T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ZEKA DENETİMİ İLE EMG SİNYALLERİNİN İŞLENMESİ VE SINIFLANDIRILMASI

CAN EROL

YÜKSEK LİSANS TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRONİK PROGRAMI

DANIŞMAN
YRD. DOÇ. DR. LALE ÖZYILMAZ

İSTANBUL, 2012

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YAPAZ ZEKA DENETİMİ İLE EMG SİNYALLERİNİN İŞLENMESİ VE SINIFLANDIRILMASI

Can EROL tarafından hazırlanan tez çalışması 19.06.2012 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç Dr. Lale ÖZYILMAZ

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Dr. Lale ÖZYILMAZ

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Tuğba KIYAN

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Gökhan BİLGİN

Yıldız Teknik Üniversitesi _____

ÖNSÖZ

Tez çalışmam sırasında her an yanımda olan ve her türlü anlayışı gösteren tez danışmanım Sayın Yrd. Doç. Dr. Lale Özyılmaz'a, tez çalışmam sırasında laboratuvar aletlerimin bir kısmını bana temin eden FARIMEX S.A., tez çalışmamda gerekli olan bazı entegrelerin temininde bana kolaylık gösteren Texas Instrument Türkiye ofisine, tez donanımındaki elektronik kartların baskı devrelerinin elde edilmesinde yardımcı olan Profilo – Telra'dan Nazım ŞENTÜRK'e ve tez çalışmam sürecinde bana maddi ve manevi her türlü desteği veren aileme sonsuz saygılarımı ve minnettarlığımı sunarım.

Haziran, 2012

Can EROL

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

Ca^{2+}	Kalsiyum İyonu
Cl^{-}	Klor İyonu
HCO^{-}_{3}	Bikarbonat
K^{+}	Pozitif Potasyum İyonu
Mg^{2+}	Magnezyum İyonu
Na^{+}	Sodyum İyonu

KISALTMA LİSTESİ

AR	Autoregressive
ARX	Auto Regressive Model
DBM	Dinamic Bayesian Network
EMG	Elektromyogram
GMM	Gizli Markov Modeli
KNN	K-Nearest Neighbor
LM	Luminiary
MSE	Mean Squared Error
NMSE	Normalised Mean Squared Error
SS-ARX	Signals Stochastic ARX Model

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

YAPAY ZEKA DENETİMİ İLE EMG SİNYALLERİNİN İŞLENMESİ VE SINIFLANDIRILMASI

Can EROL

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doç. Dr. Lale ÖZYILMAZ

Elektromiyografi (EMG) sinyalleri, elektriksel genlikleri oldukça küçük ve diğer organlar tarafından üretilen fizyolojik gürültülerden oldukça etkilenen bir yapıya sahiptir. EMG sinyalleri iğne veya yüzük elektrotlarıyla ölçülerek, karakteristik frekans spektrumlarında, filtrelenir, yükseltilir ve sonuç verilerine ulaşılır. Ölçüm ve filtreleme sırasında, frekans spektrumunun geniş tutulması istenmeyen gölgelere yol açacağı gibi, yine bu aralıkta istenmeyen gürültülerin sinyalin karakteristiğini bozmasına neden olacaktır. Bununla beraber, frekans spektrumunda yapılacak kısıtlamalar, ölçülen sinyalin özelliklerini kırparak, EMG sinyalinin yapısını bozmakta ve ölçülen sinyallerin karakterizasyonunu zorlaştırmaktadır. Dolayısıyla EMG sinyallerinin değerlendirilmesi ve bu sinyallerin karakteristiklerinin belirlenmesi için akıllı algoritmalar kullanılması şarttır.

EMG sinyallerinin birçok inceleme metodu gösterilebilir. Bunlardan biri de bölütlere ayırma yöntemidir. Bölütler ile EMG sinyalleri üzerinde analiz aralıkları oluşturulabilir, bölütten bölüte geçiş aralıkları oluşturulabilir ve bu bölütler üzerinde düğüm noktaları oluşturulabilir. Böylelikle EMG sinyallerinin karekterizasyonu probleminin çözümü için temel parçacıklar oluşturulabilir.

Bu çalışmada, önceden belirlenmiş 4 parmak hareketinden doğan EMG sinyalleri belirli zaman aralıklarında ölçülerek, bu sinyaller bölütlerle parçalandı. Her bir ölçümden sonra oluşturulan bölütler karşılaştırıldı, benzer özellikleri tespit edildi ve benzer

özelliklere göre bu bölütler üzerinde düğüm noktaları oluşturuldu. Bu düğüm noktaları yardımı ile, yapay zekâ algoritması olarak kullanılan Gizli Markov Modeli algoritması işletildi ve parmak hareketleri doğru olarak tespit edilmeye çalışıldı.

Anahtar Kelimeler: EMG Sinyallerinin Ölçümü, Elektromiyografi, Elektromiyogram, Kas Hareketlerinin Tespiti, EMG Sinyallerinin Sınıflandırılması

ABSTRACT

THE DETECTION AND CLASSIFICATION OF EMG SIGNALS WITH USING ARTIFICIAL INTELLIGENCE

Can EROL

Department of Electronics and Communications Engineering

MSc. Thesis

Advisor: Asst. Prof. Dr. Lale ÖZYILMAZ

Electromyography (EMG) signals has a low amplitude, and these signals are effected by the others physiological noise which is created by organs. EMG signals is measured with needle electrode or with surface electrodes, in their characteristic frequency spectrum, they are filtered, amplified and the result of EMG signal can be observed. During the measurement and filtering, with wide frequency spectrum it cause unwanted noise and this noise effect and change signal characteristic of EMG signal. However, the limitations to frequency spectrum cause trimming the propertied of the measured EMG signal, and it makes classification of EM signals as hard. Therefore it is necessary to use intelligent algorithm to utilize and classification EMG signals.

There are a lot of methods to utilize EMG signals. One of them is segmentation algorithm. With the segmentation of EMG signals it can be generated an analysis areas, segment to segment transition area, and can be created node points on these segments. Thus, the basic particles can be created to solution the problem of characterization of EMG signals.

In this study, four pre-determined finger movements are measured and then these signals are separated as segments. After all segmentation process these segments are compared each other, detected same properties and node points are created with using

these properties. With the help of node points, as using artifical intelligence algorithm Hidden Markov Model is run, and the movements of finger is tried to detect as correct.

Key Words: EMG Signals Measurement, Electromyography, Electromyogram, Detection Movement Of Muscles, EMG Signal Classification

BÖLÜM 1

GİRİŞ

İnsanların diğer canlılara göre farklılıklarından biri de sahip olduğu uzuvları daha verimli bir şekilde kullanabilmesidir. İlk çağlardan itibaren her zaman insanlık kendi hareket sistemini anlamaya çalışmıştır. Hareket sistemini oluşturan yapıların arasında şüphesiz ki en önemli öğe kas sistemidir. Kasların çalışma yapısını anlamak için, kaslarda ortaya çıkan elektriksel sinyallerine sık sık başvurulmaktadır.

EMG sinyallerinin ilk tespiti 1666 Redi ile başlar. İlk araştırmaların amacı, kas hareketleri sonrasında ortaya çıkan elektriksel sinyallerin varlığını tespit etmekti. 1922 yılında osiloskobun geliştirilmesi ve iğne elektrotlarıyla, kas lifleri üzerindeki elektriksel aktiviteler ve kas yapısında bulunan motor ünitelerin davranışları birçok teknik ile incelenmeye başlanmıştır.

EMG sinyalleri, kasların metabolizmik aktivite ve kasılma sonrası ortaya çıkan elektriksel sinyallerdir. EMG sinyallerinin ölçülmesine de Elektromiyografi denmektedir. EMG, kaslardaki aktivite bozukluğu ve birtakım kazalar sonucunda (kopma ve yanma gibi) ortaya çıkan işlev bozukluklarının tespiti gibi amaçlar için kullanılmanın yanında; günümüzde gelişmiş yapay organ (protez) ve robot uzuv gibi araştırmalarda sıkça kullanılmaktadır.

Gelişmiş hastalık tespiti ve robot uzuv uygulamaları için, EMG sinyalleri çok detaylı işlenmeli ve karakterize edilmelidir. Her bir kas demetinden çıkan elektriksel sinyallerin, tek tek incelenmesi mümkün değildir. Dolayısıyla, bu tür amaçlar için, kasların bulunduğu demetler üzerinden alınan bölgesel elektriksel sinyallerin karakteristikleri incelenir. Diğer bir önemli kısım ise, elektriksel aktivitelerin nasıl yorumlanacağıdır. Biyomedikal uygulamalarda yorumlama ve sınıflandırma

problemlerinin çözümleri birçok sinyal işleme, yapay zeka ve yapay sinir ağları metotları ile elde edilmeye çalışılmaktadır.

1.1 Literatür Özeti

EMG ile ilgili bilinen ilk deneyler 1666 yılındaki Francesco Redi'nin çalışmalarıdır [2]. Redi deneylerini elektrik balığı üzerinde yaptı ve elektrik balığının kas hareketleri sonucunda, kaslarda oluşan elektriklenmeleri gözlemledi (Electric Eel). 1773'de Walsh, Eel balıklarının kas dokularındaki elektriğin kıvılcım oluşturabilme yeteneğini keşfetti. 1792 yılında Luigi Galvani tarafında, "De Viribus Electritatis In Motu Musculari Commentarius" adlı makale yayınlandı. Bu makale ile bu elektriğin kas kasılmalarını başlatabileceği ispatlandı. 1849 yılında, Emil Du Bois Reymond, istemli kasların kasılmaları sırasında bu kaslardaki elektriksel aktivitenin kaydedilebileceğini ortaya koydu. İlk gerçek EMG kaydı Marey tarafından yapılmıştır ve ilk EMG tanımı Marey tarafından konmuştur. Osiloskobun bulunması ile 1924 yılında Gasser ve Erlander, EMG sinyallerini osiloskop üzerinde gözlemlemeyi başardı. 4 yıl sonra Proebster bu kaslar tarafından üretildiğini bulmuş ve klinik EMG alanını açmıştır (Proebster, R., 1928) [3]. Günümüzde de EMG çalışmaları için kullanılan eş-merkezli iğne elektrodu yöntemini 1929 yılında Adrian ve Bronk geliştirilmiştir (Adrian, E.D. ve Bronk). Devam eden yıllarda motor ünite aksiyon potansiyeli ile ilgili nicel analiz ve metotların (MUAP) geliştiricileri sayılan Kugelberg, Petersen, Buchtal, Guld, Gydikov, Kosarov, Pinelli, Rosenfalck ve Stalberg vakum tüpü yükselticileri ve daha sonra katı hal devrelerini kullandılar (solid state circuits). Denny – Brown 1949 yılında "EMG işaretlerinin gösterimi" belirledi (1984). Uchizono kurbağa sartoris kasındaki işaretlerin yayılımını tanımlamıştır (1953). Willison 1964 yılında EMG işaretlerinin genlik analizini yapmıştır (1964). 1962 yılında J.V. Basmajian tarafından yazılan "Muscles Alive" kitabı bir mihenk taşıdır. 1979 yılında De Luca tarafından yazılan ve bu konuda bir ilk olan EMG içerik bilgisi ve açılması ile ilgili makalesi EMG işaretlerinin fizyolojisi ve matematiksel metotlarını birleştiren bir klasik sayılmaktadır (1979).

Bilgisayarların kullanımı sayesinde bu işaretlerin simülasyonu ve modellerin gelişmesi daha da kolaylaştırılmıştır. 1970'ler ile 1980'lerde birçok araştırma grubu bu konuyla ilgilenmiş ve birçok yayın yayınlanmıştır. Modelleme alanında Dmitrova ve Lindstrom başı çekmişlerdir. Bu modeller EMG işaretinin içerdiği bilgilerin anlaşılması ve işaretin biyofiziğinin anlaşılmasında çok başarılı olmuştur. EMG işaretlerinin geçmiş ve doğal

bir alanı da protez kontrolüdür. Myelectric kontrol olarak adlandırılan bu saha ilk olarak 1940'larda başlamış ve 1960, 1970, 1980'lerde hızlı bir gelişme kaydetmiştir. Örüntü tanıma temelli EMG kontrolörler günümüzde güncel ve gelişen bir alandır.

1965 yılında J.V. Basmijian, S. Carlsöö, B. Johnson, M. MacConaill, J. Pauly ve L. Scheving'in yaptıkları toplantıda Uluslararası Elektormiyografi ve Kinesoloji Topluluğunu (International Society of Electromyography and Kinesiology (ISEK)) kurmayı kararlaştırdılar ve 1966 yılında kurdular. Bu topluluğun çalışma alanları arasında gelişen EMG alanı ile ilgili toplantı, konferans ve sempozyum organize etmek bulunmaktadır.

İğne EMG teknikleri bölgesel ve iğnenin batırıldığı küçük bir alanla ilgili daha geniş ve genel bilgiler verir. Yüzeysel EMG yöntemleri EMG işaretleri ile ilgili daha geniş ve genel bilgiler vermektedir.

Yüzeysel EMG ölçümlerinde, EMG işaretlerinin karakteristiğinde oluşan kayıpları en aza indirmek ve bu ölçümlerin standardizasyonu için birtakım standartlar belirlenmiştir. 1995 yılında bir grup araştırmacının önderliği ve Avrupa Birliğinin katkılarıyla elektrot, elektrot yerleşimi ve işaret işleme tekniklerinin belirlendiği Kasların Noninvassive Fikirleri İçin Yüzeysel EMG (Surface EMG For Noninvassive Assesment Muscles - SENIAM) projesi hazırlanmış ve 1996 yılında yürürlüğe girmiştir. SENIAM'ın amacı; bu konularda çalışan laboratuvarların başarılarını kabul edilebilir bir seviyeye getirmektir.

1.2 Tezin Amacı

EMG sinyallerinin ölçümü birçok amaçla yapılabilir. Son yıllarda EMG sinyallerinin incelenmesiyle, robot uzuv araştırmaları ve kas hastalıklarının teşhislerinde çok önemli mesafeler elde edilmiştir. Modern bilimin yardımıyla, karmaşık gözüken birçok yapı çözülmüştür. Artık küçük bir cihaz yardımıyla, EMG sinyalleri rahatlıkla ölçülebilir hale gelmiştir.

Robot uzuv araştırmalarında, uzuvun bulunduğu kas bölgesi analiz edilir ve bu analizler sonunda, elde edilen ölçümler sınıflandırılır, ayrıştırılır ve elde edilen sonuçlara göre akıllı robot uzuvlar elde edilmeye çalışılır.

Bu çalışmada, el parmakları üzerine yerleştirilen elektrotlar yardımıyla, elde edilen sinyallerinin ölçülmesi ve karekterizasyonu amacıyla, parmak hareketlerinden doğan

EMG sinyallerini ölçülmesi, ölçülen EMG sinyallerinin filtrelenmesi, yükseltilmesi ve sınıflandırılması üzerinde durulmuştur. Parmak üzerinde EMG sinyalleri tespiti için yüzük elektrotlar kullanılmıştır. Böylelikle mümkün oldukça EMG sinyalleri lokalize edilerek, sinyallerin bulunduğu bölge, diğer kas ve organların elektriksel aktivitelerinden uzaklaştırılmıştır.

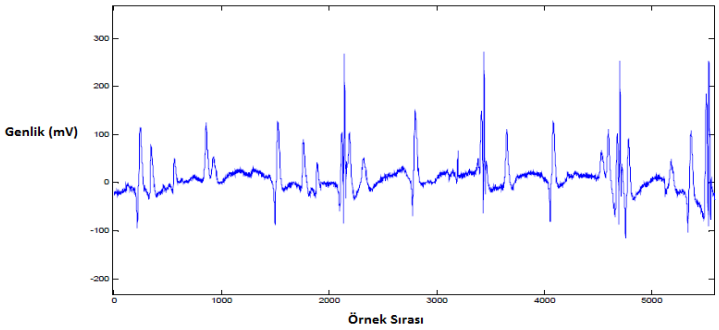
1.3 Orijinal Katkı

Bu çalışmada, kaslardaki aktivitelerden doğan EMG sinyallerinin gerçek zamanlı olarak işlenmesi ve değerlendirilmesi üzerinde durulmuştur. Bu amaçla, ses ve görüntü değerlendirme sistemlerinde sıkça kullanılan, Gizli Markov Modeli kullanılmıştır. Böylelikle Gizli Markov Modeli gibi esnek bir yapıya sahip yapay zeka algoritmalarıyla, gerçek zamanlı olarak EMG sinyallerinin tespitinin başarı oranı ve performansı araştırılmıştır.

BÖLÜM 2

EMG İŞARETLERİNİN ÖLÇÜLMESİ

EMG işareti, kasların kasılması sırasında, kas fiberlerinde oluşan küçük genlikli biyopotansiyel işaretlerdir. EMG sinyallerinin karakteristikleri, ölçülen kas kümesinin bulunduğu yer ile uyarılma şiddetine göre farklılıklar göstermektedir. Şekil 2.1'de örnek bir EMG sinyali görülmektedir.

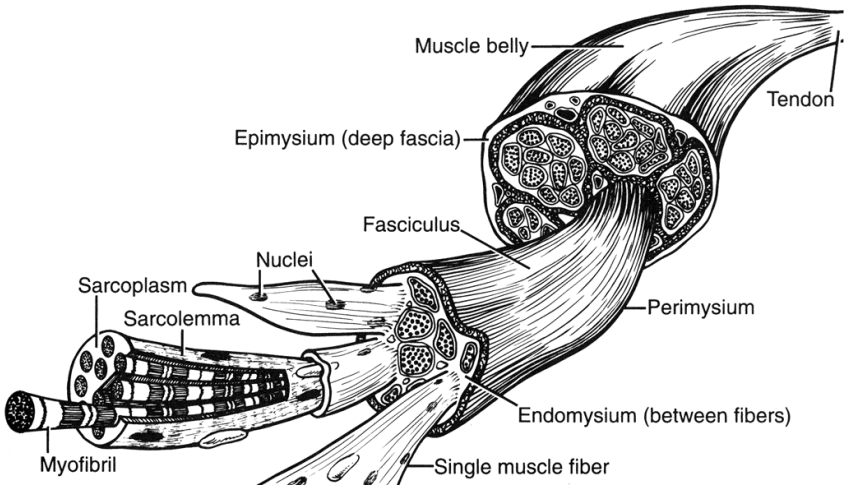


Şekil 2.1 Örnek EMG sinyali [5]

Kas kümelerinin yoğunlaştığı bölgelerde, kasların kasılma durumlarında, kaslardaki motor ünitelerinde oluşan biyopotansiyeller birbirleri üzerinde binmektedir. Bu durum, ölçülen EMG sinyallerinin görünümünü etkilemektedir. Bunun yanında kaslardaki uyarılma şiddeti de, ölçülen EMG sinyallerinin karakteristiğini doğrudan etkilemektedir. Bunun nedeni, kaslardaki uyarılma şiddeti arttıkça, kaslardaki biyopotansiyellerin frekanslarının da artmasıdır. Frekans artışı, yüksek frekanslarda EMG sinyalleri için bir gürültü teşkil etmektedir.

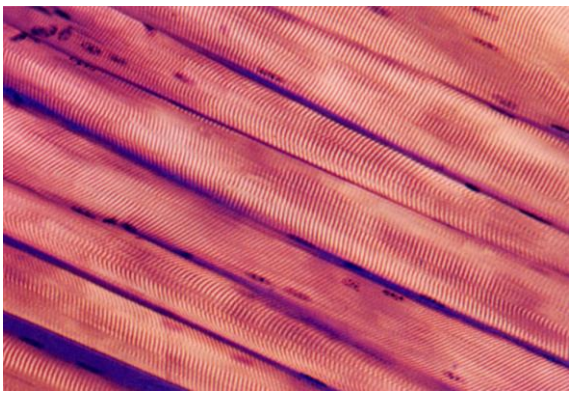
EMG işaretleri vücut üzerinde iğne veya yüzeysel elektrotlarla alınır. Bu tür işaretlerin kuvvetlendirilmesinde diferansiyel (fark) kuvvetlendiricisi kullanılır. İşaretin değerlendirilmesinde zaman domeninde ve/veya frekans domeninde yapılabilir [1].

2.1. Kasların Yapısı



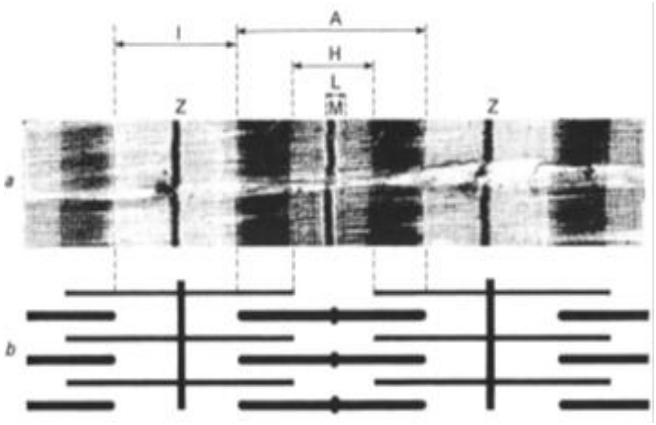
Şekil 2.2 Kas yapısının detaylı gösterimi [4]

Basit olarak kasların yapısı Şekil 2.2'de gözükmektedir. Kas gövdesi, demet demet bir yapıya sahiptir. En büyük demet fascicles ve herbir fascicle kas fiber demetidir. Kas demetlerini çevreleyen yapıya epimsium, fascicle'ları çevreleyen yapıya perimysium ve fiberleri çevreleyen yapıya ise endomsium denir. Diğer dokulardan farklı olarak insan kaslarında çok büyük miktarda kas bulunmamaktadır [4].



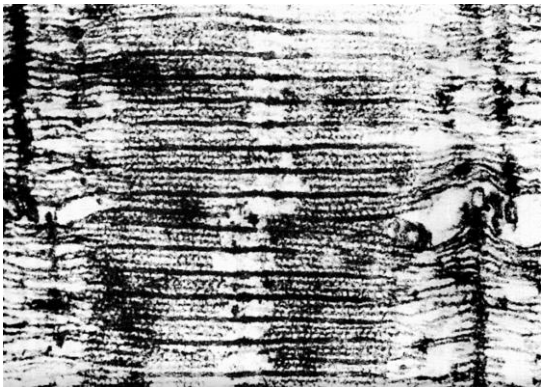
Şekil 2.3 Işın mikroskobu ile kas fiberlerinin gözlenmesi [4]

Şekil 2.3'te olduğu gibi, kas fiberlerine ışın mikroskobu altında bakıldığında, kas fiberleri çizgi çizgi görülmektedir. Böylece kas fiberleri içerisinde alt birimler olduğunu anlaşılmaktadır. Bu çizgiler polarize ışık mikroskobu altında daha iyi görüntülenebilir ancak fiberleri daha net görüntülemek için elektron mikroskobuna ihtiyaç vardır.



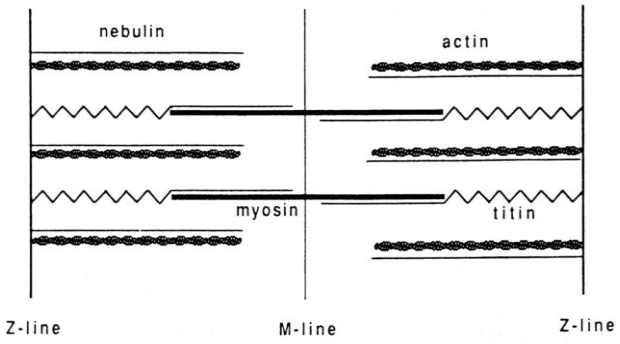
Şekil 2.4 Kas yapısındaki aydınlık ve karanlık bölgeler [4]

Şekil 2.4'te orta güç elektron mikroskobide, detayları daha iyi görebiliriz. Farklı kısımlar harfler ile ayrılmıştır. Çizgileri oluşturan karanlık ve aydınlık bantlar vardır ve bu bantlar sırasıyla I (izotopik) ve A (anizotopik) olarak bilinirler. Bir bant içinde orta tonda bir H (Heller) bölgesi vardır: Z (Zwischenscheibe) ve M (Mittelsheibe) hatları. L bölgesi bazen M hattı içerisinde hafif açık bir bölge olarak görülebilir. Bu yapı fiberin uzunluğu boyunca tekrar etmektedir ve bu uzunluk üzerindeki tekrarlar (yani iki komşu Z bölgesi arasındaki mesafe) ise sarkomer adını alır (genellikle birkaç μm). Bu yapı aşağıdaki miyografide gözüktüğü gibi filamentlerin geçiş kümesi olarak da yorumlanır.



Şekil 2.5 Kas miyografisi ile kasların incelenmesi [4]

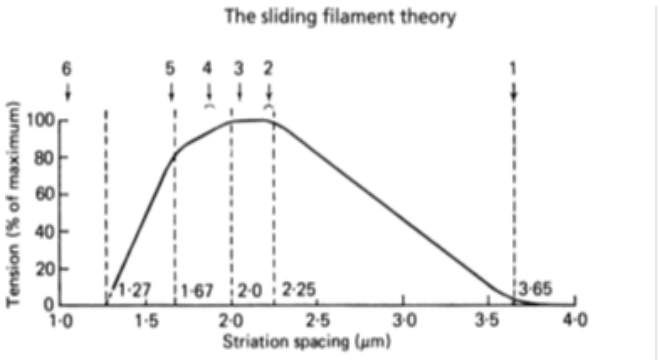
Yüksek güç ile bakıldığında (Şekil 2.5'te olduğu gibi) bu yorum daha da desteklenebilir. Bu yoruma ek olarak, filamentler arasında çapraz bağlantıların sürülebileceği gözükmektedir. Bunlarda crossbridges olarak adlandırılır.



Şekil 2.6 Kas yapısındaki çapraz bağların gösterimi [4]

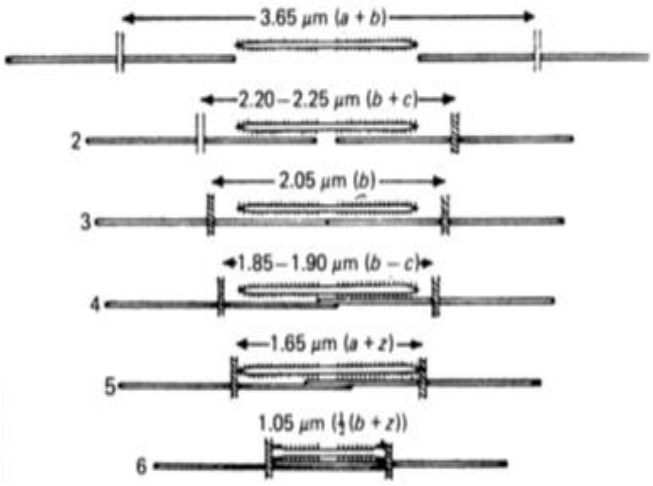
Şekil 2.6'da görüldüğü gibi, bu yapı içerisinde I (ışık) bantlarındaki actin ve A (karanlık) bantlarındaki filamentleri tespit etmek mümkündür. Bu actin ve myosin proteinlerinin öncelikli görevleri olarak, kasılma süreci olduğu görülmektedir. Nebulin ve titin ise korumaya yardımcı olarak görev yapmaktadır.

Kaslarda daha fazla bilgi edinilmek istendikçe, kaslardaki komplekslik artmakta ve kas boyutları daralmaktadır. Son yıllarda kasların yapısı hakkında kullanılan standart teori "Sliding Filament Theory"'dir. Bu teoride, aktin ve miyozinin kasılma sırasında birbiri üzerini örtmesi ve dinlenme sırasında bu örtüm miktarının azalması yani kasılma sırasında boylarının artması ve dinlenme sırasında boylarının azalmasıdır.



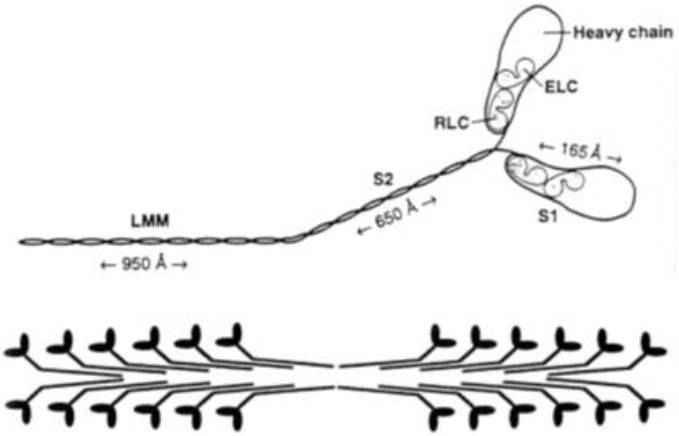
Şekil 2.7 Sliding Filament Teorisi [4]

Şekil 2.7'de ki deneysel sonuçlar ile bu teoriyi desteklemektedir. Kas farklı uzunluklarda gerildiğinde (yüzeye çıkarılmış bir kurba gastroknemius kası) üretebildiği maksimum kasılmayı göstermektedir. Kasın uzunluğu kadar sarkomer uzunluğu da değiştirilebilir ve bu çizgiler arasındaki mesafeler gözlemlenip ölçülebilir. Grafiktede görüldüğü gibi ölçülen gerilim çok kısa ve çok uzun kaslarda 0'a düşer ve ortasında en yüksek değerler alınır [4].



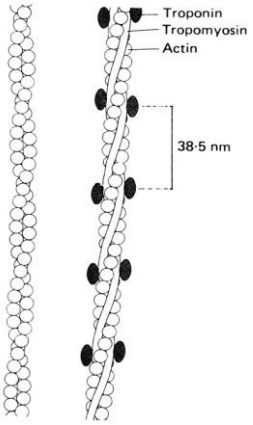
Şekil 2.8 Kasılmaya bağlı flament ve sarkomerdeki değişim [4]

Şekil 2.8'de filamentler ile sarkomerlerin birbiri üzerinde örtüşmesini göstermektedir. Maksimum gerinimde, hiçbir örtüşme ve hiçbir stresin olmadığı gözükmektedir. Minimum uzunlukta örtüşme vardır; ancak miyozin flamentleri Z hattına doğru itilir yani kas daha fazla kısalamaz ve harici güç üretemez. Orta uzunlukta üretilen kasların örtüşme miktarı değişkendir ve aktin iplikleri arasındaki etkileşime bağlı olarak değişir.



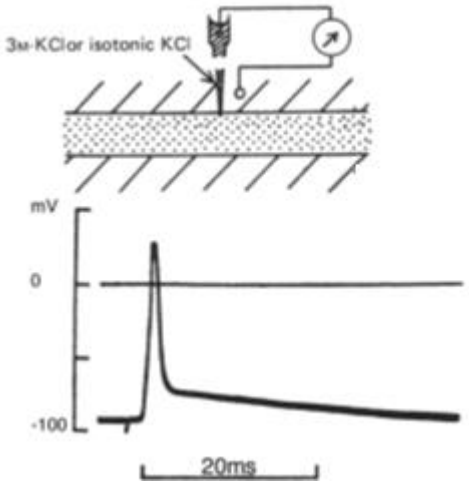
Şekil 2.9 Miyozin molekülünün yapısal analizi [4]

"Sliding Filament" iyi bir teoridir ancak, bazen kayma hareketi üretimi için moleküler metot kullanılmak zorundadır. Üretilen kuvvet öncelikle miyozin molekülleri ile ilişkili çapraz köprülerin bir fonksiyonu olarak görünür. Şekil 2.9'da molekülün daha detaylı yapısal analizi görülmektedir. Görüldüğü gibi, 4 peptit alt üniteleri ve kompleksimsi formlardan oluşur.



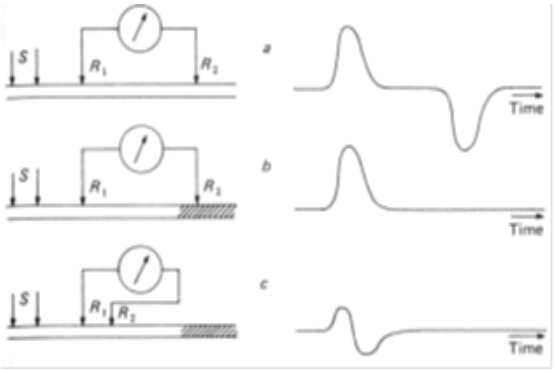
Şekil 2.10 Aktin zincirlerinin detaylı gösterimi [4]

Şekilde 2.10'dan görüldüğü gibi, aktin zincirleri çok daha basittir. Aktin küresel bir moleküldür ve troponin ile tromypsin ile birlikte spiral filementleri oluşturur. Aktin ile miyozinin birleşmesi ile viscoz komplex oluşur ve bu yağı actomysion ile adlandırılır ve bu kompleks oluşum ATPaz olarak – ATP'yi enerji türetmek amacıyla ADP (adonozindifosfat) ve P (fosfor) olarak iki parçaya ayırır. ATP -> ADP + P, hücrelerin aktiviteleri için gerekli olan enerjiyi türetmek için genel kimyasal reaksiyondur.



Şekil 2.11 Kas uyarımı sırasında değişen hücre içi potansiyel [4]

Kas lifleri, sinir hücreleri gibi uyarılabilir hücrelerdir. Şekil 2.11'de görülebildiği gibi, hücre içi, hücre dışına göre 100 mV negatiftir. Kimyasal veya fiziksel olarak hücre uyarıldığında, elektriksel potansiyelde değişiklik olur, hücre membranı polarize olmuş hale gelir ve hücre içi dışına göre pozitif 20 mV olur. Bu etki ve değişim ile hücre, hücrenin bir parçası olan ve hücreyi çevreleyen membranı uyarır ve depolorize olur. Bununla birlikte deporilize olmuş membran kısa bir süre boyunca tekrar uyarılamaz.



Şekil 2.12 Kas hücrelerinin depolarizasyonu [4]

Şekilde 2.12'de, dışarıdan elektrotlar yardımı ile alınan bir kasın depolorizasyon süreci gösterilmiştir. Şekil 2.12'den de anlaşılacağı gibi, dalga formu farklılıklar göstermektedir. Bu dalga formu tek bir kas fiberinde ve kayıt edilen tüm kaslardan alınan liflerin elektriksel sinyalleri toplamıdır ve bundan dolayı daha fazla kompleks yapı oluşturur.

Çizelde 2.1 Hücre içi iyon yoğunlukları ve elektriksel potansiyel farklılıkları [4]

	Fiber Sıvısı (mM)	Plazma Sıvısı (mM)
K^+	124	2.3
Na^+	3.6	108.8
Ca^{2+}	3.9	2.1
Mg^{2+}	14	1.3
Cl^-	1.5	77.9
HCO_3^-	12.4	26.6
Phosphocreatine	35.2	–
Organic anions	c.45	c.14

Çizelge 2.1'de, birçok iyona ait yoğunluklar verilmiştir. Elektriksel potansiyeller farklı iyonik konsantrasyonlar ile üretilir. Membran seçici geçirgen olduğu sürece, bu iyonik konsantrasyonlar elektrojeniktir. Membranın seçici geçirgen durumundan dolayı oluşan dinlenme potansiyeli potasyum iyonundan dolayıdır ve Nernst denklemi ile hesaplanır (oda sıcaklığında $E_K = 25 \log_e [K]_0 / [K]_i$ mV).

2.2 Motor Ünite

İnsan motor sistemi; iç ve dış istekler ile sınırlama ve çok geniş farklılıklarıyla baş etmek zorundadır. Bu farklılıklar; dışarıya uygulanacak kuvvetin güçlü ve hassas bir

şekilde ayarlanması, dik duruş ve birçok el kol hareketlerini içermektedir. Birçok motor sistemin özel kontrol öğelerini anlatmak çok zor olacağından burada güç ve hareket kontrolünde büyük pay sahibi olan iskelet motor sistemi kontrolünün temel prensipleri anlatılacaktır.

Merkezi Sinir Sistemi hiyerarşik bir yapıda örgütlenmiştir. Motor programlama; premotor korteks, ek motor alanı ve korteksin ilişkili olduğu diğer bölgelerde bulunur.

Bu alanlardan gelen girişler birincil motor korteksine düşer ve burada bulunan çeşitli nöronları harekete geçirir. Birincil motor korteksinin çıkışları beyin sapı ve omirilikte bulunan iç nöronlar ve motonöronlar üzerinde kuvvetli bir tesiri bulunmaktadır.

Motor üniteler (MU) omurilikte bir α-motonöron ve kas lifinden oluşmaktadır. α-motonöronlar gelen bütün girişlerin ve reflekslerin son toplanma noktasıdır. MU'ların sayısı kastan kasa değişmektedir. Mesela küçük bir el kasındaki MU'ların sayısı 100 ile 1000 arasında değişmektedir. Bu sayı kol ve bacak kaslarında daha da fazladır.

Aynı zamanda MU'ların sayısı güç üretme kapasitesine göre de değişiklik oluşturmaktadır. Yapılan ilk çalışmalar fizyolojik özelliklerine göre, 3 tip MU olduğunu göstermiştir. Bunlar:

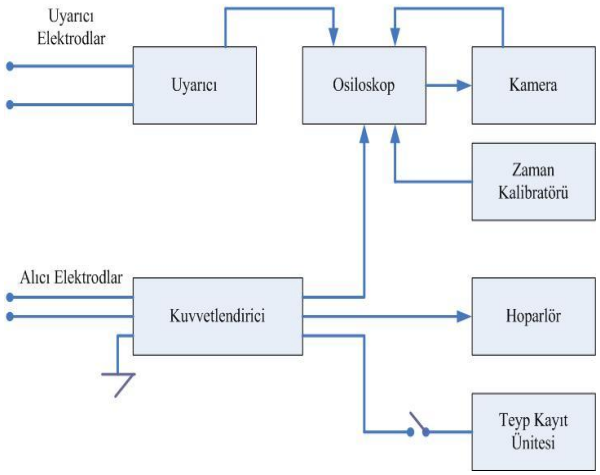
- S veya I tipi Motor Ünite: Yavaş seğirir, oksidatif enzimler içerir, en yavaş kasılma ve seğirme gerilmesi veya en küçük kuvvetle yorulmaya karşı koyan birimlerdir.
- FR veya II-a Tipi Motor Ünite: Hızlı seğirir, oksidatif ve glikolitik enzimler içerir, daha hızlı zamanları ve daha geniş kuvvetle yorulmaya karşı koyan birimlerdir.
- FF veya II-b Tipi Motor Ünite: Hızlı seğirir, glikolitik enzimler içerir, en hızlı kasılmalar ve en geniş kuvvetlerle kolayca yorulabilir [1].

2.3 EMG Ölçüm Düzeneği ve Gürültü Karakteristiği

2.3.1 EMG Ölçüm Düzeneği

Kliniklerde kullanılan EMG ölçüm düzenekleri genelde, sinyali algılayıcı olarak elektrotlar, elektrotlardan alınan sinyali yükseltmek için kuvvetlendirici devresi,

yükseltme öncesi ve sonrasında gürültünün yok edilmesi için filtreleme devreleri, çıkışları gözlemlemek için osiloskoptan oluşur. Bu yapı Şekil 2.14'de gösterilmiştir.



Şekil 2.13 Tipik bir EMG ölçüm ve değerlendirme sistemi şeması [3]

2.3.2 EMG İşaretinin Gürültü Karakteristiği

EMG sinyalleri düşük genlikli bir yapıya sahip olması itibariyle, gürültülerden oldukça etkilenebilmektedir. Düşük değerdeki EMG sinyallerini, anlamlı bir yapıya kavuşturmak için, EMG sinyallerinin gürültü karakteristiklerini çok iyi bir şekilde analiz edilmesi gerekmektedir [6].

EMG sinyalleri için, çevresel ve dönüştürücü gürültüsü olmak üzere, iki ana gürültü kaynağından söz edilebilir:

2.3.2.1 Çevresel Gürültü Kaynakları

Bu tür gürültüler, ölçüm yapılan ortamı etkileyen, bilgisayar, cep telefonu, çevrede bulunan elektrik hatları gibi faktörler kaynaklıdır. Genel olarak üzerinden akım geçen her türlü araç bir gürültü yayar. Bu tür gürültü kaynaklarının çok geniş bir frekans aralığı vardır, ancak A/C güç kaynaklarıyla alakalı olarak baskın etkileri 50 veya 60 Hz'dir.

2.3.2.2 Dönüştürücü Gürültüsü

Bu tür gürültüler, elektrot ile derinin birbiriyle birleştiği bölgede oluşur [7]. Elektrotlar kaslarda oluşan iyonik akımları elektriksel akıma çevirirler, böylece bu iyonik akımlar işlenebilir, voltaj potansiyeli formunda analog veya dijital olarak kaydedilebilirler. İyonik formdan elektriksel forma dönüştürürken iki tip gürültü kaynağı mevcuttur:

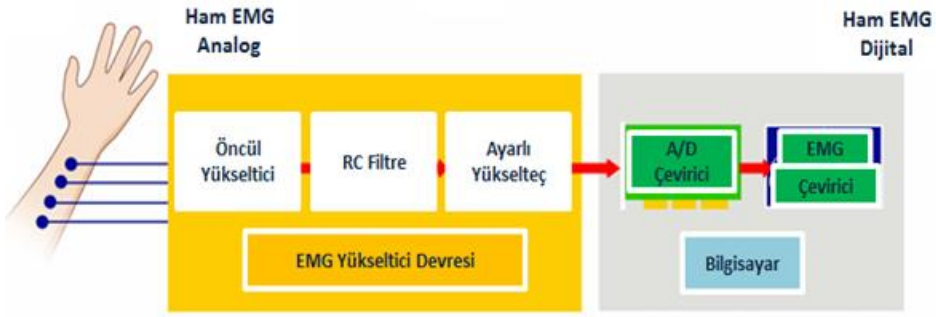
- DC (Doğru Akım) Kaynaklı Potansiyelleri: Bu tür gürültü kaynakları, deri ile elektrot sensörü arasında direnç farkı ve elektrotun bulunduğu yerdeki oksidatif ve indirgeyici kimyasal reaksiyonlar kaynaklıdır. Bu tür gürültü kaynaklarının önüne geçmek için iletken jel kullanılır.
- AC (Alternatif Akım) Kaynaklı Potansiyelleri: Dönüştürücü ve deri arasında oluşan değişken dirençten kaynaklanır. Bu tür dirençlerin etkisini azaltmak için Ag-AgCl tip elektrotların kullanılması bir efektif yol olarak gösterilebilir. Bu tür elektrotlar ince bir gümüş tabaka ve bu tabakanın üzerinde ince gümüş klorid katman bulunmaktadır.

Diğer bir gürültü kaynağı ise, ölçüm yapılan bölgeye yakın organların faaliyetleri sonrası doğan potansiyeller de gösterilebilir.

2.3.3. EMG İşaretinin Tespiti, Filtrelenmesi Ve Yükseltilmesi

EMG sinyalleri, 0 ile 10 mV (tepeden tepeye) veya 0 ile 1.5 mV (rms) arasında değere sahiptir. EMG sinyallerinin frekans aralığı 0 ile 500 Hz olmasına rağmen, kullanılabilir EMG sinyalleri aralığı baskın olarak 50 ile 500 Hz arasındadır.

EMG ölçümündeki ana hedef mümkün olduğunca gürültülerin tespit edilmesi ve bu gürültülerin yok edilmesidir. EMG ölçümünde sıkça bipolar elektrotlar kullanılır. Bipolar elektrotlar ile bastırılmak istenen sinyaller için her iki elektrotun ortak işlevi kullanılır. Böylece enstrümantasyon kuvvetlendiricileriyle bipolar elektrotlar birleştirilerek en iyi sonuç elde edilir. Bunun yanında en iyi sonucu elde edebilmek için elektrotun çıkışını yüksek empedansta tutmak yararlı olacaktır. Bu düzenek Şekilde 2.15'de gösterilmiştir [8].



Şekil 2.14 EMG ölçüm ve değerlendirme düzeneği [8]

BÖLÜM 3

EMG İŞARETLERİNİN İNCELENMESİ İÇİN YÖNTEMLER

Günümüzde EMG sinyallerinin incelenmesi için; birçok sinyal işleme, istatiksel inceleme, örüntü işleme ve yapay sinir ağı yöntemleri kullanılmaktadır. Bu çalışmada, EMG sinyallerinin karakterizasyonu için, bir örüntü tanıma yöntemi olan Gizli Markov Modeli kullanılmıştır. Bölüm 3'de genel olarak EMG sinyallerini işleme yöntemlerinden bahsedildikten sonra, Gizli Markov Modelinden bahsedilecektir.

3.1 EMG İşaretlerinin Analizi

EMG işaretlerinin analizini üç alt başlık altında incelemek mümkündür:

1. Modelleme
2. Zaman Domeni Analizi
3. Frekans Domeni Analizi

3.1.1 Modelleme

Sistem üzerindeki çalışmalar, model kurma ve sistem simülasyonu, son yıllarda büyük başarı göstermiştir. Bunun en önemli nedeni modellemeyi kolaylaştıran sistem teorisi ve bilgisayar teknolojisindeki ilerlemelerdir. Model kurma, biyomedikal mühendisleri tarafından daha geniş olarak kullanılmakta ve insan vücudundaki sistemlerin matematiksel modelleri kurularak, bu sistemlerin dinamik davranışları ve çalışmaları iyi bir şekilde incelenebilmektedir. Modeller, sistem parametrelerinin direkt olarak ölçülmediği durumlarda sistem giriş ve çıkışları ölçülerek buradan parametre kestirimine gidilmesi konusunda başarılı olarak kullanılmaktadır. Bu bağlamda

modelleme, sistem modellemesi ve işaret modellemesi (parametrik modelleme) olarak iki başlık altında incelenebilir.

Sistem modellemesi; sistemi temsil edecek modeli kurabilmek için öncelikle, sistem içindeki mekanik, kimyasal ve fiziksel olayları ve bu olaylara ilişkin temel bağlantıları bilmek gerekmektedir. Bu bağıntıların matematiksel ifadeleri oluşturularak, buradan sistemin çalışmasının ve dinamiğinin daha iyi analiz edileceği ortamdaki modeline geçişir. Model kurulduktan sonraki aşamada, model parametreleri belirlenmeli ve sistem ve model girişleri aynı olduğu anda, sistem model çıkışlarının da aynı olması gerektiğine dikkat edilmelidir. Parametreler belirlendikten sonraki adım, modelin test edilmesidir. Uygun test sonuçları sonunda model artık kullanılmaya hazırdır. Elde bulunan matematiksel model aynı zamanda bilgisayar ortamında ve analog ortamda simüle edilebilir.

Parametrik modelleme; bu modelleme çeşidinde sistemin fiziksel modeli ortada yoktur. Ancak sistemi temsil eden çıkış işaretinin, belli bir matematiksel forma uydurulması ve bu formdaki parametrelerin belirlenmesi söz konusudur. Zaman Serileri Analizi veya Doğrusal Öngörme (Linear Prediction) olarak da bilinen bu modelleme çeşidinin belli başlıcaları durağan olduğu varsayılan işaretlere uygulanan öz bağımlı modu (autoregressive mode), Yürüyen Ortalama (Moving Average) modeli ve bunların karışımı olan Öz Bağlanımlı Yürüyen Ortalama modelidir. Parametrik modellemede süreç, oransal ve nedensel olarak modellenir. Süreç modellendikten sonra parametrelerle temsil edildiği için, bu işlem, veri sıkıştırma veya sınıflama amaçlarına yönelik olarak tıp elektroniğinde yaygın olarak kullanılmaktadır.

3.1.2 Zaman Domeni Analizi

Bir işaretin, incelenebilecek olan ve işareti karakterize eden birçok zaman domeni ölçümü veya parametresi vardır. Bunların en önemlesi ortalama (3.1) ve efektif değerleridir (3.2). İşaret rastgele ise bu parametreler işaretin ortalama değeri ile varyansı (3.3) olmaktadır. Bunlar, x(n) ayrık zaman ve x(t) sürekli zaman işaretleri için aşağıdaki eşitliklerdeki gibi ifade edilebilir:

$$x_0 = \bar{x} = \frac{1}{N}\sum_{n-1}^{N} x(n) \quad \bar{x} = \frac{1}{T}\int_{t=0}^{T} x(t).dt \tag{3.1}$$

$$x_{rms} = \sqrt{\frac{1}{N}\sum_{n-1}^{N} x^2(n)} \quad , \qquad x_{rms} = \sqrt{\frac{1}{T}\int_{t=0}^{T} x^2(t).dt} \tag{3.2}$$

$$\sigma^2 = \frac{1}{N-1}\sum_{n-1}^{N}[\,x(n) - \bar{x}\,]^2 = \frac{1}{N-1}\left[\sum_{n=1}^{N} x^2(n) - N.\bar{x}^{\,2}\right] \tag{3.3}$$

İşaret ortalamasının kullanıldığı ortak amaçlardan biri de işaret/gürültü oranı iyileştirmektir. EMG işaretlerinin analiz etmek ve kasların fizyolojik durumlarını anlamak için frekans domeninde olduğu kadar; zaman domeninde de pek çok parametre kullanılmaktadır. Bu parametreler için mutlak değer ortalama, efektif değer ve mutlak değer integralin sırasıyla formülasyonları (3.4), (3.5), ve (3.6)'da ki gibidir.

$$\overline{|x|} = \frac{1}{T}\int_{t=0}^{T} |x(t)|.dt \tag{3.4}$$

$$x_{rms} = \sqrt{\frac{1}{T}\int_{t=0}^{T} x^2(t).dt} \tag{3.5}$$

$$I|x(t)| = \int_{t=0}^{T} |x(t)|dt \tag{3.6}$$

(3.4), (3.5) ve (3.6)'da ki eşitliklerle, sürekli zaman işaretleri için yazılmış olan işaretin mutlak değer ortalaması, efektif değer ve mutlak değer integrali örnek verilebilmektedir. Bu parametrelere bakılarak kas hastalıklarının incelenmesi, kas yorgunluk seviyelerinin belirlenmesi ve otomatik kol-bacak protezlerinin kontrolü gerçekleştirilebilmektedir.

3.1.3 Frekans Domeni Analizi

Sonlu bir determenistik ayrı-zaman işaretinin, x(k), enerji spektrum yoğunluğu işaretin ayrık zaman Fourier dönüşümü genliğinin karesine eşittir ($|X(e^{jw})|^2$). Bu tanım Parvesal teoreminin bir sonucudur. Geniş çağlı durağan stokastik sürecin ayrık-zaman spektral güç yoğunluğu (PSD), işaretin otokorelasyon fonksiyonunun Ayrık-Zaman Fourier dönüşümü olarak tanımlanır.

$$S_{mm}(e^{jm}) = \sum_{k=-\infty}^{\infty} r_{mm}(k)e^{-jkm} \quad (3.7)$$

e^{-jwk} **:** işaretin k. harmoniği

$r_{mm}(k)$: işaretin otokorelasyon fonksiyonu

(3.7)'de otokolerasyon dizisi sürecin bütün hesaplamalarını içeren tahminleri bulundurmaktadır. Açıkçası pratikte bu verilerin toplanması mümkün değildir. Bunula birlikte; Ergodik süreç durumunda kestirim işlemini o ana ait ortalama ile yer değiştirirsek otokolerasyon dizisi tahmin edilebilir. Sınırlı sayıda örnek olması durumunda otokorelasyon dizisi aşağıdaki şekli alır.

$$\widehat{r_{mm}}\ (k) = \frac{1}{L} \sum_{t=0}^{L-1-k} m\ (k+1)m(l)\ \ 0 \le k\ \le L \quad (3.8)$$

(3.8)'de $r_{mm}(k)$ ile $\widehat{r_{mm}}(k)$ ile yer değiştirirsek güç spektrumunu hesaplayabiliriz. Kestirilmiş otokolerasyon dizisi sistematik hatayı (bias) azaltmak için genellikle pencerelenir. Bu sayede;

$$\hat{S}_{mm}(e^{jm}) = \frac{1}{L}\ |M(e^{jw})|^2 \quad (3.9)$$

(3.9)'da belirtilen kestirici (estimator), Periodogram olarak tanımlanır. Periodogram spektrumunun asimptotik yansız kestiricisidir (unbiased estimator) [1].

3.2 EMG İşaretinin Sınıflandırılmasında Güncel Yöntem Örnekleri

Günümüzde EMG işaretinin işlenmesi ve sınıflandırılmasına ait birçok yöntem üzerinde durulmaktadır. Bu yöntemlerin birçoğu yapay zekanın iki ayrı konusu olan yapay sinir ağları ve örüntü tanıma yöntemlerine aittir.

Bu kısımda dört tane güncel tanımlama yöntemi kullanılarak yapılan örnek çalışmalar üzerinde durulacak ve bir sonraki başlıkta tezin temelini oluşturan ve bir örüntü tanıma tekniği olan Gizli Markov Modeli (GMM) üzerinde durulacaktır.

3.2.1 KNN Algoritması Kullanılarak EMG Sınıflandırılması (Classification Of Finger Activition for Use in a Robotic Prosthesis in Arm)

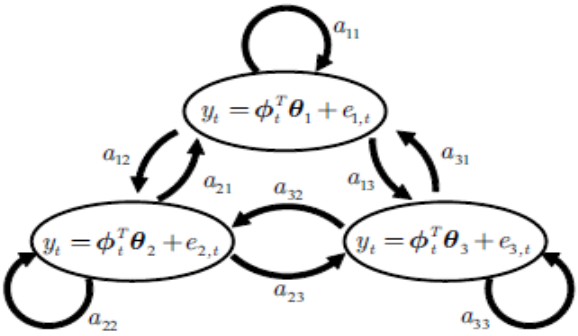
KNN (K-Nearest Neighbour) algoritması kullanılarak yapılan çalışmalardan biri olarak, gösterilebilir [9]. Bu çalışmada, el ve kol kasları üzerinde alınan EMG sinyalleri incelenmiştir. Bu amaçla elektrotlar ile alınan EMG sinyalleri kayıt edilmiş ve bu sinyaller üzerinde KNN algoritmasına uygulanmıştır.

Bu amaçla, aktive edilmiş parmakların özellikleri modifiye edilmiş KNN sınıflandırıcı ile anlaşılmaya çalışılmıştır. Standart bir KNN sınıflandırıcısı, test kümesi ile eğitim kümesi arasındaki en yakın mesafeyi bulur ve test mesafesine göre yeni sonuç oluşturur. Geliştirilmiş KNN sınıflandırıcısı ise, eğitim kümesine en yakın mesafeyi bulur ve standart KNN'den farklı olarak bunun yanında en yakın diğer iki mesafeyi de hesaba katar. Bu durumda, test kümesi ile eğitim kümesi arasındaki mesafe, ortalama uzaklık olarak bulunur.

Bu yönteme ek olarak, sınıflandırıcının hata oranı beş kat çapraz doğrulayıcı kullanılarak hesaplanmıştır: Datalar beş eşit alt kümeye ayrılmıştır. Bir alt küme test için kullanılmıştır ve diğer dört alt küme sınıflandırıcının eğitimi için kullanılmıştır. Böylece her bir hata oranı her defasında tekrar hesaplanarak ortalama bir hata oranı bulunmuştur.

3.2.2 EMG Sinyalleri Kullanılarak Yorgunluk Tanımlama ve İstatistiksel Anahtarlanmış ARX (Auto Regressive eXogenous) Model (Fatigue Recognition Using EMG Signals Stochastic Switched ARX Model)

Çalışmanın konusu olarak, EMG sinyallerine dayalı yorgunluk tanımlama ve GMM'ye dayalı istatiksel olarak anahtarlanmış ARX (SS-ARX) model üzerinde durulmuştur. SS-ARX model GMM'in bir uzantısı olarak yorumlanabilir. Her bir ARX model, GMM'nin ayrık durumları içerisinde gömülüdür. Bu çerçevede, sinyallerinin hiç birisi tek başına ölçülmemeli ve birbiri arasındaki ilişkiye dikkat edilmelidir. ARX model, anahtarlama ve istatiksel varyans içeren kompleks dinamik bir model olarak temsil edilebilir ve GMM'de kullanılan yorgunluk tanımlayıcı olarak yüksek performans beklenmektedir. Aşağıdaki Şekil 3.1'de SS-ARX model gösterilmiştir [10].



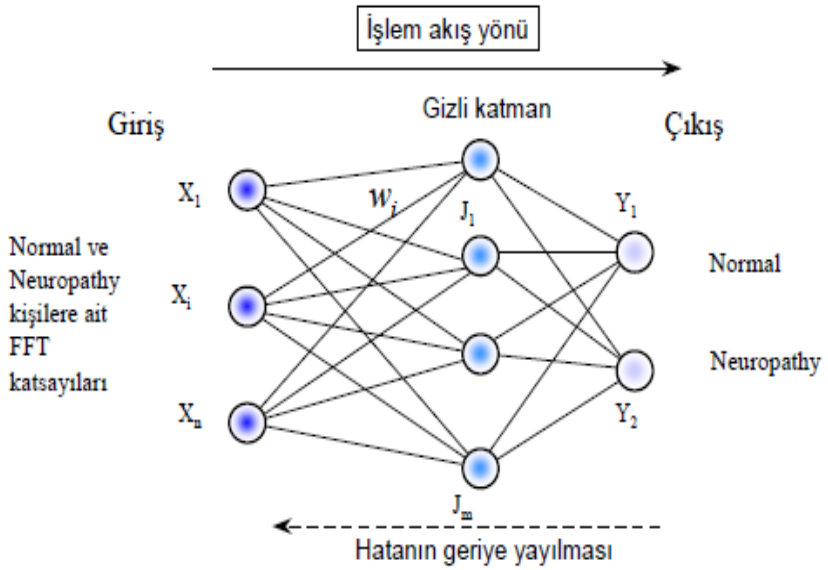
Şekil 3.1 SS-ARX Model Diyagramı [10]

3.2.3 EMG Sinyalleri Bölütlenmesi İçin Karşılaştırmalı Teknik

Bu çalışmada nöromuskular hastalıkların tespiti için EMG sinyallerinin genliklerine dayalı, seçici bir karakterizasyon algoritması olan bölütleme kullanılmıştır. Bu süreçte en düşük ve en yüksek ölçülen değer arasında bir bağıntı bulunur ve bu bağıntıya göre, bir eşik değeri belirlenir ve bu eşik değerinin altında kalan değerler sıfıra eşitlenir ve hesaba katılmaz [11].

3.2.4 Yapay Sinir Ağları Kullanılarak EMG Sinyallerinin Neuropathy Kas Hastalığının Tespiti

Bu çalışmada ileri beslemeli (feedforward), çok katmanlı idrak (Multi Layer Perceptron – MLP) YSA mimarisi kullanılmıştır. MLP mimarisi Şekil 3.2'de verilmiştir.

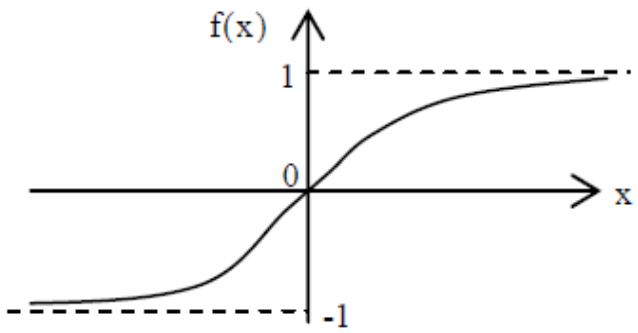


Şekil 3.2 MLP mimarisinin gösterimi [12]

YSA, yaklaşım olarak, insan beynini oluşturan fonksiyonlar arasında bir benzerlik olduğunu kabul eder. İleri beslemeli YSA ağları Şekil 3.2'de izlenildiği gibi giriş, gizli ve çıkış katmanlarındaki nöronlardan oluşmaktadır. Burada giriş, gizli ve çıkış katmanlarındaki her vektörel bağlantının ağırlık katsayıları olan w_i değerleri mevcuttur. Giriş katmanı ($x_1 \ldots x_i \ldots x_n$), EMG işaretlerinin HFD sonucu elde edilen istatikseli bilgileri hiçbir değişikliğe uğratmadan gizli katmandaki ($j_1 \ldots j_m$) hücrelere iletir ve daha sonra iletilen bu veriler, gizli ve çıkış katmanında işlenerek ağ çıkışına ait sonuçların (y_1,y_2) belirlenmesini sağlar.

Giriş, gizli ve çıkış katmanları arasındaki her vektörel bağlantının ağırlık katsayısı, katmanlar arasındaki her işlem elemanının yani bağlantının, diğer işlem elemanı üzerinde bir etkiye sahiptir.

YSA içerisinde gerçekleştirilen işlemler sonucunda, çıkış katmanında oluşan sonuç değerleri ile arzulanan sonuçlar karşılaştırılır ve aradaki farka bağlı olarak ağırlık değerleri yeniden düzenlenir. Ağın eğitilmesi sırasında ilk anda ağırlıklar 0 ile 1 arasında rastgele olarak atanır. Bu rastgele atamalardan dolayı çıkıştaki değer ile istenilen sonuç arasında bir hata olacaktır. YSA öğrenme esnasında, oluşan bu hataları kendi içerisinde değerlendirerek, sonuç en az hataya ulaşıncaya kadar bu öğrenme devam edecektir. Burada ağırlık düzeltme işlemi çıkışa yakın ağırlıklardan başlayarak, işleme ters yönde giriş seviyesine varıncaya kadar devam edecektir. Bütün bu işlemler istenilen sonucu elde etmek için, hatayı belli bir değerin altına indirinceye kadar tekrarlanacaktır. Ayrıca, nöronların girişine ve çıkışına veri aktarımı için kullanılan fonksiyonu tanjant hiperbolik (Tanh) olarak ele alınmıştır. Şekil 3.3'te görüleceği üzere Tanh transfer fonksiyonu çıkışı +1 ile -1 arasında değişmektedir.



Şekil 3.3 Tanh transfer fonksiyonu [12]

YSA'da öğrenme esnasında yapılması gereken konulardan bir tanesi de öğrenme katsayılarının ayarlanmasıdır. Ağırlıkların çok yüksek tutulması yüzünden YSA'nın

öğrenme davranışı bozulabilir. Bunu önlemek için öğrenme katsayısını tutmak gerekir. Öğrenme katsayısı 0.01 ile 10 aralığında seçilen sabit bir sayıdır. Öte yandan küçük öğrenme orantıda, öğrenme işleminin yavaşlamasına yol açacağından, bu çalışmada eşlenik gradyan ve hızlı prop öğrenme teknikleri kullanılarak bunların sınıflamadaki performansları incelenmiştir.

Eşlenik gradyan algoritması ağırlık değerlerinin en iyi şekilde karar verebilmek için performasın yüzeyinin eğiminden yararlanır. Bu algoritma yardımıyla istenen ve gerçek ağ çıktılarının arasındaki ortalama karesel hata değeri azaltılmıştır.

Hızlı prop, Soctt Fahlman tarafından gerçekleştirilen çok katmanlı ağlar için kullanılan bir öğrenme algoritmasıdır. Temel olarak ikinci derece bir yüzeydeki hata tahmini üzerine dayanır. Normal geri yayılım algoritmasına göre daha hızlı eğitmektedir.

Hızlı prop tekniği için adım büyüklüğü 0.1 ve momentum katsayısı da 0.5 ve eşlenik gradyan tekniği için ise adım büyüklüğü 0.1 olarak verilmiştir. Bu çalışmada, YSA'nın performansının değerlendirilmesinde Ortalama Karesel Hata (MSE), Normalize Edilmiş Ortalama Karesel Hata (NMSE) ve korelasyon katsayısı kullanılmaktadır.

MSE, istenen çıkışın, ağ çıkışına ne kadar iyi uyup uymadığına karar vermek için kullanılan bir değerdir. Özellikle MSE değeri 0.001'in altına inmelidir. NMSE ise ağ çıkışını normalize ederek istenen çıkışın arzulanan çıkışa uyup uymadığını kontrol etmek amacıyla kullanılan değerdir. Korelasyon katsayısı (r) yapay sinir ağı çıkışının iyi eğitilip eğitilmediğine karar vermek için kullanılan büyüklüktür. Korelasyon katsayısı istenen ağ çıkışı ile hedeflenen çıkış arasında farka bağlı olarak elde edilen bir katsayıdır. Korelasyon katsayısı -1 ile 1 arasında değişir. r=1 olması durumunda ağ çıkışı ile hedeflenen çıkış arasında mükemmel bir benzerlik olduğu kabul edilir. r = -1 olduğunda ağ çıkışı ile hedeflenen çıkış arasında ters yönde bir ilişki olduğu kabul edilir. r = 0 olduğunda ise ağ çıkışı ile hedeflenen çıkış arasında herhangi bir benzerlik bulunmamaktadır [12].

3.3 Bölütleme İşlemi ve Gizli Markov Modeli

Daha önce de belirtildiği gibi bu tezde, EMG sinyallerini karakterize etmek amacıyla, bölütleme yöntemi kullanılarak, EMG sinyalleri üzerinde düğüm noktaları elde edilmekte ve bu düğüm noktaları üzerinden GMM algoritması kullanılmaktadır. Bu kısımda bu iki yöntem ve bu iki yöntemin birleştirilmesi açıklanacaktır.

3.3.1 Bölütleme İşlemi

Bölütleme işlemi için, [11] ve [13] referanları baz alınmıştır. Bölütleme günümüzde birçok yapay zekâ destekli uygulamalarda kullanılmaktadır. EMG üzerinde bölütleme metodununun amacı, ölçülen sinyali anlamlı parçalara ayırmaktır.

Bölütleme algoritmasında, maksimum EMG genliğine göre ($max_i\{x_i\}$) (3.11) eşik değeri ve her bir EMG sinyalinin ortalama değeri,

$$\frac{1}{L}\sum_{i=1}^{L}|X_i| \tag{3.10}$$

ile hesaplanır. Burada x_i kesikli EMG değerleri ve L örnekleme sayısıdır. Eşik değeri (T) aşağıdaki algoritma ile hesaplanır:

Eğer;

$$max_i\{x_i\} > \frac{MD}{L}\sum_{i=1}^{L}|x_i| \tag{3.11}$$

ise;

$$T = \frac{TD}{L}\sum_{i=1}^{L}|x_i| \tag{3.12}$$

(MD: Maximuma göre ayarlanan değer, TD: Eşik değeri için ayarlanan değer)

formülü ile bulunur. Eğer değilse;

$$T = max_i\{x_i\}\,\frac{1}{TD} \tag{3.13}$$

formülü ile bulunur. Bu aşamadan sonra bulunan eşik değeri bir EMG sinyali genliği olarak kabul edilir ve bu genliğin altındaki genlikler yok edilir.

3.3.2 Gizli Markov Modeli

Seri olarak peş peşe gelen veriler, genel bir özellik olarak bütün bilim ve mühendislik dallarında kabul edilen gerçek dünya formlarıdır. GMM, gerçek süreçte esnek bir model davranışı sağlar. Matematiksel olarak sürekli veya kesikli süreçlere uyarlanabilir bir doğası vardır ve bu şekilde sinyal olarak karakterize edilebilir. Bu sinyaller farklı

özelliklere sahip olabilir örneğin sade veya bozulmuş, durağan veya durağan olmayan gibi. Kullanılan sinyallerin bu özellikleri birer model olarak alınabilir.

3.3.2.1 GMM'nin Kullanım Alanları

GMM'nin çok geniş bir kullanım alanı vardır, ancak genel olarak GMM'nin iki tane ana kullanım nedeni vardır. Tahmine dayalı sistemlerde GMM sıkça kullanılır (konuşma tanıma, medikal hastalık tanımlama ve finans alanında). Diğer alan ise gen tanıma, kelime tanıma, görsel ve işaret tanımlama gibi alanlarda sıkça kullanılır.

GMM aynı zamanda, gerçek dünya problemlerine dinamik tanımlama yapısı sunar. GMM gözlenmiş ve bilinen dağılımlardan yola çıkarak, bu dağılımlar ile bilinmeyen dağılımlar ilişkilendirilerek yeni bir dağılım oluşturur. Genellikle fiziksel olayların durumlar ile ilişkilendirilebildiğinde, GMM doğal sürecin modellenmesine imkan tanır.

Diğer bir önemli kullanım alanı olarak, sistemin çözüm ve modelleme potansiyeli tanımlanan yapı içerisindedir; öyle ki sistemin ana kaynaklarına ulaşılma zorunluluğu yoktur. Dolayısıyla verilen model simüle edilebilir ve kaynak hakkında daha çok bilgi edinmeye devam edilebilir ve sürecin çıktıları simüle edilebilir. Son olarak bahsedilen bu özellik, bilhassa çok önemlidir ve sinyal işleme problemlerinde sistemin daha verimli modellenmesine imkân sağlamaktadır.

3.3.2.2 GMM'nin Elemanları

GMM, λ ile ifade edilen çift istatistiksel sürece sahiptir. GMM gözlenen ve gözlemlenemeyen süreçler arasındaki ilişkiyi açıklar. t zamanında gözlenmiş çıktılar olarak Y_t tanımlansın ve t zamanındaki gizli durumlar Z_t olarak tanımlansın. GMM'nin karakteristiği aşağıdaki 5 elemana bağlıdır:

1. N model üzerindeki gizli durumların sayılarını tanımlasın. Özgün durumların olası ara bağlantıları $S = \{S_1, \dots, S_N\}$ olarak tanımlansın, öyle ki herhangi bir durum diğer durumlara ulaşabilir olsun. q_t 'de t zamanındaki durum olsun.
2. M her bir duruma ait belirgin gözlemlerin sembollerinin veya başlangıçta yapılan modellemenin cevap sürecine uyuşan sembollerin sayısı olarak tanımlansın. $V = \{v_1,....,v_m\}$ ise gözlem sembollerinin kümesi olsun.

3. A durum geçiş olasılıkları dağılımı (durum geçiş olasılıkları kümesi), aynı zamanda durum geçiş matrisi A = $\{a_{ij}\}$ olarak tanımlanır ve S_i'den S_j'ye geçiş olasılıklarını temsin eder:

$$a_{ij} = P\left[\, q_{t+1} = S_j \mid q_t = S_i\right] \qquad 1 \le i, j \le N \tag{3.14}$$

Burada q_t anlık durumu temsil eder. Geçiş olasılıklarının normal istatistiksel içeriği, aşağıdaki kısıtlamaları karşılamalıdır:

$$a_{ij} > 0, \qquad 1 \le i, j \le N \tag{3.15}$$

ve

$$\sum_{i=1}^{N} a_{ij} = 1, \quad 1 \le i, j \le N \tag{3.16}$$

4. Sembolleri gözlemleme olasılık dağılımı, aynı zamanda emisyon matrisi B=$\{b_j(k)\}$ olarak çağrılır ve sistem S_j durumundayken v_k sembolünün olasılıklarını taşır.

$$b_j(k) = P\{o_t = v_k \mid q_t = S_j\}, \qquad 1 \le j \le N\,, 1 \le j \le M \tag{3.17}$$

Burada v_k k. gözlemlenen sembol ve o_t o anki gözlem parametre vektörü. Aşağıdaki istatistiksel kısıtlama tanımlanmak zorundadır;

$$b_j(k) \ge 0, 1 \le j \le N\,, 1 \le j \tag{3.18}$$

ve

$$\sum^{M} b_j(k) = 1, 1 \le j \le N \tag{3.19}$$

5. Giriş durumu olasılık dağılımı $\pi = \{ \pi_i \}$, giriş durumları olasılık dağılımları olarak temsil edilir.

$$\pi_i = P[q_t = S_i] \;\; 1 \le j \le N, \qquad \pi_i \ge 0, \qquad \sum^{N} \pi_i = 1 \tag{3.20}$$

Dolayısıyla, geleneksel gösterim kullanılarak, GMM'nin 5 elemanı vardır. Ancak, N ve M sabit farzedilmeli ve GMM üç parametreli olarak $\lambda = (A, B, \pi)$ şeklinde ifade edilebilir.

3.3.2.3 GMM'nin Uygulanışı ve Tanımı

GMM'yi detaylı olarak anlatmadan önce, GMM'nini uygulanışını anlatmak şarttır, yani Markov süreci. Eğer o anki eylemin koşullu olasılığının yoğunluğu, verilen tüm geçmiş ve şimdiki durumlarda, sadece j şimdiki durumuna bağlıysa, bu istatistiksel süreç j. dereceden Markov süreci olarak adlandırılır.

Daha anlamlı bir ifade olarak, GMM iki istatistiksel süreç olarak tanımlanabilir. GMM birinci derecesini ve temelini teşkil eden bir istatistiksel süreci içerir. Bu istatiksel süreç Markov modelindeki olası her bir durumdaki durum geçişlerini modeller ve S_j'den S_i'ye geçiş olasılığı $P(q_{t+1} = S_j \mid q_t = S_i)$ olur, burada S_j gelecek t+1 zamanındaki durum ve S_i şu anki durumdur. GMM'nin ikinci istatistiksel katmanı, her bir durumun emisyon olasılıkları kümesinin bulunmasıdır. Örneğin, S_i durumundaki emisyon olasılıkları, gözlemlenen gerçek gözlemleri belirtir, yani S_i durumunda verilen gerçek GMM'yi verir. Olasılıkların bu ikinci katmanı sistemin görüntüsünü yaratır, yani gözlemlerin verilen dizisi ve durumların gerçek dizisi belirsizdir; gözlemciden "gizli"dir.

Daha önce tanımlanan notasyon tekrar ele alınılırsa, hem gözlemlerin üreticisi hem de gözlem dizisinin nasıl yaratıldığını gösteren model, GMM'de verilen bu beş maddeden üretilebilir. Bu prosedür, ilk defa Rabiner tarafından 1989 yılında aşağıdaki gibi verilmiştir:

1. Giriş dağılımları π'e göre, giriş durumları $q_1 = S_i$'yi seçilir.
2. t = 1 olarak ayarlanır.
3. S_i durumundaki sembol olasılık dağılımına göre $O_t = v_k$ olarak seçilir.
4. S_i durumu için, durum geçiş olasılık dağılımına göre yeni duruma geçiş $q_{t+1} = S_j$'i ayarlar.
5. Eğer $t < T$ ise üçüncü adımı döndürülür, eğer değil ise prosedür iptal edilir.

Özetle, GMM durumların sonlu bir kümesidir, burada durumların her biri olasılıksal duruma uygundur (genelde çok boyutlu). Durumlar arasında geçiş olasılıkları tarafından yönetilir ve genelde geçiş olasılıklarını işaret eder. Özellikle sonuç veya gözlem durumu

birleşmiş olasılık dağılımına göre üretilebilir. Bu sadece sürecin çıktısı ve gizli veya gözlemlenmemiş durumlar üzerinden gözlemlenen çıkış süreci olabilir.

3.3.2.4 GMM Teorisindeki Varsayımlar

GMM'nin matematiksel olarak işlenebilir formu teorik olarak desteklenebilir. Bu kısımda GMM'ye ait üç temel yaklaşım özetlenecektir. Gerçekte GMM bu üç çeşitle sınırlandırılamaz ve diğer çeşitler bu üç yaklaşımdan yola çıkılarak bulunabilir.

Muhtemelen en aşikar yaklaşım Markov yaklaşımıdır. GMM'nin tanımında da değinildiği üzere,

$$a_{ij} = P\left[\, q_{t+1} = S_j \,\middle|\, q_t = S_i\right] \qquad 1 \le i, j \le N \tag{3.21}$$

Burada bir sonraki durumun sadece o andaki duruma bağlı olduğu farzedilir. Bu Markov yaklaşımı olarak adlandırılır ve GMM'nin birinci katmanının cevabıdır. Ancak, genellikle bir sonraki durum k tane önceki duruma bağlı olabilir ve model olarak elde edilebilme ihtimali vardır, k. dereceden işaret edilen GMM'nin tanımlanması aşağıdaki gibidir:

$$a_{i1j2,\ldots,ikj} = P\left[\, q_{t+1} = S_j \,\middle|\, q_t = S_{i1}, q_{t-1} = S_{i2}, \ldots, q_{t-k} = S_{ik}\right], 1 \le i1, \ldots, ik \le N$$

(3.22)

Birinci dereceden GMM çok genel olmasına rağmen, bazı durumlarda daha fazla kompleksliği beraberinden getirmesinden dolayı; yüksek dereceden GMM kullanılır.

Teorideki bir diğer GMM yaklaşımı ise durağan GMMS (GMMS: Gizli Markov Model Sabit, HMMS: Hidden Markov Model Stadionary) yapısıdır. $\{q_t\}$ Markov geçiş olasılıkları a_{ij} ile birlikte Markov zinciridir ve giriş olasılıkları π_i ve $\{q_t\}$ durağan olarak farzedilir. Geçiş olasılıkları meydana geldikleri gerçek zamanda durum geçiş olasılıkları birbirinden bağımsız farzedilir. Bu durum matematiksel olarak aşağıdaki gibi ifade edilir:

$$P\left[\, q_{t1+1} = S_j \,\middle|\, q_{t1} = S_{i1}\right] = P\left[\, q_{t2+1} = S_j \,\middle|\, q_{t2} = S_i\right] \tag{3.23}$$

Her hangi bir t_1 ve t_2 için.

Bağımsız yaklaşım çıktısı, GMM'de belirgin olan, daha önceki çıktılardan (gözlemlerden) istatiksel olarak bağımsız olan, şimdiki durum (gözlem) çıktısı yaklaşımıdır. Bu durum gözlem dizileri tarafından aşağıdaki gibi formüle edilebilir:

$$O = o_1, o_2, \dots, o_T \tag{3.24}$$

Bundan sonra GMM'nin üçüncü parametresi,

$$P\{O|q_1, q_2, \dots, q_T, \lambda\} = \prod_{t=1}^{T} P(O_t|q_t, \lambda) \tag{3.25}$$

ile bulunur. Bu yaklaşım genelde GMM'nin zayıflığı olarak belirtilmiştir.

Kesikli zamanlı GMM'de (daha önce tartışıldığı gibi), birkaç tane ek yaklaşım daha vardır. Bunlardan bazıları sonlu N'yi içerir ve zaman noktaları t = 1, ..., n eşit aralıklarla yerleştirilmiştir. Düzenli ve eşit bir şekilde zaman noktalarını parçaya ayırmak GMM'nin kullanımını basitleştirir ancak şart değildir.

3.3.2.5 GMM Çeşitleri

Temel olarak üç çeşit GMM vardır, bunlar arasındaki farklılık çıkış olasılıkları modellemelerindeki metot farklılığıdır.

3.3.2.5.1 Sol – Sağ Modeli

GMM'yi daha önce bir ergodik model olarak tanımlamıştır. Bunun anlamı, sınırlı bir adım sayısı içinde modelin her durumu diğer, her durumda olan bir özelliğe sahip olabilir. Tüm ergodik modellerde her bir durum diğer durumlar ile bağlantılı olabilir. Bu özellik ergodik modelin çok önemli bir özelliğidir; örneğin a_{ij}'ler, basit olarak pozitif durum geçiş olasılıkları dağılımının elamanlarıdır. Ancak, GMM çeşitlerinde, a_{ij}'ler daha fazla hesap kullanılarak hesaplanır. Bu yönüyle GMM çeşitleri, çalışma alanları için cazip olmuştur.

Bu çeşitlerden en ünlüsü sol sağ modelidir (aynı zamanda Bakis modeli olarak da gösterilir) (Rabiner, 1989). Benzer bir anlam olarak, bu modelde kullanılan yöntem durum süreçlerini soldan sağa doğru uygulamaktır. Dolayısıyla, zaman arttıkça durum indeksi de artan model ile durum dizileri birleşmiş olur. Zamanla değişme özelliğine sahip olan bu süreçler, kolaylıkla GMM'de tanımlanabilir.

Ergodik formun GMM'inde, her a_{ij} katsayısı pozitif; ancak, GMM'nin soldan sağa çeşidinde, durum geçiş katsayıları aşağıdaki özelliğe sahiptir:

$$a_{ij} = 0, \qquad j < i \tag{3.26}$$

Bu içeriktekiler, şimdiki durum indeksinden daha küçük indekse sahip olan hiçbir geçiş için izinli değildir. Nispeten, ergodik model için kullanılan durum geçiş matrisi aşağındaki gibidir.

$$A_{ergodik} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} \\ a_{21} & a_{22} & \cdots & a_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{j1} & a_{j2} & \cdots & a_{ij} \end{bmatrix} \tag{3.27}$$

sol – sağ model için durum geçiş matrisi aşağıdaki formdadır:

$$A_{ergodik} = \begin{bmatrix} a_{11} & a_{12} & \cdots & 0 \\ 0 & a_{22} & \cdots & a_{2j} \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & a_{ij} \end{bmatrix} \tag{3.28}$$

Sol – sağ modelde, durum geçiş matrisi katsayılarının tanımlanması genel olarak aşağıdaki gibidir:

$$a_{NN} = 1 \tag{3.29}$$

$$a_{Ni} = 0\,,\ i < N \tag{3.30}$$

Durumlar arasında büyük sıçramaları önlemek için genellikle aşağıdaki sınırlama uygulanır:

$$a_{ij} = 0\,,\ j < i + \Delta \tag{3.31}$$

Örneğin, eğer zıplama miktarı ikiden fazla olamayacağı biliniyorsa, Δ olarak ayarlanabilir. Sol-sağ model, model ile birleşmiş durum dizileri ile ilgili istenen tüm özellikleri tutar. Durum dizileri, birinci durumdan itibaren ve N sağa doğru ilerletilirse, giriş durumları olasılığı aşağıdaki form şeklide olur:

$$\pi_i = \begin{cases} 0, & i \neq 0 \\ 1, & i = 0 \end{cases} \tag{3.32}$$

Sol-sağ modelin basitliği ve dinamik yapısı büyük bir avantajıdır. Bu avantajı sayesinde birçok alanda ve çalışmada kendine yer bulmuştur.

3.3.2.5.2 Sürekli ve Yarı Sürekli GMM

Birçok uygulamada, gözlemler sürekli bir süreç veya vektördür. Bu noktada, tartışmalar kesikli gözlemler üzerinde yoğunlaşmıştır ve bundan dolayı sınırlı kesikli semboller karakterize edilmiştir. Benzer bir davranış olarak, kod listesi kullanarak sürekli zaman cevabı parçalara ayırılabilir. Bundan dolayı, uygulanabilirlik imkanı olan sürekli GMM'nin makul ve uygulanabilir bir formülasyonunu üretmek oldukça önemlidir.

Kesikli GMM'den farklı olarak, sürekli GMM'nin her bir durumu bir olasılık dağılım fonksiyonunu bağlıdır ve böylece gözlemlenebilir çok katmanlı ve sürekli verinin olasılığı hesaplanabilir. Kesikli olasılık yoğunluk fonksiyonundan faklı olarak, sürekli olasılık yoğunluk fonksiyonu için, gözlemler devamı olduğu sürece, parametreler sürekli olarak hesaplanmak zorundadır. Bu model aşağıdaki gibi formülize edilir:

$$b_j(O) = \sum_{m=1}^{M} c_{jm} \aleph(O, \mu_{jm}, \Sigma_{jm}) \tag{3.33}$$

Burada:

c_{jm} = j durumundaki m. birleşim için birleşim katsayısı

μ_{jm} = ortalama vektörü

Σ_{jm} = j durumundaki m. birleşim elmanı için koveryans matrisi

$\aleph$ sadece gauss dağılımıyla sınırlandırılmamış olmasına rağmen, genellikle bu şekilde kullanılır. Ancak, logaritmik iç bükey veya eliptik simetrik olmalıdır (Rabiner, 1989). Daha da ötesi, birleşim katsayıları aşağıdaki istatistiksel sınırlandırılmaya uyumlu olmalıdır:

$$c_{jm} \geq 0, \ 1 \leq j \leq N, \ 1 \leq m \leq M \tag{3.34}$$

ve

$$\sum_{m=1}^{M} c_{jm} = 1, \qquad 1 \leq j \leq N \tag{3.33}$$

Dolayısıyla, sürekli zamanlı yoğunluklu GMM, $\lambda(A, c_{jm}, \mu_{jm}, \Sigma_{jm}, \pi)$'dir.

Sürekli GMM geliştirilerek, yarı sürekli GMM elde edilir. Yarı sürekli GMM'de, gözlem vektörü sınıfları, sınırlı seri içerisinde serbest parametre sayısına indirgenerek parçalanır. Ancak sürekli GMM olduğu gibi, gözlem sınıfları gauss olasılık yoğunluk fonksiyonu gibi modellenir, parçalamadan doğan tahribat ve gözlem sınıfının modellenmiş varyansının etkisi yok edilir. Aynı olasılık yoğunluk fonksiyonunu paylaşmaya zorlanmış durumların parametrelerinin bağlanmasıyla, sürekli zaman GMM'deki benzer formülasyon kullanılır. Sınıflandırmayla giriş biçimlendirilmesi yapılan örnek veriler, sınıflar içindeki GMM parametreleriyle kalıcı bir form elde edilebilmesi için, yeniden tahmin edilir.

Parametre bağlama konsepti farklı durumlardaki GMM parametreleri arasındaki ilişkiye benzer bir yapı üzerine kurulmuştur. Model içerisindeki bağımsız parametre sayısı indirgenir ve parametre tahmini kolaylaşır (Rabiner, 1989). Yoğunluğun iki ya da daha fazla olduğu uygulamalarda kullanılan her GMM benzer şekilde nitelendirilir. GMM ilginç özelliklerinden biri de, hiçbir çıkış üretmeyen geçişler ile birlikte gözlemleri ilişkilendirebilmesidir. Böylece, bir durum diğer duruma, gözlenmiş çıktı olmaksızın ilerletilebilir.

3.3.2.5.3 Özbağılımlı (AR: Autoregressive) GMM

Çıkış dağılımları üzerinde kurulmuş GMM'nin temel iki formu, kesikli veya sürekli GMM ve bunların birleşimi olan GMM'dir. Diğer bir GMM çeşidi, gerçek hayat süreçleri üzerinden belirlenmiş olan öz bağımlı GMM'dir. Tüm modellerde, gözlem vektörleri özbağılımlı süreçten gelmiştir. Basit olarak AR model, gözlem serilerine ait önceki değerlerin, lineer fonksiyonu olarak kabul edildiği yerde zaman serisi analizidir (Everitt, 2002). Birinci dereceden AR model aşağıdaki gibidir:

$$x_t = ax_{t-1} + e_t \tag{3.36}$$

burada x_t t. zamandaki gözlemdir, a model parametresi ve e_t modelin rastgele bozucu etkisidir. p. dereceden model aşağıdaki formadadır:

$$x_t = a_1x_{t-1} + a_2x_{t-2} + \cdots + a_px_{t-p} + e_t \tag{3.37}$$

burada a'nın p parametrelerini içermektedir.

Gözlem vektörü O'yu x_0, x_1, …, x_{k-1} elemanlarıyla göz önüne alalım. Gauss dağılımı, gözlem vektörüne p. dereceden gauss özbağılımlıdır ve gözlem vektörü elemanlarının arasında aşağıdaki ilişki vardır:

$$O_k = -\sum_{i=1}^{p} a_i O_{k-i} + e_k \tag{3.38}$$

burada e_k, k = 0,1,2,…, K -1 sıfır ortalamalı gauss bağımsız rastgele dağılımı ve varyansı σ^2 ve a_i özbağılımlı dağılım katsayılarıdır. O için olasılık yoğunluk fonksiyonu aşağıdaki gibidir:

$$f(O) = (2\pi\sigma^2)^{-K/2} exp\left\{-\frac{1}{2\sigma^2}\,\delta(O,a)\right\} \tag{3.39}$$

burada;

$$\delta(O,a) = r_a(0)r(0) + 2\sum_{i=1}^{p} r_a(i)r(i) \tag{3.40}$$

$$a' = [1, a_1, a_2, \dots, a_p] \tag{3.41}$$

Zaman serisi içerisindeki seri yapıya bağımlılığın önemi, otokorelasyon fonksiyonu için de öne çıkmaktadır. r(i)'nin gözlemlenmiş örneklerinin otokolerasyon fonksiyonu ve r_a(i)'in özbağımlılık katsayısı olduğunu göz önünde bulundurarak, otokolerasyon fonksiyonu aşağıdaki formda oluşur:

$$r(i) = \sum_{n=0}^{K-i-1} X_n\, X_{n+i} \qquad 0 \le i \le p \tag{3.42}$$

$$r_a(i) = \sum_{n=0}^{K-i-1} a_n\, a_{n+i} \qquad (a_0 = 1),\ 0 \le i \le p \tag{3.43}$$

Gauss özbağımlı GMM tanımlamak için, emisyon matrisi elemanları birleşik yoğunluklu olarak farz edilir:

$$b_j(O) = \sum_{m=1}^{M} c_{jm} b_{jm}(O) \tag{3.44}$$

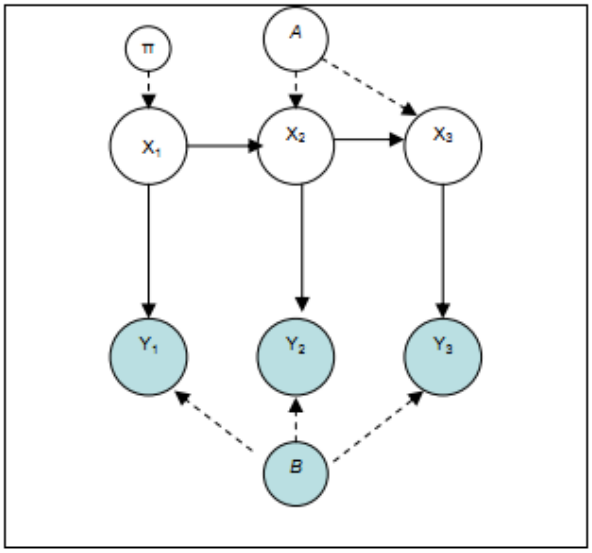
burada b_{jm}(O) özbağımlılık vektörü $\mathbf{a_{jm}}$ birlikte yoğunluğu temsil eder:

$$b_{jm}(O) = (\frac{2\pi}{K})^{-\frac{K}{2}} exp\left\{-\frac{K}{2}\,\delta(O, a_{jm})\right\} \quad (3.45)$$

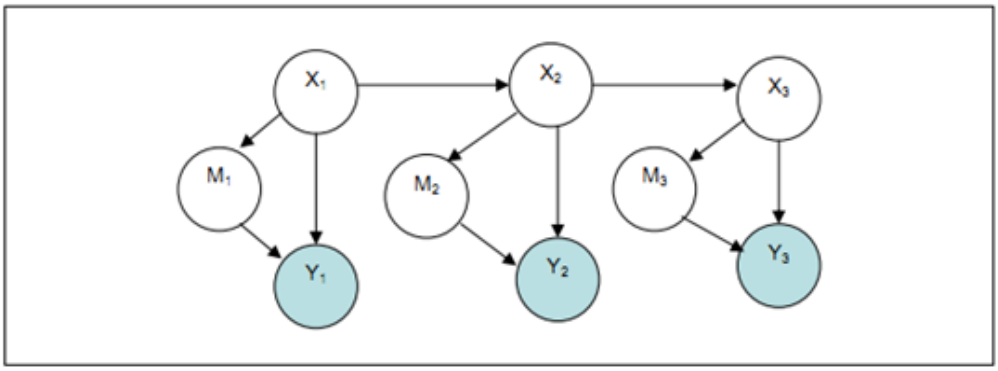
GMM'deki standart koşullu yaklaşım oldukça güçlü bir yapıdır ve özbağımlılık algoritmasının kullanılabilirliğini kolaylaştırmaktadır. Önceki gözlemlerin, şu anki durumun önceden gözlenmesine yardım etmesini sağlayarak, bozucu noktaların etkisini azaltır. Modeldeki sonuçlar gerçek sonuçlara oldukça yaklaşıktır. Genelleştirilmiş özbağımlı GMM algoritması aynı zamanda, gözlemlerin cevapları arasındaki lineer olmayan bağımlılıkları da içermektedir. Bu model genelde ses tanımada, finans ve diğer birçok mühendislik alanında kullanılır [14], [15], [16], [17].

3.3.2.6 GMM'nin Temsili

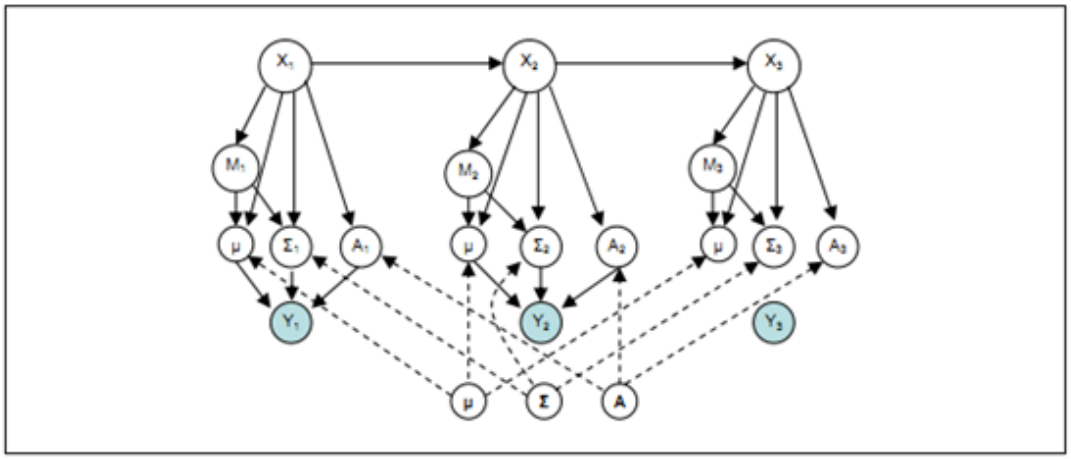
Birçok GMM uygulamasında, rastgele sürecin dinamik yapısını anlatmak şarttır. GMM'nin dinamik bayes ağı (DBN: Dinamik Bayesian Network) olarak temsili orijinal metotlardan biridir. GMM'ye ait dinamik yapıların bazı temsilleri Şekil 3.4, Şekil 3.5, Şekil 3.6 ve Şekil 3.7'de gösterilmiştir:



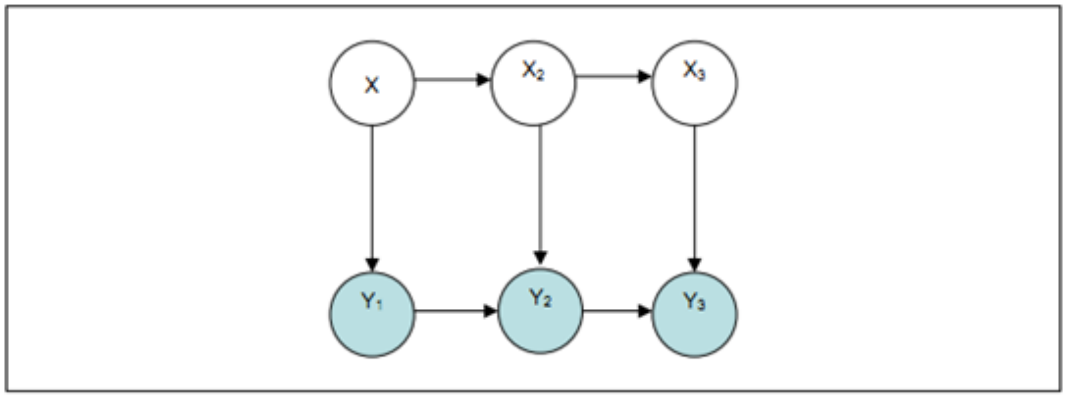
Şekil 3.4 GMM'de parametrelerin yayılımının belirgin şekilde gösterimi [14]



Şekil 3.5 Gauss dağılımına sahip GMM gösterimi [14]



Şekil 3.6 Yarı sürekli GMM dağılımının gösterimi [14]



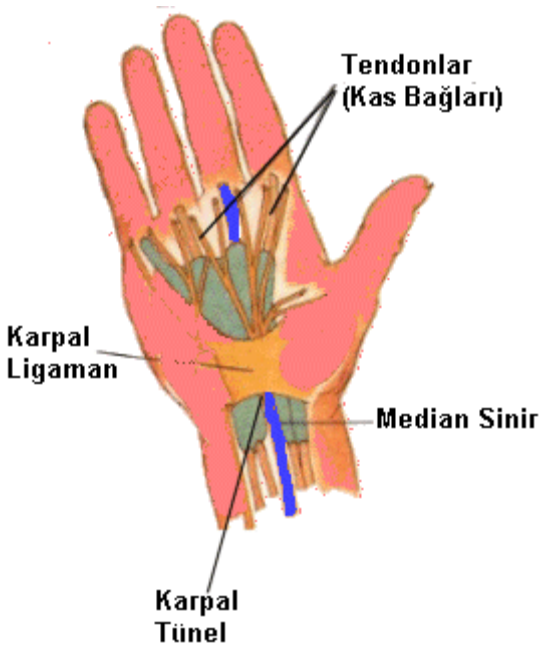
Şekil 3.7 Özbağılımlı GMM gösterimi [14]

BÖLÜM 4

GİZLİ MARKOV MODEL İLE DESTEKLENEN EMG ÖLÇÜM ve SINFLANDIRMA SİSTEMİ

4. 1 Üzerinde Çalışılan Kas Grubu

Tüm parmakların kökünde yerleşmiş Tenar kasları uygulama için seçilen kas gruplarıdır. Bu kasların işleyişini anlamak için, ön kol ve el kaslarının işleyişini bilmek gerekir. Şekil 4.1'de el kaslarının yapısı görülmektedir:



Şekil 4.1 El kaslarının yapısı

Önkol Kasları: Dirsek eklemi (art. cubiti) üzerine etki eden ve kol üzerinde lokalize olan üç adet bükücü (flexor) ve bir tane de açıcı (extensor) kas bulunur. Bunun yanında dirsek ekleminde, flexion ve extention hareketlerinin yanı sıra, önkolun dışarıya doğru dönmesi (supinasyon) ve içe dönmesi (pronasyon) hareketleri de ortaya çıkar. Elin supinasyon ve pronasyon hareketlerinde ise m.supinator ve m.pronator teres ile bir miktar da m.pronator quadratus etkili olur.

Önkolda ele ve bileğe doğru uzanan çok sayıda kas vardır. Önkol kasları, el ve parmak hareketlerini sağlayan kaslardır.

El Kasları: Elin hareketleri ile ilgili kaslar fonksiyonlarına göre, flexor ve extensor kaslar olmak üzere ayrılırlar. El hareketleri ile ilgili kasların hepsi uzun, ince ve kuvvetli kirişlere sahiptirler. Bu ince kirişler el bileğinden geçip ilgili yerlere uzanırlarken, özel kiriş yapılı (aponevrotik) kılıflar içinde bulunurlar. Elin flexionu ile ilgili kasların başlangıç yerleri, humerus'un distal ucunda görülen içteki çıkıntı (apicondylus medialis) üzerindedir.

Ekstensor kaslar ise dıştaki büyük çıkıntıdan (epicondylus lateralis) başlar. Bu antegonist kasların çalışmaları ile elin palmar veya dorsal fleksiyonu yapılabilir. Ancak, radius tarafındaki fleksor ve ekstensor kasların aynı anda çalışmaları ile elin radius yönünde olmak üzere (dışa doğru) uzaklaştırılması mümkün olur. Ulnar tarafındakiler ise, içe doğru hareketi meydana getirirler (Ulnar abduction).

El parmaklarının hareketleri, uzun ve kısa kaslar tarafından meydana getirilir. Kirişleri parmaklara kadar uzanan uzun kasların grup olarak bulundukları yer esas itibariyle önkoldur.

Burada çeşitli kemik ve kemikler arası sağlam aponevrotik yapılara tutunarak başlayıp, aşağılarda ortaya çıkan ince sonuç kirişleri, el bileğinden geçerek ilgili parmaklara kadar uzanırlar. Kısa kaslar ise avuç içinde ve küçük kemikler arasında lokalize olmuşlardır.

Parmaklar arasında yer alan dört adet kısa kas ile (Mm. lumbricales), kemikler arasında lokalize olmuş yedi adet kas (Mm. interossei) el parmaklarının kısa kaslarını meydana getirirler.

Bunlara ek olarak, insan elinin baş parmağının özel hareketlerini yapabilmesi için, sadece bu parmağa ait olan bir kas daha vardır (m. opponens pollicis). Eldeki, kombine

ve komplike hareketlerin ortaya çıkışı göz önüne alındığında, diğer parmaklardan birisinin kaybında fonksiyonel olarak büyük boyutlarda bir aksama görülmemesine rağmen, başparmağın kaybında durum değişir. Zira bu durumda, elin kullanılma kabiliyeti büyük ölçülerde kaybedilmiş olur.

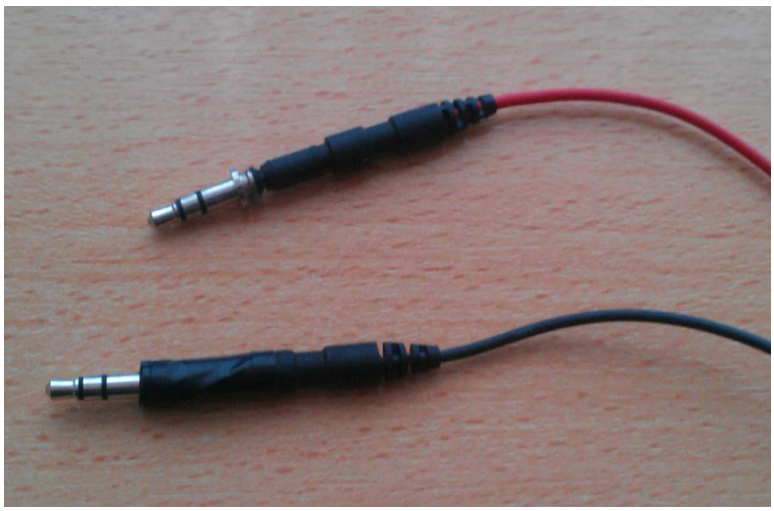
4. 2 Yüzey Elektrotları ve Yüzük Elektrot

EMG sinyallerinin yükseltici devreye bağlantısı yüzey elektrot çeşitlerinden, yüzük elektrotları ile sağlanmıştır. Genel amaçlı birçok EMG uygulamasında güvenirlilik açısından, iğne elektrotları kullanılmaktadır. İğne elektrotları ile kasa doğrudan ulaşılabilir ve gerçeğe en yakın sonuçlar elde edilebilir. Ancak iğne elektrotlarının kullanımı oldukça ciddi bir iştir. Son derece hijyen gerektiren ve can acıtıcı bir işlemdir. Aynı zamanda bu tür sosyal hayatla doğrudan ilişkili uygulamalarda, uygulanabilirliği çok azdır. Bu yüzden bu çalışmada, EMG ölçümü için yüzey elektrotları seçilmiştir.

Yüzey elektrotları ile yapılan ölçümlerde çok geniş bir alandaki elektriksel aktivite ile ilgili bilgi elde edilebilir. Özel olarak bir motor ünitesinin veya üniteler grubunun incelenmesinde, elektrotların bilgi topladıkları alan çok geniş olabilir. Ayrıca, yüzeydeki kasların faaliyeti alttan gelen bilgiyi maskelediğinden, yüzey elektrotlar sadece yüzeydeki kasların incelenmesinde kullanılabilir. Şekil 4.2'de yüzük elektrotları görülmektedir.

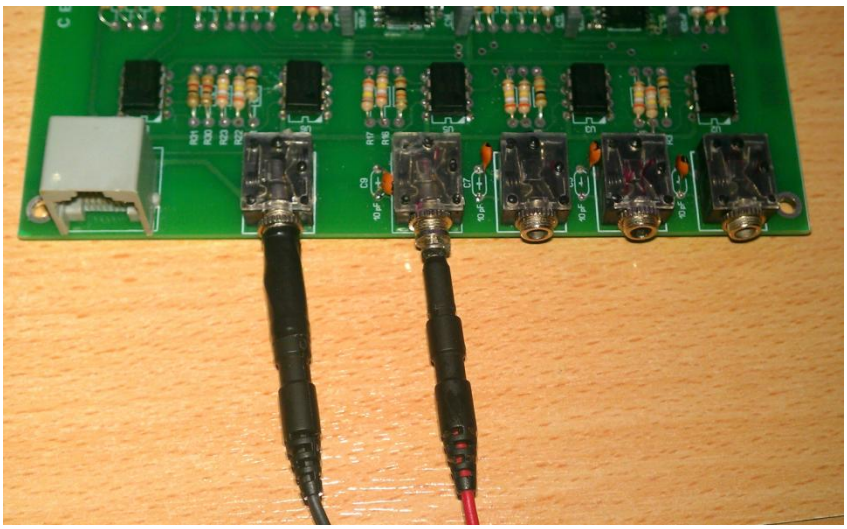
Şekil 4.2 Yüzük elektrot

Yüzük elektrotları yüzük şekli verilmiş bobinlerden ve bu bobinlerin bağlı olduğu kablolardan oluşur. Yüzük elektrotlarının sabitlenmesi için sabitleyici krem ve elektriksel iletkenliğin artması için iletken jel ile kullanılır. Yüzük elektrotlarının yükseltici devre ile iletişimi, kulaklıklarda kullanılan konnektör ile sağlanmıştır. Şekil 4.3'de elektrotların kulaklık girişi ile bütünleştirilmesi, Şekil 4.4 elektrotların parmaklara yerleştirilmesi ve Şekil 4.5'de elektrotların yükseltici devre ile birleştirilmesi görülmektedir:



Şekil 4.3 Kulaklık girişleriyle birleştirilmiş yüzük elektrotlar

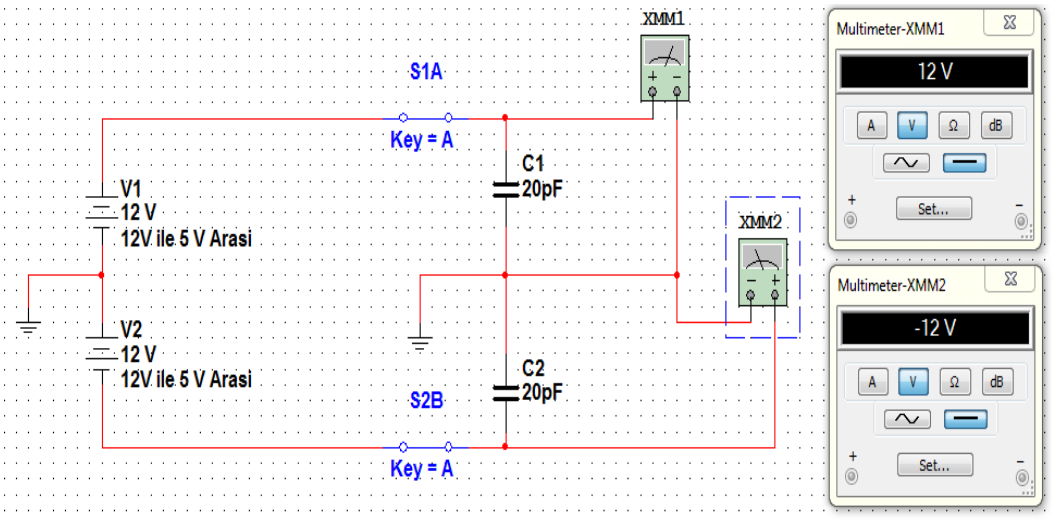
Şekil 4.4 Yüzük elektrotlarının kullanımı



Şekil 4.5 Yüzük elektrotların EMG sinyallerini yükseltici devre ile bağlantısı

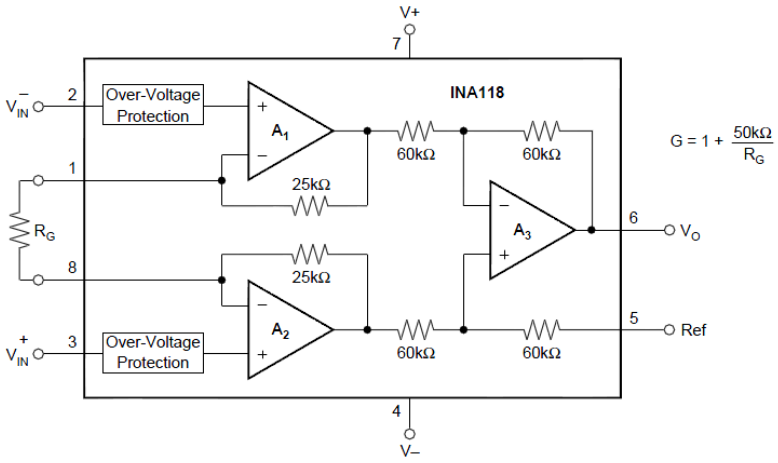
4. 3 EMG Yükseltici Devresi

Bu çalışmada amaç, EMG sinyallerini yükseltme ve değerlendirme işlemlerini yapan, hızlı ve akıllı bir sistem geliştirmekti. Bu sistemi gerçekleştirmek amacıyla, EMG sinyallerini daha anlamlı hale getirecek EMG yükseltici devresi yapılmıştır. Devrede güç kaynağı olarak, 2 adet adaptör kullanılarak, kesintisiz ve düzenli güç elde edilmiştir. Güç kaynaklarından biri pozitif gerilim referansı, diğeri ise negatif gerilim referansı olarak kullanılmıştır. Bu adaptörlerin her ikisi de gerilimleri ayarlanabilir olarak seçilmiştir. Güç kaynaklarındaki olası bir dalgalanmayı regüle etmek için, güç kaynaklarının çıkışları ile ortak toprak arasında 20pF'lık kapasitör eklenmiştir.



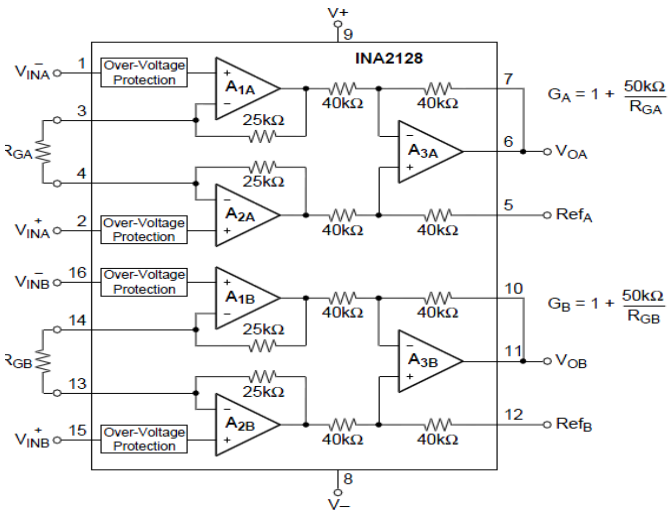
Şekil 4.6 EMG yükseltici devresinin +/- referansları

EMG sinyallerinin yükseltilmesi için iki basamaklı yükseltici devre tasarlanmıştır. EMG yükseltici devre, endüstriyel ve operasyonel olmak üzere iki çeşit kuvvetlendiriciye sahiptir. EMG yükseltici devresinin ilk tasarımında, enstrümantasyon kuvvetlendirici olarak INA118 kullanılmıştır. Bunun nedeni INA118'in medikal uygulamalarda sıkça kullanılması ve başarılı bir kuvvetlendirici olmasıdır. INA118'in içyapısı aşağıdaki gibidir:



Şekil 4.7 INA118'in iç yapısı [19]

INA118 ile yapılan devre tasarımından sonra, montajı açısından daha kolay olan ve benzer özelliklere sahip olan INA2128 kullanılmıştır. INA2128 için aynı kalıpta çift INA118 denilebilir. INA2128'in içyapısı aşağıdaki gibidir:

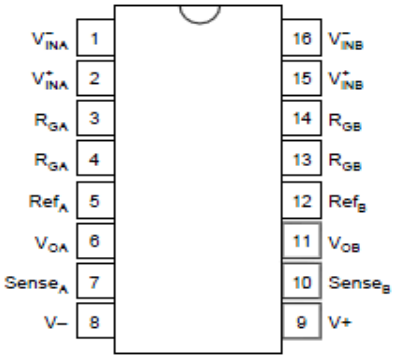


Şekil 4.8 INA2128'in iç yapısı [20]

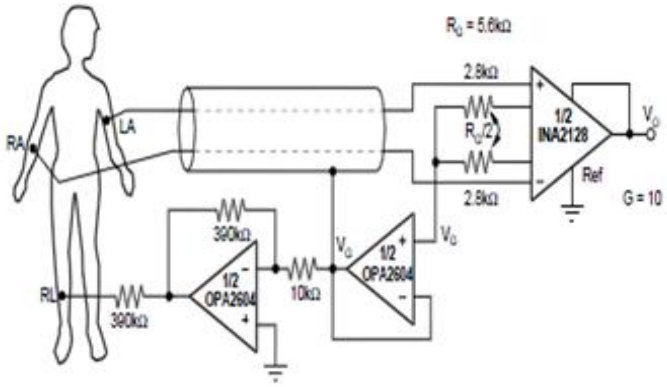
INA2128 düşük güç gereksinimi olan, genel amaçlı ve yüksek başarıya sahip bir enstrümantasyon kuvvetlendiricisidir. Çift giriş ve çift çıkıştan oluşan INA2128, üç tane opamp çıkışı ile tasarlanmıştır ve küçük boyutları ile çok geniş bir uygulama alanına

sahiptir. Akım-geri besleme devresi ile çok geniş bir bant aralığında yüksek kazanç vermektedir (200 kHz'de G = 100 (G: Gain - Kazanç)). Basit bir harici direnç uygulamasıyla, kazanç 1 ile 10000 arasında değiştirilebilir. INA2128 aynı zamanda, çipi kaybetme gerilimi için, ±40V giriş korumasına sahiptir.

INA2128 çok düşük offset gerilimlerinde (50uV) tetiklenebilmektedir ve ortak mod reddetmede oldukça başarılıdır (G ≥ 100'de 120dB). ±2.5V gibi düşük gerilimlerde ve 700uA gibi düşük akımlarda çalışabilmektedir. INA2128'in pin konfigürasyonu ve referans devresi aşağıdaki gibidir:

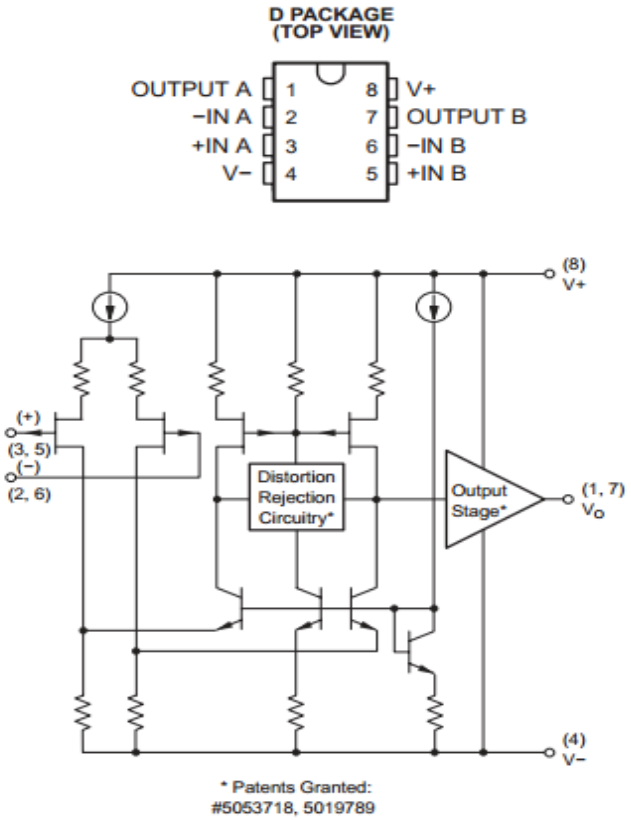


Şekil 4.9 INA2128'in pin konfigürasyonu [20]



Şekil 4.10 INA2128 referans devresi [20]

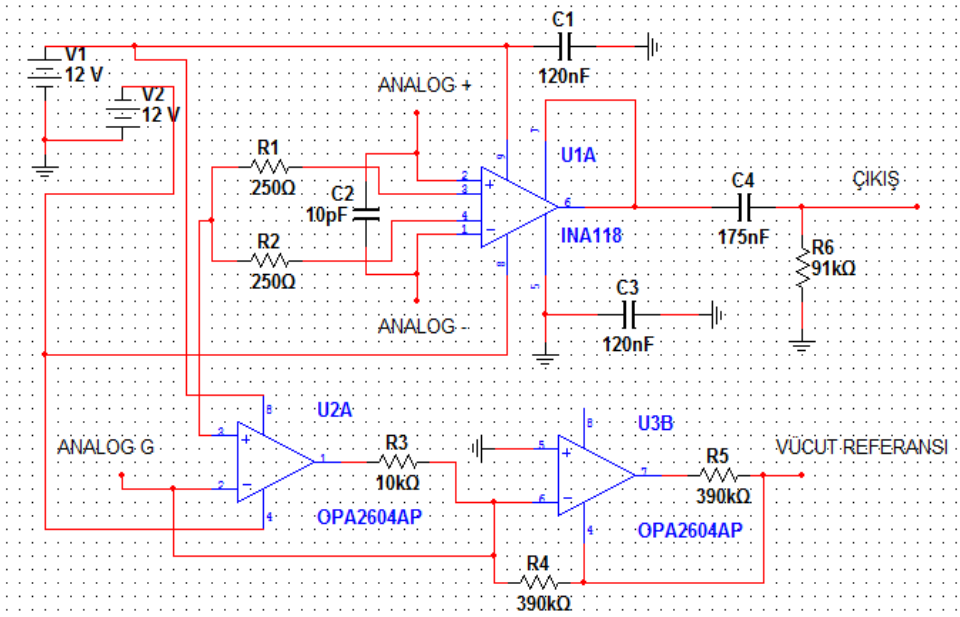
Tezde yapılan tasarımda ve birçok tasarımda olduğu gibi, enstrümantasyon kuvvetlendiricisinin çıkışında, vücuttan ayrı bir noktadan referans alınara ortak gürültüyü reddetme devresi yapılmıştır. Tasarımda bu amaçla referans devre şemasında olduğu gibi OPA2604 operasyonel yükseltici kullanılmıştır. OPA2604 birçok eski tasarımda olduğu gibi, günümüz tasarımlarında da sıkça karşımıza çıkmaktadır. OPA2604'ün şeması aşağıdaki gibidir:



Şekil 4.11 OPA2604'ün pin konfigürasyonu ve iç yapısı [21]

OPA2604 çift, başarılı AC performanslar için geliştirilmiş FET-girişli operasyonel bir yükselticidir. Çok düşük sapma, düşük gürültü ve geniş bant aralığıyla birçok ses ve medikal uygulamada kullanılmaktadır.

EMG yükseltici devresi için S. Sakrit'in EMG Amplifier çalışması referans alınmıştır. EMG sinyalleri öncelikli olarak INA118 ile yükseltilmiş ve OPA2604 operasyonel yükselticisi kullanılarak, yükseltilmiş EMG sinyali ve vücuttan ikinci bir elektrot ile alınan referans sinyali değeri, ortak mod gürültüsünün yok edilmesi için farklandırılmıştır. C4 ve R6 dirençleriyle 10Hz'lik yüksek geçiren filtre eklenmiştir. Bunun nedeni EMG sinyallerinin minimum 10 Hz'e sahip olmasıdır. Şekil 4.12'te öncül yükseltici devresi verilmiştir:



Şekil 4.12 Öncül yükseltici devresi

Devre üzerindeki hesaplamalar aşağıdaki gibidir:

Kazanç öncül yükseltici için 100 alınmıştır:

INA118'in özellikler kılavuzuna bakındığında Kazanç Formülü:

$$Kazanç = 1 + \frac{50k}{R_G} \tag{4.1}$$

ve bu formülden;

100 * R_G = R_G + 50k çıkar;

99R_G = 50000 Ω yani R_G = 505 Ω'dur. (R_G / 2 = R_1 = R_2) (R_1 = R_2 = 250 Ω alınmıştır. Bu durumda kazanç 101 civarı çıkmaktadır.)

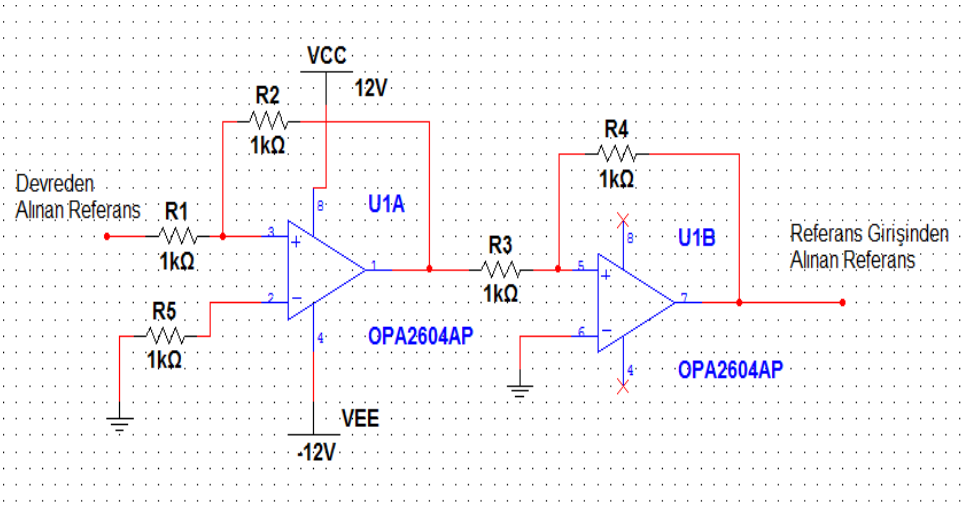
10 Hz yüksek geçiren filtre devresi hesabı aşağıdaki gibidir:

$$f_{kesim} = \frac{1}{2\pi RC} \tag{4.2}$$

$$f_{kesim} = \frac{1}{2\pi\ 91k\Omega\ 175\ nF}$$

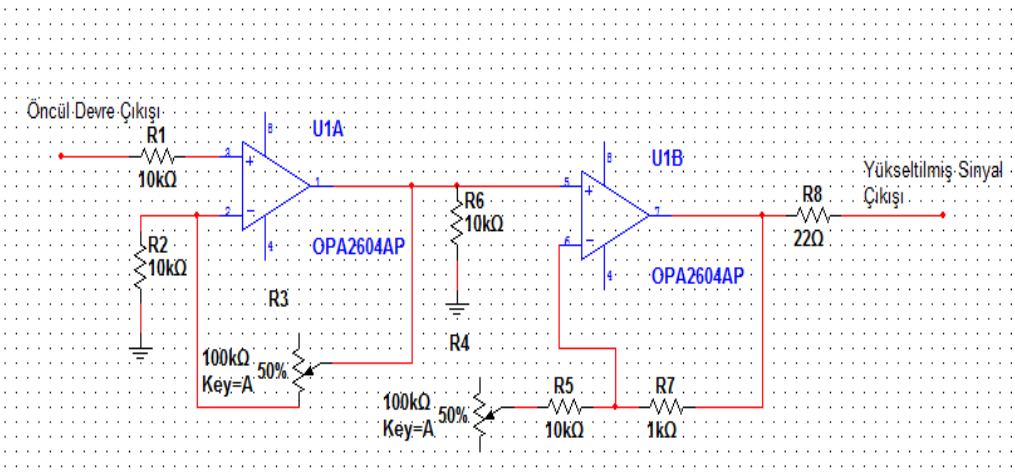
f_{kesim} = 9.99 Hz'dir.

Ortak mod gürültüsünü engellemek için; devre üzerinde alınan vücut referansı ile vücut üzerinde alınan referans farklandırılmıştır:



Şekil 4.13 Toprak gürültüsünü farklandırıcı devre

Ön yükselticiden çıkan yükseltilmiş sinyal, daha sonra ikinci bir yükseltici devreye tabi tutuldu. Bu işlemdeki amaç, elde edilen gerilimi ve gücü yükseltmektir. Bunun yanında devrede iki tane ayarlı direnç kullanılmıştır. Ayarlı dirençler kullanılarak, devrenin kazanç oranı ayarlanabilme hedeflenmiştir. İkincil yükseltici devre ile alakalı devre şemasına Şekil 4.14'de yer verilmiştir:



Şekil 4.14 İkincil yükseltici devre

Devre ile üzerindeki hesaplamalar aşağıdaki gibidir:

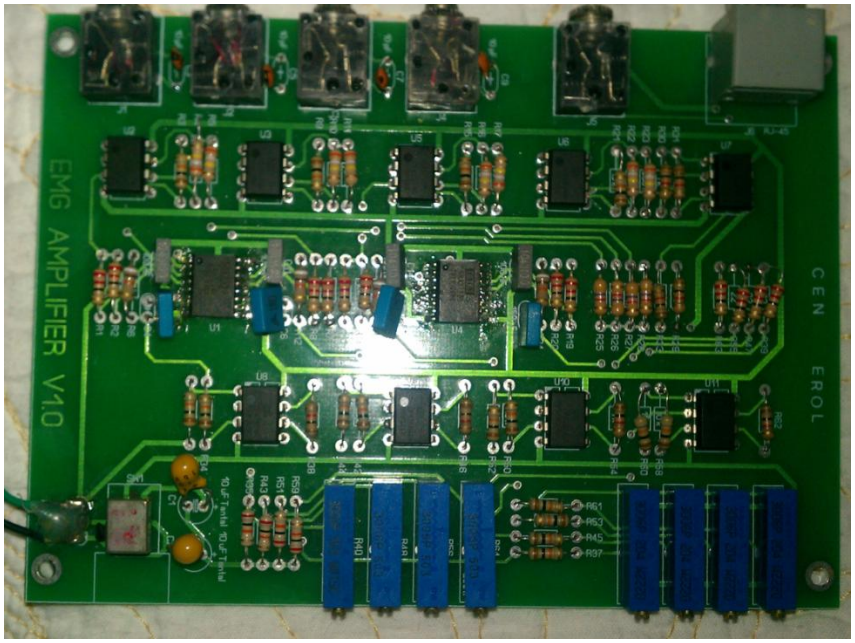
Birinci opamp'ın çıkışındaki gerilim kazancı aşağıdaki formülle hesaplanır:

$$Kazanç1 = 1 + \frac{R_3}{R_1} \quad (4.3)$$

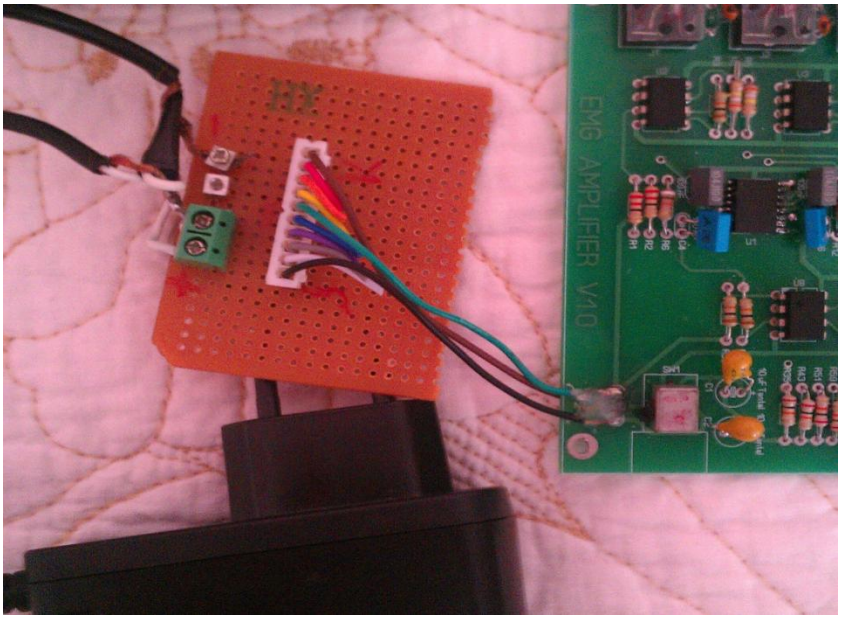
İkinci opamp'ın çıkışındaki gerilim kazancı da aşağıdaki formülle hesaplanır:

$$Kazanç2 = 1 + \frac{R_7}{R_5 + R_4} \quad (4.4)$$

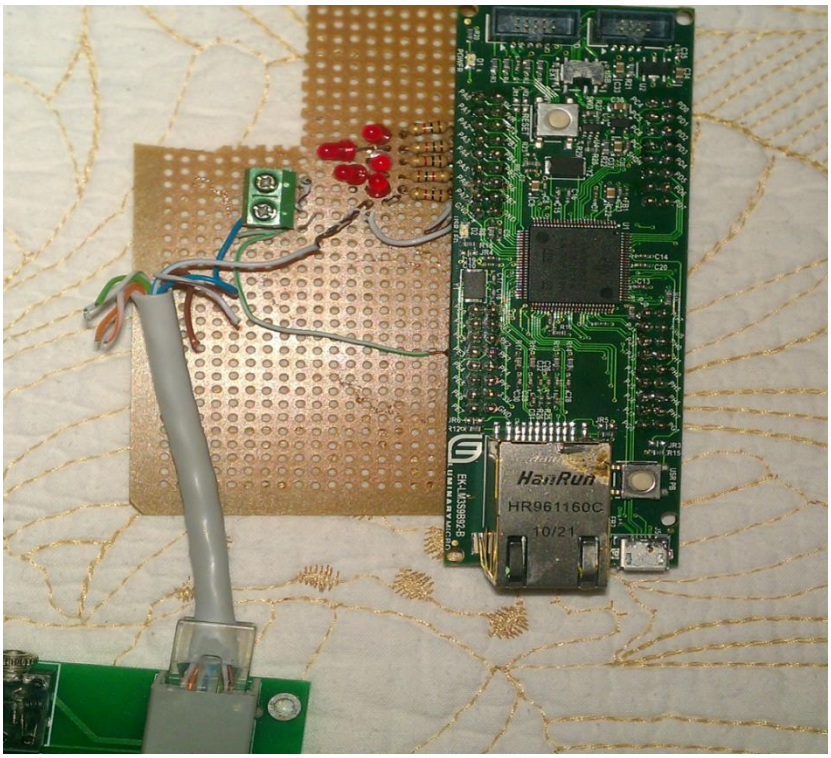
Yükseltilen EMG sinyalleri akıllı algoritmanın çalıştığı yükseltici devresine, ethernet konnektörü ile aktarılmıştır. Şekil 4.15'te yükseltici devre, Şekil 4.16'de adaptör bağlantıları görülmektedir, Şekil 4.17'de akıllı devre ünitesi ile yükseltici devrenin bağlantısı görülmektedir:



Şekil 4.15 EMG sinyallerini yükseltici devre



Şekil 4.16 EMG yükseltici devresi ile adaptörlerin bağlantısı

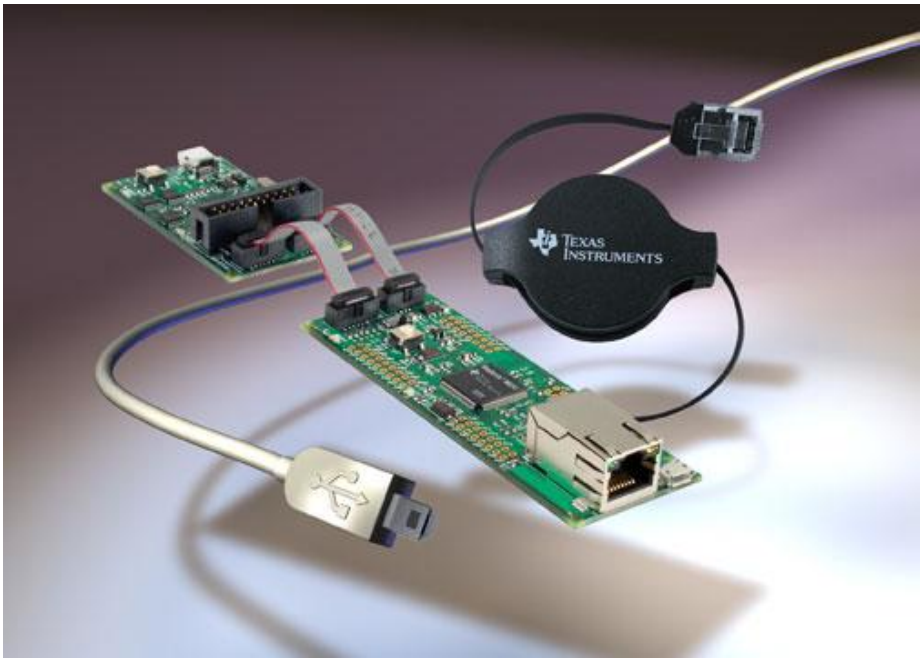


Şekil 4.17 EMG yükseltici devresi ile akıllı devrenin bağlantısı

EMG yükseltici devresi için genel bir devre şeması ve yükseltici devreye ait diyagram EK-D'de verilmiştir.

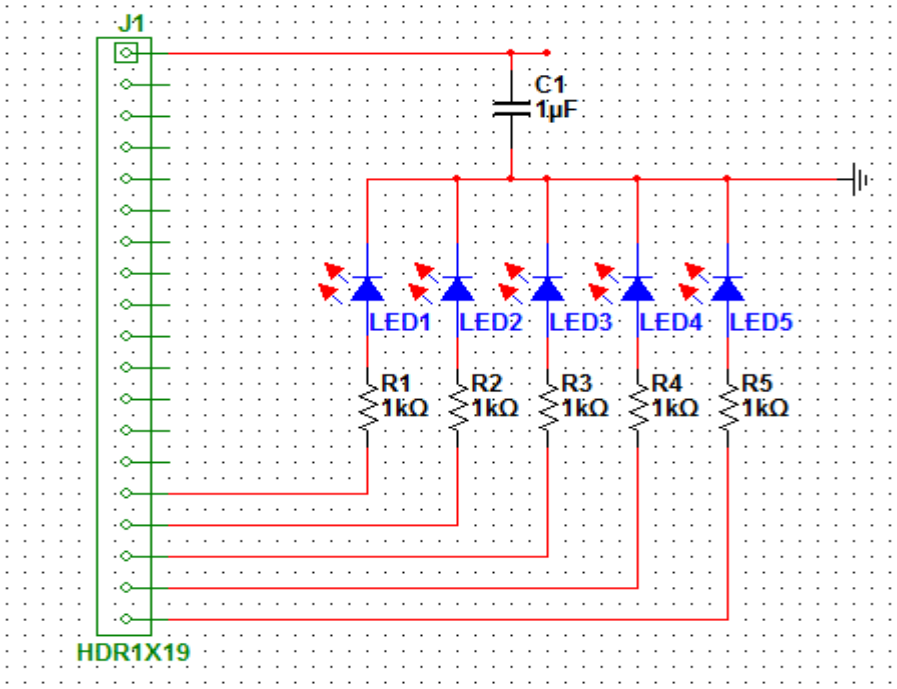
4. 4 Mikrodenetleyicili Akıllı Devre Ünitesi

GMM tabanlı akıllı algoritmaya ait yazılımının uygulandığı mikro denetleyici ünitesi olarak, Texas Instrument'ın LM3S9B92 mikrodenetleyicisi tabanlı, EKS-LM3S9B92 geliştirme devresi kullanılmıştır. LM3S9B92'nin ürün özellikleri EK A verilmiştir. Genel olarak LM3S9B92, ARM7 Cortex-M3 Stellaris tabanlı bir mikrodenetleyicidir. 256 kB dahili flash hafıza, 96 kB SRAM ünitesine, 10 bitlik ADC çözünürlüğüne, 16 MHz dahili osilatör'e, dahili +/- ADC referansına ve 80 MHz'e kadar varabilen işlem hızını sahiptir. EKS-LM3S9B92 geliştirme devresinin üzerinden bir adet LM3S9B92, bir adet USB çıkışı, 1 adet UART/USB ve ICDI programlama ve yazılım izleme devresi bağlantısı bulmaktadır. Şekil 4.18'de EKS-LM3S9B92 ve ICDI devresi görülmektedir [18]:

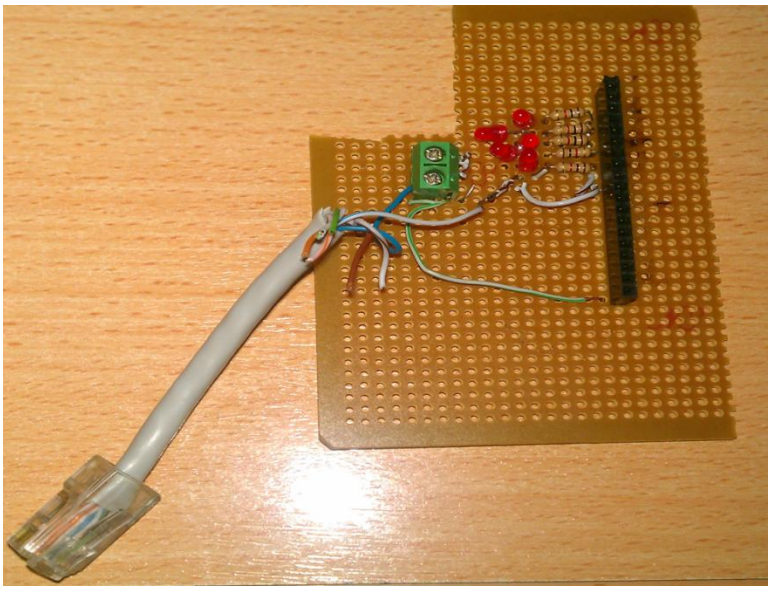


Şekil 4.18 EKS-LM3S9B92 ve ICDI Devresi [18]

EKS-LM3S9B92'e ek olarak, sonuçları gözlemlemek amacıyla çıktıların gözlemlendiği bir devre üzerine oturtulmuştur ve aynı zamanda ADC girişi bu devre üzerinden sağlanmıştır. Şekil 4.19 ve Şekil 4.20'de sonuç döndürücü devreye ait devre şeması ve resim verilmiştir:



Şekil 4.19 Sonuç gözlemleme devresinin şeması



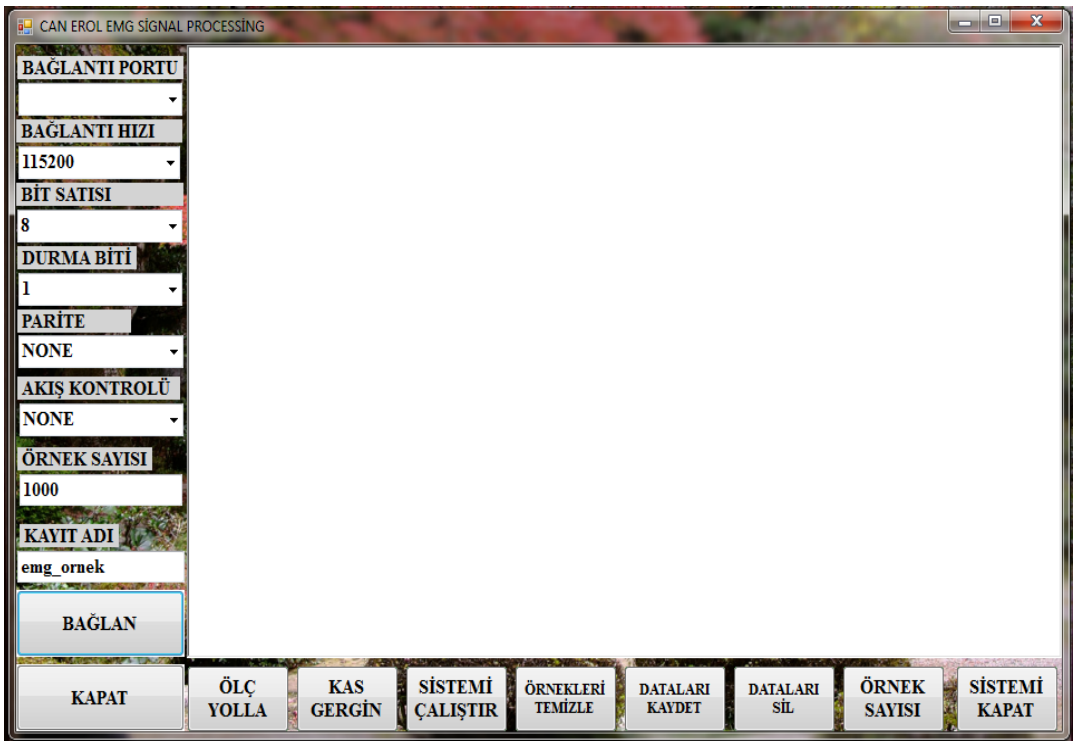
Şekil 4.20 Sonuç gözlemleme devresi

Devre üzerindeki ledler sırasıyla aşağıdaki görevlere sahiptir:

- 1. Led: Devre çalıştığında yanmaktadır.
- 2. Led: Gerginlik testinde 3. – 4. – 5. ledler ile birlikte sırasıyla yanar. Hareket testinde ise; aşağı doğru hareketi temsil eder.
- 3. Led: Gerginlik testinde 2. – 4. – 5. ledler ile birlikte sırasıyla yanar. Hareket testinde ise; yukarı doğru hareketi temsil eder.
- 4. Led: Gerginlik testinde 2. – 3. – 5. ledler ile birlikte sırasıyla yanar. Hareket testinde ise; sağa doğru hareketi temsil eder.
- 5. Led: Gerginlik testinde 2. – 3. – 4. ledler ile birlikte sırasıyla yanar. Hareket testinde ise; sola doğru hareketi temsil eder.

4. 5 Bilgisayar Arayüzü ve Gömülü Yazılımın Geliştirilme Ortamı

Sistemin bilgisayar arayüzü için Visual Studio 2010 ve C# yazılım dili kullanılmıştır. Bilgisayar arayüzü Şekil 4.20'de gösterilmiştir (bilgisayar arayüzü kodları EK B'de verilmiştir):



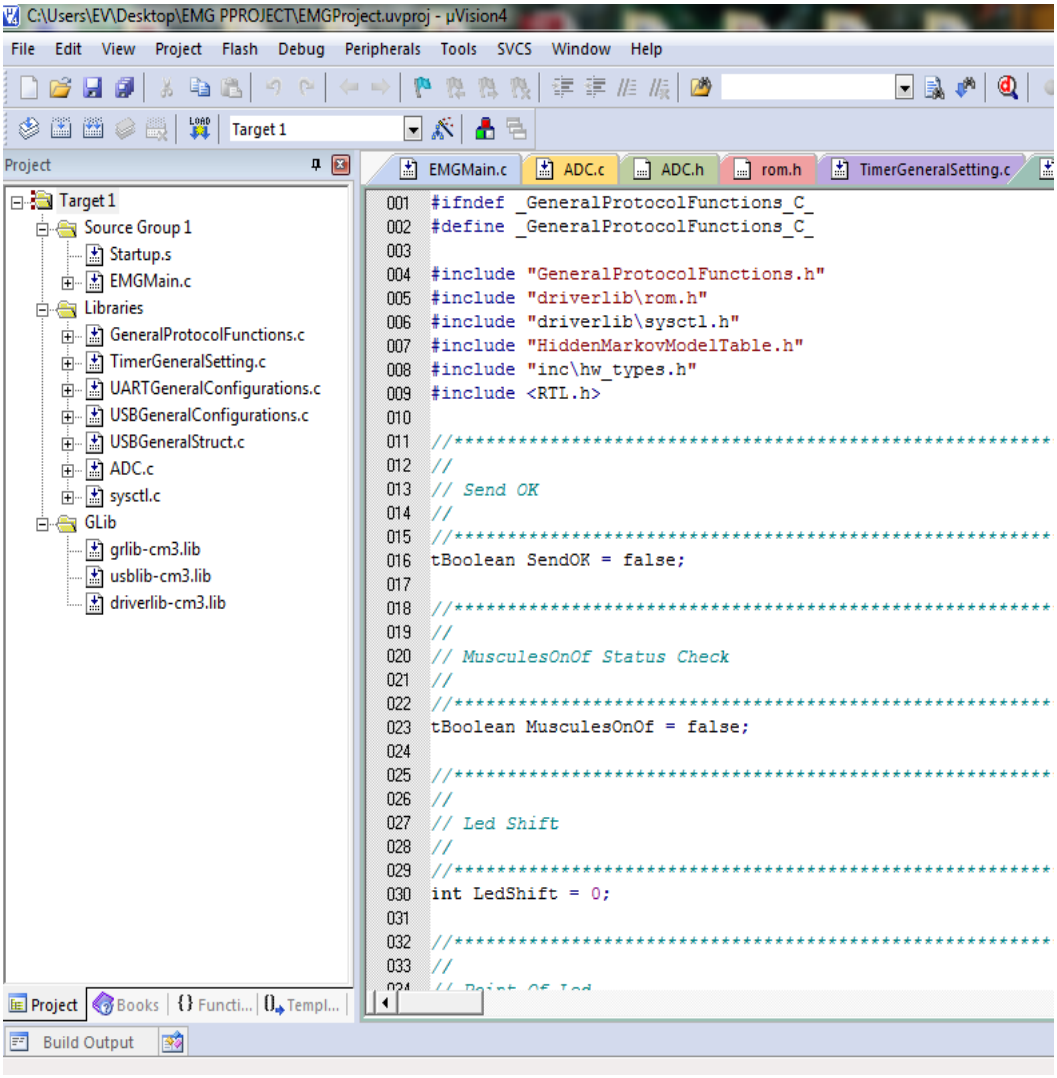
Şekil 4.21 EMG yükseltici devresinin bilgisayar arayüzü

Bilgisayar arayüzü seri-usb veya seri (UART) bağlantı kurulabilecek şekilde tasarlanmıştır. Programdan da görülebileceği gibi toplam altı adet iletişim ayarı

(“Bağlantı Portu”, “Bağlantı Hızı”, “Bit Sayısı”, “Durma Biti”, “Akış Kontrolü”), gelen verileri isimlendirmek için “Kayıt Adı” bölümü ve örnekleme sayısını ayarlama için “Örnek Sayısı” bölümü yapılmıştır. “Bağlan” tuşu ile bu ayarlar kullanılarak, elektronik devre ile iletişim kurulur. “Kapat” tuşu ile cihaz ile bağlantı kesilir. Diğer tuşların amaçları aşağıdaki gibidir:

- **Ölç Yolla:** Bu tuşa basıldığında, tek seferlik EMG ölçümü yapılır, bilgiler elektronik devreden gelir ve zengin yazı bölümüne kaydedilir.
- **Kas Gergin:** Bu tuşa basıldığında, ölçülen kasın gergin olup olmadığını anlayan algoritma çalışır ve sonuçları ledlerin bulunduğu devre üzerinde döndüren ve sürekli çalışan algoritma başlatılır.
- **Sistemi Çalıştır:** Bu tuşa basıldığında, elektronik devre sürekli bir şekilde parmakların yaptığı hareketin sağ-sol-ön-arka mı olduğunu tespit etmeye çalışır ve sonuçları ledlerin bulunduğu devre üzerinden döndürülür.
- **Örnekleri Temizle:** Bu tuş ile zengin yazı bölümündeki eski örneklemelere ait bilgiler silinir.
- **Dataları Kaydet:** Bu tuş ile zengin yazı bölümündeki bilgiler bir text dosyasına kaydedilir.
- **Dataları Sil:** Bu tuş ile elektronik kartın RAM’indeki bilgiler silinir.
- **Sistemi Kapat:** Bu tuş ile herhangi sürekli dönen algoritmalar sonlandırılır.

Gömülü yazılım geliştirme ortamı olarak “Keil uVision 4” programı kullanılmıştır. Yazılımın yazıldığı kod sayfası ve yazılan kütüphaneler Şekil 4.21‘de verilmiştir:



Şekil 4.22 Keil uVision 4'te EMG sinyallerini işleyen ve değerlendiren kodlara ait kütüphaneler

Daha öncede söylendiği gibi, cihaz ile bilgisayar arayüzü arasında seri-usb bağlantısı da kurulabilmektedir. Bu bağlantı için, USB Windows sürücüsü ile sanal COM port oluşturulmuştur. USB Windows sürücüsü Texas Instrument tabanlı olup, aşağıdaki yükleme dosyası ile bilgisayara yüklenmektedir:

```
;
;   Stellaris USB CDC (serial) driver installation file.
;
[Version]
Signature="$Windows NT$"
Class=Ports
ClassGuid={4D36E978-E325-11CE-BFC1-08002BE10318}
Provider=%MFGNAME%
CatalogFile.NTx86=usb_dev_serial_x86.cat
LayoutFile=layout.inf
CatalogFile.NTamd64=usb_dev_serial_amd64.cat
DriverVer=03/10/2010,1.2.0

[Manufacturer]
%MFGNAME%=VirComDevice,NT,NTamd64

[DestinationDirs]
DefaultDestDir = 12

[VirComDevice.NT]
%DESCRIPTION%=DriverInstall,USB\Vid_1CBE&Pid_0002

[VirComDevice.NTamd64]
%DESCRIPTION%=DriverInstall,USB\Vid_1CBE&Pid_0002

[DriverInstall.NT]
Include=mdmcpq.inf
CopyFiles=FakeModemCopyFileSection
AddReg=DriverInstall.NT.AddReg

[DriverInstall.NT.AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,usbser.sys
HKR,,EnumPropPages32,,"MsPorts.dll,SerialPortPropPageProvider"

[DriverInstall.NT.Services]
AddService=usbser, 0x00000002, DriverServiceInst

[DriverServiceInst]
DisplayName=%SERVICE%
ServiceType=1
StartType=3
ErrorControl=1
ServiceBinary=%12%\usbser.sys

[Strings]
MFGNAME     = "Texas Instruments, Inc."
DESCRIPTION = "Stellaris USB serial port"
SERVICE     = "Stellaris USB CDC serial port"
```

Gömülü yazılıma ait demo kodlar EK-C'de verilmiştir.

4. 6 Hesaplama İşlemleri ve Algoritmalar

Algoritmaların hesaplanması için, her bir hareket grubu ve parmak için iki yüzer defa ölçüm alınmış, eğitim kümeleri oluşturulmuş ve oluşturulan eğitim kümelerine göre algoritmalar hesaplanmıştır. 4.5.1'de bölütleme işlemine ait hesaplamalar ve 4.5.2'de ise GMM modeline ait hesaplamalar verilmiştir.

4.6.1 Bölütleme İşlemi Hesaplamaları

Bölütleme işlemi için 3.3.1'deki (3.11) formülü revize edilerek kullanılmıştır. Bu amaçla eşik şiddetinin bulunması için tüm örnek setlerindeki EMG data'ları incelenmiştir. Genellikle bölütleme algoritmasıyla, incelenen sinyal üzerinde anlamlı parçaların bulunması amaçlanmaktadır. Bu çalışmada, bölütleme algoritması GMM için düğüm noktaları bulunması amacıyla kullanılmıştır. Aşağıdaki hesaplamalar bölütleme hesaplamalarına aittir:

1. Tüm veri seti incelendi. Tüm veri setlerinde, kasın herhangi bir konumda kasılı olduğu zaman, anlamlı en yüksek değerin 0.30 V ile 0.36 V olduğu gözlemlendi.
2. Tüm veri seti incelendi, kasın hiçbir türlü kasılı olmadığı zaman, kastan elde edilen en yüksek değer 0.17 V ile 0.20 V arasında olduğu gözlemlendi.
3. En uç noktalar yani 0.17 V değeri 0.36 V ile eşleştirildi. Aradaki fark bu sayıların ortalaması yani "**Ortalama** = (0.36 + 0.17) / 2 = 0.265" olarak bulundu.
4. Herhangi bir kasılma sırasında ölçülen sinyaller arasında, 0.265 V üzerinde olan değerlerin maksimum sayısı 27'dir.
5. Herhangi bir kasılma sırasında ölçülen sinyaller arasında, 0.265 V üzerinde olan değerlerin minimum sayısı 5'dir.

Formülasyon revize edilirse:

Eğer;

$$max_i\{x_i\} > \frac{27}{L}\sum^{L}|x_i| \tag{4.5}$$

ise;

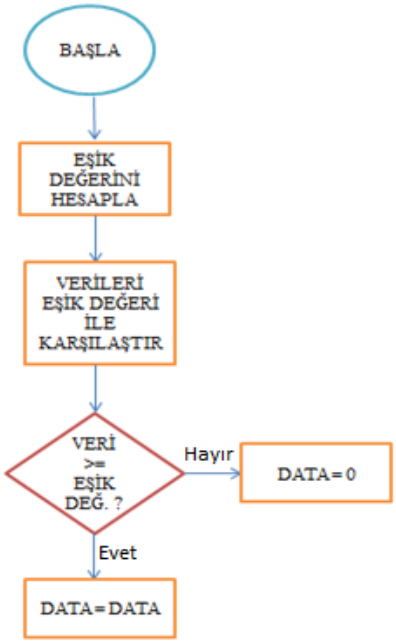
$$T = \frac{5}{L}\sum_{i=1}^{L}|x_i| \tag{4.6}$$

formülü ile bulunur. Eğer değilse;

$$T = max_i\{x_i\}\,\frac{1}{5} \tag{4.7}$$

şeklini alır.

Gömülü yazılım içinde kullanılan, EMG'ye ait düğüm noktalarını tespit algoritması aşağıdaki gibidir:



Şekil 4.23 Bölütleme işleminin algoritmik gösterimi

Bölütleme işleminin yalancı kodlar ile gösterimi:

- Bölütleme işlemine başla;
- Eşik değerini hesapla;
- Döngünün örnek sayısını ayarla ve döngüyü başlat;
- Döngü sayısına göre örneklenen sinyal değerleri ile eşik değerini karşılaştır;
- Örnek değer, eğer eşik değerinden küçük ise sıfıra eşitle;
- Değil ise değer kendisidir;
- Döngü sayacını bir arttır;
- Eğer döngü sayacı ve örnekleme sayısı eşitse döngüyü bitir;
- Değilse döngüye devam et;

4.6.2 GMM ile Alakalı Hesaplamalar

Bölütleme algoritması ile hesaplanan EMG düğüm noktaları ile GMM çizelgesi oluşturulmuştur. GMM için ileri yönlü ergodik model kullanılmıştır. Bu çizelgeler her bir parmak için ayrı ayrı oluşturulmuştur. Bu amaçla;

1. Ortak Bir Düğüm Noktası:
- Sağa doğru harekette en yüksek değerin 0.30 V veya üzeri olma olasılığı: $P_{SEY}(s_1)$ = %80
- Sola doğru harekette en yüksek değerin 0.30 V veya üzeri olma olasılığı: $P_{SLEY}(sl_1)$ = %96
- Yukarı doğru harekette en yüksek değerin 0.30 V veya üzeri olma olasılığı: $P_{YEY}(y_1)$ = %100
- Aşağı doğru harekette en yüksek değerin 0.30 V veya üzeri olma olasılığı: $P_{AEY}(a_1)$ = %98

Görüldüğü gibi her bir hareket sırasında ölçülen değerler arasında; en yüksek değerin 0.30 V ve daha yüksek olma olasılığı oldukça yüksektir. Böylece 0.30 V'luk değer ortak bir düğüm noktası olarak alınabilir.

2. Hareketlere bağlı geçiş matrisleri elemanları ve geçiş matrisleri aşağıdaki gibidir:
Sağa doğru hareket için; hareketten elde edilen ve en yüksek koşullu olasılığa sahip düğüm noktaları aşağıdaki gibidir:

d_1 = 0.2900 – 0.3500 (volt)
d_2 = 0.2356 – 0.2654 (volt)
d_3 = 0.2654 – 0.2899 (volt)
d_4 = 0.2053 – 0.2355 (volt)
d_5 = 0.1753 – 0.2052 (volt)

Bu koşullu olasılık değerlerinden yola çıkarak, oluşturulan geçiş ve olasılık dağılım matrisi aşağıda verilmiştir. Olasılık dağılım matrisi hesaplamalarında, eksik parametre tahmini hesapları, incelenen örneklere göre olasılıklar hesaplanarak yapılmıştır.

Geçiş matrisi aşağıdaki gibidir:

$$S = \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} & s_{15} \\ s_{21} & s_{22} & s_{23} & s_{24} & s_{25} \\ s_{31} & s_{32} & s_{33} & s_{34} & s_{35} \\ s_{41} & s_{42} & s_{43} & s_{44} & s_{45} \\ s_{51} & s_{52} & s_{53} & s_{54} & a_{55} \end{bmatrix}$$

Geçiş olasılıkları matrisi:

$$P(s_{ij}) = \begin{bmatrix} 0.3 & 0.08 & 0.06 & 0.06 & 0.04 \\ 0.02 & 0.18 & 0.02 & 0.04 & 0 \\ 0 & 0.06 & 0.08 & 0.04 & 0 \\ 0 & 0.12 & 0.02 & 0.14 & 0 \\ 0 & 0 & 0 & 0 & 0.02 \end{bmatrix}$$

$$\sum_{i=1}^{5}\sum_{j=1}^{5} P(s_{ij}) = 1'dir.$$

Bu işlemler benzer olarak sırasıyla sol – yukarı – aşağı ve durgun hareketlerde de uygulanmıştır:

Sol hareketi için geçiş matrisi aşağıdaki gibidir:

$$SL = \begin{bmatrix} sl_{11} & sl_{12} & sl_{13} & sl_{14} & sl_{15} \\ sl_{21} & sl_{22} & sl_{23} & sl_{24} & sl_{25} \\ sl_{31} & sl_{32} & sl_{33} & sl_{34} & sl_{35} \\ sl_{41} & sl_{42} & sl_{43} & sl_{44} & sl_{45} \\ sl_{51} & sl_{52} & sl_{53} & sl_{54} & sl_{55} \end{bmatrix}$$

Geçiş olasılıkları matrisi:

$$P(sl_{ij}) = \begin{bmatrix} 0.18 & 0.16 & 0.06 & 0.04 & 0 \\ 0.06 & 0.12 & 0.02 & 0.02 & 0 \\ 0.1 & 0.06 & 0.02 & 0.02 & 0 \\ 0 & 0.02 & 0.06 & 0.04 & 0 \\ 0.02 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\sum_{i=1}^{5}\sum_{j=1}^{5} P(sl_{ij}) = 1'dir.$$

Yukarı hareketi için geçiş matrisi aşağıdaki gibidir:

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15} \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25} \\ y_{31} & y_{32} & y_{33} & y_{34} & y_{35} \\ y_{41} & y_{42} & y_{43} & y_{44} & y_{45} \\ y_{51} & y_{52} & y_{53} & y_{54} & y_{55} \end{bmatrix}$$

Geçiş olasılıkları matrisi:

$$P(y_{ij}) = \begin{bmatrix} 0.24 & 0.18 & 0.08 & 0.02 & 0 \\ 0.1 & 0.06 & 0.02 & 0.04 & 0 \\ 0.02 & 0.06 & 0.02 & 0.06 & 0 \\ 0.04 & 0.02 & 0.02 & 0.02 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\sum_{i=1}^{5}\sum_{j=1}^{5} P(y_{ij}) = 1'dir.$$

Aşağı hareketi için geçiş matrisi aşağıdaki gibidir:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

Geçiş olasılıkları matrisi:

$$P(a_{ij}) = \begin{bmatrix} 0.1 & 0.06 & 0.12 & 0.1 & 0.04 \\ 0.02 & 0.02 & 0.04 & 0 & 0.02 \\ 0.12 & 0.04 & 0.06 & 0.02 & 0 \\ 0.02 & 0.06 & 0.02 & 0 & 0 \\ 0 & 0.02 & 0.06 & 0.06 & 0 \end{bmatrix}$$

$$\sum_{i=1}^{5}\sum_{j=1}^{5} P(a_{ij}) = 1'dir.$$

3. Tüm çizelgeler olasılık değerleriyle birlikte gömülü yazılıma işlenmiştir. Gömülü yazılım, ölçtüğü değerlerin offset'li aralığını çizelgelerde arar, olasılıklarını bulur ve bulduğu olasılıkları logaritmik olarak hesaplar:

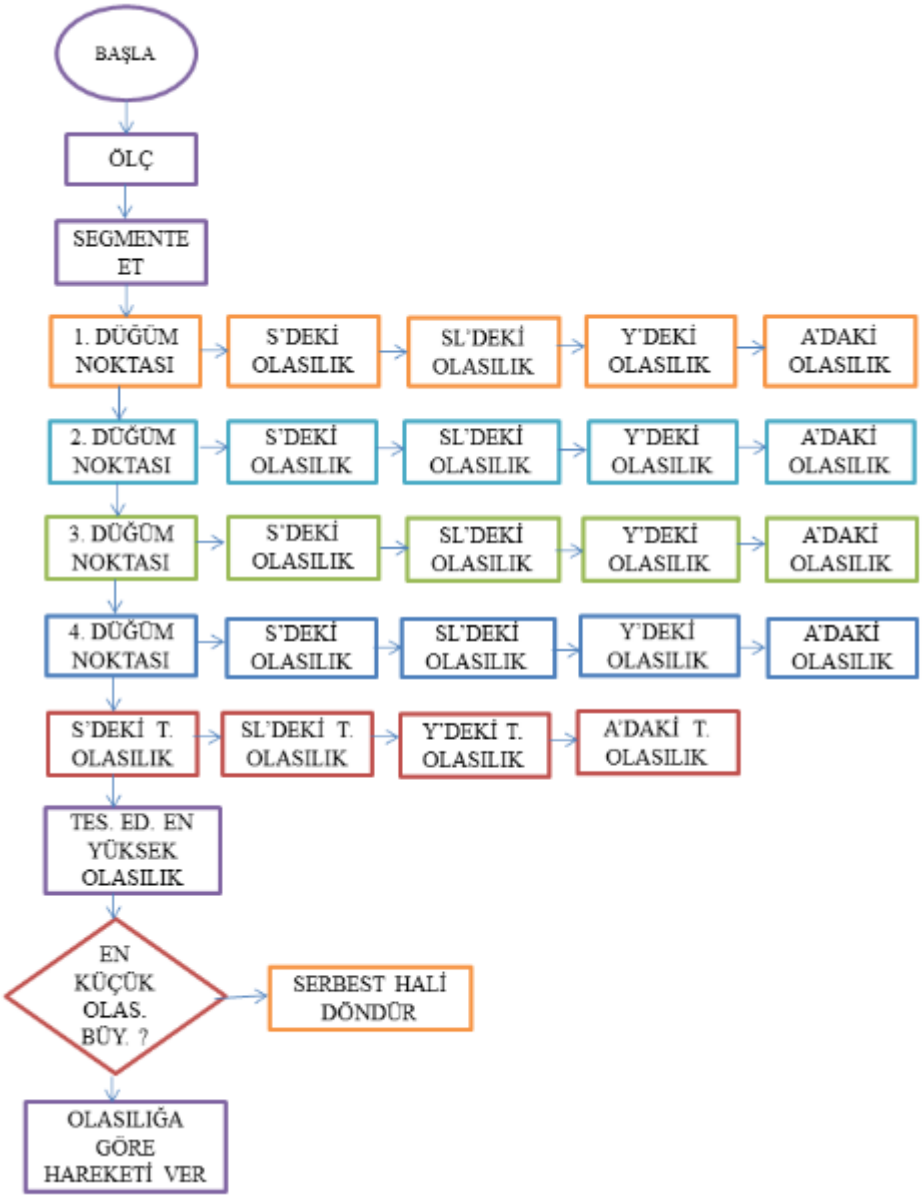
Örneğin, ölçülen beş sinyal değerinin olasılığı A'ya göre aşağıdaki gibi hesapları: Tespit Edilen Düğüm Noktaları: Birinci – Beşinci – Üçüncü – Dördüncü =>

a_{11}, a_{15}, a_{53}, a_{34}

$\log P(A) = \log a_{11} + \log a_{15} + \log a_{53} + \log a_{34}$ şeklinde hesaplanır.

4. Çizelgelere göre bulunan 4 tane sonuç, beklenen en düşük değerden de düşük ise bu hareket serbest hal hareketidir.

Algoritmanın akış diyagramı aşağıdaki gibidir:



Şekil 4.24 GMM'nin algoritmasına ait akış diyagramı

BÖLÜM 5

SONUÇ ve YORUMLAR

5.1. Kasılı Durum Sonuçları

Kasılı durum sonuçları, aşağıdaki gibidir:

Çizelge 5.1 Parmaklara göre kasılma değerlerinin doğru tespit yüzdeleri

PARMAK	BAŞ	İŞARET	ORTA	YÜZÜK	SERÇE
BAŞARI ORANI	%100	%100	%92	%90	%100

Görüldüğü gibi, ölçüm yapılan parmağın kasılı olup olmadığının tespiti oldukça yüksek sonuçlar ile tespit edilmiştir. Bundaki en büyük katkı, parmağın kasılı olmadığı durumlarda ölçülen EMG sinyali seviyelerinin oldukça düşük olması. Böylelikle, kasılı olmayan durumda elde edilen EMG sinyallerinin, GMM çizelgelerinde tespit edilen olasılık değerleri oldukça düşük olmaktadır.

5.2. Sağ-Sol-Yukarı-Aşağı Konum Testleri Sonuçları

Bu testler sol el ve her bir parmak için ayrı ayrı yapılmıştır ve eşit süreli olarak denenmiştir. Her bir denemeden sonra, parmaklar dinlendirilmiştir. Toplam 100'er deneme sonrasındaki test sonuçları aşağıdaki çizelgede verilmiştir:

Çizelge 5.2 Parmaklara göre sağ-sol-yukarı-aşağı hareketlerin doğru tespitlerinin yüzdeleri

	SAĞ	SOL	YUKARI	AŞAĞI
BAŞ	%92	%83	%95	%84
İŞARET	%67	%63	%92	%92
ORTA	%24	%28	%78	%71
YÜZÜK	%36	%29	%64	%83
SERÇE	%25	%69	%72	%69

Görüldüğü gibi başparmaktan alınan verilerin sonuçlarının doğru tespit oranı en yüksektir. Bunun nedeni başparmağın hareket kabiliyetinin diğer parmaklara göre daha yüksek olmasıdır. Böylece daha fazla kastan bilgi elde edilebilmiştir. Parmaklar arasında doğru tespitin, en düşük olduğu parmak yüzük parmağıdır. Bu da beklenen bir sonuçtur, çünkü bu parmağın hareket kabiliyeti diğer parmaklara göre oldukça düşüktür.

5.3 Sonuçların Yorumlanması ve Düşünülen Geliştirmeler

Bu çalışmada belirlenen yöntem ile hareketlerin doğru tespiti yukarıdaki çizelgelerde verilmiştir. Sonuçların bazılarının, hatırı sayılabilir bir doğruluk oranı olduğu söylenebilir. Özellikle başparmak sonuçları oldukça doğruluk yüzdesine sahiptir.

Elde edilen sonuçlarda, hareketlerin yüksek kas gerginliği ile yapılması, doğru tespit olasılığını artırdığı görülmüştür.

Başarı yüzdelerinin artırımı için bir yöntem olarak GMM çizelgelerindeki parametre ve kaslardan elde edilen düğüm noktası sayısının arttırılması gösterilebilir. Böylece EMG sinyalleri üzerinde daha fazla ayrıntı incelenebilir. Ancak, bu yöntem sistemde gecikmeye yol açacaktır.

Bu çalışmada her bir parmak ayrı ayrı incelenmiştir. Tüm parmaklardan aynı anda veri alıp değerlendirilmesi durumunda, bu yöntemle başarı oranının düşeceği tahmin edilmektedir, çünkü bu yöntem belirli bir zaman diliminde tekbir hareket üzerinden tasarlanmıştır.

Çalışmada dört çeşit hareket incelenmiştir. Ancak gerçek hayat düşünülecek olursa, bir parmakta tespit edilecek birçok hareket ve konum olabilir. Eğer bir protez parmak yapılması düşünülürse, bir parmakta tespit edilmek istenecek dörtten çok daha fazla konum vardır ve bir el hareketinin doğru tespiti ancak elin çevresine yayılmış bir sensör ağıyla yapılabilir. Örneğin, kalem tutma hareketi düşünülürse, başparmak ve işaret parmak yatay bir konumda kalemi sıkıştırır ve diğer parmakla mümkün olduğunca içe bükülür. İşaret parmağı kopmuş bir hasta düşünülürse, protez diğer parmakların konumunu algıladığında, yani başparmağın yatay bir konumda büküldüğünü ve diğer parmakların mümkün olduğunca içe bükülmelerini algıladığında, yapılan hareket doğru bir şekilde tespit edilebilir ve protez doğru bir şekilde istenilen konuma yönlendirilebilir.

KAYNAKLAR

[1] Yazgan, E. ve Korkürek, M., (1996). Tıp Elektroniği, İstanbul Teknik Ünivertsitesi Yayınları, 386, İstanbul

[2] Clarys, J. P., (2000), "EMG History", Ergonomics, 43:1760-1762.

[3] Yazıcı, İ, (2008). EMG İşaretlerinin İşlenmesi Ve Sınıflandırılması, Yüksek Lisans Tezi, Sakarya Üniversitesi Fen Bilimleri Enstitüsü, Sakarya.

[4] Seller, B., Muscle Physiology, chrisevans3d.com/files/reference/muscle_physiology_sellers.pdf, 1 Haziran 2012

[5] Kaur, G., Arora, A.S., Jain, V.K. ve Sidhu, G.S., (2010). "EMG Diagnosis Via Time Domain Features And Binary Support Vector Machine Classification", Internastional Journal Of Engineering Science and Techology, 2(10):5192-5196

[6] Luca, C.J. Surface Electromyography:Detection And Recording, www.delsys.com/Attachments_pdf/WP_SEMGintro.pdf, 1 Haziran 2012

[7] Day, S., Important Factors in Surface EMG Measurement, www.health.uottawa.ca/biomech/courses/apa4311/semg.pdf, 1 Haziran 2012

[8] Siriprayoonsak, S., (2005). Real – Time Measurement Of Prehensile EMG Signals, Yüksek Lisans Tezi, Department of Computer Science San Diego State University, San Diego.

[9] Peleg, D., Brainman, E. ve Yom-Yov, E., (2002). "Classification Of Finger Activation For Use In A Robotic Prosthesis Arm", IEEE, 10(4):290-293

[10] Okuda, H., Kometani F., Inagaki S. ve Suzuli, T., (2009). "Fatigue Recognition Using EMG Signals And Stohastic Switched ARX Model", ICINCO, 202-207.

[11] Kaur, G., Arora, A.S. ve Jain, V.K., (2009). "Comparison Of The Techniques Used For Segmentation Of EMG Signals", MACMESE'09 Proceeding of the 11th WSEAS international conferense, 7-9 November 2009, Baltimore.

[12] Hardalaç, F. ve Poyraz, M., (2002). "Yapay Sinir Ağları Kullanarak EMG Sinyallerinin Sınıflandırılması ve Neuropathy Hastalığının Teşhisi", Journal of Polytechnic, 5(1):75-82.

[13] Azami, H., Sanei, S. ve Mohammadi, K., (2011). "A Novel Signal Segmentation Method Based On Standard Deviation And Variable Treshold", International Journal Of Computer Applications, 34(2):27-34.

[14] Mazumder, A., A Model-Based Indicator Of Musculoskeletal Disorders Via Hidden Markov Model As An EMG Pattern Recognition Method, http://probability.ca/jeff/ftpdir/anjali1.pdf, 1 Haziran 2012.

[15] Evans, J.S. ve Krishnamurthy, V., (1999). "Hidden Markov Model State Estimation With Randomly Delayed Observations", IEEE Transactions On Signal Processing, 47(8):2157-2166.

[16] Juang, B.H. ve Rabiner, L.R., (1991). "Hidden Markov Models For Speech Receognition", Technometrics, 33(3):251-272.

[17] Rabiner, L.R., (1989). "A Tutorial On Hidden Markov Models And Selected Applıcatıons In Speech Recognition", IEEE, 77(2):257-286.

[18] Texas Instrument, LM3S9B92, www.ti.com/lit/ds/symlink/lm3s9b92.pdf, 1 Haziran 2012.

[19] Texas Instrument, INA118, www.ti.com/lit/ds/symlink/ina118.pdf, 1 Haziran 2012.

[20] Texas Instrument, INA2128, www.ti.com/lit/ds/symlink/ina2128.pdf, 1 Haziran 2012.

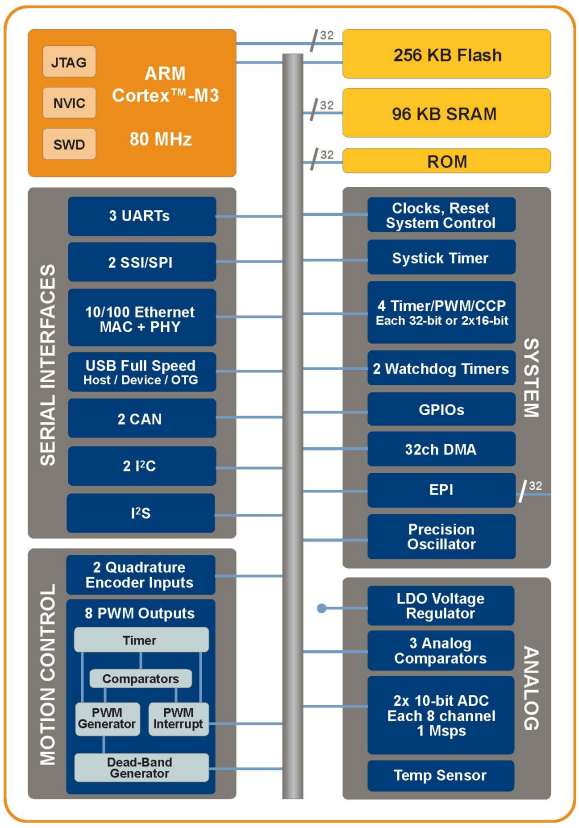
[21] Texas Instrument, OPA2604, www.ti.com/lit/ds/symlink/opa2604.pdf, 1 Haziran 2012.

EK-A

LM3S9B92 ÜRÜN ÖZETİ

LM3S9B92 Microcontroller

LUMINARY MICRO®



Product Features

- ARM® Cortex™-M3 Processor Core
  - 80-MHz operation; 100 DMIPS performance
  - ARM Cortex SysTick Timer
  - Nested Vectored Interrupt Controller (NVIC)
- On-Chip Memory
  - 256 KB single-cycle Flash
  - 96 KB single-cycle SRAM
  - Internal ROM loaded with StellarisWare™ software:
    - Stellaris® Peripheral Driver Library
    - Stellaris® Boot Loader
    - Advanced Encryption Standard (AES) cryptography tables
    - Cyclic Redundancy Check (CRC) error detection functionality
- External Peripheral Interface (EPI)
  - 8/16/32-bit dedicated parallel bus for external peripherals
  - Supports SDRAM, SRAM/Flash, FPGAs, CPLDs
- Advanced Serial Integration
  - 10/100 Ethernet MAC and PHY
  - Two CAN 2.0 A/B controllers
  - USB 2.0 OTG/Host/Device
  - Three UARTs with IrDA and ISO 7816 support (one UART with full modem controls)
  - Two I^2C modules
  - Two Synchronous Serial Interface modules (SSI)
  - Integrated Interchip Sound (I^2S) module
- System Integration
  - Direct Memory Access Controller (DMA)
  - System control and clocks including on-chip precision 16-MHz oscillator
  - Four 32-bit timers (up to eight 16-bit)
  - Eight Capture Compare PWM pins (CCP)
  - Real-Time Clock
  - Two Watchdog Timers
    - One timer runs off the main oscillator
    - One timer runs off the precision internal oscillator
  - Up to 65 GPIOs, depending on configuration
    - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
    - Independently configurable to 2, 4 or 8 mA drive capability
    - Up to 4 GPIOs can have 18 mA drive capability
- Advanced Motion Control
  - Eight advanced PWM outputs for motion and energy applications
  - Four fault inputs to promote low-latency shutdown
  - Two Quadrature Encoder Inputs (QEI)
- Analog
  - Two 10-bit Analog-to-Digital Converters (ADC) with sixteen analog input channels and sample rate of one million samples/second
  - Three Analog Comparators
  - 16 Digital Comparators
  - On-chip voltage regulator
- JTAG and ARM Serial Wire Debug (SWD)
- 100-pin LQFP package
- Industrial (-40°C to 85°C) Temperature Range

Target Applications

- Motion control
- Factory automation
- Fire and security
- HVAC and building control
- Power and energy
- Transportation
- Test and measurement equipment
- Medical instrumentation
- Remote monitoring
- Electronic point-of-sale (POS) machines
- Network appliances and switches
- Gaming equipment

Ordering Information

Orderable Part Number	Description
LM3S9792-IQC80	Stellaris® LM3S9792-IQC80 Microcontroller Industrial Temperature (128 KB Flash)
LM3S9B92-IQC80	Stellaris® LM3S9B92-IQC80 Microcontroller Industrial Temperature (256 KB Flash)

Luminary Micro, Inc. • 108 Wild Basin, Suite 350 • Austin, TX 78746
Main: +1-512-279-8800 • Fax: +1-512-279-8879 • http://www.luminarymicro.com

PB-LM3S9B92-00

EK-B

BİLGİSAYAR ARAYÜZÜ KODLARI

Form1.Designer.Cs

```
namespace EMGKayıtArayuzu
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.splitter1 = new System.Windows.Forms.Splitter();
            this.button1 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.comboBox1 = new System.Windows.Forms.ComboBox();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.comboBox2 = new System.Windows.Forms.ComboBox();
            this.label3 = new System.Windows.Forms.Label();
            this.comboBox3 = new System.Windows.Forms.ComboBox();
            this.label4 = new System.Windows.Forms.Label();
            this.comboBox4 = new System.Windows.Forms.ComboBox();
            this.label5 = new System.Windows.Forms.Label();
            this.comboBox5 = new System.Windows.Forms.ComboBox();
            this.label6 = new System.Windows.Forms.Label();
            this.comboBox6 = new System.Windows.Forms.ComboBox();
            this.shapeContainer1 = new
Microsoft.VisualBasic.PowerPacks.ShapeContainer();
            this.lineShape1 = new Microsoft.VisualBasic.PowerPacks.LineShape();
            this.button3 = new System.Windows.Forms.Button();
            this.panel1 = new System.Windows.Forms.Panel();
            this.button10 = new System.Windows.Forms.Button();
            this.button9 = new System.Windows.Forms.Button();
            this.button8 = new System.Windows.Forms.Button();
            this.button7 = new System.Windows.Forms.Button();
            this.button6 = new System.Windows.Forms.Button();
            this.button5 = new System.Windows.Forms.Button();
            this.button4 = new System.Windows.Forms.Button();
            this.textBox1 = new System.Windows.Forms.TextBox();
```

```
            this.label7 = new System.Windows.Forms.Label();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.label8 = new System.Windows.Forms.Label();
            this.saveFileDialog1 = new System.Windows.Forms.SaveFileDialog();
            this.serialPort1 = new System.IO.Ports.SerialPort(this.components);
            this.richTextBox1 = new System.Windows.Forms.RichTextBox();
            this.panel1.SuspendLayout();
            this.SuspendLayout();
            //
            // splitter1
            //
            this.splitter1.BackColor = System.Drawing.Color.Lime;
            this.splitter1.Location = new System.Drawing.Point(0, 0);
            this.splitter1.Name = "splitter1";
            this.splitter1.Size = new System.Drawing.Size(154, 563);
            this.splitter1.TabIndex = 0;
            this.splitter1.TabStop = false;
            //
            // button1
            //
            this.button1.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button1.Location = new System.Drawing.Point(0, 447);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(151, 55);
            this.button1.TabIndex = 1;
            this.button1.Text = "BAĞLAN";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button2.Location = new System.Drawing.Point(0, 508);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(151, 55);
            this.button2.TabIndex = 2;
            this.button2.Text = "KAPAT";
            this.button2.UseVisualStyleBackColor = true;
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // comboBox1
            //
            this.comboBox1.BackColor = System.Drawing.Color.White;
            this.comboBox1.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Location = new System.Drawing.Point(2, 31);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(147, 27);
            this.comboBox1.TabIndex = 3;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.BackColor = System.Drawing.Color.Gainsboro;
            this.label1.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.label1.Location = new System.Drawing.Point(1, 9);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(149, 19);
```

```
            this.label1.TabIndex = 4;
            this.label1.Text = "BAĞLANTI PORTU";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.BackColor = System.Drawing.Color.LightGray;
            this.label2.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.label2.Location = new System.Drawing.Point(0, 61);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(149, 19);
            this.label2.TabIndex = 5;
            this.label2.Text = "BAĞLANTI HIZI     ";
            //
            // comboBox2
            //
            this.comboBox2.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.comboBox2.FormattingEnabled = true;
            this.comboBox2.Items.AddRange(new object[] {
            "9600",
            "14400",
            "19200",
            "115200"});
            this.comboBox2.Location = new System.Drawing.Point(2, 83);
            this.comboBox2.Name = "comboBox2";
            this.comboBox2.Size = new System.Drawing.Size(145, 27);
            this.comboBox2.TabIndex = 6;
            this.comboBox2.Text = "115200";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.BackColor = System.Drawing.Color.LightGray;
            this.label3.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.label3.Location = new System.Drawing.Point(0, 113);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(150, 19);
            this.label3.TabIndex = 7;
            this.label3.Text = "BİT SATISI               ";
            //
            // comboBox3
            //
            this.comboBox3.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.comboBox3.FormattingEnabled = true;
            this.comboBox3.Items.AddRange(new object[] {
            "8",
            "9"});
            this.comboBox3.Location = new System.Drawing.Point(1, 135);
            this.comboBox3.Name = "comboBox3";
            this.comboBox3.Size = new System.Drawing.Size(148, 27);
            this.comboBox3.TabIndex = 8;
            this.comboBox3.Text = "8";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.BackColor = System.Drawing.Color.LightGray;
```

```
            this.label4.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.label4.Location = new System.Drawing.Point(1, 165);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(107, 19);
            this.label4.TabIndex = 9;
            this.label4.Text = "DURMA BİTİ";
            //
            // comboBox4
            //
            this.comboBox4.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.comboBox4.FormattingEnabled = true;
            this.comboBox4.Items.AddRange(new object[] {
            "1",
            "1.5",
            "2"});
            this.comboBox4.Location = new System.Drawing.Point(2, 187);
            this.comboBox4.Name = "comboBox4";
            this.comboBox4.Size = new System.Drawing.Size(147, 27);
            this.comboBox4.TabIndex = 10;
            this.comboBox4.Text = "1";
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.BackColor = System.Drawing.Color.LightGray;
            this.label5.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.label5.Location = new System.Drawing.Point(1, 217);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(102, 19);
            this.label5.TabIndex = 11;
            this.label5.Text = "PARİTE         ";
            //
            // comboBox5
            //
            this.comboBox5.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.comboBox5.FormattingEnabled = true;
            this.comboBox5.Items.AddRange(new object[] {
            "NONE",
            "ODD",
            "EVEN",
            "MARK",
            "SPACE"});
            this.comboBox5.Location = new System.Drawing.Point(2, 238);
            this.comboBox5.Name = "comboBox5";
            this.comboBox5.Size = new System.Drawing.Size(148, 27);
            this.comboBox5.TabIndex = 12;
            this.comboBox5.Text = "NONE";
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.BackColor = System.Drawing.Color.LightGray;
            this.label6.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.label6.Location = new System.Drawing.Point(1, 272);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(146, 19);
            this.label6.TabIndex = 13;
```

```
            this.label6.Text = "AKIŞ KONTROLÜ ";
            //
            // comboBox6
            //
            this.comboBox6.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.comboBox6.FormatString = "NONE";
            this.comboBox6.FormattingEnabled = true;
            this.comboBox6.Items.AddRange(new object[] {
            "NONE",
            "RTS-CTS"});
            this.comboBox6.Location = new System.Drawing.Point(3, 294);
            this.comboBox6.Name = "comboBox6";
            this.comboBox6.Size = new System.Drawing.Size(147, 27);
            this.comboBox6.TabIndex = 14;
            this.comboBox6.Text = "NONE";
            //
            // shapeContainer1
            //
            this.shapeContainer1.Location = new System.Drawing.Point(0, 0);
            this.shapeContainer1.Margin = new System.Windows.Forms.Padding(0);
            this.shapeContainer1.Name = "shapeContainer1";
            this.shapeContainer1.Shapes.AddRange(new
Microsoft.VisualBasic.PowerPacks.Shape[] {
            this.lineShape1});
            this.shapeContainer1.Size = new System.Drawing.Size(938, 563);
            this.shapeContainer1.TabIndex = 15;
            this.shapeContainer1.TabStop = false;
            //
            // lineShape1
            //
            this.lineShape1.BorderColor = System.Drawing.Color.Maroon;
            this.lineShape1.Name = "lineShape1";
            this.lineShape1.X1 = 156;
            this.lineShape1.X2 = 157;
            this.lineShape1.Y1 = 0;
            this.lineShape1.Y2 = 610;
            //
            // button3
            //
            this.button3.Font = new System.Drawing.Font("Times New Roman", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.button3.Location = new System.Drawing.Point(392, 11);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(92, 55);
            this.button3.TabIndex = 16;
            this.button3.Text = "DATALARI KAYDET";
            this.button3.UseVisualStyleBackColor = true;
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
            // panel1
            //
            this.panel1.BackColor = System.Drawing.Color.Lime;
            this.panel1.Controls.Add(this.button10);
            this.panel1.Controls.Add(this.button9);
            this.panel1.Controls.Add(this.button8);
            this.panel1.Controls.Add(this.button7);
            this.panel1.Controls.Add(this.button6);
            this.panel1.Controls.Add(this.button5);
            this.panel1.Controls.Add(this.button4);
            this.panel1.Controls.Add(this.button3);
            this.panel1.Location = new System.Drawing.Point(153, 498);
```

```
            this.panel1.Name = "panel1";
            this.panel1.Size = new System.Drawing.Size(785, 65);
            this.panel1.TabIndex = 17;
            //
            // button10
            //
            this.button10.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button10.Location = new System.Drawing.Point(98, 11);
            this.button10.Name = "button10";
            this.button10.Size = new System.Drawing.Size(92, 55);
            this.button10.TabIndex = 24;
            this.button10.Text = "KAS GERGİN";
            this.button10.UseVisualStyleBackColor = true;
            this.button10.Click += new System.EventHandler(this.button10_Click);
            //
            // button9
            //
            this.button9.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button9.Location = new System.Drawing.Point(690, 11);
            this.button9.Name = "button9";
            this.button9.Size = new System.Drawing.Size(92, 55);
            this.button9.TabIndex = 22;
            this.button9.Text = "SİSTEMİ KAPAT";
            this.button9.UseVisualStyleBackColor = true;
            this.button9.Click += new System.EventHandler(this.button9_Click);
            //
            // button8
            //
            this.button8.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button8.Location = new System.Drawing.Point(196, 11);
            this.button8.Name = "button8";
            this.button8.Size = new System.Drawing.Size(92, 55);
            this.button8.TabIndex = 21;
            this.button8.Text = "SİSTEMİ ÇALIŞTIR";
            this.button8.UseVisualStyleBackColor = true;
            this.button8.Click += new System.EventHandler(this.button8_Click);
            //
            // button7
            //
            this.button7.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button7.Location = new System.Drawing.Point(0, 11);
            this.button7.Name = "button7";
            this.button7.Size = new System.Drawing.Size(92, 55);
            this.button7.TabIndex = 20;
            this.button7.Text = "ÖLÇ YOLLA";
            this.button7.UseVisualStyleBackColor = true;
            this.button7.Click += new System.EventHandler(this.button7_Click);
            //
            // button6
            //
            this.button6.Font = new System.Drawing.Font("Times New Roman", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.button6.Location = new System.Drawing.Point(294, 11);
            this.button6.Name = "button6";
            this.button6.Size = new System.Drawing.Size(92, 55);
            this.button6.TabIndex = 19;
            this.button6.Text = "ÖRNEKLERİ TEMİZLE";
            this.button6.UseVisualStyleBackColor = true;
```

```
            this.button6.Click += new System.EventHandler(this.button6_Click);
            //
            // button5
            //
            this.button5.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button5.Location = new System.Drawing.Point(588, 11);
            this.button5.Name = "button5";
            this.button5.Size = new System.Drawing.Size(92, 55);
            this.button5.TabIndex = 18;
            this.button5.Text = "ÖRNEK SAYISI";
            this.button5.UseVisualStyleBackColor = true;
            this.button5.Click += new System.EventHandler(this.button5_Click);
            //
            // button4
            //
            this.button4.Font = new System.Drawing.Font("Times New Roman", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.button4.Location = new System.Drawing.Point(490, 11);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(92, 55);
            this.button4.TabIndex = 17;
            this.button4.Text = "DATALARI SİL";
            this.button4.UseVisualStyleBackColor = true;
            this.button4.Click += new System.EventHandler(this.button4_Click);
            //
            // textBox1
            //
            this.textBox1.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.textBox1.Location = new System.Drawing.Point(3, 352);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(146, 26);
            this.textBox1.TabIndex = 18;
            this.textBox1.Text = "1000";
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.BackColor = System.Drawing.Color.LightGray;
            this.label7.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.label7.Location = new System.Drawing.Point(1, 330);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(121, 19);
            this.label7.TabIndex = 19;
            this.label7.Text = "ÖRNEK SAYISI";
            //
            // textBox2
            //
            this.textBox2.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.textBox2.Location = new System.Drawing.Point(1, 415);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(150, 26);
            this.textBox2.TabIndex = 22;
            this.textBox2.Text = "emg_ornek";
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.BackColor = System.Drawing.Color.LightGray;
```

```
            this.label8.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
            this.label8.Location = new System.Drawing.Point(3, 393);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(89, 19);
            this.label8.TabIndex = 23;
            this.label8.Text = "KAYIT ADI";
            //
            // richTextBox1
            //
            this.richTextBox1.Location = new System.Drawing.Point(153, 0);
            this.richTextBox1.Name = "richTextBox1";
            this.richTextBox1.Size = new System.Drawing.Size(785, 503);
            this.richTextBox1.TabIndex = 21;
            this.richTextBox1.Text = "";
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.White;
            this.ClientSize = new System.Drawing.Size(938, 563);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.richTextBox1);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.panel1);
            this.Controls.Add(this.comboBox6);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.comboBox5);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.comboBox4);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.comboBox3);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.comboBox2);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.comboBox1);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.splitter1);
            this.Controls.Add(this.shapeContainer1);
            this.Name = "Form1";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "CAN EROL EMG SİGNAL PROCESSİNG";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.panel1.ResumeLayout(false);
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Splitter splitter1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.ComboBox comboBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
```

```
        private System.Windows.Forms.ComboBox comboBox2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.ComboBox comboBox3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.ComboBox comboBox4;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.ComboBox comboBox5;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.ComboBox comboBox6;
        private Microsoft.VisualBasic.PowerPacks.ShapeContainer shapeContainer1;
        private Microsoft.VisualBasic.PowerPacks.LineShape lineShape1;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.Button button6;
        private System.Windows.Forms.Button button5;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.SaveFileDialog saveFileDialog1;
        private System.IO.Ports.SerialPort serialPort1;
        private System.Windows.Forms.Button button7;
        private System.Windows.Forms.Button button8;
        private System.Windows.Forms.Button button9;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private System.Windows.Forms.Button button10;
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace EMGKayıtArayuzu
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Threading;
using System.IO;

namespace EMGKayıtArayuzu
{
    public partial class Form1 : Form
    {
        string portName = "";
        int baudRate = 115200;
        int dataBits = 8;
        Parity parity = Parity.None;
        StopBits stopBit = StopBits.One;
        Handshake handShake = Handshake.None;
        int characters_counter = 0;
        string temp_string = "";

        public Form1()
        {
            InitializeComponent();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            richTextBox1.Clear();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            string Save_File_Name = "";
            saveFileDialog1.InitialDirectory = "C:";
            saveFileDialog1.Title = "EMG Datalarını Kaydet";

            saveFileDialog1.FileName = textBox2.Text;
            saveFileDialog1.Filter = "Text Files|*.txt|All Files|*.*";
            if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)
            {
                Save_File_Name = saveFileDialog1.FileName;
                richTextBox1.SaveFile(Save_File_Name,
RichTextBoxStreamType.PlainText);
            }

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //BackColor = Color.White;
            TransparencyKey = Color.Lime;
            //FormBorderStyle = FormBorderStyle.None;
            foreach (string ComPorts in SerialPort.GetPortNames())
                comboBox1.Items.Add(ComPorts);

            serialPort1.DataReceived += new
SerialDataReceivedEventHandler(serialPort1_DataReceived);
        }

        private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
```

```
    // This event handler fires each time data is received by the serial port.
    // Read available data from the serial port and display it on the form.
    // This event does not run in the UI thread, so need to
    // use delegate function

    string strErg = serialPort1.ReadExisting();
    this.BeginInvoke(new EventHandler(delegate
    {
        temp_string = temp_string + strErg;
        SetTheText(temp_string);
        characters_counter = 0;
        temp_string = null;


    }));
    Application.DoEvents();
}

private void SetTheText(string strText)
{
    richTextBox1.Text += strText;
}

private void button1_Click(object sender, EventArgs e)
{
    portName = comboBox1.Text;

    if (portName != null)
    {
        baudRate = Convert.ToInt32(comboBox2.Text);
        dataBits = Convert.ToInt32(comboBox3.Text);

        //Set Stop Bits
        switch (comboBox4.SelectedIndex)
        {
            case 0: stopBit = StopBits.One; break;
            case 1: stopBit = StopBits.OnePointFive; break;
            case 2: stopBit = StopBits.Two; break;
            default: break;
        }
        //Set Parity
        switch (comboBox5.Text)
        {
            case "EVEN": parity = Parity.Even; break;
            case "MARK": parity = Parity.Mark; break;
            case "NONE": parity = Parity.None; break;
            case "ODD": parity = Parity.Odd; break;
            case "SPACE": parity = Parity.Space; break;
            default: break;
        }
        //Set Flow Control
        switch (comboBox6.Text)
        {
            case "NONE": handShake = Handshake.None; break;
            case "RTS-CTS": handShake = Handshake.RequestToSend; break;
            default: break;
        }

        serialPort1.PortName = portName;
        serialPort1.DataBits = dataBits;
        serialPort1.Parity = parity;
        serialPort1.StopBits = stopBit;
```

```
                serialPort1.BaudRate = baudRate;
                serialPort1.Handshake = handShake;

                // Try to open the serial port
                try
                {
                    serialPort1.Open();
                }
                catch (System.Exception)
                {
                    MessageBox.Show("Failed to open selected port.\nIs it open in
another application?\n(" + ")");
                }
                finally
                {
                    // If we have successfully opened the port, enable/disable buttons
                    if (serialPort1.IsOpen)
                    {
                        serialPort1.RtsEnable = true;
                        serialPort1.DtrEnable = true;

                        // Set write timeout
                        serialPort1.WriteTimeout = 5000;
                    }
                }
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            char[] SendBuffer = new char[11];

            SendBuffer[0] = 'C';
            SendBuffer[1] = 'A';
            SendBuffer[2] = 'N';
            SendBuffer[3] = '@';
            SendBuffer[4] = 'S';
            SendBuffer[5] = 'A';
            SendBuffer[6] = 'Y';
            SendBuffer[7] = textBox1.Text[0];
            SendBuffer[8] = textBox1.Text[1];
            SendBuffer[9] = textBox1.Text[2];
            SendBuffer[10] = textBox1.Text[3];

            try
            {
                serialPort1.Write(SendBuffer, 0, 11);
            }
            catch (System.Exception)
            {
                MessageBox.Show("FAILED TO WRITE TO PORT");
            }

        }

        private void button6_Click(object sender, EventArgs e)
        {
```

```
    char[] SendBuffer = new char[11];

    SendBuffer[0] = 'C';
    SendBuffer[1] = 'A';
    SendBuffer[2] = 'N';
    SendBuffer[3] = '@';
    SendBuffer[4] = 'B';
    SendBuffer[5] = 'T';
    SendBuffer[6] = 'L';
    SendBuffer[7] = '0';
    SendBuffer[8] = '0';
    SendBuffer[9] = '0';
    SendBuffer[10] = '0';

    try
    {
        serialPort1.Write(SendBuffer, 0, 11);
    }
    catch (System.Exception)
    {
        MessageBox.Show("FAILED TO WRITE TO PORT");
    }
}

private void button7_Click(object sender, EventArgs e)
{
    char Text;
    int counter = 0;
    char[] SendBuffer = new char[11];


    SendBuffer[0] = 'C';
    SendBuffer[1] = 'A';
    SendBuffer[2] = 'N';
    SendBuffer[3] = '@';
    SendBuffer[4] = 'O';
    SendBuffer[5] = 'L';
    SendBuffer[6] = 'Y';
    SendBuffer[7] = '0';
    SendBuffer[8] = '0';
    SendBuffer[9] = '0';
    SendBuffer[10] = '0';


    try
    {
        serialPort1.Write(SendBuffer, 0, 11);
    }
    catch (System.Exception)
    {
        MessageBox.Show("FAILED TO WRITE TO PORT");
    }

    //serialPort1.DiscardOutBuffer();
    //Text = serialPort1.ReadLine();

   /* while (counter < 9000)
    {
        Text = (char)serialPort1.ReadChar();
        richTextBox1.Text = richTextBox1.Text + Text;
        counter++;
    }*/
```

```
}

private void button8_Click(object sender, EventArgs e)
{
    char[] SendBuffer = new char[11];

    SendBuffer[0] = 'C';
    SendBuffer[1] = 'A';
    SendBuffer[2] = 'N';
    SendBuffer[3] = '@';
    SendBuffer[4] = 'S';
    SendBuffer[5] = 'C';
    SendBuffer[6] = 'L';
    SendBuffer[7] = '0';
    SendBuffer[8] = '0';
    SendBuffer[9] = '0';
    SendBuffer[10] = '0';

    try
    {
        serialPort1.Write(SendBuffer, 0, 11);
    }
    catch (System.Exception)
    {
        MessageBox.Show("FAILED TO WRITE TO PORT");
    }
}

private void button9_Click(object sender, EventArgs e)
{
    char[] SendBuffer = new char[11];

    SendBuffer[0] = 'C';
    SendBuffer[1] = 'A';
    SendBuffer[2] = 'N';
    SendBuffer[3] = '@';
    SendBuffer[4] = 'S';
    SendBuffer[5] = 'K';
    SendBuffer[6] = 'T';
    SendBuffer[7] = '0';
    SendBuffer[8] = '0';
    SendBuffer[9] = '0';
    SendBuffer[10] = '0';

    try
    {
        serialPort1.Write(SendBuffer, 0, 11);
    }
    catch (System.Exception)
    {
        MessageBox.Show("FAILED TO WRITE TO PORT");
    }
}

private void button10_Click(object sender, EventArgs e)
{
    char[] SendBuffer = new char[11];

    SendBuffer[0] = 'C';
    SendBuffer[1] = 'A';
    SendBuffer[2] = 'N';
    SendBuffer[3] = '@';
```

```
            SendBuffer[4] = 'K';
            SendBuffer[5] = 'S';
            SendBuffer[6] = 'Y';
            SendBuffer[7] = '0';
            SendBuffer[8] = '0';
            SendBuffer[9] = '0';
            SendBuffer[10] = '0';

            try
            {
                serialPort1.Write(SendBuffer, 0, 11);
            }
            catch (System.Exception)
            {
                MessageBox.Show("FAILED TO WRITE TO PORT");
            }
        }

    }
}
```

EK-C

GÖMÜLÜ YAZILIM DEMO KODLARI

TimerSetting.c

```
#ifndef _TimerGeneralSetting_C_
#define _TimerGeneralSetting_C_

#include "TimerGeneralSetting.h"
#include "ADC.h"

extern tBoolean SendOK;
extern tBoolean MusculesOnOf;
extern int LedShift;
extern int PointCount;
extern tBoolean OpOnce;
extern tBoolean SendComputer;
extern double StoreADCSample[1000];
unsigned long ulADC0_Value;
extern long SampleSize;
extern double* PointsForHMM;
extern int SampleCounterTM;
unsigned long timer0_flag = 0;
extern tBoolean StartGetSample;
extern tBoolean OperateSystem;


void Timer0IntHandler(void)
{
  ROM_TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
  //ROM_TimerDisable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
  //
  // Trigger the ADC conversion.
  //
  ROM_ADCProcessorTrigger(ADC0_BASE, 0);

  //
  // Wait for conversion to be completed.
  //
  while(!ROM_ADCIntStatus(ADC0_BASE, 0, false))
  {
  }

  //
  // Read ADC Value.
  //
  ROM_ADCSequenceDataGet(ADC0_BASE, 0, &ulADC0_Value);
```

```
        //
// Clear the timer interrupt.
//
        if(CalculateADC(ulADC0_Value) <= MaxADCValue)
        {
                //
                // Store ADC Sample
                //
                StoreADCSample[timer0_flag] = (double)((double) (ulADC0_Value)  * 3 / 1024);
}
        else
        {
                //
                // Store ADC Sample
                //
                StoreADCSample[timer0_flag] = MaxADCValue;
        }

        //
// Toggle the flag for the first timer.
//
timer0_flag++;



        //
        // Check timer flag
        //
        if(timer0_flag == SampleSize)
        {
         ROM_TimerDisable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
         ROM_TimerIntDisable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
         if(OperateSystem == true)
         {
                 //SendADCSampledDatas();
                 GetPointsOfEMG();
                 GetPoints();
                 GetResultsFromHMMTable();
           ROM_TimerEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
                 ROM_TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
         }
         else if(SendComputer == true)
         {
                 SendADCSampledDatas();
                 CloseTimer();
         }
```

```
            else if(MusculesOnOf == true)
            {
                    GetMediumAndShiftLed();
                    ROM_TimerEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
                    ROM_TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
            }
            //
            // Clear timer0 flag
            //
            timer0_flag = 0;

            if(OpOnce == true)
            {
                    //
                    // Close Timer
                    //
                    CloseTimer();

                    //
                    // Clear OpOnce
                    //
                    OpOnce = false;

                    //
                    // Clear operate system
                    //
                    OperateSystem = false;

                    //
                    // Clear Send Computer
                    //
                    SendComputer = false;
    }

          }

}

extern void
TimerInitialize(void)
{
 ROM_IntEnable(INT_TIMER0A);
 ROM_TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
 ROM_TimerEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
 //ROM_IntMasterEnable();
```

```
 StartGetSample = true;
}

extern void
CloseTimer(void)
{
        timer0_flag = 0;

        OperateSystem = false;

        MusculesOnOf = false;

        StartGetSample = false;

  ROM_TimerDisable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);

  ROM_TimerIntDisable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
}

#endif
```

TimerSetting.h

```
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/adc.h"
#include "driverlib/rom.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"

void ADCInitialize(void);
```

ADC.c

```
#include "ADC.h"

//! - GPIO Port E peripheral (for ADC0 pins)
//! - AIN0 - PE7
//! - AIN1 - PE
extern void
ADCInitialize(void)
{
  ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
  ROM_SysCtlPeripheralReset(SYSCTL_PERIPH_ADC0);
  ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
  GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_3);
```

```
  ROM_ADCSequenceConfigure(ADC0_BASE, 0, ADC_TRIGGER_PROCESSOR, 0);
  ROM_ADCSequenceStepConfigure(ADC0_BASE, 0, 0, ADC_CTL_CH8 | ADC_CTL_IE |
ADC_CTL_END);
  ROM_ADCSequenceEnable(ADC0_BASE, 0);

}
```

ADC.h

```
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/adc.h"
#include "driverlib/rom.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"


void ADCInitialize(void);
```

GeneralProtocolFunctions.c

```
#ifndef _GeneralProtocolFunctions_C_
#define _GeneralProtocolFunctions_C_

#include "GeneralProtocolFunctions.h"
#include "driverlib\rom.h"
#include "driverlib\sysctl.h"
#include "HiddenMarkovModelTable.h"
#include "inc\hw_types.h"
#include <RTL.h>


tBoolean SendOK = false;
tBoolean MusculesOnOf = false;
int LedShift = 0;
int PointCount = 0;
double* PointsForHMM;
int SampleCounterTM = 0;
double StoreADCSample[1000];
unsigned long SampleSize = 1000;
char dataBuf[PACKET];
unsigned long dBuf_counter = 0;

tBoolean StartGetSample = false;
```

```
tBoolean SendComputer = false;
tBoolean OperateSystem = false;
tBoolean OpOnce = false;


void
GPIO_Configuration(void)
{
        //
        // Define default USB configuration status
        //
        g_bUSBConfigured = false;

        //
        // Set the clocking to run from the PLL at 80 MHz.
        //
        ROM_SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
SYSCTL_OSC_MAIN);

        //
  // Enable GPIO port A which is used for UART0 pins.
  // TODO: change this to whichever GPIO port you are using.
  //
        ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

        ROM_GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_2);
        ROM_GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_3);
        ROM_GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_4);
        ROM_GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_5);
        ROM_GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_6);

        ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2, GPIO_PIN_2);

  //
  // Initialize the UART for console I/O".
  //
        ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);

  //
  // Configure the pin muxing for UART0 functions on port A0 and A1.
  // This step is not necessary if your part does not support pin muxing.
  // TODO: change this to select the port/pin you are using.
  //
  GPIOPinConfigure(GPIO_PA0_U0RX);
  GPIOPinConfigure(GPIO_PA1_U0TX);
```

```
        //
    // Set the GPIO pins to UART functionality.
    // TODO: change this to select the ports/pins you are using.
    //
    ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_1 | GPIO_PIN_0);


        //
    // Enable the peripherals used by this example.
    //
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);

    //
    // Configure Timer0B as a 16-bit periodic timer.
    //
        ROM_TimerConfigure(TIMER0_BASE, TIMER_CFG_16_BIT_PAIR | TIMER_CFG_A_PERIODIC);

        //
    // Set the Timer0A load value to 1ms.
    //
    ROM_TimerLoadSet(TIMER0_BASE, TIMER_A, ROM_SysCtlClockGet()/1000);

    //
    // Set Prascale with using starting value as 10
    //
        //ROM_TimerPrescaleSet(TIMER0_BASE, TIMER_A, 10);

    //
    // Enable processor interrupts.
    //
        ROM_IntMasterEnable();

        //
    // Configure the Timer0B interrupt for timer timeout.
    //
    ROM_TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);

}

//*****************************************************************************
//
// The Buffer Clear function is called after of (USB - UART)
// data recieving and sending operation.
//
```

```
//*****************************************************************************
void
Buffer_Clear(char* buffer)
{
	int size;

	//
	// Clear buffer with using for loop and null characters
	//
	for	(size = 0; size < PACKET; size++)
	{
		 *(buffer + size) = '\0';			//Clearing buffer with this loop
	}

}

//*****************************************************************************
//
// Select sending response type with using the Sending_Response function.
//
//*****************************************************************************
void
Sending_Response(char *message, char way, long size)
{
	//
	// Select command sending way
	//
	switch(way)
	{
		//
		// If way is 'U' ,send via USB
		//
		case USB:
		USBSend(message, size);
		break;

		//
		// If way is 'S' ,send via UART
		//
		case SERIAL:
		UARTSend(message, size);
		break;

		default:
		break;
```

```
        }
}

//*****************************************************************************
//
// The decode_direction          function is used to decode direction end do this
// direction.
//
//*****************************************************************************
void
decode_direction(char* dBuf, char way)
{
        int command_index_counter = 0;

        //
        // Decode Command
        //
        while(command_index_counter < COMMAND_HEADER_SIZE)
        {
                //
                // Store command in command structure
                //
                *(command.header + command_index_counter) = *(dBuf +
command_index_counter);

                //
                // Increase command index counter
                //
                command_index_counter++;
        }

        //
        // Find command input and output number
        //
        command.inputnum = ((*(dBuf + 7) - '0') * 10) +  (*(dBuf + 8) - '0');
        command.outputnum = ((*(dBuf + 9) - '0') * 10) +  (*(dBuf + 10) - '0');

        //
        // Clear command counter
        //
        command_index_counter = 0;

        //
        // Clerar dBuf
        //
```

```
Buffer_Clear(dataBuf);

//
// Check command from command array
//
if(strstr(command.header, direction_table[0]) != 0 )
{
        //
        // Measure And Send
        //
        MeasureAndSend();
}
else if(strstr(command.header, direction_table[1]) != 0 )
{
//
        // Operate System
        //
   OperateSystemFunction();
}
else if(strstr(command.header, direction_table[2]) != 0 )
{
//
        // Close System
        //
        CloseSystem();
}
else if(strstr(command.header, direction_table[3]) != 0 )
{
//
        // Clear Buffer
        //
        ClearOutputs();
}
else if(strstr(command.header, direction_table[4]) != 0 )
{
//
        // Operate System
        //
        OperateOnceTime();
}
else if(strstr(command.header, direction_table[5]) != 0 )
{
//
        // Change Sample Size
        //
```

```
                ChangeSampleSize(command.inputnum * 100 + command.outputnum, way);
        }
        else if(strstr(command.header, direction_table[6]) != 0 )
        {
        //
                // Send Ok true
                //
                SendOK = true;
        }
        else if(strstr(command.header, direction_table[7]) != 0 )
        {
        //
                // Check Muscules And Shift Lamps
                //
                MusculesOnOffStatusCheck();
        }
        else
        {
           ERR(way);
        }

}

//*****************************************************************************
//
// Error Sending Function
//
//*****************************************************************************
void
ERR(char way)
{
  Sending_Response("CAN@ERR0001", way, COMMAND_SIZE);
}

void
SendDataToComputerFromCard(double data, char way)
{
  char SendDataInfo[9];

  sprintf(SendDataInfo, "%.6f\n", data);

  Sending_Response(SendDataInfo, way, 9);

}
```

```
//*****************************************************************************
//
// Change Sample Size
//
//*****************************************************************************
void
ChangeSampleSize(long value, char way)
{
  if(StartGetSample == false)
  {
            SampleSize = value;
  }
  else
  {
            ERR(way);
  }
}

//*****************************************************************************
//
// ADC Bilgilerini Yolla
//
//*****************************************************************************
void
SendADCSampledDatas(void)
{
 while( SampleCounterTM < SampleSize)
 {

            SendDataToComputerFromCard(StoreADCSample[SampleCounterTM], USB);
            SampleCounterTM++;

            SysCtlDelay(10000);
 }
 SampleCounterTM = 0;
}

//*****************************************************************************
//
// Clear Outputs
//
//*****************************************************************************
extern void
ClearOutputs(void)
{
```

```
 static int ClearCounter;

 if(StartGetSample == false)
 {
        for(ClearCounter = 0; ClearCounter < SampleSize; ClearCounter++)
        {
          *(StoreADCSample + ClearCounter) = 0.00;
        }
 }
 else
 {
          ERR(USB);
 }
}

//*****************************************************************************
//
// Operate System As Once
//
//*****************************************************************************
extern void
OperateOnceTime(void)
{
 if(StartGetSample == false)
 {
   OpOnce = true;
        OperateSystem = true;
        TimerInitialize();
 }
 else
 {
        ERR(USB);
 }
}

//*****************************************************************************
//
// Operate System As Once
//
//*****************************************************************************
extern void
MeasureAndSend(void)
{
 if(StartGetSample == false)
 {
```

```
   OpOnce = true;
        SendComputer = true;
        TimerInitialize();
 }
 else
 {
        ERR(USB);
 }
}

//*****************************************************************************
//
// Operate System As Once
//
//*****************************************************************************
extern void
OperateSystemFunction(void)
{
 if(StartGetSample == false)
 {
   OperateSystem = true;
        TimerInitialize();
 }
 else
 {
        ERR(USB);
 }
}

//*****************************************************************************
//
// Close System
//
//*****************************************************************************
extern void
CloseSystem(void)
{
 StartGetSample = false;

 ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_3, 0x00);
 ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4, 0x00);
 ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5, 0x00);
 ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6, 0x00);
 CloseTimer();
```

```
}

//*****************************************************************************
//
// Check Muscules Status And Shift Leds
//
//*****************************************************************************
extern void
MusculesOnOffStatusCheck(void)
{
 if(StartGetSample == false)
 {
  MusculesOnOf = true;
        TimerInitialize();
 }
 else
 {
        ERR(USB);
 }
}

extern void
GetMediumAndShiftLed(void)
{
  double Medium = 0.0;
  int counter = 0;

  for( counter = 0; counter < 1000; counter++)
  {
         Medium = Medium + StoreADCSample[counter];
  }

  Medium = Medium / 1000;

  ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_3, 0x00);
  ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4, 0x00);
  ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5, 0x00);
  ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6, 0x00);
  SysCtlDelay(1000);
  if(Medium > 0.017)
  {
        LedShift++;

        if(LedShift == 5)
        {
```

```
            LedShift = 1;
          }

          if(LedShift == 1)
          {
           ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_3, GPIO_PIN_3);
          }
          else if(LedShift == 2)
          {
           ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4, GPIO_PIN_4);
          }
          else if(LedShift == 3)
          {
           ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5, GPIO_PIN_5);
          }
          else if(LedShift == 4)
          {
           ROM_GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6, GPIO_PIN_6);
          }
  }

  SysCtlDelay(1000);
}
#endif
```

Startup.s

```
;*******************************************************************************
;
; <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
;
;*******************************************************************************
Stack   EQU     0x00000400

;*******************************************************************************
;
; <o> Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
;
;*******************************************************************************
Heap    EQU     0x00000000

;*******************************************************************************
;
; Allocate space for the stack.
;
```

```
;*******************************************************************************
    AREA    STACK, NOINIT, READWRITE, ALIGN=3
StackMem
    SPACE   Stack
__initial_sp

;*******************************************************************************
;
; Allocate space for the heap.
;
;*******************************************************************************
    AREA    HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
HeapMem
    SPACE   Heap
__heap_limit

;*******************************************************************************
;
; Indicate that the code in this file preserves 8-byte alignment of the stack.
;
;*******************************************************************************
    PRESERVE8

;*******************************************************************************
;
; Place code into the reset code section.
;
;*******************************************************************************
    AREA    RESET, CODE, READONLY
    THUMB

;*******************************************************************************
;
; External declarations for the interrupt handlers used by the application.
;
;*******************************************************************************
    EXTERN  USB0DeviceIntHandler
    EXTERN  USBUARTIntHandler
    EXTERN  UART0_IRQHandler
    EXTERN  Timer0IntHandler

;*******************************************************************************
;
; The vector table.
```

```
;
;*******************************************************************************
    EXPORT  __Vectors
__Vectors
    DCD    StackMem + Stack          ; Top of Stack
    DCD    Reset_Handler             ; Reset Handler
    DCD    NmiSR                     ; NMI Handler
    DCD    FaultISR                  ; Hard Fault Handler
    DCD    IntDefaultHandler         ; The MPU fault handler
    DCD    IntDefaultHandler         ; The bus fault handler
    DCD    IntDefaultHandler         ; The usage fault handler
    DCD    0                   ; Reserved
    DCD    0                   ; Reserved
    DCD    0                   ; Reserved
    DCD    0                   ; Reserved
    DCD    IntDefaultHandler         ; SVCall handler
    DCD    IntDefaultHandler         ; Debug monitor handler
    DCD    0                   ; Reserved
    DCD    IntDefaultHandler         ; The PendSV handler
    DCD    IntDefaultHandler         ; The SysTick handler
    DCD    IntDefaultHandler         ; GPIO Port A
    DCD    IntDefaultHandler         ; GPIO Port B
    DCD    IntDefaultHandler         ; GPIO Port C
    DCD    IntDefaultHandler         ; GPIO Port D
    DCD    IntDefaultHandler         ; GPIO Port E
    DCD    UART0_IRQHandler          ; UART0 Rx and Tx
    DCD    USBUARTIntHandler         ; UART1 Rx and Tx
    DCD    IntDefaultHandler         ; SSI0 Rx and Tx
    DCD    IntDefaultHandler         ; I2C0 Master and Slave
    DCD    IntDefaultHandler         ; PWM Fault
    DCD    IntDefaultHandler         ; PWM Generator 0
    DCD    IntDefaultHandler         ; PWM Generator 1
    DCD    IntDefaultHandler         ; PWM Generator 2
    DCD    IntDefaultHandler         ; Quadrature Encoder 0
    DCD    IntDefaultHandler         ; ADC Sequence 0
    DCD    IntDefaultHandler         ; ADC Sequence 1
    DCD    IntDefaultHandler         ; ADC Sequence 2
    DCD    IntDefaultHandler         ; ADC Sequence 3
    DCD    IntDefaultHandler         ; Watchdog timer
    DCD    Timer0IntHandler          ; Timer 0 subtimer A
    DCD    IntDefaultHandler         ; Timer 0 subtimer B
    DCD    IntDefaultHandler         ; Timer 1 subtimer A
    DCD    IntDefaultHandler         ; Timer 1 subtimer B
    DCD    IntDefaultHandler         ; Timer 2 subtimer A
    DCD    IntDefaultHandler         ; Timer 2 subtimer B
```

```
    DCD    IntDefaultHandler        ; Analog Comparator 0
    DCD    IntDefaultHandler        ; Analog Comparator 1
    DCD    IntDefaultHandler        ; Analog Comparator 2
    DCD    IntDefaultHandler        ; System Control (PLL, OSC, BO)
    DCD    IntDefaultHandler        ; FLASH Control
    DCD    IntDefaultHandler        ; GPIO Port F
    DCD    IntDefaultHandler        ; GPIO Port G
    DCD    IntDefaultHandler        ; GPIO Port H
    DCD    IntDefaultHandler        ; UART2 Rx and Tx
    DCD    IntDefaultHandler        ; SSI1 Rx and Tx
    DCD    IntDefaultHandler        ; Timer 3 subtimer A
    DCD    IntDefaultHandler        ; Timer 3 subtimer B
    DCD    IntDefaultHandler        ; I2C1 Master and Slave
    DCD    IntDefaultHandler        ; Quadrature Encoder 1
    DCD    IntDefaultHandler        ; CAN0
    DCD    IntDefaultHandler        ; CAN1
    DCD    IntDefaultHandler        ; CAN2
    DCD    IntDefaultHandler        ; Ethernet
    DCD    IntDefaultHandler        ; Hibernate
    DCD    USB0DeviceIntHandler     ; USB0
    DCD    IntDefaultHandler        ; PWM Generator 3
    DCD    IntDefaultHandler        ; uDMA Software Transfer
    DCD    IntDefaultHandler        ; uDMA Error
    DCD    IntDefaultHandler        ; ADC1 Sequence 0
    DCD    IntDefaultHandler        ; ADC1 Sequence 1
    DCD    IntDefaultHandler        ; ADC1 Sequence 2
    DCD    IntDefaultHandler        ; ADC1 Sequence 3
    DCD    IntDefaultHandler        ; I2S0
    DCD    IntDefaultHandler        ; External Bus Interface 0
    DCD    IntDefaultHandler        ; GPIO Port J

;*******************************************************************************
;
; This is the code that gets called when the processor first starts execution
; following a reset event.
;
;*******************************************************************************
    EXPORT  Reset_Handler
Reset_Handler
    ;
    ; Call the C library enty point that handles startup.  This will copy
    ; the .data section initializers from flash to SRAM and zero fill the
    ; .bss section.
    ;
    IMPORT  __main
```

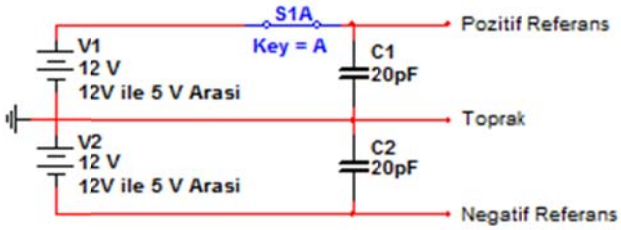
```
    B      __main

;*******************************************************************************
;
; This is the code that gets called when the processor receives a NMI.  This
; simply enters an infinite loop, preserving the system state for examination
; by a debugger.
;
;*******************************************************************************
NmiSR
    B      NmiSR

;*******************************************************************************
;
; This is the code that gets called when the processor receives a fault
; interrupt.  This simply enters an infinite loop, preserving the system state
; for examination by a debugger.
;
;*******************************************************************************
FaultISR
    B      FaultISR

;*******************************************************************************
;
; This is the code that gets called when the processor receives an unexpected
; interrupt.  This simply enters an infinite loop, preserving the system state
; for examination by a debugger.
;
;*******************************************************************************
IntDefaultHandler
    B      IntDefaultHandler

;*******************************************************************************
;
; Make sure the end of this section is aligned.
;
;*******************************************************************************
    ALIGN

;*******************************************************************************
;
; Some code in the normal code section for initializing the heap and stack.
;
;*******************************************************************************
    AREA    |.text|, CODE, READONLY
```

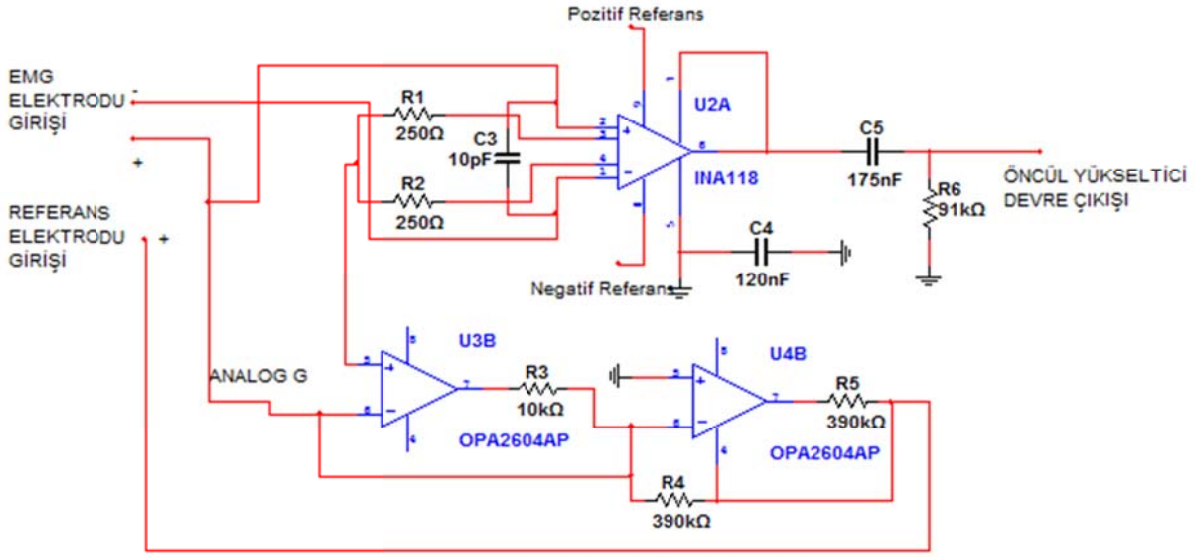
```
;*******************************************************************************
;
; The function expected of the C library startup code for defining the stack
; and heap memory locations.  For the C library version of the startup code,
; provide this function so that the C library initialization code can find out
; the location of the stack and heap.
;
;*******************************************************************************
  IF :DEF: __MICROLIB
    EXPORT  __initial_sp
    EXPORT  __heap_base
    EXPORT  __heap_limit
  ELSE
    IMPORT  __use_two_region_memory
    EXPORT  __user_initial_stackheap
__user_initial_stackheap
    LDR    R0, =HeapMem
    LDR    R1, =(StackMem + Stack)
    LDR    R2, =(HeapMem + Heap)
    LDR    R3, =StackMem
    BX     LR
  ENDIF

;*******************************************************************************
;
; Make sure the end of this section is aligned.
;
;*******************************************************************************
    ALIGN

;*******************************************************************************
;
; Tell the assembler that we're done.
;
;*******************************************************************************
    END
```

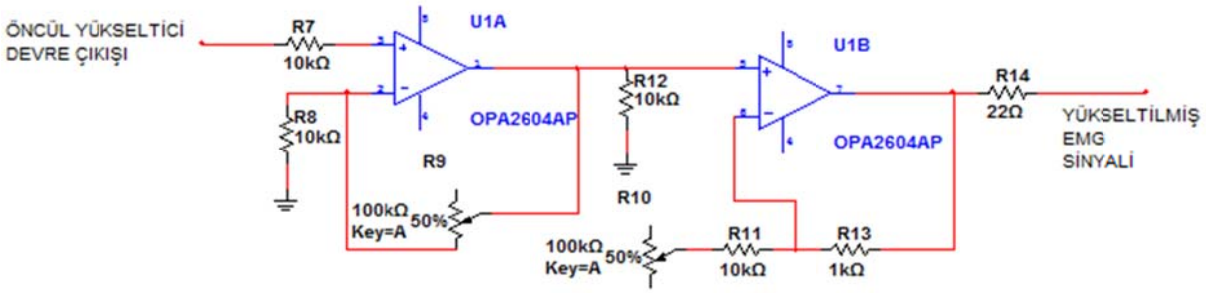
EK-D

EMG YÜKSELTİCİ DEVRENİN ŞEMASI VE DEVRE DİYAGRAMI

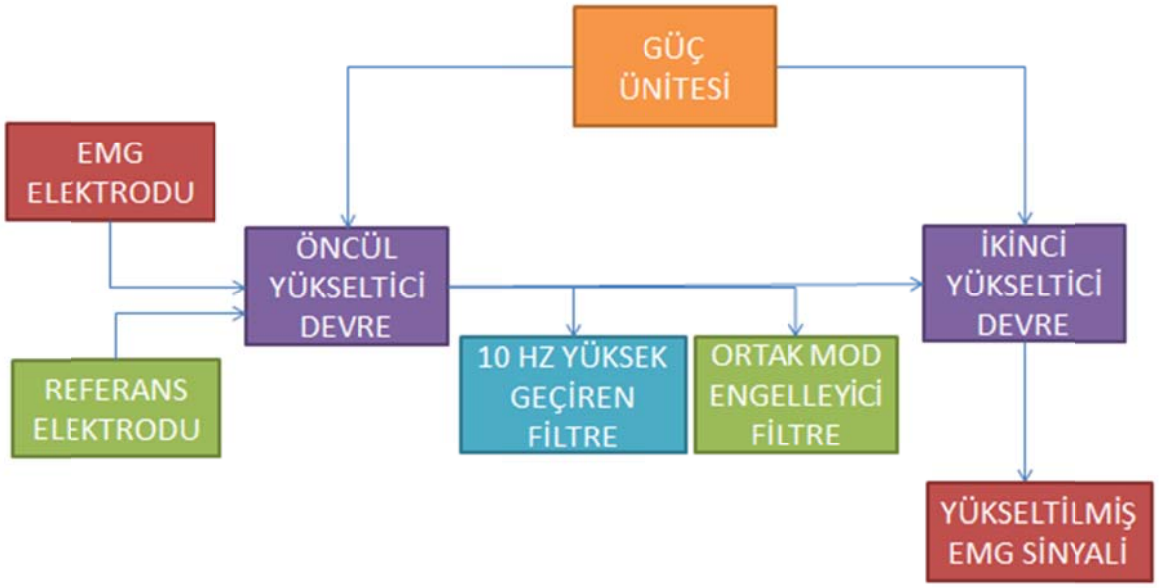
EMG YÜKSELTİCİ DEVRE ŞEMASI







DEVRE DİYAGRAMI



ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Can EROL

Doğum Tarihi ve Yeri : 28.07.1986, Şişli

Yabancı Dili : İngilizce, Almanca

E-posta : canerol2004@yahoo.com

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Y. Lisans	Elektronik	Yıldız Teknik Üniversitesi	-
Lisans	Elektronik Mühendisliği	Fatih Üniversitesi	2009
Lise	Fen	Bahçelievler Anadolu Lisesi	2004

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2011	EDT Elektronik	Yazılım Mühendisi
2009	Farimex S.A.	Yazılım Mühendisi
2008	Fatih Üniversitesi	Öğrenci Asistanı

Bildiri

1. Erol, C., ÖZYILMAZ, E., (2012). "EMG Sinyallerinin Gizli Markov Modeli ile Sınıflandırılması", ASYU 2012, sf. 182-186, 3-4 Temmuz 2012, Trabzon