# HAL\_Delay(3);

# HAL\_GPIO\_TogglePin(Modbus\_Control\_GPIO\_Port,Modbus\_Control\_Pin);

# HAL\_UART\_Receive\_IT(huart, (uint8\_t\*)responseData, 7);

# HAL\_Delay(3);

# Verilen kod parçası cihazdan bilgi okumak için kullanılmaktadır. Yollanan veriye bu sefer cevap olarak istenilen adreste bulunan değer geri döndürülür. Bu veri responseData’nın içerisindedir ve tek bir adres okunduğu sürece her zaman 7 bit’lik bir cevap dönmektedir. Veri alma işlemi bir kesmedir. Yani işlemci gerekli veri adeti sağlanana kadar her hangi bir şekilde bekleme yapmaz. Gerekli veri adeti sağlandığında yine HAL içerisinde bulunan CallBack fonksiyonu çağrılır.

# CallBack fonksiyonunun içerisinde gerekli verinin ayrıştırılması gerekmektedir ve bu işlem aşağıda verilen kod ile yapılır.

# if(readContol == 1){

# readContol = 0;

# int retL,retH;

# retL = (modbusResponse[3] & 0x0F) \* (16\*16);

# retH = ((modbusResponse[3] & 0xF0) >> 4) \* (16\*16\*16);

# resultValue = retL + retH + (int)modbusResponse[4];

# }

# Hem veri gönderme hem de veri almada görülebileceği üzere Modbus\_Control pininin set-reset durumu değişmektedir. Bunun sebebi kullanılan TTS-RS485 entegresinin üzerinde bulunan DE pininin veri yazma esnasında set durumunda yani 1(açık) durumunda olması gerekmektedir. Aynı şekilde RE pininin veri okuma esnasında ise reset konumunda yani 0(kapalı) durumda olması gerekmektedir. Aynı anda hem yazım hem okuma işlemi gerçekleşmeyeceği için ayrı pinler atamanın bir anlamı yoktur.

# CRC16 ALGORİTMASI

# CRC (Cyclic Redundancy Check) dijital dünyada bir çok aygıtta kullanılan ve veri alışverişi sırasında verideki değişikliği algılamak için kullanılan bir yöntemdir. Checksum yöntemine göre daha kullanışlıdır. Çünkü checksum toplam farkına göre hata ayıklar iken, CRC ise 16 bit ya da 32 bit kodlamaya dayanan matematik bir formül ile hata ayıklar. CRC16 ise Modbus için kullanılan hata ayıklama yöntemidir.

# Modbus bölümünde anlatıldığı gibi CRC16 modbus verisinin en sonunda 16 bit şeklinde haberleşme gerçekleştirilecek cihaza gönderilir.

# Gerekli algoritmanın yazılması 3 temel basamağa dayanmaktadır. Bu basamakların ilki oluşan modbus verisinin hex haline yani 16 tabanında bir karakter dizinine dönüştürülmesidir. Oluşturulan hex karakter dizini CRC’si bulunmak üzere yukarıda verilen koda gönderilir. Oluşan CRC 16 bit olduğundan bir integer değişkende saklanır. Fakat modbus haberleşme protokolü CRC16’yı 2’ye bölünmüş 8 bitler halinde istemektedir. Bu yüzden integer olan CRC aşağıdaki kod bütünü ile 8’er bit olan 2 değişkene ayrılır.

# // CRC’nin hesaplanması

# int calculateCRC(char\* data, int lenght){

# int crc = 0xFFFF;

# int pos,i;

# 

# for(pos = 0; pos < lenght; pos++){

# crc ^= (int)data[pos];

# for(i = 8; i > 0; i--){

# if((crc & 0x0001) != 0){

# crc >>= 1;

# crc ^= 0xA001;

# }

# else{

# crc >>= 1;

# }

# }

# }

# return crc;

# }

# // Integer olan CRC’nin iki ayrı 8 bit değişkene ayrılması

# void divideCRC(int crc, uint8\_t\* crcLow, uint8\_t\* crcHigh){

# int temp;

# temp = (crc & 0xff00) >> 8;

# \*crcLow = (uint8\_t)crc;

# \*crcHigh = (uint8\_t)temp;

# }

# RASPBERRY Pİ’IN GÖNDERMİŞ OLDUĞU VERİLERİN ALGILANMASI VE İŞLENMESİ

# Raspberry Pi tarafından gönderilen veriler yine UART haberleşmesi tarafından STM32F4’e iletilir. Fakat bu sefer STM32F4 Discovery kitinin dizaynından dolayı tek bir UART portu ile değil 2 UART portu ile gerçekleşmektedir.

# İlk planlamalara göre UART1 bu iş için ayrılarak, STM32F4 ile Raspberry arasındaki haberleşmeyi sağlayacaktı fakat yapılan testler sonunda UART1’in TX bitinin başka bir özelliğe daha bağlı olduğu görüldü. Bu yüzden Raspberry Pi ile yapılan bağlantıda 2 farklı UART portu kullanıldı. Raspberry Pi’ın TX biti UART1’in RX bitine, yine Raspberry Pi’ın RX biti bu sefer UART2’nin TX bitine bağlandı. UART2 her ne kadar iki farklı haberleşme içi kullanılmış olsa da Raspberry Pi tarafında yani Python kodu tarafında bu düzensizlik giderildi. Bu kısım Raspberry Pi bölümünde anlatılmıştır.

# Algılama işlemi 3 farklı bölüme bölünmüştür. Sistemin çalışma ve algılama algoritması Şekil 2.15’de verilmiştir.

# Perde sinyalinin çözümlenmesi

# Işıklandırma sistemi ile alakalı sinyallerin çözülmesi

# Klima sistemi ile alakalı sinyallerin çözülmesi

# https://documents.lucidchart.com/documents/d974c886-59a2-42fb-974f-71d59e24d3fb/pages/0_0?a=612&x=530&y=-24&w=660&h=968&store=1&accept=image%2F*&auth=LCA%209053b502b0856098621a1db84fcb32cc0045dbe2-ts%3D1528065125

# Şekil 2.15 – Algılama Algoritması

# STM32F4’ün veri beklemesi işlemcinin o anda başka hiçbir şey ile ilgilenmeyip veriyi beklemesi anlamına gelmektedir. Bu işlemci için gereksiz bir bekleme olacağından, bu durum için Raspberry Pi’dan gelecek veri STM32F4 tarafından bir interrupt yani bir kesme ile beklenmektedir. İşlemci verilen bit sayısı tamamlanana kadar CallBack üretmemektedir. Fakat gerekli bit sayısı sağlandığında CallBack fonksiyonu işlemci tarafından çağırılır. Çağrılan bu fonksiyona 2 adet değişken değerleri değiştirme işlemi atanmıştır. Bunlardan biri kesmeyi kontrol etmek için açılmış olan raspDataControl(aynı zamanda alınan veriye uygulanacak işlemin bitip bitmediğini belirtir) ikincisi ise verinin tam olarak alınıp alınmadığını kontrol eden waitCounter’dır. Gerekli data okunduktan sonra gelen sinyalin işlenmesine başlanmaktadır.

# Fakat sinyal işlemeye geçmeden önce Raspberry Pi tarafından gönderilen verinin işlenmesi gerekmektedir. Raspberry Pi veriyi belirli bir kodlama ile göndermektedir ve veri her zaman 6 bit olarak içermektedir. Verideki sayılar arasında bir adet “a” karakteri bulunmaktadır. Veri eğer 6 karakterden az tutacak ise de yine “a” karakterleri ile 6’ya tamamlanmaktadır. STM32F4 gelen sinyaldeki ilk sayıyı fonksiyon kodu olarak, ikinci sayıyı ise ayarlanmak istenen değer olarak ayarlar. Fakat ikinci sayı aynı zamanda “b” karakteri de olabilir. Bu STM32F4 tarafından veri okunmak isteniyor olarak algılanır ve buna göre gerekli değer değişkenini -1 olarak ayarlar bu STM32F4’e veri okuma olarak tanıtılmıştır. Yazılan algoritmada dikkat edilmesi gereken noktalardan biri de STM32F4 bu verileri char değişkeni yani karakter olarak almaktadır. Bu yüzden değer her seferinde bir bit olarak okunarak 10 tabanına dönüştürülmelidir. İlk sayının 2 bit gelmesi halinde yapılan işlem aşağıda verilmiştir.

# else if(s == 2){

# int temp1 = raspData[0] - '0';

# temp1 = temp1 \* 10;

# int temp2 = raspData[1] - '0';

# functionCode = temp1 + temp2;

# }

# Sayının kaç haneli geldiği ise sayı aralarında ve sonunda bulunan “a” karakterlerinin yer tespiti ile yapılmaktadır. Yukarıda verilen kod ilk değeri yani fonksiyon kodunu temsil etmektedir. Bu yüzden s olarak adlandırılan değişken ilk “a” karakterinin karakter dizininde bulunduğu yeri göstermektedir.

# İkinci sayının algılanması için gerekli kod aşağıda verilmiştir.

# else if((s - temp) == 2){

# int temp1 = raspData[s - 1] - '0';

# temp1 = temp1 \* 10;

# int temp2 = raspData[s] - '0';

# value = temp1 + temp2;

# }

# Temp olarak adlandırılan değişken ikinci sayının son rakamının bulunduğu yeri belirtmektedir. Daha öncede bahsedildiği gibi bu ikinci sayı olarak belirtilen value değişkeni “b” olarak da gelebilir. Bunu algılayan kod aşağıda verilmiştir.

# else if((s - temp) == 1){

# if(raspData[s] == 'b'){

# value = -1;

# }

# else{

# value = raspData[s] - '0';

# }

# }

# Gerekli fonksiyon kodu ve değer(value) değişkeni Raspberry Pi tarafından gönderilen veride ayıklandıktan sonra, geçerli fonksiyon değerine göre STM32F4 gerekli işlemi uygulamaktadır. Tablo 2.4 ve Tablo 2.5’de geçerli fonksiyonların anlamları verilmiştir.

# Tablo 2.4 – Fonksiyon Kodları

|  |  |
| --- | --- |
| Fonkiyon Kodları | Açıklama |
| 1'den 49'a kadar | Modbus cihazının sahip olduğu adresler |
| 50 | Perde |
| 51 | Işıklandırma |

# Tablo 2.5 – Projede kullanılan modbus cihazının adresleri

|  |  |
| --- | --- |
| Adresler | Açıklama |
| 0 | Ekran koruyucu |
| 1 | Isıtma/Soğutma Seçimi |
| 2 | Fan Hız Ayarı |
| 3 | Set Değeri |
| 4 | Ekran Kilidi |
| 8 | Ekonomi Modundaki Set Ayarı |
| 10 | Ekonomi Modu Aç/Kapa |

# Gerçek hayatta modbus adresleri 16 bit’in elverdiği son değere kadar çıkabilir fakat proje içerisinde kullanılan modbus cihazında deneme yanılma metodu ile toplam 7 adet adres bulunmuştur. Fakat Raspberry Pi kısmında anlatıldığı gibi eğer modbus fonksiyonlarını içeren dosya değiştirilip anlamlı ve geçerli anahtar kelimeler seçilir ise 49’a kadar olan tüm adresler algılanıp, gerekli işlem STM32F4 tarafından yapılacaktır.

# PERDE SİNYALİNİN ÇÖZÜMLENMESİ

# Tablo 2.4’de de görüldüğü gibi eğer fonksiyon kodu 50 ise STM32F4 bunu perde sinyali olarak algılar. Perde sinyali bir adet servo motor ile kontrol edilmektedir. Bu servo motorun kontrolü için STM32F4 tarafından PWM üretilir. PWM üretilmesi için öncelikle STM32Cube programından PWM üretilmesi adına TIMER1 seçilmiştir. TIMER1 için yapılan ayarlar Şekil 2.16’da verilmiştir.

# 

# Şekil 2.16 – TIMER1 için gerekli ayarlar sayfası

# TIMER mikroişlemciler içerisinde bulunan bir çeşit sayıcıdır. Mikro denetleyiciler içerisinde direk olarak analog çıkış pinleri bulunmamaktadır. Bu yüzden belirli süre içerisinde belirli bir görev döngüsü belirlenerek sinyal kesimi uygulanarak analog sinyal üretilir. Bu yönteme PWM adı verilir. Belirli süre olarak bahsedilen süreleri ise TIMER ayarlar. TIMER’ın ayarlanması için gerekli 4 değer vardır.

# Prescaler – Ön bölücü

# Counter Mode – Yukarı veya aşağı sayıcı olarak ayarlama

# Counter Period – AutoReload Hesabı

# Duty Cycle – Görev döngüsü

# Bu özelliklerden ilk 3’ü STM32Cube tarafından seçilerek otomatik olarak kod haline çevrilebilir. Duty Cycle ise mikroişlemcinin çalışma süresinde belirlenir. Gerekli PWM frekansı aşağıda verilen formül ile hesaplanabilir.

# Daha öncede bahsedildiği gibi proje içerisinde istenilen PWM frekansı 50 Hz’dir ve Clock sinyali 16 MHz’dir. Bu yüzden gerekli counter periyodu 19999 ve prescaler 15 olarak ayarlanmıştır.

# HAL kütüphanesi gerekli DC’nin seçilebilmesi için aşağıdaki fonksiyonu kütüphanesi içerisinde barındırmaktadır.

# \_\_HAL\_TIM\_SET\_COMPARE(&htim1,TIM\_CHANNEL\_1,Duty Cycle Değeri);

# Fakat geçerli fonksiyon içerisine 0’dan counter period değerine kadar bir sayı beklemektedir ve DC normal şartlar altında yüzdelik bir ifadedir. Servo motor bölümünde bahsedildiği gibi, proje içerisindeki servo motor DC 1ms’de tam kapalı konumunda, 2ms’de iken tam açık konumumdadır. Daha öncede 19999 olarak ayarlanan counter periyod değeri 20ms DC’ye eşittir. Bu yüzden basit bir içler dışlar çarpımı yapıldığında kullanılan servo motor için DC değerinin 1000 ile 2000 arasında olması gerektiği görülmüştür. Perde sistemini tam açmak istediğimizde 2000 değeri verilen kod parçasında Duty Cycle Değeri kısmına yazılmaktadır, aynı şekilde tam kapalı konumuna getirmek için ise 1000 değeri yazılmalıdır.

# Raspberry Pi tarafından gelen sinyal 2 çeşit sinyale ayrılabilir. Bunlardan biri mevcut perde konumunu öğrenmek için gelecek “b” karakteri iken diğeri perdeyi istenilen konuma çekmek için gelecek value değeridir.

# Perde sistemi için geçerli olan algoritma Şekil 2.17’de verilmiştir.

# https://documents.lucidchart.com/documents/e1086c9b-47c1-439b-91b3-f87bad828f71/pages/0_0?a=718&x=463&y=7&w=814&h=517&store=1&accept=image%2F*&auth=LCA%20dd2a1f019e9f47f21084771c466ded58725aad94-ts%3D1528147262

# Şekil 2.17 – Perde sisteminin algılama algoritması

# Değer yazma kısmında STM32F4 gerekli veriyi iki adet kontrolden geçirmektedir. Bunlardan biri eğer Raspberry Pi sinyali 1 olarak yollamış ise bu perdeyi tam aç anlamındadır. Buna özel olarak kontrol atanmıştır çünkü eğer 1 olarak gelen veri direk değer olarak işlenseydi bu değer tam olarak açığa değil yüzde 1 açığa denk gelmiş olacaktı bu da sistemin istenileni yapmamasına yol açacaktı. Diğer bir kontrol ise eğer değer 1 değil ise gelen değeri önce 10’a bölmekle başlar. Bunun sebebi Raspberry Pi veriyi göndermeden önce 10 ile çarpmaktadır. Sebebi sıcaklık değerinin 3 haneli bir sayı istenmesidir. 10 ile çarpılan bu yeni değer artık gerekli orana sıkıştırılabilecek durumdadır. Bu oran gelen değerin 10 ile çarpılık daha sonrasında 1000 eklenmesi ile gerçekleşir. Önce 10’a bölünüp daha sonra 10 ile çarpılmasının asıl amacı ise integer bir değişken olduğu için gereksiz küsurattan kurtulmak istenmesidir.

# Değer okuma kısmında ise STM32F4 öncelikle Rasberry Pi’ya anlamsız (dummy olarak adlandırılmış) bir bilgi yollar. Yollanmaması durumunda sorun bu kısımda değil Raspberry Pi’ın okuma kısmında çıkmaktadır. Çıkan sorunun sebebi ise modbus haberleşmesinden sorumlu olan UART2’nin aynı zamanda STM32F4’ün Raspberry Pi’a veri atmasından da sorumlu olmasından kaynaklanmaktadır.

# Atılan veriden sonra 2ms’lik bir bekleme verilmiştir. Bunun sebebi ise sisteme gerekli UART hattını tamamen boş hale getirmesi süre verilmesidir. Bekleme bittikten sonra STM32F4 kısmında storeLoc olarak adlandırılan değişkenden perdenin mevcut durumumdaki konumu alınır ve bu veri 3 basamaklı hale sokulup Raspberry Pi’a gönderilmektedir. 3 basamak olması için alınan değerin önüne fazladan anlamı olmayan 0’lar eklenmektedir. Bunun sebebi Raspberry Pi’ın veriyi 3 basamaklı beklemesinden dolayıdır. Aksi taktirde veri gelmedi olarak değerlendirerek işleme devam etmeyecektir. Mevcut durumda 1 basamaklı bir storeLoc değişkenine sahip ise geçekleştirilen kod bütünü aşağıda verilmiştir.

# if(tempr >= 0 && tempr <= 9){

# HAL\_UART\_Transmit(&huart2, (uint8\_t\*)dummy, 8, 1000);

# HAL\_Delay(2);

# sprintf(raspRet, "00%d", tempr);

# HAL\_UART\_Transmit(&huart2, (uint8\_t\*)raspRet, 3, 1000);

# }