T.C.

HİTİT ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ENERJİ SİSTEMLERİ MÜHENDİSLİĞİ ANABİLİM DALI

ROBOTİK SİSTEMLERDE GÖRÜNTÜ İŞLEME TABANLI NESNE TANIMA İÇİN AKILLI ORTAM AYDINLATMASI

Yüksek Lisans

Uğur AKIŞ

Çorum - 2023

ROBOTİK SİSTEMLERDE GÖRÜNTÜ İŞLEME TABANLI NESNE TANIMA İÇİN AKILLI ORTAM AYDINLATMASI

Uğur AKIŞ

Lisansüstü Eğitim Enstitüsü

Enerji Sistemleri Mühendisliği Anabilim Dalı

Yüksek Lisans

TEZ DANIŞMANI

Dr. Öğr. Üyesi Serkan DİŞLİTAŞ

Çorum 2023

Uğur AKIŞ tarafından hazırlanan "Robotik Sistemlerde Görüntü İşleme Tabanlı Nesne Tanıma İçin Akıllı Ortam Aydınlatması" adlı tez çalışması 20/01/2023 tarihinde aşağıdaki jüri üyeleri tarafından oy birliği ile Hitit Üniversitesi Lisansüstü Eğitim Enstitüsü Enerji Sistemleri Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Doç. Dr. Hayati MAMUR ……

Jüri Başkanı

Dr. Öğr. Üyesi Serkan DİŞLİTAŞ ……

Üye (Danışman)

Dr. Öğr. Üyesi Murat Alparslan GÜNGÖR ……

Üye

Hitit Üniversitesi Lisansüstü Eğitim Enstitüsü Yönetim Kurulunun …/…/…… tarih ve …….. sayılı kararı ile Uğur AKIŞ'ın Enerji Sistemleri Mühendisliği Anabilim Dalında Yüksek Lisans derecesi alması onanmıştır.

Prof. Dr. Muhammed Asıf YOLDAŞ

Lisansüstü Eğitim Enstitüsü Müdür V.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını beyan ederim.

Uğur AKIŞ

ROBOTİK SİSTEMLERDE GÖRÜNTÜ İŞLEME TABANLI

NESNE TANIMA İÇİN AKILLI ORTAM AYDINLATMASI

Uğur AKIŞ

ORCID: 0000-0001-7095-7770

HİTİT ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

Yüksek Lisans

Ocak 2023

ÖZET

Görüntü işleme tabanlı uygulamalarda ortam aydınlatması, nesnelerin doğru tespit edilmesi ve enerji kullanımı açısından önemli bir husustur. Klasik yöntemlerde ortam aydınlatması ışık kaynağı sürekli açık tutularak, ihtiyaca göre elle açılıp kapatılarak, zamanlayıcı kullanarak veya çeşitli sensörler yardımıyla aydınlık düzeyi ölçülerek otomatik olarak yapılmaktadır. Bu tez çalışmasında, robotik elleçleme uygulamalarına yönelik nesne tespiti ve pozisyon belirlenmesi amacıyla görüntü işlemeye dayalı referans-renk-tabanlı akıllı ortam aydınlatması yapan bir gömülü sistem tasarımı yapılmış ve performansı incelenmiştir. Sistemde asgari aydınlık düzeyinin elde edilmesi için harici bir sensöre ihtiyaç duyulmadan pozisyonları daha önceden belirlenmiş kırmızı, yeşil, mavi ve sarı renkli nesneler referans alınarak aydınlatma sisteminin gücü ayarlanmaktadır. Tasarlanan sistemde görüntü işleme, aydınlatma ve 3-eksen elleçleme işlemleri için Raspberry Pi 4B mini bilgisayar ve GRBL yazılım yüklü Arduino CNC shield kartı kullanılmıştır. Görüntü alma işlemi Raspberry Pi kamera modülü üzerinden, nesnelerin elleçlenmesi ise bir elektromıknatıs ile yapılmıştır. Aydınlatma için PWM sinyali ile sürülen 12V DC beyaz şerit LED lamba kullanılmış ve aydınlık düzeyi oransal olarak kontrol edilerek akıllı ortam aydınlatması yapılmıştır. Sistem arayüz yazılımı Python dilinde kodlanmıştır ve görüntü işleme ile nesne tanıma ve pozisyon belirlenme için OpenCV kütüphanesi kullanılmıştır. Tasarlanan sistemin performansının incelenmesine yönelik deneysel çalışmalar kapsamında beyaz ışık ile aydınlatma yapılmış, platform zemini için yansıma yapmayan beyaz mat renkli plastik fon kullanılmış ve platform üzeri siyah örtü ile örtülmüştür. Renk tabanlı görüntü işleme ile nesne tespitinde Kırmızı, Yeşil, Mavi ve Sarı olmak üzere farklı renklere sahip referans nesneler kullanılarak gerekli olan minimum platform aydınlık düzeyi değerleri

araştırılmıştır. Elde edilen deneysel sonuçlara göre; tespit işleminde renk seçimin önemli olduğu anlaşılmış ve bu renkler içerisinde en az aydınlık düzeyine ihtiyaç duyulan rengin sarı olduğu tespit edilmiştir. Sarı rengi ise sırasıyla, yeşil, kırmızı ve mavi renklerin takip ettiği görülmüştür. Tasarlanan sistemde yaklaşık 50 lx aydınlık düzeyinde renk tabanlı nesne tespitinin başarılı bir şekilde yapılabileceği ve çevresel aydınlatma durumuna bağlı olarak yaklaşık 0-225 mW arasında değişen bir güce ihtiyaç duyulduğu anlaşılmıştır. Sonuç olarak; akıllı ortam aydınlatması yapan gömülü sistemin çevresel koşullara bağlı olarak değişen enerji tasarrufu ile asgari aydınlık düzeyini sağladığı görülmüştür.

Anahtar Kavramlar **:** Akıllı Ortam Aydınlatması, Görüntü İşleme, Renk Tabanlı Nesne Tespiti, Robotik

Bilim Kodu **:** 92417

INTELLIGENT AMBIENT LIGHTING FOR IMAGE PROCESSING BASED OBJECT RECOGNITION IN ROBOTIC SYSTEMS

Uğur AKIŞ

ORCID: 0000-0001-7095-7770

HITIT UNIVERSITY

GRADUATE SCHOOL

Master of Science Thesis

January 2023

ABSTRACT

In image processing-based applications, ambient lighting is an important issue in terms of accurate detection of objects and energy use. Ambient lighting in classical methods; It is done by keeping the light source on all the time, turning it on and off manually as needed, using a timer or automatically measuring the illuminance level with the help of various sensors. In this thesis, an embedded system designed for reference-color-based intelligent ambient lighting based on image processing for object detection and position determination for robotic handling applications has been designed and its performance has been examined. In order to obtain the minimum illuminance level in the system, the power of the lighting system is adjusted by reference to red, green, blue and yellow objects whose positions are determined beforehand, without the need for an external sensor. Raspberry Pi 4B mini computer and Arduino CNC shield board with GRBL software is used for image processing, lighting and 3-axis handling processes in the designed system. Image acquisition was done via the Raspberry Pi camera module, and the handling of objects was done with an electromagnet. For lighting, 12V DC white strip LED lamp driven by PWM signal was used and smart ambient lighting was made by controlling the illuminance level proportionally. System interface software is coded in Python language and OpenCV library is used for image processing and object detection and position determination. Within the scope of experimental studies to examine the performance of the designed system, white light was illuminated, a non-reflective white matte plastic background was used for the platform floor, and the platform was covered with a black cover. In object detection with color-based image processing, the minimum required platform

illuminance values were investigated by using reference objects with different colors such as Red, Green, Blue and Yellow. According to the experimental results obtained; it has been understood that color selection is important in the detection process and it has been determined that the color that needs the least illumination level among these colors is yellow. It was observed that yellow color was followed by green, red and blue colors, respectively. It has been understood that in the designed system, color-based object detection can be done successfully at an illuminance level of about 50 lx and a power ranging from 0 to 225 mW is needed depending on the environmental lighting situation. In conclusion; It has been observed that the embedded system, which makes smart ambient lighting, provides the minimum illumination level with energy saving that changes depending on environmental conditions.

Key Terms **:** Smart Ambient Lighting, Image Processing, Color-Based Object Recognition, Robotics

Science Code **:** 92417

TEŞEKKÜR

Çalışmalarım boyunca ilgi ve bilimsel katkılarıyla beni yönlendiren, maddi ve manevi desteğini esirgemeyen tez danışmanım Dr. Öğr. Üyesi Serkan DİŞLİTAŞ'a, çalışmalarıma katkı sağlayan Dr. Öğr. Üyesi Yusuf KANCA ve Bilgisayar Öğretmeni Veysel SARI'ya, Pamukkale Üniversitesi idari amirim Dr. Öğr. Üyesi Umut TEPEKULE'ye, emeği geçen tüm arkadaşlarıma ve manevi destekleriyle beni hiçbir zaman yalnız bırakmayan aileme teşekkürlerimi sunarım.

Uğur AKIŞ

İÇİNDEKİLER

Sayfa

Sayfa

3. BÖLÜM

ANALİZ VE BULGULAR

TABLOLAR DİZİNİ

Tablo **Sayfa**

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

RESİMLER DİZİNİ

Resim **Sayfa**

SİMGELER VE KISALTMALAR

Simgeler

A	Akım birimi (Amper)
E	Aydınlık Düzeyi (lx)
E_{min}	Sisteme yönelik görüntü işleme için gerekli minimum aydınlık düzeyi (lx)
e	Pozisyon hatası
f	Frekans (Hz)
u	Kontrol sinyali
V	Potansiyel Fark (Volt)
W_{min}	Sisteme yönelik görüntü işleme için harcanan minimum elektriksel aydınlatma gücü (lx)
X_d	Tespit edilen referans X eksen pozisyonu
X_r	Gerçek referans X eksen pozisyonu
Y_d	Tespit edilen referans Y eksen pozisyonu
Y_r	Gerçek referans Y eksen pozisyonu
φ	Yansıtma faktörü

Kısaltmalar

BGR	Mavi - Yeşil - Kırımızı (Blue – Green - Red) renk formatı
CIE	Uluslararası Aydınlatma Komisyonu
CNC	Bilgisayarlı Sayısal Kontrol (Computer Numerical Control)
CSI	Kamera Seri Arabirimi (Camera Serial Interface)
DC	Doğru Akım (Direct Current)
GPIO	Genel Amaçlı Giriş - Çıkış (General Purpose Input-Output)
HSV	Ton – Doygunluk - Değer (Hue – Saturation - Value) renk uzayı
LED	Işık Yayan Diyot (Ligth Emitting Diode)

MOSFET	Metal Oksit Yarıiletken Alan Etkili Transistör (Metal Oxide Semiconductor Field Effect Transistor)
OpenCV	Açık Kaynak Bilgisayarlı Görü Kütüphanesi (Open Source Computer Vision Library)
PWM	Pals Genişlik Modülasyonu (Pulse Width Modulation)
RGB	Kırmızı - Yeşil - Mavi (Red – Green - Blue) renk uzayı
RMS	Kare Ortalamasının Karekökü (Root Mean Square)
USB	Evrensel Seri Veriyolu (Universal Serial Bus)

GİRİŞ

Dünyada elektrik enerjisine olan ihtiyacın hızla artması, elektrik enerjisinin üretimi, iletimi, dağıtımı sürecinde küresel, ulusal ve bireysel anlamda ekonomik bir yük oluşturmaktadır. Bu sebeple de elektrik enerjisi üretim maliyetlerinin düşürülmesi ve verimli kullanılmasına yönelik olarak dünyada yoğun bir şekilde çalışmalar yapılmaktadır. Bu kapsamda enerji kaynaklarının ve enerjinin kullanımında verimliliğin artırılmasına yönelik 5627 sayılı "Enerji Verimliliği Kanunu" kabul edilerek 26510 sayılı resmi gazetede yayımlanarak yürürlüğe girmiştir (Enerji Verimliliği Kanunu, 2007).

Elektrik enerjisinin en önemli kullanım alanlarından biri olan aydınlatma, özellikle fabrika, laboratuvar ve iş yerlerinde kaçınılmaz bir gereklilik olmuştur. Bu bağlamda elektrik enerjisinden tasarruf edilmesi ve enerjinin verimli kullanılması açısından akıllı aydınlatma sistemleri büyük önem taşımaktadır.

Robotik sistemler sahip oldukları seri üretim, verimlilik, iş güvenliği gibi birçok avantajından dolayı elleçleme başta olmak üzere kaynak, boyama, montaj, test vb. çeşitli işlemlerin yapılması amacıyla yaygın olarak kullanılmaktadır. Robot hücrelerinin yapılacak işe uygun olarak belirli birtakım özelliklere sahip olması gerekir (Dişlitaş, 2015). Görüntü işleme tabanlı olarak elleçleme işlemlerinin sağlıklı bir şekilde yapılabilmesi ve kamera ile elde edilen anlık görüntülerde renklerin dolayısıyla nesnelerin doğru bir şekilde algılanabilmesi için ortam aydınlatmasının yeterli düzeyde olması gerekir. Ayrıca renklerin doğru oluşmasında en iyi referans renk beyazdır (Bayram, 2009).

Genel olarak ışık, nesnelerin görülmesi ve renklerin ayırt edilmesine yarayan bir enerji şeklidir. Renk ise ışığa bağlı olarak ortaya çıkan, insan gözünün belli bir dalga boyunda algılayabildiği ışık spektrumu olarak tanımlanabilir. Ortam aydınlatmasında ışık kaynağının yönü ve miktarı, nesnelerin doğru algılanabilmesi için önemli bir unsurdur. Bununla birlikte ortam aydınlatmasının kötü olmasına bağlı olarak gölge ve parlama gibi nesne algılanmasını olumsuz yönde etkileyen unsurlar da vardır. Işığın zayıf olması gölgelere, güçlü olması ise çok fazla yansıma sonucu nesne renginin bozulmasına yol açar ve parlamalara sebep olur. Ayrıca ışığın tek bir noktadan uygulanması, gölge etkisini artıracağından, farklı noktalardan aydınlatma yapılması tercih edilir. Aydınlatmada bir nesne yeterli miktarda ışık aldığında yeterince yansıma yapacağı için orijinal renginde görülür(Zettl, 1999).

Ortam aydınlatılması amacıyla doğal ve yapay ışık kaynakları kullanılmaktadır. Aydınlatmada en büyük doğal ışık kaynağı olan Güneşten azami ölçüde yararlanılması enerji tüketiminin azaltılması açısından oldukça önemli bir konudur. Günışığı ile aydınlatma, kaliteli bir ışık kaynağı olup, en iyi renk görüntüleme indeksine sahiptir (Singh ve Garg, 2010). Ancak gün ışığı zamana, hava durumuna veya ortamdaki gölge oluşturacak diğer bozucu etkenlere bağlı olarak yeterli olmayabilmektedir. Gün ışığının olmadığı veya yetersiz olduğu mekanlarda ortam aydınlatması için yapay aydınlatma sistemleri kullanılmaktadır. Hem kaliteli bir aydınlatmanın

yapılması hem de enerjinin yerinde kullanılması amacıyla iyi bir aydınlatma sisteminden ihtiyaç duyulan yerde yeterli düzeyde ışık sağlaması istenir. Aydınlatmanın yetersiz olması durumunda konfor ve emniyet açısından sıkıntı yaşanabileceği gibi aşırı aydınlatma da ortamda parlama gibi olumsuz sonuçlara yol açabilmektedir (Onaygil, 2001).

Enerjinin üretilmesi kadar verimli kullanılması ve tasarruf edilmesi de bir o kadar önemlidir. Enerji tasarrufu alınan önlemlere bağlı olarak harcanan enerji miktarının azalması iken enerji verimliliği yeni ve akıllı teknolojiler kullanarak üretim, kalite, performans ve konforu düşürmeden enerji tasarrufu sağlanmasıdır (Enerji Verimliliği Kanunu, 2007). Enerjinin iyi yönetilmesi ve tasarruf edilmesi ile gereksiz kullanım ortadan kaldırılarak enerji tüketimi azaltılabilmektedir (Skaria, 2014). Enerji tasarrufu amacıyla gün ışığından maksimum düzeyde yararlanmak ve otomatik kontrol sistemlerini kullanarak yapay aydınlatma sistemlerinin kullanım süresi ve gücünün azaltılması gerekmektedir.

Aydınlatmada enerji tasarrufu sağlanması amacıyla elle, açık çevrim ve kapalı çevrim olmak üzere çeşitli kontrol yöntemleri kullanılmaktadır. Elle aydınlatma kontrolü, ihtiyaç olması durumuna göre elektrik anahtarlarının elle veya uzaktan açılıp kapatılarak aydınlanmanın sağlanması şeklinde olmaktadır. Açık çevrim aydınlatma kontrolünde, herhangi bir geribildirim olmaksızın elektrik anahtarlarının belirlenen zaman veya programa göre açılıp kapatılarak aydınlanmanın sağlanması şeklinde olmaktadır. Kapalı çevrim aydınlatma kontrolünde ise; aydınlanma ihtiyacını belirleyen ışık miktarı, yansıma, gölge vb. bir geribildirim durumuna göre elektrik anahtarlarının insan olmaksızın otomatik olarak açılıp kapatılarak aydınlanmanın sağlanması şeklinde olmaktadır.

Sektörel anlamda ihtiyaçlara yönelik olarak ticari, enerji tasarruflu ve gelişmiş özelliklere sahip akıllı aydınlatma sistemleri mevcuttur (Chew, 2017). Ticari akıllı aydınlatma sistemleri, temel anlamda kullanıcılar tarafından açma/kapama, loşlaştırma, izleme vb. Işlevsel özelliklere sahip sistemlerdir. Enerji tasarrufu sağlayan akıllı aydınlatma sistemleri, genellikle bir algoritma ve yazılım dahilinde birtakım sensörler yardımıyla çalışarak daha az elektrik enerjisi harcayan sistemlerdir (Neida vd., 2001). Gelişmiş akıllı aydınlatma sistemlerinde ise enerji tasarrufunun yanında çeşitli algoritma ve yapay zeka tabanlı uygulamalar ile ışığın şiddeti, yönü, açısı, mesafesi vb. özellikleri ayarlanarak ışık kalite kontrolü de yapılabilmektedir (Oh vd., 2014).

Literatür araştırmasına göre aydınlatmada elektrik enerjisinin verimli kullanılmasına yönelik çeşitli çalışmaların olduğu görülmüştür. Gökmen (2010) tez çalışmasında, endüstriyel tesislerde enerji verimliliği sağlayan aydınlatma teknikleri üzerine bir çalışma yapmış ve aydınlatmanın iş performansına etkilerini incelemiştir. Yücel (2019), aydınlatma sistemlerine yönelik özel bir kontrol sistemi tasarlayarak ortalama %59 oranında enerji verimliliği sağlanabileceğini göstermiştir. Bilici (2019) tez çalışmasında, LED aydınlatma araçları ve gün ışığından faydalanarak aydınlatma kalitesinin artırılması ve elektrik enerjisinin verimli kullanılmasına yönelik Pals Genişlik Modülasyonu (PWM – Pulse Width Modulation) yöntemi ile akıllı aydınlatma sistemi tasarlamış ve manuel kullanımla karşılaştırıldığında ortama bağlı

olarak %97 oranlarına varan elektrik enerjisi tasarrufu sağlamıştır. Kapusuzoğlu (2020) tez çalışmasında, aydınlatmada enerji tasarrufu amacıyla PWM yöntemini kullanarak mikrodenetleyici tabanlı uzaktan kontrollü LED sürücü devresi tasarlamıştır. Aydoğduoğlu (2021) tez çalışmasında, kamu binalarında enerji verimliliği sağlanması amacıyla akıllı aydınlatma üzerine LabVIEW ile bir yazılım geliştirmiş ve aydınlatma standartlarına göre sağlanması gereken asgari aydınlık düzeyi için %85 daha az enerji tüketilerek aydınlatma yapılabildiği sonucuna ulaşmıştır.

Endüstriyel alanlarda nesnelerin hızlı ve doğru bir şekilde tespit, takip ve sınıflandırması amacıyla görüntü işleme tabanlı robotik sistemler yaygın olarak kullanılmaktadır. Genel bir tanımlama ile görüntü işleme, gerçek bir görüntünün sayısal ortama aktarılarak üzerinde çeşitli algoritmalar yardımıyla iyileştirme, sadeleştirme, analiz, çıkarım gibi birtakım işlemlerin yapılmasını sağlayan teknolojidir (Yılmaz, 2007; Gonzales ve Woods, 2008; Hanbay ve Üzen, 2017). Görüntü işleme tabanlı olarak nesnelerin sınıflandırılması şekil, hareket, renk ve doku tabanlı yöntemler kullanılarak yapılabilmektedir (Tiwari ve Singhai, 2017). İşlem hızı ve başarı oranının çok yüksek olmasından dolayı renk tabanlı sınıflandırma yöntemleri yaygın olarak kullanılmaktadır (Fan vd., 2016). Görüntü işleme algoritmalarında tespit edilen bir nesnenin görüntü üzerindeki konumunun belirlenmesi ise ağırlık merkezinin bulunması prensibine dayanmaktadır (Minichino ve Howse, 2015).

Görüntü işleme tabanlı robotik sistemlerde, tespit edilecek nesnenin şekil, renk gibi fiziksel özelliklerine bağlı olarak ışık şiddetinin yeterli düzeye getirilmesinin yanı sıra gölge, yansıma, parlama vb. olumsuz etkilerin ortadan kaldırılarak aydınlatmanın iyi bir şekilde yapılması gerekmektedir. Aydınlatma uygulamalarında uzun ömürlü oluşu, enerji verimliliği sağlaması, hızlı anahtarlama vb. avantajlarından dolayı LED lambalar yaygın olarak kullanılmaktadır (Barwar vd., 2022). Ayrıca görüntü işleme tabanlı uygulamalarda ışık yoğunluğu vb. kontrollerin kolayca yapılabilmesi nedeniyle LED'ler ideal bir ışık kaynağı durumundadır (Li-li vd., 2013; Linnartz vd., 2008).

Literatür araştırmasına göre; aydınlatma ve robotik sistemlere yönelik görüntü işleme tabanlı endüstriyel uygulamaların hızla yaygınlaştığı görülmüştür. Uzer, Yılmaz ve Bayrak (2010), görüntü işleme ve robotik görme teknikleri kullanılarak farklı renklerdeki nesnelerin gerçek zamanlı takip edilmesine yönelik bir mobil robot tasarımı yapmış ve özelliklerini incelemişlerdir. Altınkurt ve Kahriman (2011), görüntü işlemeye dayalı renk tabanlı nesne tespiti ve takibi amacıyla, gerçek zamanlı kamera görüntüsünün MATLAB programı ile adaptif bir biçimde işlendiği mikrodenetleyici tabanlı nesne takip sistemi geliştirmiştir. Adelkhani, Beheshti, Minaei ve Javadikia (2012) çalışmalarında narenciyeye yönelik görüntü işlemede ışık tipi, ışık şiddeti, ışık kaynağının yüksekliği ve kamera yüksekliği gibi parametreleri ayarlanabilir bir şekilde kullanarak aydınlatma koşullarının, kamera yüksekliğinin ve renklerin optimizasyonunu ele almışlardır. Büyükarıkan (2014) tez çalışmasında, görüntü işleme yöntemiyle PWM tekniğini kullanarak ışık seviyesinin ayarlanmasını sağlayarak

cisimlerin optimum aydınlatma koşullarının belirlenmesine yönelik çalışmalar yapmıştır. Şenel ve Çetişli (2015), robotik sistem kurulu üretim bandından geçen hatalı ürünlerin görüntü işleme tabanlı olarak tespit edilerek tasnif edilmesine yönelik bir gömülü sistem yazılımı geliştirmiş ve %100 başarıyla sınıflandırma yapmıştır. Büyükkoçak (2018) tez çalışmasında, görüntü işleme tekniklerini kullanarak iç mekan aydınlatma şiddeti ölçümünün yapılmasına yönelik çalışmalar yapmış ve önerilen ölçüm yöntemi ile uygulanabilir sonuçlar elde etmiştir.

Büyükarıkan ve Üncü (2019) bilgisayarlı görü sistemleri için aydınlatma kontrolünün yapılması ve aydınlık düzeyinin ölçülmesi amacıyla PWM sinyali ile sürülen bilgisayar tabanlı bir sistem tasarımı gerçekleştirmişlerdir. Yıldırım (2019) tez çalışmasında, 3-eksen robot kol yardımıyla elleçleme uygulaması için renk tabanlı görüntü işleme ile gerçek zamanlı nesne tespiti yapılmasına yönelik çalışma yapmış ve başarılı sonuçlar almıştır. Sarıyıldız ve Demirhan (2021) tarafından renklerine göre nesneleri kategorilere ayıran görüntü işleme tabanlı robotik sistem tasarımı yapılmış ve %100 başarı sağlanmıştır. Yılmaz ve Sungur (2021) görüntü işlem tabanlı bir robotik sistem kurulumu yaparak akan banttaki çikolataların yüksek doğrulukta tespiti ve hatasız paketlenmesine yönelik çalışma gerçekleştirmişlerdir. Zeynel (2021), çalışmasında otomasyon sistemlerine yönelik görüntü işleme tabanlı ürün tanıma ve kalite kontrolü amacıyla Raspberry Pi geliştirme kartı ve Python dili kullanılarak bir gömülü sistem tasarımı yapmıştır. Dersuneli, Gündüz ve Kutlu (2021), klasik görüntü işleme ve derin öğrenme yöntemlerini kullanarak Python programlama dilinde OpenCV kütüphanesinden yararlanarak renk tabanlı nesne tespit edilmesine yönelik çalışma yapmış ve başarılı sonuçlar almıştır.

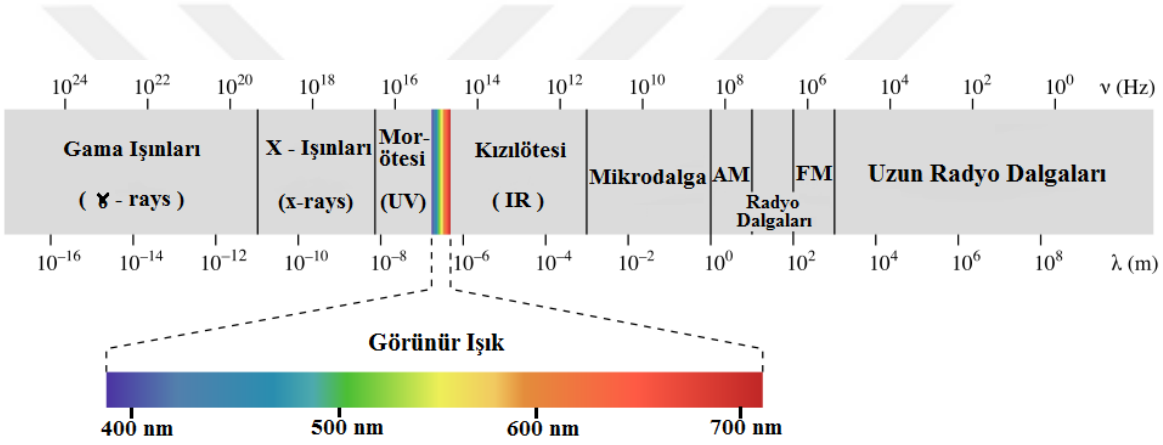
Bu tez çalışmasında, robotik elleçleme uygulamalarına yönelik görüntü işlemeye dayalı nesne tespiti ve pozisyon belirlemesi için referans renk-tabanlı akıllı ortam aydınlatması yapan bir gömülü sistem tasarımı yapılmış ve performansı incelenmiştir. Tez çalışması beş bölümden oluşmaktadır. Tezin giriş bölümünde, enerji, robotik ve görüntü işlemeye değinilerek çalışma ile ilgili literatür bilgisi verilmiştir. Tezin birinci bölümünde, aydınlatma temel kavramları ve renk uzayları konusunda bilgiler verilmiştir. Tezin ikinci bölümünde, tasarlanan ve gerçekleştirilen akıllı ortam aydınlatması yapan gömülü sistem donanım ve yazılım yönüyle anlatılmıştır. Tezin üçüncü bölümünde, gerçekleştirilen sistemin deneysel analizi yapılarak performansı incelenmiştir. Tezin sonuç bölümünde, tasarlanan sistemin genel bir değerlendirilmesi yapılarak kullanılabilirliği ve geliştirilebilirliği hakkında bilgiler verilmiştir.

1. BÖLÜM

AYDINLATMA

1.1. Aydınlatma Temel Kavramları

Işık; dalga şeklinde yayılan elektromanyetik dalga enerjisinin (radyasyon) gözle görülebilir halidir. Şekil 1.1'de elektromanyetik spektrum içerisinde ışığın dalga boyuna göre radyo, mikrodalga, kızılötesi, görünür ışık, ultraviyole, X-ışınları, gama ışınları şeklinde sınıflandırılması görülmektedir. Elektromanyetik spektrumun insanlar tarafından çıplak gözle görülebilen 380 nm ile 780 nm arasındaki kısmına görünür ışık denir (Onaygil, 2000; Gonzales ve Woods, 2008).



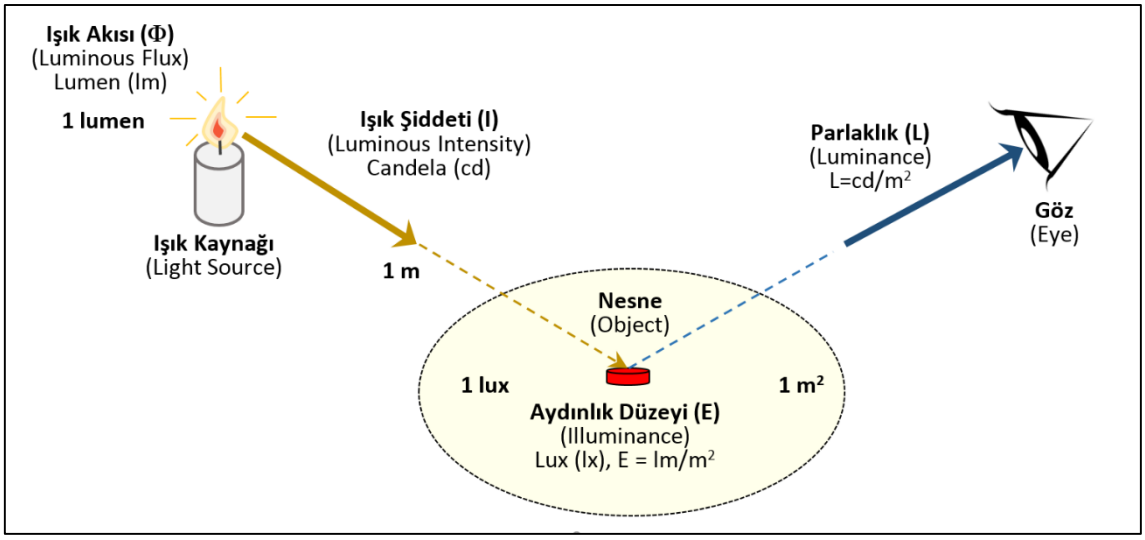
Şekil 1.1. Işığın dalga boyuna göre sınıflandırılması

Uluslararası Aydınlatma Komisyonuna (CIE) göre; aydınlatma, çevre veya bir nesnenin gereği gibi görülebilmesi için ışık uygulanmasıdır (IESNA, 2000). Başka bir tanıma göre; aydınlatma, ışığın ve ışığa bağlı olarak oluşan gölgenin ihtiyaçlar çerçevesinde kontrol edilmesidir (Zett1, 1999). Aydınlatma, asgari görme ihtiyacını sağlayan ışığın, ortamdaki dağılımını kontrol eden ve görme konforuna bağlı olarak iş verimini yükseltmeyi amaç edinen özel bir bilim dalı haline gelmiştir (Demirdeş, 1993). Aydınlatma ile ilgili ışık şiddeti, aydınlık düzeyi gibi temel kavramlar Tablo 1.1'de özetlenmiştir (Onaygil, 2000; Ünal, 2014; Özkaya vd., 2011).

Tablo 1.1. Aydınlatma temel kavramları

Terim	Sembol	Birim	Açıklama
Işık Şiddeti	I	cd (Kandela)	Bir ışık kaynağından belirli bir açıda yayılan görünebilir ışık miktarıdır.
Işık Akısı	Φ	lm (Lümen)	Bir ışık kaynağından 1 saniyede yayılan ışık miktarıdır.
Aydınlık Düzeyi	E	lx (Lüks)	Bir yüzeyin birim alanına (S) düşen ışık akısı miktarıdır.
Parıltı	L	cd/m^2	Belirli bir doğrultuda aydınlatılan yüzeye ait birim alana düşen ışık şiddetidir. Yüzeyin algılanan parlaklığını verir.
Işık Dalga Boyu	λ	m	Işığın birim frekanstaki hızını ifade eder.

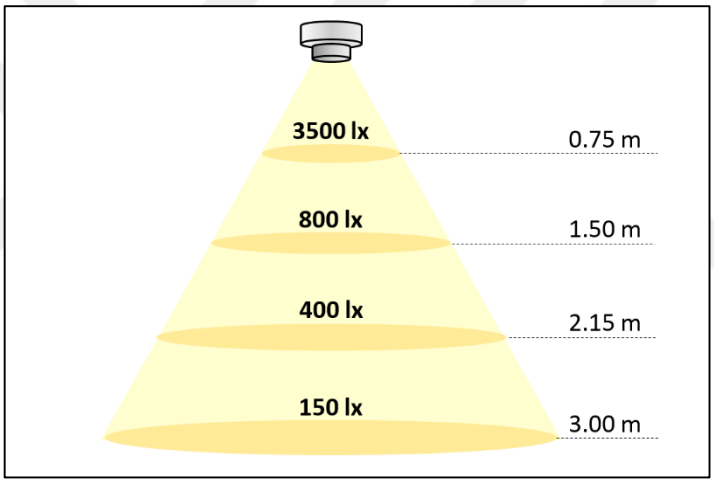
Bir nesnenin ışık kaynağına bağlı olarak görülmesinde etkili olan ışık akısı, ışık şiddeti, aydınlık düzeyi ve parıltı arasındaki ilişkisi Şekil 1.2'de görülmektedir. Aydınlık düzeyi için sadece kaynağın ışık akısı, açısı, yüzeyin kaynağa olan mesafesi belirleyici iken parıltıda ise bunlara ilaveten yüzeyin renk durumu ve yapısı da belirleyici rol üstlenmektedir. Aynı aydınlık düzeyinde açık renkler koyu renklere oranla daha fazla ışık verirler ve bu nedenle beyaz renkli yüzeyler ışığı daha iyi yansıttığı için parlak görünürken, siyah renkli yüzeyler ışığı daha az yansıttığı için mat olarak görünür (Onaygil, 2000; Ünal, 2014; Özkaya vd., 2011).



Şekil 1.2. Işık akısı, ışık şiddeti, aydınlık düzeyi ve parıltı ilişkisi

Aydınlık düzeyi, ışık kaynağından gelen ışık akısının yüzeyle olan açısına ve yüzeyin ışık kaynağından uzaklığına bağlıdır (Şekil 1.3). Aydınlık düzeyi, ışık kaynağından uzaklaştıkça yaklaşık uzaklığın karesi ile orantılı bir şekilde azalmaktadır. Işık akısının yüzeyle olan açısına bağlı olarak, açı azaldıkça yüzeydeki aydınlık düzeyi de açının sinüs fonksiyonu ile orantılı olarak azalmaktadır. Ayrıca yüzeyin yansıtma durumu da aydınlık düzeyine etki eden önemli bir faktördür. Koyu renge sahip yüzeyler gelen ışık akısını soğururken, açık renge sahip yüzeyler ışık akısını yansıtır. Bir ışık kaynağına aynı mesafe ve açıda bulunan iki farklı yüzeyden koyu renkli olanın, açık renkli olana göre daha az aydınlık düzeyine sahip olduğu görülmektedir (Aydoğduoğlu, 2021).

TS EN 12464-1 standardına göre; fabrika, laboratuvar, ofis vb. kapalı çalışma alanlardaki ortalama aydınlık düzeyinin yapılan işin muhteviyatına göre yaklaşık 50 ile 2000 lüks arasında değiştiği görülmektedir (TS EN 12464-1, 2021).



Şekil 1.3. Aydınlık düzeyinin uzaklık ile ilişkisi (Barnett, 2013)

Yansıma, ışığın bir yüzeye çarpması sonucu yön ve doğrultu değiştirerek geldiği ortama geri dönmesi olayıdır. Yansıma cisim yüzeyinin fiziksel özelliklerine göre değişebilmektedir ve bu nedenle cisimler üzerlerine düşen ışığın tamamını yansıtabilmekte veya bir kısmını absorbe edebilmektedir. Bir cisimden yansıyan ışık akısının, cisme gelen ışık akısına oranına Yansıtma Faktörü (φ) denir. Işık yansıtma faktörü koyu renklerde, açık renklere göre daha düşüktür. Bu nedenle aydınlatma sisteminin veriminin yansıtma açısından olumsuz etkilenmemesi için çalışma yüzeylerinin parlak olmayan ve koyu renkli malzemeden seçilmesine özen gösterilmelidir (Dehoff vd., 2005). Tablo 1.2'de renklerin sahip oldukları ışık yansıtma değerleri (LRV - Light Reflectance Value) görülmektedir (Imamguluyev, 2021).

Tablo 1.2. Renk ışık yansıtma değerleri (LRV)

Renk	Işık Yansıtma Değeri (LRV)
Beyaz	% 85
Sarı	% 68
Açık Yeşil	%60
Açık Mavi	%50
Açık Kırmızı	%35
Koyu Yeşil	% 30
Koyu Mavi	%20
Koyu Kırmızı	% 18
Mor	%10
Siyah	% 7

1.2. Renk Uzayı Modelleri

Renk ışığa bağlı olarak ortaya çıkan, insan gözünün belli bir dalga boyunda algılayabildiği ışık spektrumu olarak tanımlanabilir (Gonzales ve Woods, 2008)**.** Tablo 1.3'te görüldüğü gibi insanlar görünür ışık olarak adlandırılan 380 nm – 780 nm arasındaki ışıkları dalga boylarına göre farklı renklerde görebilmektedir. Cisimler uygulanan ışığa bağlı olarak yansıttığı renkte görünürler. Bu durumda ışık uygulanan bir cisim tüm renkleri yansıtıyorsa beyaz, hiçbir rengi yansıtmıyorsa siyah, sadece bir rengi yansıtıyorsa yansıttığı renkte görünmektedir (Onaygil, 2000; Özcan, 2010).

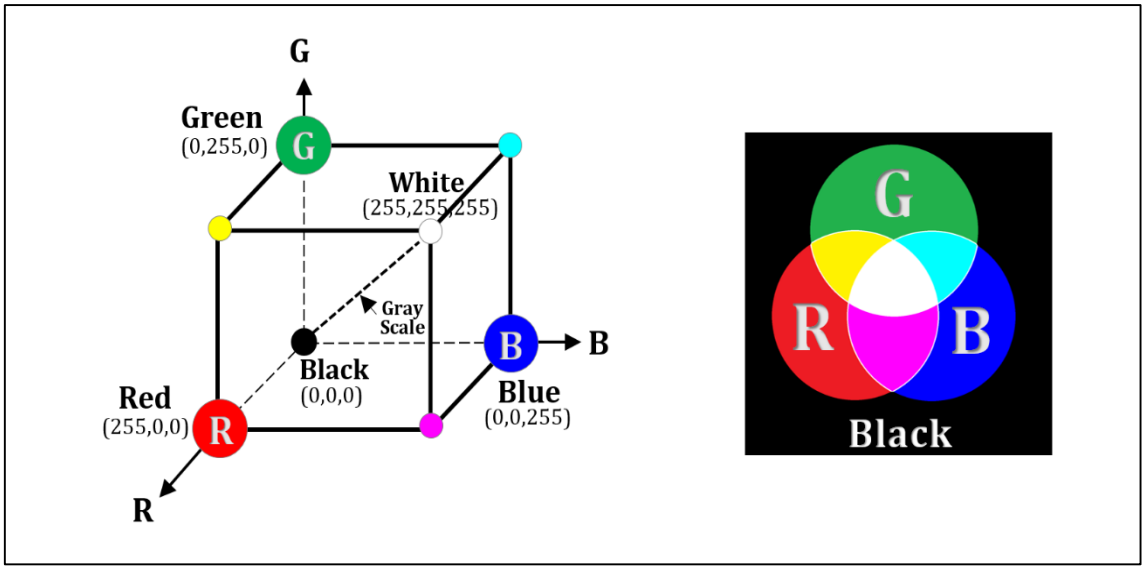
Tablo 1.3. Renk dalga boyları

Renk	Dalga Boyu	Frekans
Kırmızı	~ 625-740 nm	~ 480-405 THz
Turuncu	~ 590-625 nm	~ 510-480 THz
Sarı	~ 565-590 nm	~ 530-510 THz
Yeşil	~ 500-565 nm	~ 600-530 THz
Camgöbeği	~ 485-500 nm	~ 620-600 THz
Mavi	~ 450-485 nm	~ 680-620 THz
Mor	~ 380-450 nm	~ 790-680 THz

Renk uzayları, bütün renklerin birtakım standartlara göre belirlemesi ve sınıflandırılması amacıyla 3D olarak tasarlanan matematiksel modellerdir. Uygulamada yaygın olarak kullanılan RGB, CMYK, HSV ve HSL gibi çeşitli renk uzayı modelleri mevcuttur (Yılmaz, 2002; Lucas, 2022).

1.2.1. RGB renk uzay modeli

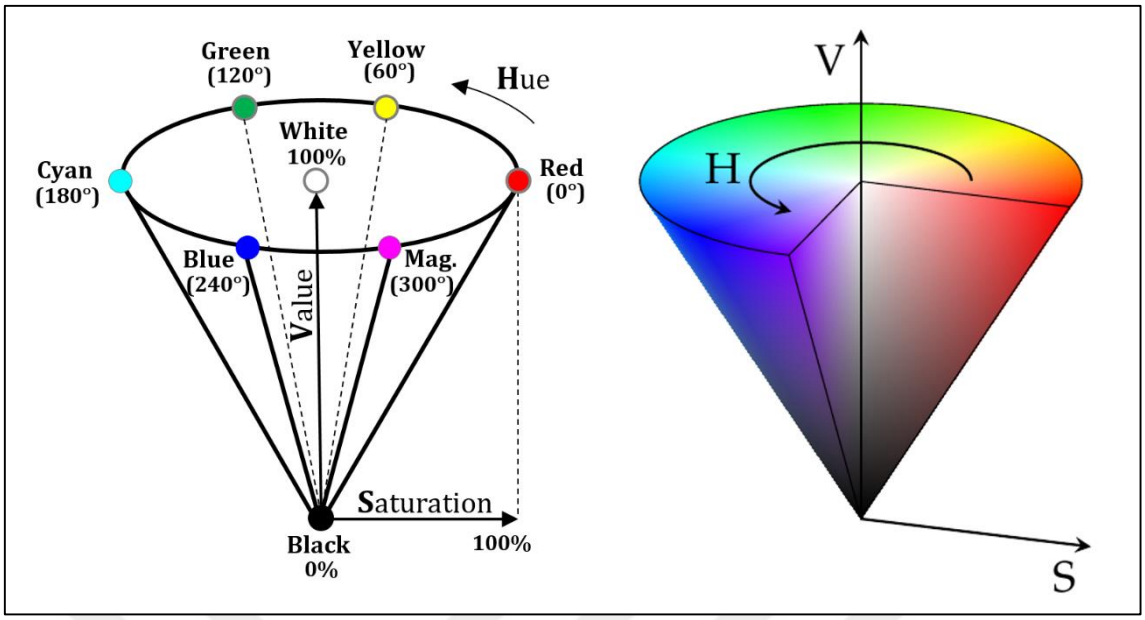
Şekil 1.4'te RGB renk uzay modeli görülmektedir. RGB (Red – Green - Blue) renk uzayı, her eksende bir renk olmak üzere Kırmızı, Yeşil ve Mavi renklerin oluşturduğu üç boyutlu bir koordinat düzlemi olarak temsil edilebilmektedir. Üç rengin 0-255 arası farklı kombinasyonları ile farklı renkler elde edilir (Hema ve Kannan, 2019; Yılmaz vd., 2002). Örneğin RGB (0, 0, 0) için Siyah, RGB (255, 255, 255) için Beyaz, RGB (255, 0, 0) için Kırmızı, RGB (0, 255, 0) için Yeşil, RGB (0, 0, 255) için Mavi ve farklı değerler için diğer renkler elde edilmektedir. RGB renk uzay modeli özellikle bilgisayar ekranı, televizyon gibi aktif göstergelerde yaygın olarak kullanılmaktadır (Çetin, 2008).



Şekil 1.4. RGB renk uzayı

1.2.2. HSV renk uzay modeli

Şekil 1.5'te HSV renk uzay modeli görülmektedir. HSV (Hue – Saturation - Value) renk uzayı modelinde renkler; renk tonu, doygunluk ve değer olarak tanımlanır. Diğer renk uzaylarına göre HSV renk uzayı, insan görü düzenine daha yakın bir yapıdadır ve renkli nesnelerin tespit edilmesinde daha iyi sonuçlar vermesinden dolayı görüntü işleme uygulamalarında HSV renk uzayı kullanılmaktadır. HSV renk uzay modeli temel alt ve üst renk aralıkları Tablo 1.4'te görülmektedir (Hema ve Kannan, 2019; Manipriya vd., 2014; Albayrak, 2001; Macit, H. B., 2020).



Şekil 1.5. HSV renk uzayı

Tablo 1.4. HSV renk uzay modeli temel renk aralıkları

Renk	**Alt Limit (H, S, V)**	**Üst Limit (H, S, V)**
Siyah	(0, 0, 0)	(179, 255, 30)
Beyaz	(0, 0, 231)	(179, 18, 255)
Turuncu	(0, 100, 100)	(22, 255, 255)
Sarı	(22, 100, 100)	(38, 255, 255)
Yeşil	(38, 100, 100)	(75, 255, 255)
Mavi	(75, 100, 100)	(130, 255, 255)
Mor	(130, 100, 100)	(160, 255, 255)
Kırmızı	(160, 100, 100)	(179, 255, 255)

2. BÖLÜM

TASARLANAN SİSTEM

2.1. Sistemin Genel Yapısı

Tasarlanan sistemde enerji tasarrufu sağlanması açısından akıllı ortam aydınlatması yapılmaktadır. Tasarlanan robotik sistemde referans, robot ve nesnelerin tespiti ve koordinatlarının belirlenmesi amacıyla görüntü işleme tekniği kullanılmaktadır. Görüntü işleme esnasında ortam aydınlatması nesnelerin doğru bir şekilde tespiti için önemli bir husustur. Bu açıdan görüntü işlemeye dayalı olarak nesnelerin tespit edilmesi ve koordinatlarının doğru olarak belirlenebilmesi için gerekli olan minimum ışık şiddeti yoğunluğunun uygulanması gerekir.

Deneysel çalışmalar yapmak amacıyla tasarlanan 3-eksen hareket yeteneğine sahip kartezyen robot hücresi Resim 2.1'de görülmektedir. Renk tabanlı görüntü işleme ile nesne tespiti ve tasnifi yapan robotik sistemde enerji tasarrufu sağlanması açısından akıllı ortam aydınlatması yapılmaktadır. Tasarlanan sistemin genel karakteristik özellikleri Tablo 2.1'de verilmiştir.

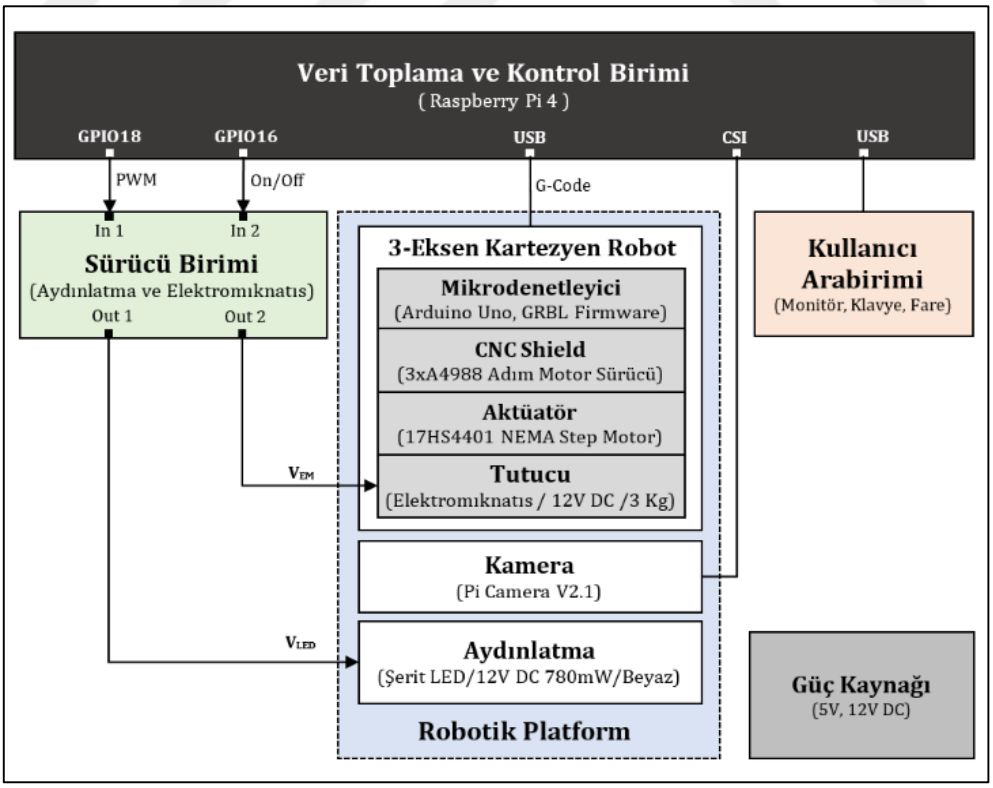
Resim 2.1. Tasarlanan sistemin genel görünümü

Tablo 2.1. Tasarlanan sistemin genel karakteristik özellikleri

Karakteristik	**Özellik**
Uygulama Alanı	Nesne Tespiti ve Sınıflandırma
Çalışma Yöntemi	Akıllı ortam aydınlatması ile renk tabanlı görüntü işleme
Kullanıcı Arabirimi	Raspberry Pi Tabanlı Grafik Arayüz (Python)
Renk Tanımları	Referans : Kırmızı, Mavi, Yeşil, Sarı Nesne (1,5 cm x 1,5 cm kare) Robot : Sarı Nesne : Yeşil (İyi), Kırmızı (Kötü)
Robot	3-eksen kartezyen robot
Robot Kontrolörü	Arduino Uno (GRBL firmware / G-Kodu) ve CNC Shield
Platform Aydınlatması	12 V DC 780 mW Beyaz Şerit LED (PWM ile 0-780 mW ayarlanabilir)
Tutucu Tasarımı	Aç/Kapat Kontrollü Elektromıknatıs (12 V DC, 3 Kg)
Platform Ölçüleri	50 cm x 40 cm x 40 cm

2.2. Sistem Donanımı

Tasarlanan sistemin genel blok şeması Şekil 2.1'de, iç ve dış görünümleri ise Resim 2.2'de görülmektedir. Sistem temel olarak veri toplama ve kontrol birimi, robotik platform, sürücü birimi, kullanıcı arabirimi ve güç beslemelerinden oluşmaktadır.

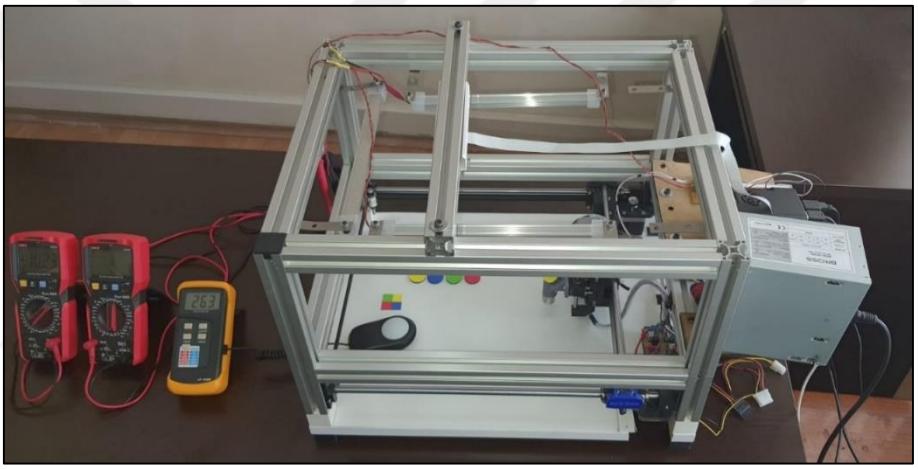


Şekil 2.1. Tasarlanan sistemin genel blok şeması

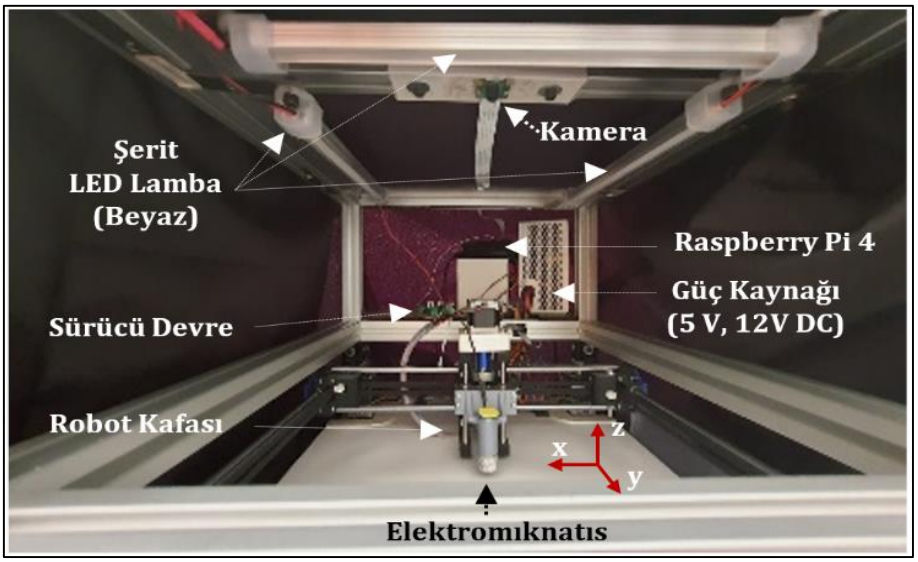
2.2.1. Aydınlatma kontrol sistemi

Şekil 2.2'de platform aydınlık düzeyinin belirlenmesine yönelik kapalı çevrim kural tabanlı oransal kontrol sistemi blok şeması görülmektedir. Kapalı çevrime özgü geribildirim ile hataya bağlı gerekli düzeltmeler yapılarak sistemin elektrik gücünün seviyesi ayarlanmaktadır ve bu şekilde aydınlık düzeyi istenilen değerde elde edilebilmektedir.

Tasarlanan sistemde renk tabanlı görüntü işleme için gerekli olan asgari aydınlık düzeyinin belirlenmesinde, referans nesne koordinatları ile görüntü işleme sonucu elde edilen koordinatlar arasındaki hata oranına bağlı olarak oransal kontrol işlemi yapılmaktadır. Sistemde harici bir sensör veya cihaza ihtiyaç duyulmadan pozisyonları daha önceden belirlenmiş kırmızı, yeşil, mavi ve sarı renkli nesneler referans alınarak aydınlatma sisteminin gücü ayarlanmaktadır. Yapay aydınlatma sisteminde yer alan şerit LED lambanın güç kontrolü MOSFET tabanlı devre üzerinden PWM sinyali ile sürülerek yapılmaktadır.

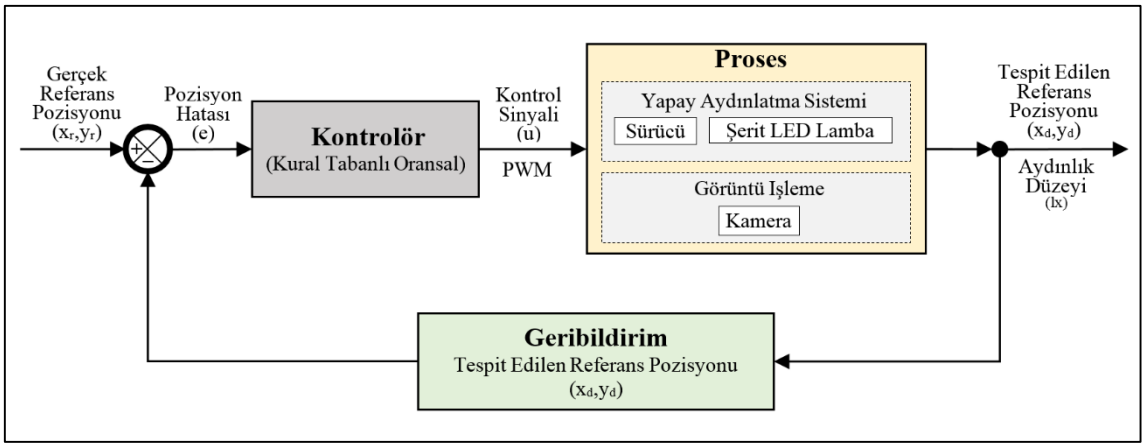


(a)



(b)

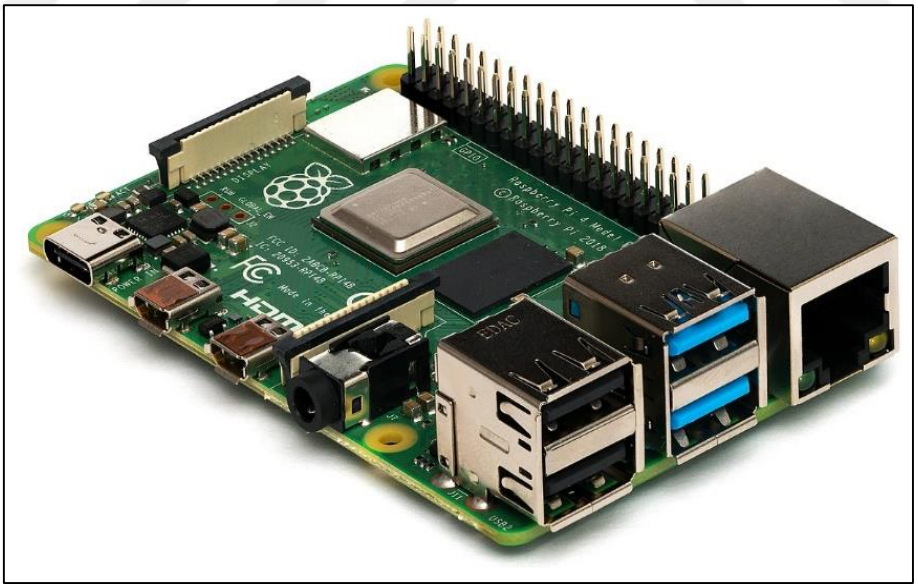
Resim 2.2. Tasarlanan robotik sistemin dış (a) ve iç (b) görünüşü



Şekil 2.2. Kapalı çevrim aydınlatma kontrol sistemi

2.2.2. Veri toplama ve kontrol birimi

Tasarlanan robot hücresinde veri toplama ve kontrol amacıyla Resim 2.3'te görülen ve teknik özellikleri Tablo 2.2'de verilen Raspberry Pi 4 mini bilgisayar geliştirme kartı kullanılarak renk tabanlı görüntü işleme, G-kodları ile 3-eksen robot elleçlemesi ve PWM tekniği ile ortam aydınlık düzeyinin ayarlanması sağlanmaktadır (Raspberry Pi, 2022).



Resim 2.3. Raspberry Pi geliştirme kartı

Tablo 2.2. Raspberry Pi geliştirme kartı teknik özellikleri

Model	Raspberry Pi Model B
İşlemci	Quad 1.5 Ghz CPU
Bellek	2 GB DDR4 RAM
Port	USB 2.0 : 2 adet USB 3.0 : 2 adet Mikro HDMI : 4K Display (2 adet) CSI Kamera True Gigabit Ethernet Dual-Band 2.4/5 GHz Wireless LAN BLE Bluetotth 5.0 microSD Kart Slot
Giriş/Çıkış Pinleri	Genel amaçlı 40 pin
Pin Başına DC Akım	Maksimum 16 mA (3.3 V)
Güç Girişi	USB-C (5V/3A)
İletişim Protokolleri	UART, SPI, I^2C
Ebat	68.63 mm x 94.09 mm x 26.63 mm
Ağırlık	45 g

2.2.3. 3-eksen kartezyen robot

Robotun 3-eksen hareketi için 3 adet adım motor (17HS4401 NEMA Step Motor), A4988 Step Motor Sürücü ve kayış takımı kullanılmıştır. 3-eksen robot konumlandırma için pozisyon bilgilerine bağlı olarak G-Kodu (RS-274, Geometrik Kod) bilgisayar sayısal kontrol (CNC) programlama dili talimatları üretilmektedir. G-Kodu talimatlarının derlenmesi amacıyla USB bağlantılı Arduino Uno mikrodenetleyici kart üzerinde yüklü açık kaynak kodlu ücretsiz GRBL donanım yazılımı (firmware) kullanılmıştır ve CNC Shield sürücü kart yardımıyla adım motorların hareket yönü ve hızı belirlenerek robotun 3-eksende istenilen pozisyona konumlanması sağlanmaktadır (Gandhi ve Sangeetha, 2018; Sarguroh ve Rane, 2018; Deniz, 2019).

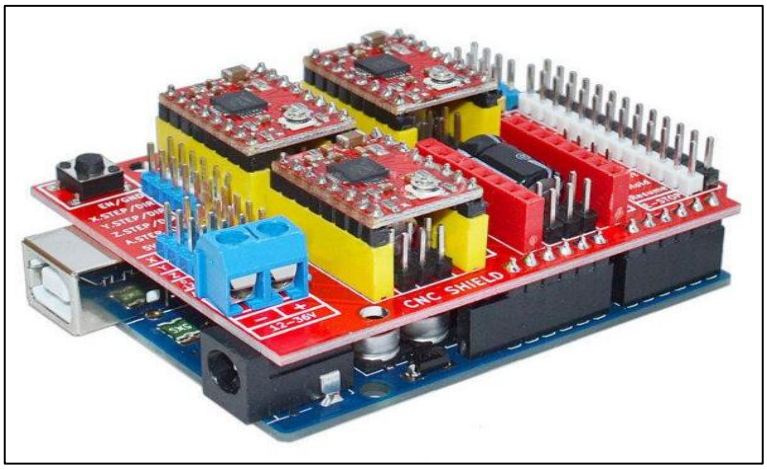
Resim 2.4'te görülen Arduino Uno mikrodenetleyici kart teknik özellikleri Tablo 2.3'te, Resim 2.5'te görülen CNC Shield kart teknik özellikleri ise Tablo 2.4'te verilmiştir (Arduino, 2022, Kruger, 2015). Robot hücresinin tabanında **30 cm x 30 cm** alana sahip bir platform oluşturularak robotun hareket alanı belirlenmiştir. Ayrıca sistemde elleçleme işlemleri için

robotun Z-eksenine yerleştirilmiş bir elektromıknatıs kullanılmıştır (Resim 2.6). Nüve üzerine sarılı bobin üzerinden elektrik akımı geçmesine bağlı olarak oluşturulan manyetik alan ile metalleri çekme özelliğine sahip elektromıknatıs temel özellikleri Tablo 2.5'te verilmiştir (Stump, 2004).

Resim 2.4. Arduino Uno R3 geliştirme kartı

Tablo 2.3. Arduino Uno R3 geliştirme kartı özellikleri

Mikrodenetleyici	ATmega328p 8-bit
Saat Frekeansı	16 MHz
Voltaj	Çalışma V. : 5 V Giriş V. : 7 ~ 12 V (Limit 6 ~ 20 V)
Pin Başına DC Akım	5.5 V : 40 mA 3.3 V : 50 mA
Giriş/Çıkış Pinleri	Dijital : 14 adet (6 adet PWM) Analog : 6 adet (sadece giriş)
LED_BUILTIN pin	13 nolu pin
Bellek	FLASH : 32 KB SRAM : 2 KB EEPROM : 1 KB
İletişim Protokolleri	UART, SPI ve I^2C
Programlama	USB (USB - B), ICSP
Ebat	68.6 mm x 53.4 mm
Ağırlık	25 g



Resim 2.5. Arduino CNC Shild kartı

Tablo 2.4. Arduino CNC Shild kartı özellikleri

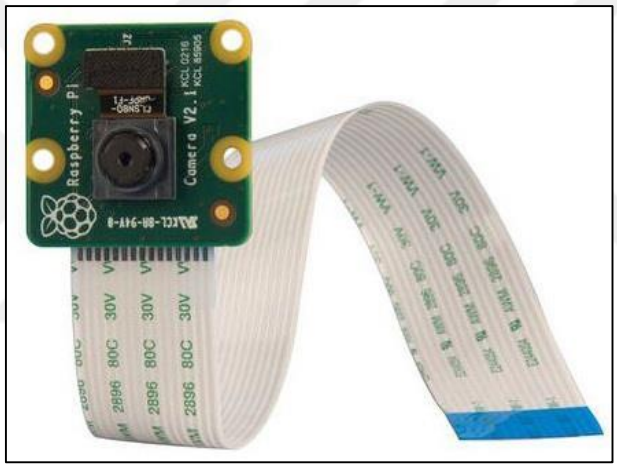
Model	Arduino CNC Shield v3.0 (GRBL uyumlu)
Eksen Kontrol	4-eksen (X - Y - Z - A)
Çalışma Gerilimi	12 - 36 V DC
Limit Anahtarı	Her eksen için 2 adet pin

Resim 2.6. Elleçleme işleminde kullanılan elektromıknatıs

Tablo 2.5. Elektromıknatıs teknik özellikleri

Model	Elektromıknatıs
Çalışma Gerilimi	12 V DC
Kontrol	Aç/Kapat
Taşıma Kapasitesi	3 Kg
Ebat	Silindirik (h : 2 cm h : 2 cm)

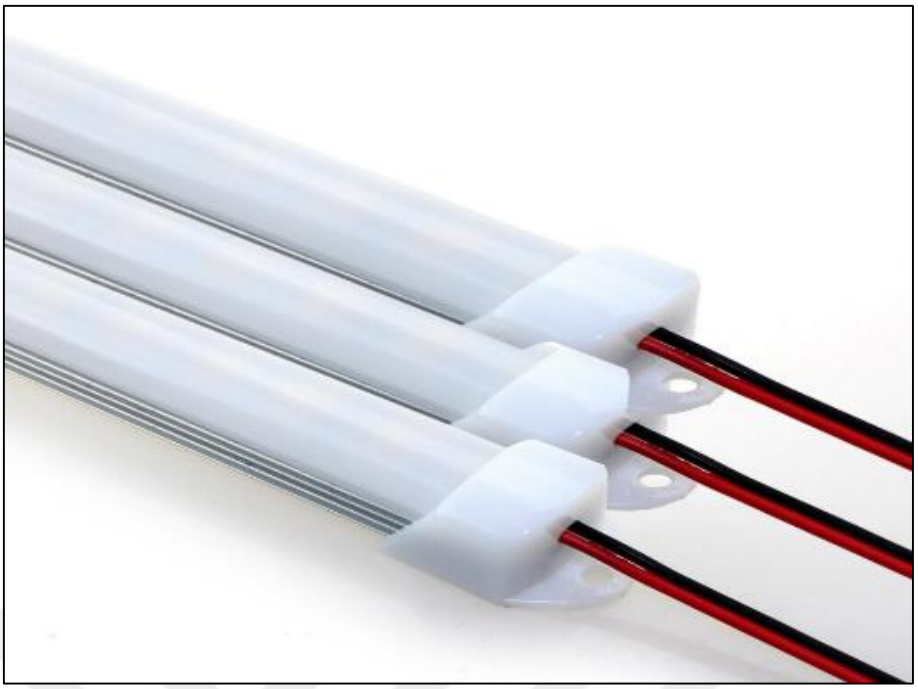
Sistemde görüntü işleme için Raspberry Pi 4 geliştirme kartına 15-pinli CSI (Kamera Seri Arabirimi) konnektör üzerinden bağlantılı Resim 2.7'de görülen Pi Camera V2.1 model kamera modülü kullanılmıştır. Raspberry Pi kamera modülünün teknik özellikleri Tablo 2.6'da verilmiştir (Raspberry Pi, 2022). Kamera modülü platform yüzeyinden 32 cm yükseklikte ve yukarıdan kuş bakışı görecek şekilde robot hücresinin tavan merkezine yerleştirilmiştir. Ortam aydınlatması için PWM sinyali ile ışık şiddeti ayarlanarak sürülen beyaz renkli 12V 780 W gücünde difüzörlü DC şerit LED kullanılmıştır (Resim 2.8.). Şerit LED lamba hem kamera altında kalacak hem de robot kafası ve platformu ön cepheden aydınlatacak şekilde platform yüzeyinden 28 cm yükseklikte U şeklinde robot hücresine sabitlenmiştir. Robot hücresinin taban platformu için fotoğrafçılıkta kullanılan yansıma yapmayan beyaz mat renkli plastik fon kullanılmıştır. Ayrıca deneysel çalışmalar amacıyla ortam aydınlatmasının çevre doğal ve yapay ışık kaynaklarından etkilenmemesi için siyah bir örtü kullanılmıştır.



Resim 2.7. Raspberry Pi kamera modülü

Tablo 2.6. Kamera özellikleri

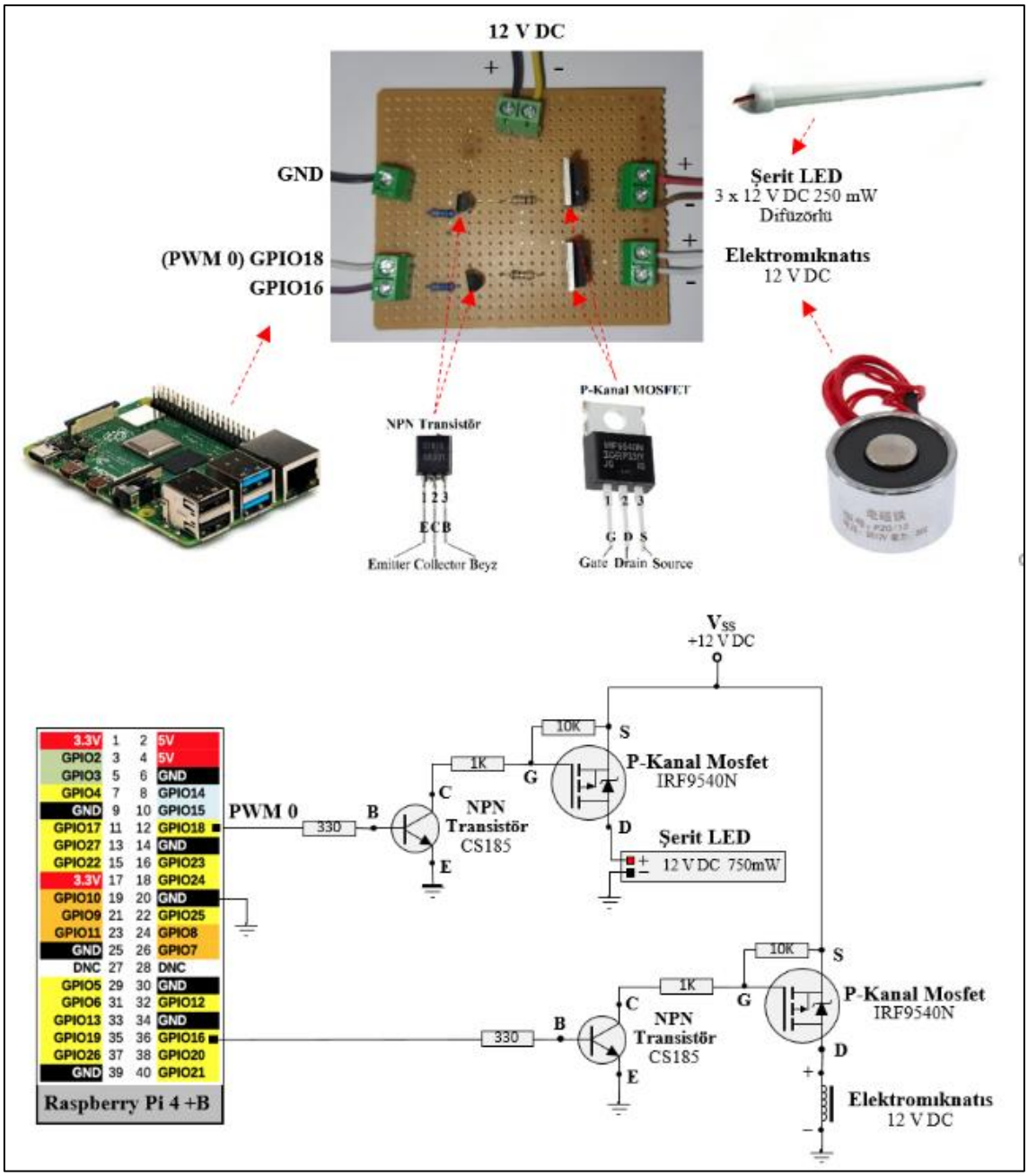
Model	Raspberry Pi Camera v2.1
Çözünürlük	Statik Fotoğraf : 8 mp (3280 x 2464 pixel) Video : 1080p, 720p60 ve VGA90
Bağlantı Birimi	CSI (Camera Serial Interface) 15-pin şerit kablo
Ebat	25 mm x 23 mm x 9 mm
Ağırlık	≅ 3 g



Resim 2.8. Difüzörlü **ş**erit LED lamba

2.2.4. Sürücü devre

Tasarlanan robotik sistemde şerit LED lamba ile ortam aydınlatmasının ayarlanmasının yanı sıra elleçleme işlemlerinde elektromıknatıs yardımıyla nesne alıp bırakma amacıyla Raspberry Pi ile kontrol edilen MOSFET tabanlı bir sürücü devre tasarımı yapılmıştır (Şekil 2.3). Yapay aydınlatma sisteminde yer alan şerit LED lambanın güç kontrolü MOSFET tabanlı devre üzerinden PWM sinyali ile sürülerek yapılmaktadır. PWM modülasyonunda LED lamba için sağlanan güç hızlı bir şekilde açılıp kapatılarak ışık şiddeti ayarlanmaktadır. Bu açıdan PWM kullanımında sinyal frekansı ve görev çevrimi (duty cycle) önemli parametrelerdir. 100 Hz'in üzerindeki frekanslarda PWM kullanımına bağlı olarak LED aydınlatmada meydan gelen titreşim etkisi azalmakta ve görsel olarak algılanamamaktadır (Hsu, 2016). Elleçleme için 12 V DC ile çalışan bir elektromıknatıs kullanılmıştır. Elektromıknatıs için ise GPIO16 pin çıkışından Aç/Kapat kontrolü ile nesne alma ve bırakma işlemi yapılmaktadır.



Şekil 2.3. Raspberry Pi tabanlı şerit LED lamba ve elektromıknatıs sürücü devresi

2.3. Sistem Yazılımı

Tasarlanan sistem yazılımı temel olarak görüntü işleme, aydınlatma ve 3-eksen kartezyen hareket sağlayan arayüz ve gömülü sistem yazılımlarından oluşmaktadır. Sistemde veri toplama ve kontrol birimi olarak kullanılan Raspberry Pi 4 minibilgisayara Debian tabanlı Raspberry Pi OS (Raspbian) işletim sistemi yüklenmiştir. Sistemin çalışmasına yönelik arayüz yazılımı Python dilinde kodlanmıştır ve görüntü işleme ile nesne tespiti ve koordinat belirlenme için OpenCV

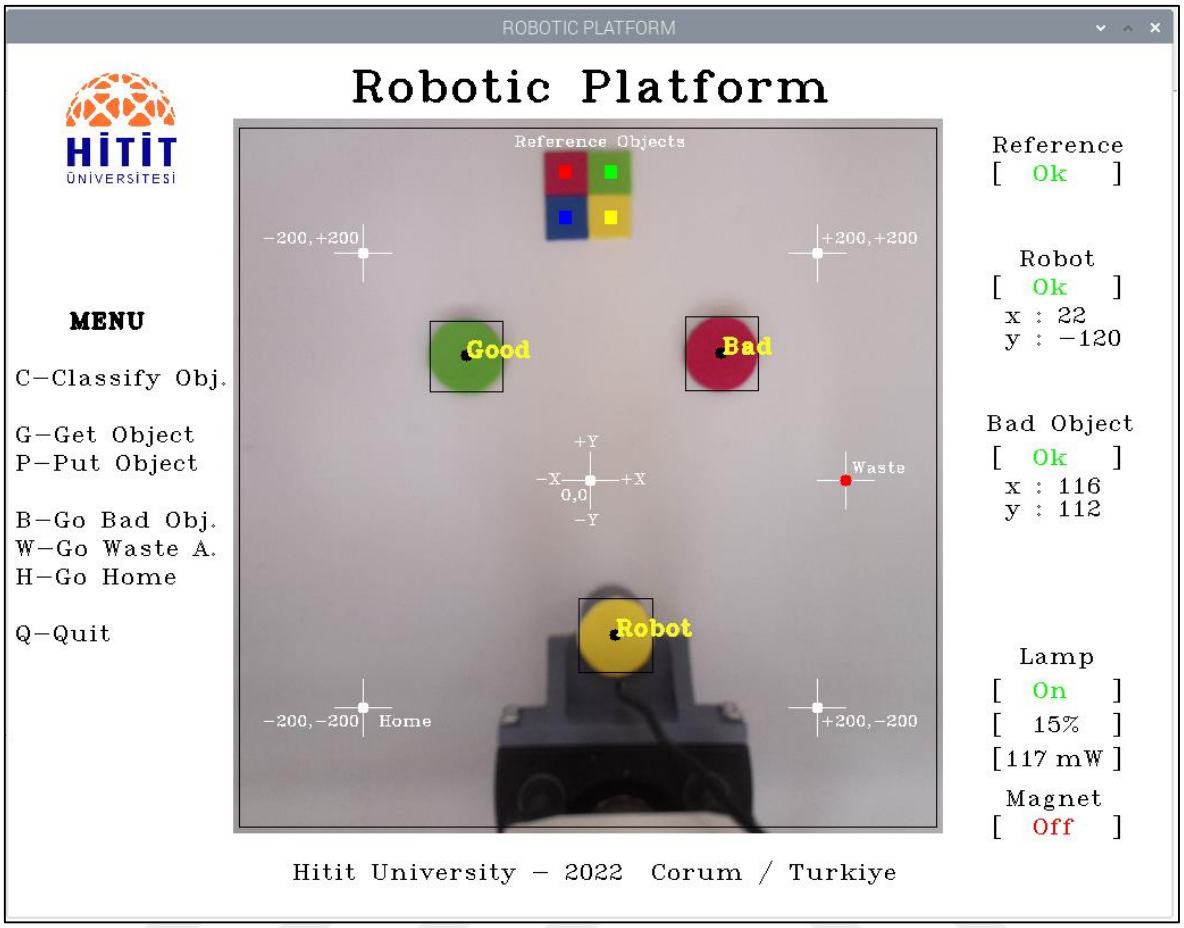
(Open Source Computer Vision Library) kütüpanesi kullanılmıştır (Minichino ve Howse, 2015; Işık, 2022). Sistem arayüz yazılımı kaynak kodları Ek-1'de verilmiştir.

Sistemde G-Kodu talimatlarının işlenerek 3-eksen robotun pozisyonlandırılması amacıyla CNC kontrolörü olarak çalışan, içerisine GRBL firmware yazılımı yüklü Arduino Uno mikrodenetleyici kart kullanılmıştır. Çalışma kapsamında 3-eksen Kartezyen robotun pozisyonlandırılması için kullanılan G-Kodu talimatları Tablo 2.7'de verilmiştir.

Tablo 2.7. 3-eksen kartezyen robotun pozisyonlandırılmasına yönelik G-Kodu talimatları

G-Kodu	Açıklama
G21	Tüm birimler "mm" olarak belirlenir. (G20 olması durumunda birim "inç" olarak belirlenir.)
G91	Artan Pozisyonlandırma Modu ile koordinata gidilir. (G90 olması durumunda Mutlak Pozisyonlandırma Modu geçerlidir)
G00	Hızlı Pozisyonlandırma Modu ile koordinata gidilir. (G01 olması durumunda Hızı Ayarlanabilir Doğrusal Pozisyonlandırma Modu geçerlidir)
G01 X10 F260	+X ekseninde 10mm mesafe 260 mm/dk. hızında doğrusal pozisyonlandırma hareketi ile gidilir.
G01 X-10 F260	-X ekseninde 10mm mesafe 260 mm/dk. hızında doğrusal pozisyonlandırma hareketi ile gidilir.
G01 X10 Y20 Z5 F260	+X, +Y, +Z ekseninde (10, 20, 5) mm mesafe 260 mm/dk. hızında doğrusal pozisyonlandırma hareketi ile gidilir.

Resim 2.9'da geliştirilen yazılımın robotik platforma yönelik ekran alıntısı görülmektedir. Kamera ile 1024x768 çözünürlüklü alınan görüntü üzerine X, Y koordinatları yerleştirilmiştir. Platform koordinatları orijin X_0, Y_0 (0,0), sol üst –X, +Y (-200, +200), sağ üst +X, +Y (+200, +200), sol alt –X, -Y (-200, -200) ve sağ alt +X, -Y (+200, -200) olarak belirlenmiştir. Platform üzerinde ayrıca Park ve Atık Bırakma koordinatları da belirlenmiştir. Platform üzerinde Tablo 2.8'de görüldüğü gibi sabit pozisyonlu Kırmızı, Yeşil, Mavi ve Sarı renkten oluşan referans nesneler aydınlık düzeyi ayarlamalarında kullanılmaktadır. Alınan görüntünün renk tabanlı işlenmesine bağlı olarak tespit edilen Robot (Sarı), İyi (Yeşil) ve Kötü (Kırmızı) olarak ifade edilmektedir. Ayrıca nesnenin elleçlenebilmesi için robota yerleştirilen elektromıknatıs dikey eksende (Z) hareket edebilmektedir.



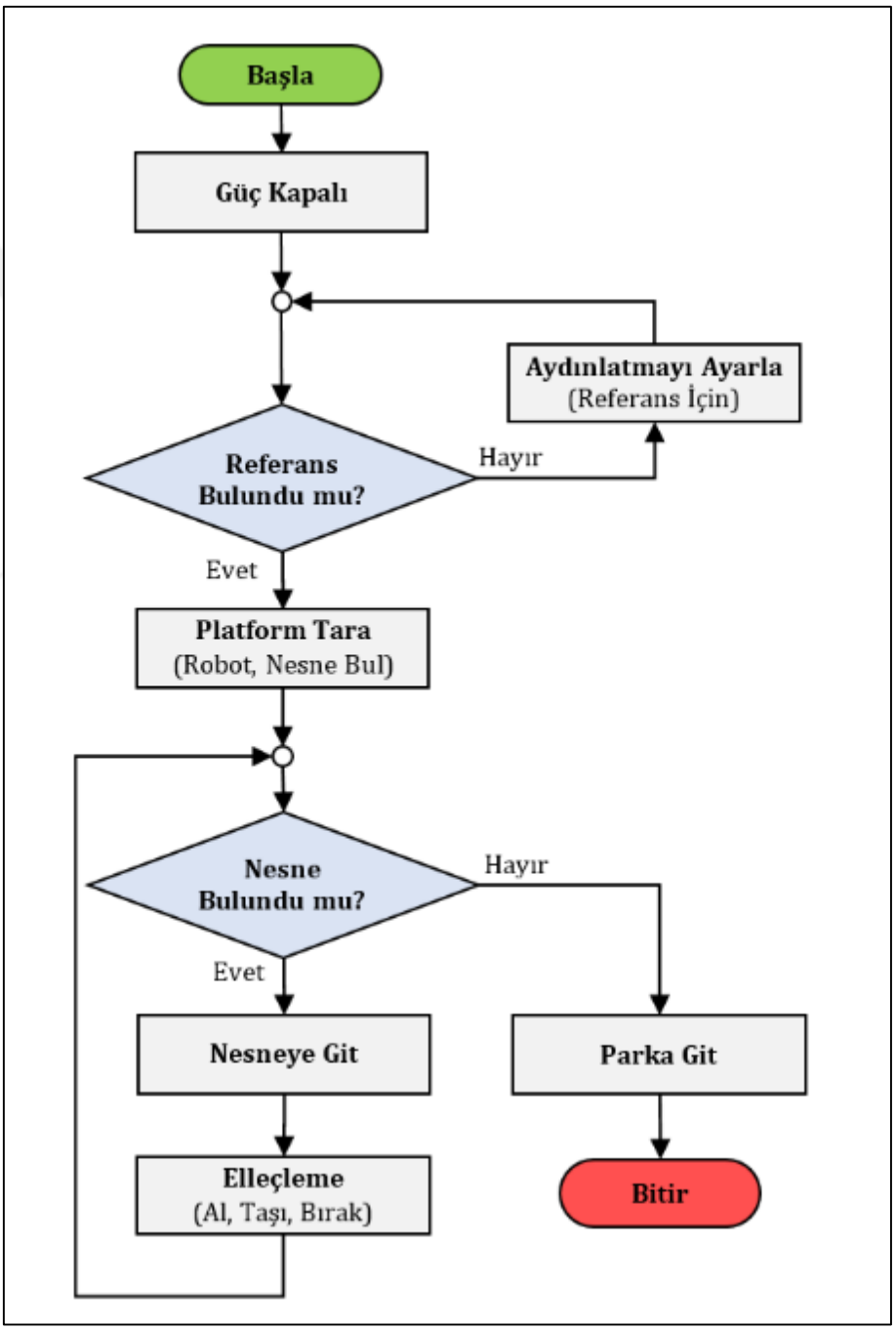
Resim 2.9. Robotik platform ve X-Y-Z koordinatları

Tablo 2.8. Tasarlanan sisteme yönelik referans renkler ve sabitlenen pozisyonları

Referans Renk		**Gerçek Pozisyonu**	
		X_r	**Y_r**
Kırmızı	(R-Red)	-20	250
Yeşil	(G-Green)	20	250
Mavi	(B-Blue)	-20	230
Sarı	(Y-Yellow)	20	230

Şekil 2.4'te arayüz yazılımı algoritmasının işleyişine yönelik akış şeması yer almaktadır. Geliştirilen algoritmada sistemin çalıştırılmasıyla birlikte; robotik platform için temelde 4 renk için referans kontrolü yapılarak ihtiyaç duyulan aydınlatma yapılmaktadır. Bu işlem için şerit LED lambanın gücü PWM ile ayarlanarak görüntü işleme için yeterli olan aydınlatma miktarı belirlenmektedir. Referansların bulunmasıyla birlikte platformda robot ve nesne tespiti

yapılmaktadır. Platform üzerinde nesne bulunmasıyla birlikte görüntü işleme tabanlı olarak merkez noktası belirlenmekte ve elde edilen G-Kodu talimatları CNC kontrolöre gönderilmektedir. Bu sayede robotun tespit edilen nesnenin merkez noktasına gitmesi sağlanmaktadır. Elleçleme işlemi kapsamında elektromıknatıs yardımıyla nesne alınıp tasnif yerine götürülerek bırakılmaktadır. Nesne bulunmadığı durumda robotun önceden belirlenen Park noktasına giderek beklemesi sağlanmaktadır.

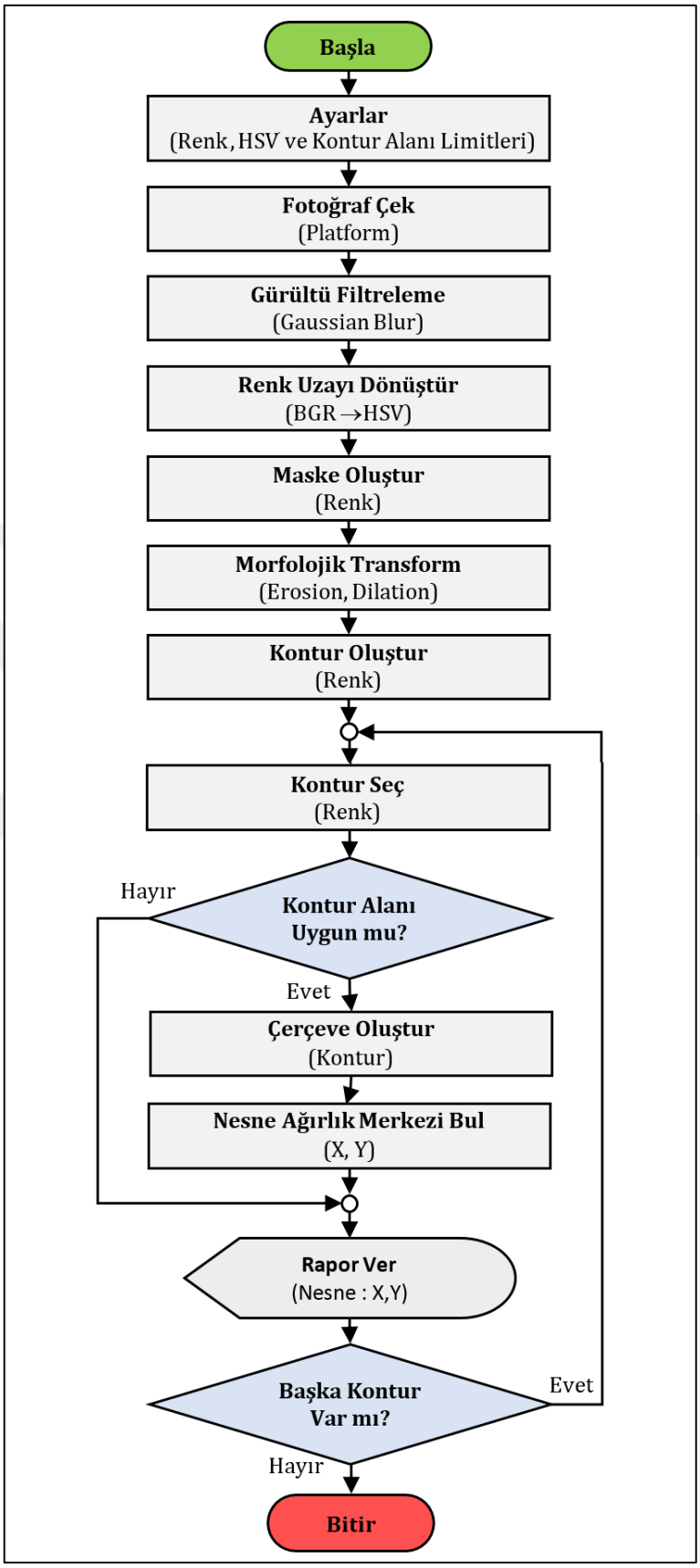


Şekil 2.4. Robotik sistem proses akış şeması

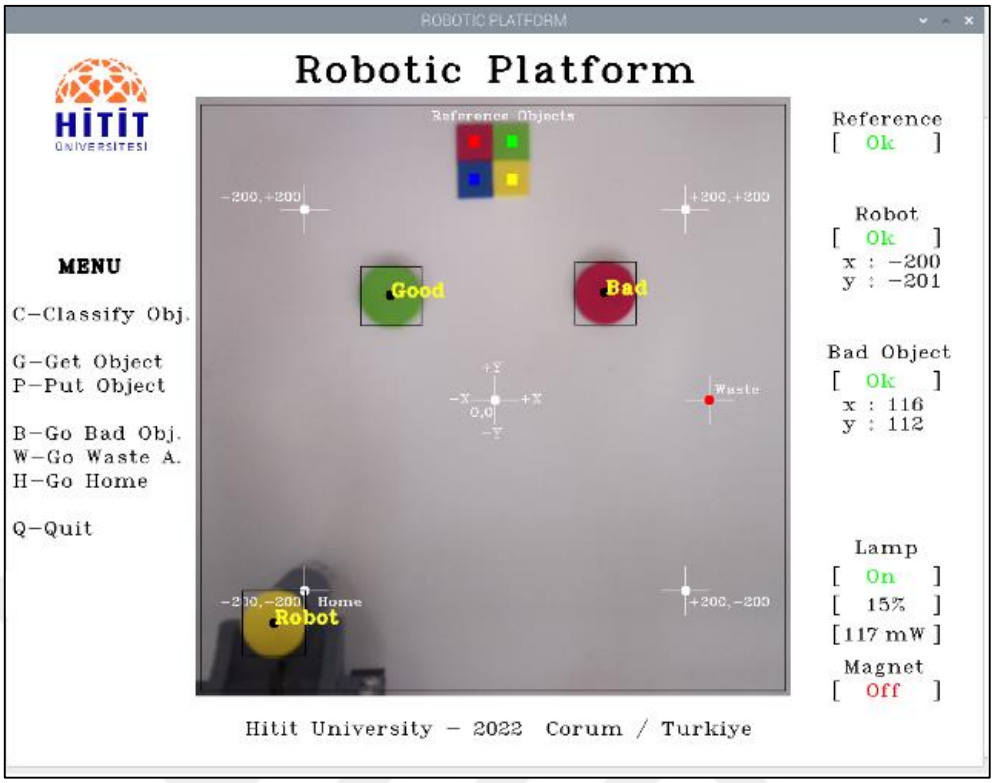
Şekil 2.5'te renk tabanlı görüntü işleme yöntemiyle nesne tespiti ve merkez noktasının bulunmasına yönelik akış şeması görülmektedir. Algoritmanın çalıştırılmasıyla birlikte öncelikle renk seçimi, ilgili renk için HSV renk uzayı alt ve üst sınır değerleri ve nesne tespitine yönelik olarak asgari alan büyüklüğü ayarları yapılır.

Başlangıç ayarlarından sonra sisteme bağlı kamera yardımıyla video üzerinden robotik platforma ait bir görüntü alınır ve sonrasında görüntüdeki ayrıntıları ve gürültüyü ortadan kaldırmak için Gaussian Blur filtreleme işlemi yapılır. OpenCV'de RGB renk formatı yerine renklerin farklı olarak dizildiği BGR (Blue-Green-Red) renk formatı kullanılmaktadır (Minichino ve Howse, 2015). Ancak görüntü işleme uygulamalarında nesne tanıma için daha iyi sonuçlar vermesinden dolayı görüntünün BGR renk uzayından HSV renk uzayına dönüşümü yapılır. Tanımlanan renk ve aralığına bağlı olarak maskeleme işlemi yapılır. Morfolojik Transform aşamasında, görüntünün gürültüden arındırılması için kenar erozyonu (erosion) ve genişletme (dilation) yapılarak görüntü yeniden düzenlenir. Tespit edilen renkli bölgelerin ayırt edilebilirliğini artırmak için kontur oluşturulur. Bir çevrim içerisinde nesne tespitine yönelik asgari alan büyüklüğünü sağlayan her bir kontur için öncelikle çerçeve çizilmekte ve moment hesaplamaları yardımıyla ağırlık merkezi bulunurak 2-eksen için (x, y) koordinatı belirlenerek raporlama işlemi gerçekleştirilmektedir.

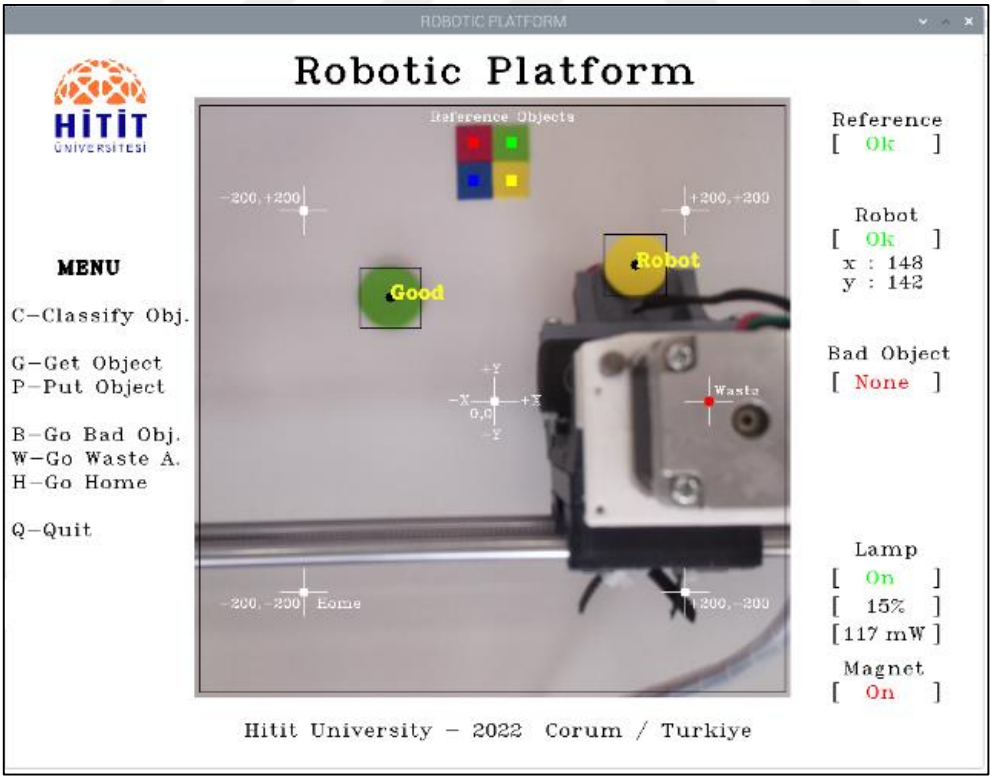
Sistemde renk tabanlı görüntü işleme ile nesne sınıflandırmasına yönelik örnek ekran görüntüleri Resim 2.10'da görülmektedir. Yazılım arayüzü menüsünde bulunan Sınıflandırma (C - Classify Obj.) seçeneği kullanıldığında, öncelikle pozisyonları bilinen referans renklerin tespitine göre şerit LED lambanın elektriksel aydınlatma gücü ayarlanarak sistem kullanıma hazır hale getirilmektedir. Platform aydınlatmasının sağlanmasının ardından platformdaki tespit edilen Kırmızı (Kötü/Bad) nesnenin merkez noktasına robot hareketi sağlanmaktadır. Robot tarafından elektromıknatıs yardımıyla Kırmızı (Bad) nesne alınıp Atık (Waste) Alanına götürülerek bırakılmaktadır. Kırmızı (Bad) nesnenin bırakılmasının ardından Robot Park (Home) Alanına gelerek işlem sonlandırılmaktadır.



Şekil 2.5. Renk tabanlı görüntü işleme ile nesne tespiti akış şeması

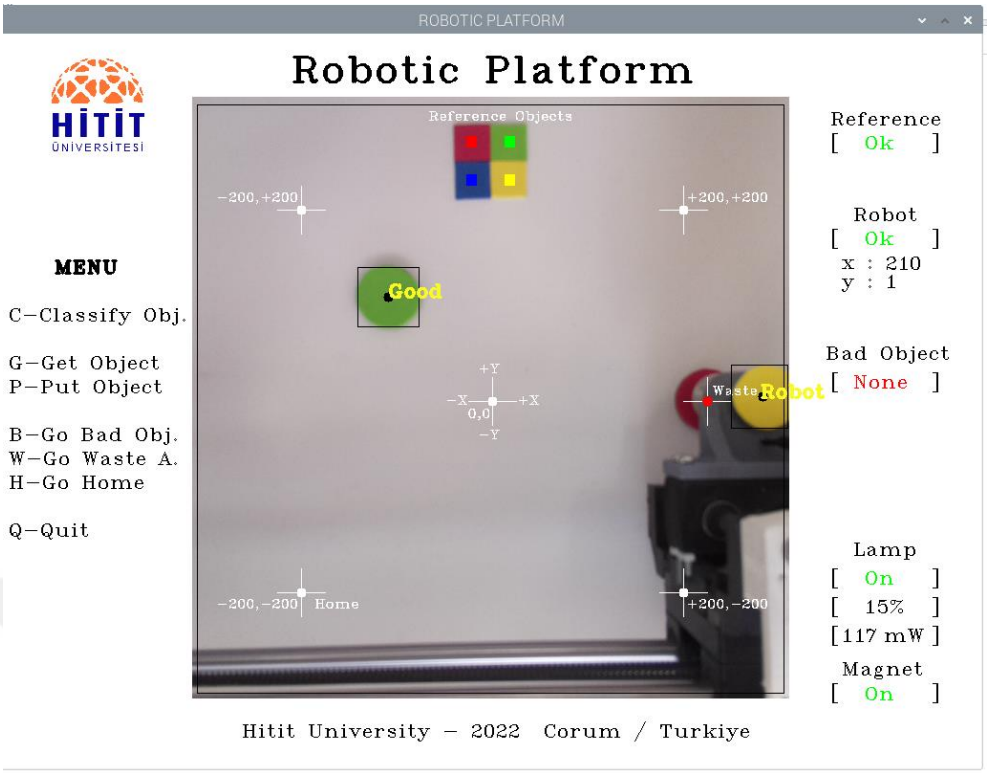


(a) Robot Pozisyonu - 1: Park (Home) Alanı

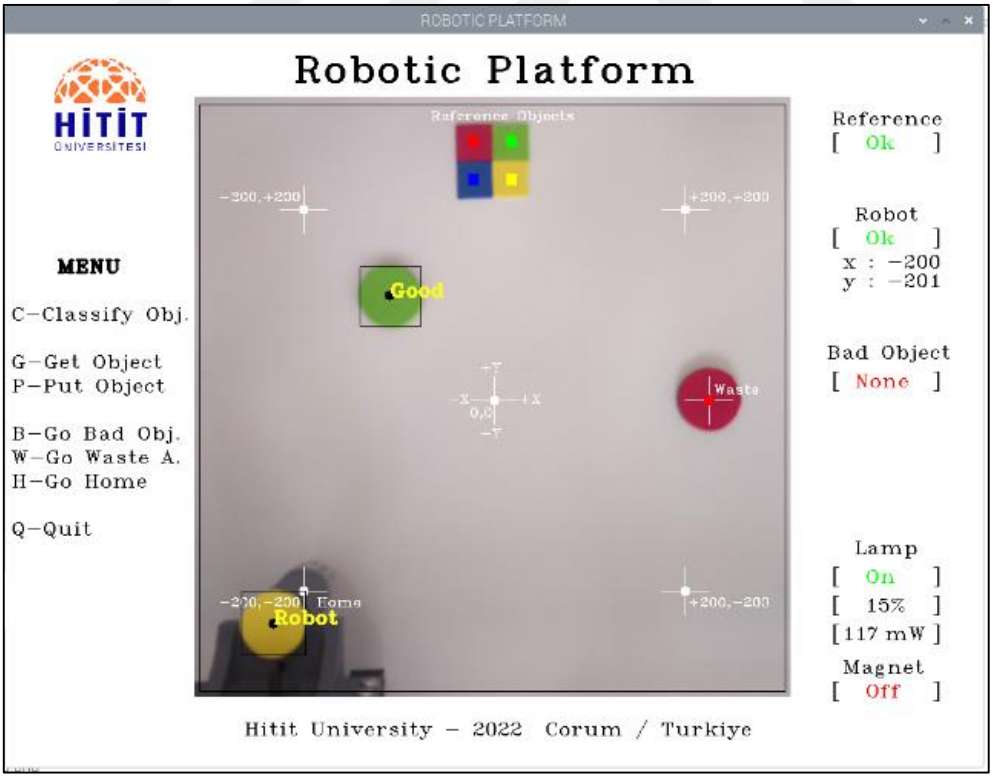


(b) Robot Pozisyonu- 2 : Kırmızı (Bad) Nesne

Resim 2.10. Renk tabanlı görüntü işleme ile nesne sınıflandırma



(c) Robot Pozisyonu - 3: Atık (Waste) Alanı



(d) Robot Pozisyonu - 4: Park (Home) Alanı

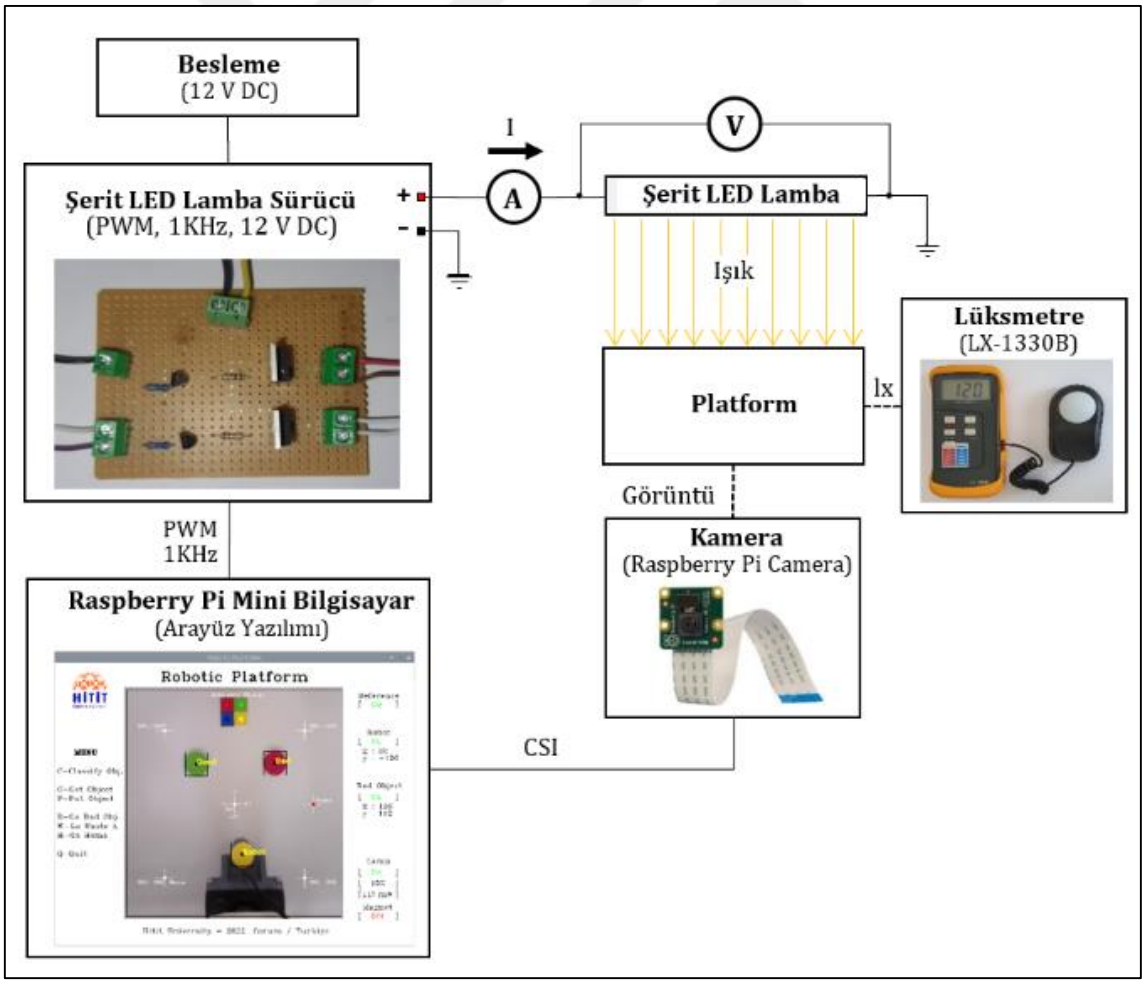
Resim 2.10. Renk tabanlı görüntü işleme ile nesne sınıflandırma (Devam)

3. BÖLÜM

ANALİZ VE BULGULAR

3.1. Deneysel Düzenek

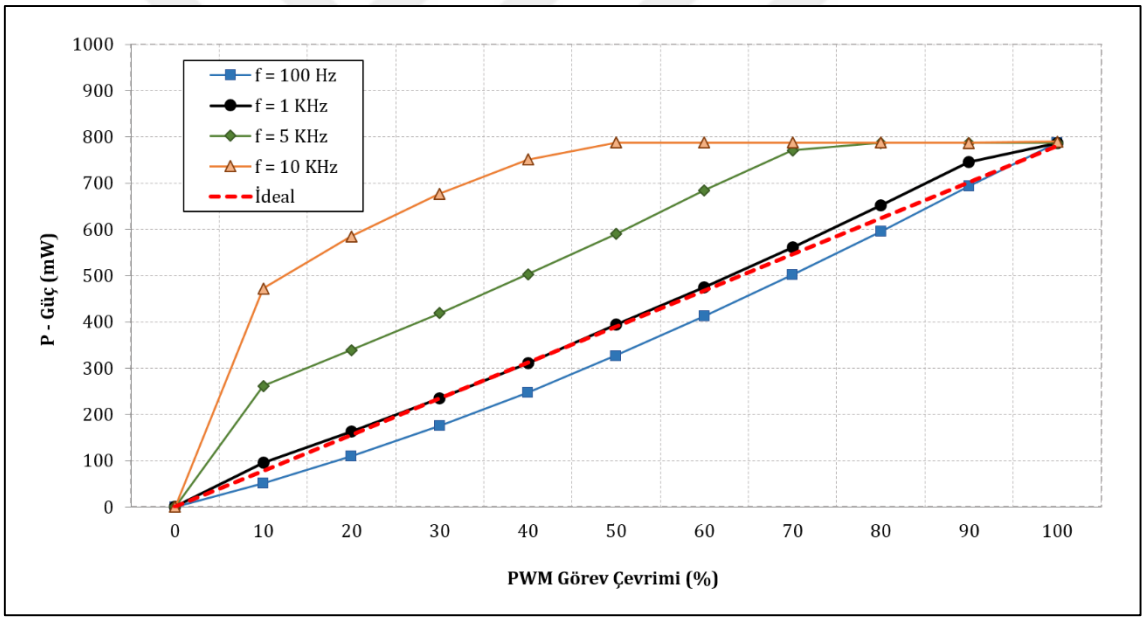
Tasarlanan sistemin performansının incelenmesine yönelik deneysel çalışmalar yapmak üzere Şekil 10'da görülen düzenek oluşturulmuştur. Düzenekte ortam aydınlatması için beyaz ışık veren 12 V DC 780 mW difüzörlü şerit LEDlamba ve MOSFET tabanlı sürücü devre kullanılmıştır. Kamera ile alınan görüntülerin işlenmesi için Raspberry Pi mini bilgisayar kullanılmıştır. Deneysel düzenekte harici olarak robotik platforma yönelik aydınlık düzeyinin ölçülmesi amacıyla LX-1330B model dijital lüksmetre ile şerit LED lamba gücünün belirlenmesi amacıyla akım ve gerilim ölçümlerine yönelik için True RMS dijital multimetre kullanılmıştır. Deneysel çalışmada beyaz ışık ile aydınlatma yapılmış, platform tabanında yansıması düşük beyaz fon kullanılmış ve dış ortam etkilerini ortadan kaldırmak amacıyla sistemin üzeri siyah kumaş kılıf ile örtülmüştür.



Şekil 3.1. Deneysel ölçüm düzeneği

3.2. Şerit LED Lamba Sürücüsü PWM Performans İncelemesi

Deneysel çalışmada öncelikle patform aydınlık düzeyinin ayarlanması amacıyla kullanılan sürücü devrenin anahtarlama performansı incelenmiştir. Bu amaçla PWM sinyali ile sürülen MOSFET tabanlı devrenin frekans ve görev çevrimi değişimine bağlı olarak şerit LED lambanın güç değişimi araştırılmıştır. Deneysel düzenekte PWM frekansı 100 Hz, 1 KHz, 5 KHz ve 10 KHz için görev çevriminin %0 - %100 arası değişimine bağlı olarak şerit LED lamba ya uygulanan güç değişimi elde edilmiştir (Şekil 3.2). Aydınlatma sisteminde kırpışmayı en aza indirmek amacıyla, şerit LED lamba Raspberry Pi GPIO18 pin çıkışından donanımsal PWM sinyali ile sürülmüştür. Tasarlanan sistemde aydınlatmadaki kırpışmanın önlenmesi amacıyla sürücü devrenin PWM frekansı ideale en yakın olduğu için 1KHz olarak belirlenmiştir. Sistemde daha yüksek frekanslarda görev çevrimine bağlı olarak anahtarlamanın düzgün yapılabilmesi amacıyla, palsin verilmesi ve kesilmesi için ayrı ayrı olmak üzere çift MOSFET ile sürme işleminin gerçekleştirilmesi gerekmektedir.

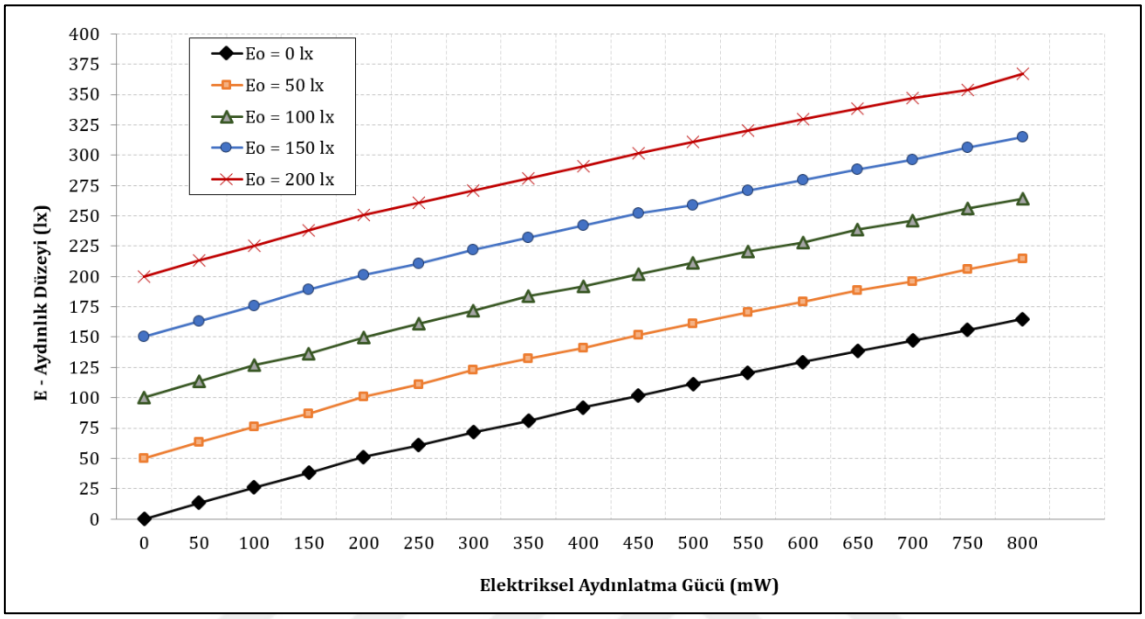


Şekil 3.2. Farklı PWM frekansı ve görev çevrimleri için şerit LED lamba sürücüsünün anahtarlama performansına bağlı çıkış gücü değişimleri

3.3. Sistemde Güce Bağlı Aydınlık Düzeyi Değişimlerinin İncelenmesi

Deneysel çalışmanın ikinci aşamasında doğal ve çevresel faktörlere göre değişiklik gösteren farklı mevcut aydınlık düzeylerinde aydınlatma gücüne bağlı olarak platform aydınlık düzeyi değişimleri araştırılmıştır. Bu kapsamda çevresel aydınlık düzeyi 0 lx, 50 lx, 100 lx, 150 lx ve 500 lx iken; LED sürücü devre ile PWM sinyali (1 KHz) görev çevrimi %0 ile %100 arasında

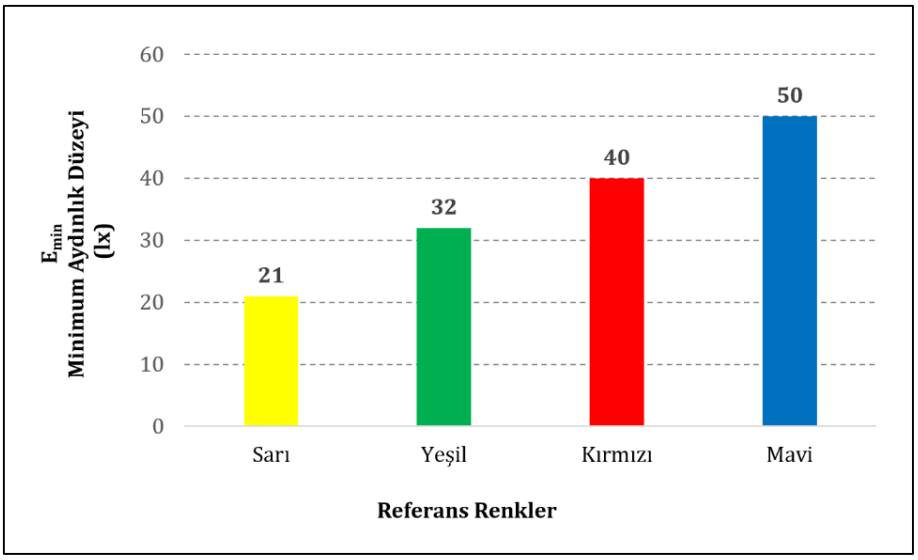
ayarlanarak elde edilen elektriksel aydınlatma gücüne bağlı platform aydınlık düzeyi değerleri dijital lüksmetre ile ölçülmüştür (Şekil 3.3). Aydınlatma sisteminde elektriksel güç değişimine bağlı olarak aydınlık düzeyinin yaklaşık doğrusal olarak arttığı görülmektedir.



Şekil 3.3. Elektriksel aydınlatma gücüne bağlı olarak platform aydınlık düzeyi değişimleri

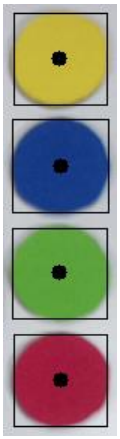
3.4. Renk Tabanlı Görüntü İşlemede Asgari Aydınlık Düzeyi Değerlerinin İncelenmesi

Deneysel çalışmanın üçüncü aşamasında; nesne tespit işlemlerine yönelik renk tabanlı görüntü işleme için farklı renklere göre ihtiyaç duyulan minimum aydınlık düzeyleri araştırılmıştır. Bu amaçla beyaz renkli robotik platform siyah örtü ile örtülerek mevcut aydınlık düzeyi 0 lx iken; 2,5 cm çaplı Kırmızı, Yeşil, Mavi ve Sarı renkli nesneler kullanılarak PWM çıkış sinyali ile elektriksel aydınlatma gücü ayarlanarak görüntü işlemede başarılı sonuçların alındığı minimum aydınlık düzeyleri dijital lüksmetre ile ölçülmüştür (Şekil 3.4). Renk tabanlı görüntü işleme ile nesne tanımada rengin önemli olduğu ve en az aydınlık düzeyine dolayısıyla en az enerji harcamasına ihtiyaç duyulan rengin sarı (E_{min} = 21 lx) olduğu ve bunu sırasıyla yeşil (E_{min} = 32 lx), kırmızı (E_{min} = 40 lx) ve mavi (E_{min} = 50 lx) renklerin takip ettiği görülmüştür. Tablo 3.1'de farklı aydınlık düzeylerinde Kırmızı, Yeşil, Mavi ve Sarı renkli nesneler için elde edilen görüntüler ve tespit edilme durumları görülmektedir.



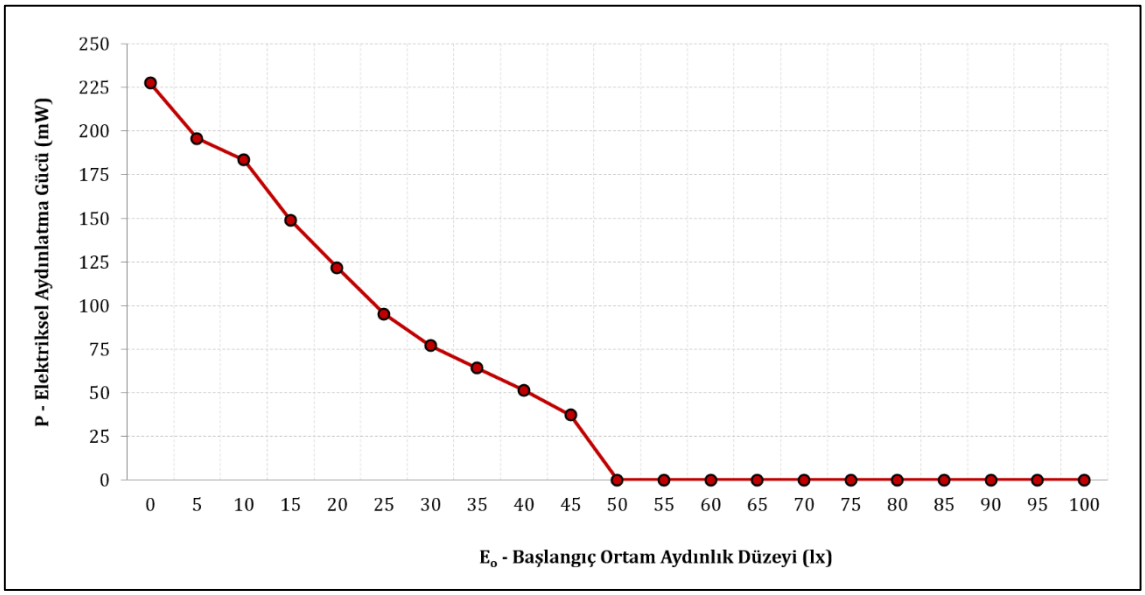
Şekil 3.4. Renk tabanlı görüntü işleme için farklı renklere göre ihtiyaç duyulan minimum aydınlık düzeyleri

Tablo 3.1. Farklı renge sahip nesneler için çeşitli aydınlık düzeylerinde elde edilen görüntüler ve tespit edilme durumları

Nesne	**Tespit Edilme Durumu**						
	E = 0 lx	E = 10 lx	E = 21 lx	E=32 lx	E=40 lx	E=50 lx	E=100 lx
Referans RGBY Nesne							
Sarı Nesne Mavi Nesne Yeşil Nesne Kırmızı Nesne							

3.5. Ortam Aydınlık Düzeyine Bağlı Harcanan Elektriksel Aydınlatma Gücü İncelemesi

Çalışmanın dördüncü aşamasında, nesne tespiti ve pozisyon belirlenmesi için çevresel aydınlık düzeyine bağlı olarak sistem tarafından harcanan elektriksel aydınlatma gücü değerleri araştırılmıştır. Bu amaçla 0 – 100 lx arası farklı ortam aydınlık düzeyleri için sistem çalıştırılarak asgari PWM görev çevrimi değerine bağlı şerit LED lambanın harcadığı elektriksel güç değerleri elde edilmiştir (Şekil 3.5). Sistemde çevresel aydınlık düzeyi 0 lx iken aydınlatma için harcanan güç yaklaşık olarak 225 mW iken bu değer ortam aydınlık düzeyiyle ters orantılı olarak düşmekte ve 50 lx üzerinde aydınlatma gücü kapatılarak 0 W olmaktadır.



Şekil 3.5. Farklı ortam aydınlık düzeyleri için tasarlanan sisteme yönelik gerekli olan elektriksel aydınlatma gücü değerleri

SONUÇ VE ÖNERİLER

Bu tez çalışmasında, robotik elleçleme uygulamalarına yönelik görüntü işlemeye dayalı nesne tespiti ve pozisyon belirlenmesi için referans renk tabanlı akıllı ortam aydınlatması yapan bir gömülü sistem tasarlanmıştır. Çalışmada kamera ile elde edilen görüntüler Python programlama dilinde OpenCV kütüphanesi kullanılarak işlenmiş, renk tabanlı nesne tespiti ve gerekli ortam aydınlatması yapılarak tasnif işlemi gerçekleştirilmiştir. Çalışma kapsamında aydınlatma için platform üzerinde U şeklinde konumlandırılmış beyaz ışık veren 12 V DC 780 mW şerit LED lamba kullanılmıştır. Beyaz ışık kullanımı ile nesne renklerinin doğru sıcaklıkta elde edilmesi sağlanmıştır. Işığın platform üzerinde homojen dağılımı ve parlama yapmaması amacıyla LED önüne difüzör plakalar konulmuştur. Ayrıca taban üzerinden yansıma olmaması için fotoğrafçılıkta kullanılan mat beyaz renkli fon perdesi kullanılmıştır. Aydınlatmada ışık kırpışmasını engellemek amacıyla Raspberry Pi minibilgisayarın donanım tabanlı PWM çıkışı kullanılmıştır. Ayrıca düşük frekanslarda PWM sinyali ile şerit LED lambanın sürülmesi ışık kırpışmasına yol açtığından PWM frekansı 1 KHz olarak ayarlanmıştır.

Geleneksel yöntemlerde görüntü işleme için gerekli ortam aydınlatması ışık kaynağı sürekli açık bırakılarak, elle aç/kapat yapılarak, zamanlayıcı kullanılarak veya aydınlık düzeyi ölçülerek otomatik olarak yapılmaktadır. Bu çalışmada ise renk tabanlı görüntü işleme ile referans renklerin tanınmasına dayalı uyarlamalı aydınlatma işlemi yapılmıştır. Sistemde nesne tespitine yönelik görüntü işleme için gerekli olan ışık şiddeti seviyesi asgari düzeye getirilerek hem ortam aydınlatmasında enerji tasarrufu sağlanmakta hem de aşırı aydınlatmanın olumsuzlukları giderilmiş olmaktadır.

Çalışma kapsamında tasarlanan sistemin performansının belirlenmesi amacıyla akım, gerilim ve aydınlık düzeyi ölçümlerinin yapılabileceği bir deneysel düzenek oluşturulmuştur. Deneysel çalışmada çevresel aydınlık düzeyi 0 lx iken PWM ile sürülen aydınlatma sisteminde görev çevriminin %0 - %100 arası değişimine bağlı olarak elektriksel aydınlatma gücü 0 – 780 mW arasında ayarlanmış ve elde edilen aydınlık düzeyinin 0-163 lx arasında yaklaşık doğrusal artış gösterdiği görülmüştür. Sistemde renk tabanlı görüntü işleme ile nesne tanımada Kırmızı, Yeşil, Mavi ve Sarı olmak üzere farklı renklere sahip referans nesneler kullanılarak gerekli olan ortam aydınlatması değerleri araştırılmıştır. Çevresel aydınlık düzeyi 0 lx iken renk tabanlı görüntü işleme ile nesne tespiti için yeterli olan harcanan minimum güç (W_{min}) ve minimum aydınlık düzeyi (E_{min}) değerleri ölçülmüştür. Elde edilen bu değerler sırasıyla; sarı renkli nesne için (80 mW, 21 lx), yeşil renkli nesne için (125 mW, 32 lx), kırmızı renkli nesne için (160 mW, 40 lx) ve mavi renkli nesne için (202 mW, 50 lx) olmuştur.

Elde edilen sonuçlara göre; renk tabanlı görüntü işleme ile nesne tanımada renk seçiminin önemli olduğu ve bununla birlikte en az aydınlık düzeyine dolayısıyla en az enerji kullanımına ihtiyaç duyulan rengin ışık yansıtma değerine bağlı olarak sarı olduğu ve bunu sırasıyla yeşil, kırmızı ve mavi renklerin takip ettiği görülmüştür. Beyaz ışık ile beyaz fon üzerine yapılan aydınlatmada, ışık yansıtma değerine bağlı olarak mavi renkli nesnelerin tespiti için daha fazla

aydınlık düzeyine ihtiyaç duyulduğu anlaşılmıştır. Bu sebeple nesne tespitine yönelik optimum aydınlık düzeyinin belirlenmesi için referans olarak mavi renkli nesne kullanılmasının yeterli olduğu sonucuna ulaşılmıştır. Tasarlanan sistemde yaklaşık 50 lx aydınlık düzeyinde renk tabanlı nesne tespitinin başarılı bir şekilde yapılabileceği ve çevre aydınlatmasına bağlı olarak yaklaşık 0-225 mW arasında değişen bir güce ihtiyaç duyulduğu görülmüştür.

Sonuç olarak, akıllı ortam aydınlatması yapan sistemin çevresel koşullara bağlı olarak platform için asgari aydınlık düzeyi sağladığı ve endüstriyel uygulamalarda enerji tüketimini azaltacağı anlaşılmıştır. Gelecekte insan gözüyle takibin yapılmadığı sadece robotların kullanıldığı platformlarda daha fazla enerji tasarrufu sağlamak amacıyla flaş şeklinde anlık aydınlatma yapılarak elde edilen görüntüler üzerinde nesne tespiti ve koordinat belirleme işlemleri yapılabilir. Ayrıca farklı çözüm yöntemleri geliştirilerek ışık kaynağının şiddetinin yanı sıra yönü, açısı, bölgeselliği ve rengi de ayarlanarak ortam aydınlatmasının performansı artırılabilir ve daha iyi sonuçlar alınabilir.

KAYNAKLAR

Adelkhani, A., Beheshti, B., Minaei, S. ve Java, P. (2012). Optimization of Lighting Conditions and Camera Height for Citrus Image Processing. *World Applied Sciences Journal*, 18(10), 1435-1442.

Albayrak, S., (2001). *Renk Özelliği ile İçerik Tabanlı Görüntü Erişimi,* (Doktora Tezi), İstanbul: Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü.

Altınkurt, Ö. ve Kahriman, M. (2011). *Gerçek Zamanlı Olarak, Anfis İle Renk Tabanlı Nesne Tespit ve Motorlu Sistem ile Takip Edilmesi.* Teknik Bilimler Dergisi, 1(1), 1-5.

Arduino, (2022). Arduino web sayfası. Erişim Tarihi: 01 Haziran 2021. https://www.arduino.cc

Aydoğdu, E. (2019*). Mevcut Ticari Binaların Aydınlatma Sistemlerinde Enerji Verimliliği Analizi İçin Örnek Bir Çalışma, (Yüksek Lisans Tezi),* İstanbul: İstanbul Teknik Üniversitesi Enerji Enstitüsü.

Aydoğduoğlu, O. (2021*). Enerji Verimliliği Sağlamada Akıllı Aydınlatma Sistemi Uygulaması Manisa Celal Bayar Üniversitesi Rektörlük Binası Örneği*, (Yüksek Lisans Tezi), Manisa: Manisa Celal Bayar Üniversitesi Fen Bilimleri Enstitüsü.

Barnett, C. (2013). Lumen, Lux & Candela – an introduction. *Lyco*. Erişim Tarihi: 01 Aralık 2022. https://www.lyco.co.uk/advice/lumen-lux-candela-intro/

Barwar, M.K., Sahu, L.K. ve Bhatnagar, P. (2022). Reliability analysis of flicker-free LED driver based on five-level rectifier, *Optik*, 268, 169762, 1-13.

Bayram, F. (2009). Işık ve Aydınlatma: Işığın Televizyon ve Sinemada İşlevsel Kullanımı Üzerine Bir Değerlendirme. *Erciyes İletişim Dergisi*, 1(2), 122-131.

Bilici, A. (2019*). Led Lambalarla Verimli Bir Aydınlatma Sisteminin Gerçekleştirilmesi*, (Yüksek Lisans Tezi), Konya: Selçuk Üniversitesi Fen Bilimleri Enstitüsü.

Büyükarıkan, B. (2014*). Görüntü İşleme Teknikleri Kullanarak Işık Havuzunda ki Cisimlerin Optimum Aydınlatma Koşullarının Belirlenmesi ve Uygulanması,* (Yüksek Lisans Tezi), Isparta: Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü.

Büyükarıkan, B. ve Üncü, İ. S. (2019). Bilgisayarlı Görü Sistemleri İçin Sistem Tasarımı ve Kontrolü. *Selçuk Üniversitesi Mühendislik, Bilim ve Teknoloji Dergisi*, 7(1) 228-240.

Büyükkoçak, Y. (2018). *Görüntü İşleme Tabanlı Aydınlatma Ölçüm Sistemi Tasarımı ve Uygulaması*, (Yüksek Lisans Tezi), Bilecik: Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Enstitüsü.

Chew I., Karunatilaka D., Tan C. P. ve Kalavally V. (2017). Smart lighting: The way forward? Reviewing the past to shape the future. *Energy and Buildings*, 149, 180–91.

Çan, Z. (2021). *Otomasyon Sistemlerinde Görüntü İşleme Tekniklerini Kullanan Ürün Tanımı Uygulaması,* (Yüksek Lisans Tezi), Sakarya: Sakarya Üniversitesi Fen Bilimleri Enstitüsü.

Çetin, Ö. (2008). *Hareketli Görüntü Uygulamaları İçin Sırörtme Yaklaşımı ile Veri Gömme Algoritması Tasarımı*, (Doktora Tezi), Sakarya: Sakarya Üniversitesi Fen Bilimleri Enstitüsü.

Dehoff P., Di Fraia, L., Henderson, R., Julian, W., Juslen, H., Kaplan, H., Katayama, S., Lillelien, E., Schierz C., Stockmar, A., Vonnak, I., Wisniewski, A. Ve Zonneveldt, L. (2005). Guide On The Maintenance Of Indoor Electric Lighting Systems. *CIE*, 97, 11-14.

Demirdeş, H. (1993). Uygun Aydınlatma Bileşenleri. *Kaynak Elektrik Dergisi*. 6, 67-68.

Deniz, H. İ. (2019). *Mikroişlemci Tabanlı 3 Boyutlu Cnc Platform Geliştirilmesi*, (Yüksek Lisans Tezi), Ankara: Milli Savunma Üniversitesi Alparslan Savunma Bilimleri Enstitüsü.

Dersuneli, M., Gündüz, T. ve Kutlu Y. (2021). Bul-Tak Oyuncağı Şekillerinin Klasik Görüntü İşleme ve Derin Öğrenme Yöntemleri ile Tespiti*. Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, *10*(4), 1290-1303.

Dişlitaş, S. (2015). *Endüstriyel Robot Programlama*, AB Projesi / TRH2.2.IQVETII/P-03/921, ISBN: 978-605-344-294-3, Çorum, Türkiye.

Enerji Verimliliği Kanunu. (2007). T.C. Resmi Gazete (26510, 5 Mayıs 2002).

Fan, L., Wang, Z., Cail, B. and Tao, C. (2016). A Survey on Multiple Object Tracking Algorithm. *Proceedings of the IEEE International Conference on Information and Automation*, 1855-1862, Ningbo, China.

Gandhi, A. ve Sangeetha, M. (2018). Development of an Image Processing Algorithm for Smart CNC Machines. *IEIE Transactions on Smart Processing & Computing*, *7*(3), 232-235.

Gonzales, R. C. and Woods, R. E. (2008). *Digital Image Processing (3rd Edition).* Prentice Hall.

Gökmen, M. R. (2010). *Endüstri Tesislerinde Enerji Verimli Aydınlatma Teknikleri ve Örnek Çalışma,* (Yüksek Lisans Tezi), İstanbul: İstanbul Teknik Üniversitesi Enerji Enstitüsü.

Halis, M. (2000). İş Yaşamı Kalitesi Açısından Çalışma Ortamında İnsan-Renk Etkileşimi. *Verimlilik*. (2000/2), 65-81.

Hanbay, K. ve Üzen, H. (2017). Nesne tespit ve takip metotları: Kapsamlı bir derleme*. Türk Doğa ve Fen Dergisi*, *6*(2), 40-49.

Hema, D. ve Kannan, S. (2019). Interactive Color Image Segmentation using HSV Color Space. *Science and Technology Journal*, 7(1), 37-41.

Hsu, I. (2016). LED brightness adjustment: high-frequency PWM dimming. Texas Instruments. Erişim Tarihi: 10 Mayıs 2022. https://e2e.ti.com/blogs_/b/powerhouse/posts/led-brightness-adjustment-high-frequency-pwm-dimming

IESNA, (2000). Light and Optics. Rea, M.S. (Ed*.), The Illuminating Engineering Society of North America Lighting Handbook*. New York: Illuminating Engineering Press.

Imamguluyev, R. (2021). Optimal Solution Based On Fuzzy Logic Of Light Reflectance Value In Interior Lighting, *Technical sciences, 12th International Scientific and Practical Internet Conference "Modern Movement of Science",* Ukrayna, 69-71,

Işık, M. (2022), OpenCV ve Python ile Görüntü İşleme, Seçkin Yayıncılık, 41-341.

Kapusuzoğlu, F. (2020). *Uzaktan Kontrollü Mikroişlemci Tabanlı Akıllı Aydınlatma Devresi,* (Yüksek Lisans Tezi), Kocaeli: Kocaeli Üniversitesi Fen Bilimleri Enstitüsü.

Kruger, B. (2015), Arduino CNC Shield. Electronic Prototyping Specialists. Erişim Tarihi: 15 Ekim 2022. https://blog.protoneer.co.nz/arduino-cnc-shield/

Linnartz, J. P. M. G., Feri, L., Yang, H., Colak, S. B., & Schenk, T.C.W. (2008, May). Communications and Sensing of Illumination Contributions in a Power LED Lighting System, *ICC'08 IEEE International Conference*, 5396-5400, Beijing, China.

Li-Li, Z., Yan-Hua, W., Xue-Feng, Z., & Hong-Yu, L. (2013, July). Implementation of a Novel LED Backlight Device Used for Glass Bottle Detection, *2013 Seventh International Conference on Image and Graphics*, 766-769, Qingdao, China.

Lucas, S. (2022). Color Systems. Erişim Tarihi: 13 Haziran 2022. https://color.lukas-stratmann.com/color-systems.html

Macit, H. B. (2020), Python ve OpenCV ile RGB üzerinde HSV renk kodu tespiti. Erişim Tarihi: 16 Kasım 2022. https://www.hbmacit.com/2020/09/15/python-ve-opencv-ile-rgb-uzerinde-hsv-renk-kodu-tespiti

Manipriya, S., Mala, C. ve Mathew, S. (2014). Performance Analysis of Spatial Color Information for Object Detection Using Background Subtraction. *IERI Procedia*, 10, 63-69.

Minichino, J. ve Howse, J. (2015). *Learning OpenCV 3 Computer Vision with Python*. Second Edition, Packt Publishing, Birmingham-Mumbai, ISBN 978-1-78528-384-0.

Neida, B. V., Manicria, D. ve Tweed, A. (2001). An analysis of the energy and cost savings potential of occupancy sensors for commercial lighting systems*. J. Illum. Eng. Soc.* 30 (2), 111–125.

Oh, J. H., Yang, S. J. and Do, Y. R. (2014). Healthy, natural, efficient and tunable lighting: four package white leds for optimizing the circadian effect, color quality and vision performance*. Light Sci. Appl.* 3 (2), 141.

Onaygil, S. (2000). Aydınlatma Aygıt Tasarımı Temel İlkeleri Dersi. Erişim Tarihi: 15 Ekim 2022. http://web.itu.edu.tr/~onaygil/eut339/isik_fotometrik_buyuklukler.doc

Onaygil, S. (2001). Kent İçi Aydınlatma. *Kaynak Elektrik Dergisi*. 107-112.

Özcan, T. (2010). Hareketli Nesnelerde Yüz Tespitine Yönelik Bir Uygulama, (Yüksek Lisans Tezi), Edirne: Trakya Üniversitesi Fen Bilimleri Enstitüsü.

Özkaya, M. ve Tüfekçi, T. (2011). Aydınlatma Tekniği, 1. Baskı, İstanbul: Birsen Yayınevi.

Raspberry Pi, (2022). Raspberry Pi Foundation web sayfası. Erişim Tarihi: 10 Eylül 2021. https://www.raspberrypi.com

Sarguroh, S. S. ve Rane, A. B. (2018). Using GRBL-Arduino-based controller to run a two-axis computerized numerical control machine. In *2018 International Conference on Smart City and Emerging Technology (ICSCET)* (pp. 1-6). IEEE.

Sarıyıldız, S. Ö. ve Demirhan, A. (2021). Görüntü İşleme Teknikleri ve Robot Kol şle Nesneleri Kategorilerine Ayırma. *Uludağ Üniversitesi Mühendislik Dergisi*, *26*(2), 547-556.

Singh, M. ve Garg, S. (2010), Illuminance estimation and daylighting energy savings for Indian regions. *Renewable Energy*, 35 (3), 703-711.

Skaria, S., John, M., Paul, B. (2014). Automatic lighting controller. *International Journal of Engineering Research and Development*, 10(2), 29-36.

Stump, D. R. (2004). Electromagnet. *Encyclopedia of Energy Reference Works,* Ed. Cutler J. Cleveland, (ss. 319-328), United States: Elsevier Science.

Şenel, F. A. ve Cetişli, B. (2015*).* Görüntü İşleme ve Beş Eksenli Robot Kol İle Üretim Bandında Nesne Denetimi. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 21(5), 158-161.

Tiwari, M. ve Singhai, R. (2017). A Review of Detection and Tracking of Object from Image and Video Sequences. *International Journal of Computational Intelligence Research*. 13(5), 745-765.

TS EN 12464-1. (2001). Işık ve Aydınlatma - Çalışma Yerlerinin Aydınlatılması - Bölüm 1: Kapalı Çalışma Alanları, 30.09.2021.

Uzer, M., Yılmaz, N. ve Bayrak, M. (2010). Görme Tabanlı Mobil Robot ile Farklı Renklerde Nesnelerin Gerçek Zamanlı Takibi. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 25(4), 759-766.

Ünal A. (2014). *Aydınlatma Tasarımı ve Proje Uygulamaları*, 2. Baskı, İstanbul: Birsen Yayınevi.

Yıldırım, M. Y. (2019). *Gerçek Zamanlı Görüntü İşleme Temelli Al - Bırak Yapabilen Endüstriyel Robot Kol,* (Yüksek Lisans Tezi), Karabük: Karabük Üniversitesi Fen Bilimleri Enstitüsü.

Yılmaz, A. (2007). *Kamera Kullanılarak Görüntü İşleme Yoluyla Gerçek Zamanlı Güvenlik Uygulaması*, (Yüksek Lisans Tezi), İstanbul: Haliç Üniversitesi Fen Bilimleri Enstitüsü.

Yılmaz, İ., Güllü, M., Baybura, T. ve Erdoğan, O. (2002). Renk Uzayları ve Renk Dönüşüm Programı (RDP), *Afyon Kocatepe Üniversitesi Fen Bilimleri Dergisi*, 2(2), 19-35.

Yılmaz, M. A. ve Sungur, C. (2021*).* Görüntü İşleme ve Robot Kol Tabanlı Çikolata Toplama ve Paketleme Sistemi. *Avrupa Bilim ve Teknoloji Dergisi*, (30), 79-82.

Yücel, U. (2019). *Aydınlatma Sistemlerinde Yüksek Enerji Verimliliği İçin Kontrol Sistemi Tasarımı ve Uygulaması*. Kocaeli: Kocaeli Üniversitesi Fen Bilimleri Enstitüsü.

Zettl, H. (1999). *Sight, Sound Motion. Applied Media Aesthetics*. USA: Wadsworth Publication. 54-59.

EKLER

EK-1. Sistem Yazılımı (Python)

```
# ------------------------------------------------------------------------------------------------------------
# ------------------------------------------------------------------------------------------------------------
# Program Adı      : prj_Robotik_Sistem.py
# Programlama Dili : Python 3.7.3
# Tarih            : 30 Aralık 2022
# Tez Türü         : Yüksek Lisans Tezi
# Tez Adı          : Robotik Sistemlerde Görüntü İşleme Tabanlı Nesne Tanıma İçin Akıllı Ortam Aydınlatması
# Üniversite       : Hitit Üniversitesi
# Enstitü          : Lisansüstü Eğitim Enstitüsü
# Ana Bilim Dalı   : Enerji Sistemleri Mühendisliği ABD
# Öğrenci          : Uğur AKIŞ
# Danışman         : Dr. Öğr. Üyesi Serkan DİŞLİTAŞ
# ------------------------------------------------------------------------------------------------------------
# ------------------------------------------------------------------------------------------------------------

# Paketleri Yukle ---------------------------------------------------------------------
import cv2                                  # Open CV cv2
import numpy as np
from   time import sleep
import serial                               # Serial Port
import RPi.GPIO as GPIO                     # GPIO Mode aktif olsun
from   rpi_hardware_pwm import HardwarePWM  # Hardware PWM
# -------------------------------------------------------------------------------------

# HİTİT LOGO Yükleme ------------------------------------------------------------------
# Logo Yükle ve boyutlandır
img_Logo           = cv2.imread('Hitit_Logo.jpeg')
Logo_Size          = 100
img_Logo           = cv2.resize(img_Logo, (Logo_Size, Logo_Size))

# Logo için Maske oluştur
img2gray           = cv2.cvtColor(img_Logo, cv2.COLOR_BGR2GRAY)
ret, img_Logo_Mask = cv2.threshold(img2gray, 1,255, cv2.THRESH_BINARY)

# -------------------------------------------------------------------------------------
```

```
# Sistem Değişkenleri ------------------------------------------------------------
Duty   = 0                      # LED Lamba PWM Duty Cycle değeri

# Platform Özellikleri -----------------------------------------------------------
# Kamera Çözünürlüğü          : 1024, 768
# Platform Orijini            : xo, yo = 512,384
# Platform Köşe Koordinatları : Sol Üst (312,184)   Sag Ust (712,184)   Sol Alt (312,384)  Sag Alt (712,384)

xo            = 512           # Platform Origin x koordinatı
yo            = 384           # Platform Origin y koordinatı
xm            = 200           # Platform Originden dışarıya x ekseninde maksimum uzaklık
ym            = 200           # Platform Originden dışarıya y ekseninde maksimum uzaklık

# RGBY Referans Renk Koordinatları
xrr           =  -20          # Referans Renk RED    X
yrr           = -250          # Referans Renk RED    Y   #Eksen ters kullanılmıştır
xrg           =   20          # Referans Renk GREEN  X
yrg           = -250          # Referans Renk GREEN  Y   #Eksen ters kullanılmıştır
xrb           =  -20          # Referans Renk BLUE   X
yrb           = -230          # Referans Renk BLUE   Y   #Eksen ters kullanılmıştır
xry           =   20          # Referans Renk YELLOW X
yry           = -230          # Referans Renk YELLOW Y   #Eksen ters kullanılmıştır

# Home Area Koordinatı
xh            = -250          # Home area x koordinatı
yh            = 200           # Home area y koordinatı

# Waste Area Koordinatı
xw            =  250          # Waste area x koordinatı
yw            =  0            # Waste area y koordinatı

# Board ve Pin Ayarları ----------------------------------------------------------
pin_LED       = 12            # LED Lamba         pin bağlantısı
pin_Magnet    = 16            # Elektro Mıknatıs  pin bağlantısı

GPIO.setwarnings(False)                         # Uyarılar PASİF
GPIO.setmode(GPIO.BOARD)                        # Raspberry Pi Board pinleri AKTİF
GPIO.setup(pin_Magnet,GPIO.OUT)                 # Elektro Mıknatıs pin OUT

pwm_LED = HardwarePWM(pwm_channel=0, hz=1000)   # PWM0, Fiziksel 12 nolu pin
pwm_LED.start(Duty)                             # pwm Başlat
```

```
# HSV Renk Uzayı    (H:Hue, S:Saturaion,  V: Value) Alt-Ust Aralıkları ----------------

HSV_blackUpper  = (179, 255,  30)
HSV_blackLower  = (  0,   0,   0)
HSV_whiteUpper  = (179,  18, 255)
HSV_whiteLower  = (  0,   0, 231)
HSV_yellowUpper = ( 38, 255, 255)
HSV_yellowLower = ( 22, 100, 100)
HSV_redUpper    = (179, 255, 255)
HSV_redLower    = (160, 100, 100)
HSV_greenUpper  = ( 75, 255, 255)
HSV_greenLower  = ( 38, 100, 100)
HSV_blueUpper   = (130, 255, 255)
HSV_blueLower   = ( 75, 100, 100)


# capture Tanımları ------------------------------------------------------------------
cap             = cv2.VideoCapture(0)
cap.set(3,1024)
cap.set(4,768)

# Serial Port Tanımları --------------------------------------------------------------
sp_cnc=serial.Serial('/dev/ttyUSB0',115200)
sleep(.5)                                                   # Bekle


# ------------------------------------------------------------------------------------
# FONKSİYON TANIMLARI ----------------------------------------------------------------
# ------------------------------------------------------------------------------------

# Serial_Port_Ac ---------------------------------------------------------------------
def Serial_Port_Ac():
    if (sp_cnc.isOpen==False): sp_cnc.open()                # Serial Port Aç

# Serial_Port_Kapat ------------------------------------------------------------------
def Serial_Port_Kapat():
    sp_cnc.close()                                          # Serial Port Kapat

# LED Lamba Yak ----------------------------------------------------------------------
def Lamba_Yak(Duty):
    pwm_LED.change_duty_cycle(Duty)
    sleep(0.001)
```

```
# LED Güç Kontrolu ----------------------------------------------------------------
def Lamba_Guc_Kontrolu():
    ret_Power = "None"
    Duty = 0
    while (Duty<45):              # 0, 10, 20 ,30 ,100 için (110 dahil değil)

        Lamba_Yak(Duty)

        success, img_Platform = cap.read()
        cv2.imshow("ROBOTIC PLATFORM",img_Platform)
        cv2.waitKey(1)

        _Rxr, _Ryr, _Rxg, _Ryg, _Rxb, _Ryb, _Rxy, _Ryy    = Referans_XY(img_Platform)

        if (abs(_Rxr-xrr)<3) and (abs(_Ryr-yrr)<3):    # Ref. Renk Kontrol : Red
              ret_Ref_R = "O"
        else: ret_Ref_R = "x"

        if (abs(_Rxg-xrg)<3) and (abs(_Ryg-yrg)<3):    # Ref. Renk Kontrol : Green
              ret_Ref_G = "O"
        else: ret_Ref_G = "x"

        if (abs(_Rxb-xrb)<3) and (abs(_Ryb-yrb)<3):    # Ref. Renk Kontrol : Blue
              ret_Ref_B = "O"
        else: ret_Ref_B = "x"

        if (abs(_Rxy-xry)<3) and (abs(_Ryy-yry)<3):    # Ref. Renk Kontrol : Yellow
              ret_Ref_Y = "O"
        else: ret_Ref_Y = "x"

        print ("Duty Cycle ( %{} )  )".format(  Duty ) )
        print ("RED    X (Ref, Find, Err) = ({}, {}, {} ) Result :{} )".format(  xrr, _Rxr, abs(_Rxr-xrr),ret_Ref_R ) )
        print ("       Y (Ref, Find, Err) = ({}, {}, {} ) Result :{} )".format(  yrr, _Ryr, abs(_Ryr-yrr),ret_Ref_R ) )
        print ("GREEN  X (Ref, Find, Err) = ({}, {}, {} ) Result :{} )".format(  xrg, _Rxg, abs(_Rxg-xrg),ret_Ref_G ) )
        print ("       Y (Ref, Find, Err) = ({}, {}, {} ) Result :{} )".format(  yrg, _Ryg, abs(_Ryg-yrg),ret_Ref_G ) )
        print ("BLUE   X (Ref, Find, Err) = ({}, {}, {} ) Result :{} )".format(  xrb, _Rxb, abs(_Rxb-xrb),ret_Ref_B ) )
        print ("       Y (Ref, Find, Err) = ({}, {}, {} ) Result :{} )".format(  yrb, _Ryb, abs(_Ryb-yrb),ret_Ref_B ) )
        print ("YELLOW X (Ref, Find, Err) = ({}, {}, {} ) Result :{} )".format(  xry, _Rxy, abs(_Rxy-xry),ret_Ref_Y ) )
        print ("       Y (Ref, Find, Err) = ({}, {}, {} ) Result :{} )".format(  yry, _Ryy, abs(_Ryy-yry),ret_Ref_Y ) )
        print ("--------")
```

```
        if (ret_Ref_R == "O") and (ret_Ref_G == "O") and (ret_Ref_B == "O") and (ret_Ref_Y == "O"):    # Kontrol : POWER OK
              ret_Power = "Ok"
              print ("POWER : OK")
              break
        else: ret_Power = "None"

        # kp Düzenle
        if    (ret_Ref_Y == "O") : kp=1
        elif  (ret_Ref_G == "O") : kp=2
        elif  (ret_Ref_R == "O") : kp=3
        else : kp=5
        Duty = Duty + kp
        ret_Power = "Ok"
    return ret_Power, ret_Ref_R, ret_Ref_G, ret_Ref_B, ret_Ref_Y, Duty, _Rxr, _Ryr, _Rxg, _Ryg, _Rxb, _Ryb, _Rxy, _Ryy

# Nesne Al (Mıknatıs : ON) ------------------------------------------------------------
def Get_Object():
    sp_cnc.write(b"G91\n");
    sp_cnc.write(b"G01 Z-12 F260\n");
    sleep(4)
    GPIO.output(pin_Magnet, 1)               # Magnet ENABLE
    sleep(.1)
    sp_cnc.write(b"G91\n");
    sp_cnc.write(b"G01 Z+12 F260\n");

ret_Magnet = "On"

    return ret_Magnet

# nesne Bırak (Mıknatıs : OFF) ------------------------------------------------------------
def Put_Object():
     sp_cnc.write(b"G91\n");
     sp_cnc.write(b"G01 Z-12 F260\n");
     sleep(4)
     GPIO.output(pin_Magnet, 0)               # Magnet DISABLE
     sleep(.1)
     sp_cnc.write(b"G91\n");
     sp_cnc.write(b"G01 Z+12 F260\n");
     sleep(.1)

     ret_Magnet = "Off"

     return ret_Magnet
```

```
# Platform Hazırla ------------------------------------------------------------------------
def Platform_Hazirla(img_Platform, ret_Robot, Rx, Ry, ret_Bad, Ox, Oy, ret_Power, ret_Ref_R, ret_Ref_G, ret_Ref_B,
ret_Ref_Y, Duty, _Rxg, _Ryg, ret_Magnet):

       # Dış Çerçeve
       cv2.rectangle(img_Platform, (xo-(xm+109),yo-(ym+105)-5),  (xo+(xm+105),yo+(ym+105)), (0,0,0), 1 )
       cv2.rectangle(img_Platform, (0,0  ),  (1024,65),  (255,255,255), cv2.FILLED )
       cv2.rectangle(img_Platform, (0,695),  (1024,775), (255,255,255), cv2.FILLED )
       cv2.rectangle(img_Platform, (0,0  ),  (197,768),  (255,255,255), cv2.FILLED )
       cv2.rectangle(img_Platform, (823,0),  (1024,768), (255,255,255), cv2.FILLED )

      # Waste Area
       cv2.line(img_Platform,      (xo+200,yo), (xo+250,yo), (255,255,255), 1)
       cv2.line(img_Platform,      (xo+225,yo-25), (xo+225,yo+25), (255,255,255), 1)
       cv2.circle(img_Platform,    (xo+225,yo), 5, (0,0,255),cv2.FILLED)

       cv2.putText(img_Platform,"Waste ",(xo+230,yo - 7), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 1)

       # Sol Üst Koordinat
       cv2.circle(img_Platform, (xo-xm,yo-ym), 5, (255,255,255),cv2.FILLED)
       cv2.line(img_Platform,   (xo-xm,yo-ym-25), (xo-xm,yo-ym+25), (255,255,255), 1)
       cv2.line(img_Platform,   (xo-xm-25,yo-ym), (xo-xm+25,yo-ym), (255,255,255), 1)

       # Sağ Üst koordinat
       cv2.circle(img_Platform, (xo+xm,yo-ym), 5, (255,255,255),cv2.FILLED)
       cv2.line(img_Platform,   (xo+xm,yo-ym-25), (xo+xm,yo-ym+25), (255,255,255), 1)
       cv2.line(img_Platform,   (xo+xm-25,yo-ym), (xo+xm+25,yo-ym), (255,255,255), 1)

       #Sol Alt Koordinat
       cv2.circle(img_Platform, (xo-xm,yo+ym), 5, (255,255,255),cv2.FILLED)
       cv2.line(img_Platform,   (xo-xm,yo+ym-25), (xo-xm,yo+ym+25), (255,255,255), 1)
       cv2.line(img_Platform,   (xo-xm-25,yo+ym), (xo-xm+25,yo+ym), (255,255,255), 1)

       #Sağ Alt Koordinat
       cv2.circle(img_Platform, (xo+xm,yo+ym), 5, (255,255,255),cv2.FILLED)
       cv2.line(img_Platform,   (xo+xm,yo+ym-25), (xo+xm,yo+ym+25), (255,255,255), 1)
       cv2.line(img_Platform,   (xo+xm-25,yo+ym), (xo+xm+25,yo+ym), (255,255,255), 1)

       #Origin Koordinat
       cv2.circle(img_Platform, (xo,yo), 5, (255,255,255),cv2.FILLED)
       cv2.line(img_Platform,   (xo-25,yo),    (xo+25,yo), (255,255,255), 1)
       cv2.line(img_Platform,   (xo   ,yo-25), (xo,yo+25), (255,255,255), 1)
```

```
#Koordinatları yaz
cv2.putText(img_Platform, "+X",       (538,387), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 0)
cv2.putText(img_Platform, "-X",       (463,387), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 0)
cv2.putText(img_Platform, "+Y",       (497,354), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 0)
cv2.putText(img_Platform, "-Y",       (497,423), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 0)

cv2.putText(img_Platform, "0,0",           (486,401), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 0)
cv2.putText(img_Platform, "-200,+200",     (223,175), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 0)
cv2.putText(img_Platform, "-200,-200  Home", (223,600), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 0)

cv2.putText(img_Platform, "+200,+200",     (715,175), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 0)
cv2.putText(img_Platform, "+200,-200",     (715,600), cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7, (255,255,255), 0)


# Yazı Ekle
cv2.putText(img_Platform, "Robotic Platform",(300,50), cv2.FONT_HERSHEY_COMPLEX_SMALL, 2, (0,0,0), 2)
cv2.putText(img_Platform, "    MENU   ",     (5,250),  cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 2)

cv2.putText(img_Platform, "C-Classify Obj.", (5,300), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)
cv2.putText(img_Platform, "G-Get Object",    (5,350), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)
cv2.putText(img_Platform, "P-Put Object",    (5,375), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)
cv2.putText(img_Platform, "B-Go Bad Obj.",   (5,425), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)
cv2.putText(img_Platform, "W-Go Waste A.",   (5,450), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)
cv2.putText(img_Platform, "H-Go Home ",      (5,475), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)
cv2.putText(img_Platform, "Q-Quit    ",      (5,525), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)

cv2.putText(img_Platform, "  Hitit University - 2022  Corum / Turkiye",
                                    (225,735), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)

cv2.putText(img_Platform, "Reference Objects ",(445,90),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 0.7,(255,255,255), 1)
cv2.putText(img_Platform, "Reference " ,       (865,95),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)
cv2.putText(img_Platform, "[        ] " ,      (865,120),  cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0), 1)

if (ret_Power=="Ok"):
    cv2.putText(img_Platform,   "   Ok " ,     (865,120),  cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,255,0), 1)
else: cv2.putText(img_Platform, "  None " ,    (865,120),  cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,255), 1)

if (ret_Ref_R=="O"): cv2.rectangle(img_Platform, (485,107),  (495,118), (0,0,255),   cv2.FILLED )
if (ret_Ref_G=="O"): cv2.rectangle(img_Platform, (525,107),  (535,118), (0,255,0),   cv2.FILLED )
if (ret_Ref_B=="O"): cv2.rectangle(img_Platform, (485,147),  (495,158), (255,0,0),   cv2.FILLED )
if (ret_Ref_Y=="O"): cv2.rectangle(img_Platform, (525,147),  (535,158), (0,255,255), cv2.FILLED )
```

```
cv2.putText(img_Platform,       "  Robot  " ,       (865,195),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
cv2.putText(img_Platform,       "[        ] " ,     (865,220),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
if (ret_Robot=="Ok"):
      cv2.putText(img_Platform, "   Ok " ,          (865,220),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,255,0), 1)
      cv2.putText(img_Platform, " x : " + str(Rx),  (865,245),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
      cv2.putText(img_Platform, " y : " + str(-Ry), (865,265),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
else: cv2.putText(img_Platform, "  None " ,         (865,220),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,255), 1)

cv2.putText(img_Platform,       "Bad Object " ,     (860,340),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
cv2.putText(img_Platform,       "[        ] " ,     (865,370),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)

if (ret_Bad=="Ok"):
      cv2.putText(img_Platform, "   Ok " ,          (865,370),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,255,0), 1)
      cv2.putText(img_Platform, " x : " + str(Ox),  (865,395),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
      cv2.putText(img_Platform, " y : " + str(-Oy), (865,415),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
else: cv2.putText(img_Platform, "  None " ,         (865,370),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,255), 1)

cv2.putText(img_Platform,       "  Lamp  " ,        (865,545),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
cv2.putText(img_Platform,       "[        ] " ,     (865,575),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)

if (Duty>0):
    cv2.putText(img_Platform,   "   On   ",         (865,575),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,255,0), 1)
else: cv2.putText(img_Platform, "   Off " ,         (865,575),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,255), 1)

cv2.putText(img_Platform,       "[        ] " ,     (865,605),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)

cv2.putText(img_Platform,       "   " +
                              str(Duty) + "%",      (865,605),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
cv2.putText(img_Platform,       "[        ] " ,     (865,635),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
cv2.putText(img_Platform,       " " +
                              str(round(Duty*780/100)) + " mW",
                                                    (865,635),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)

cv2.putText(img_Platform,       " Magnet  " ,       (865,670),   cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)
cv2.putText(img_Platform,       "[        ] " ,     (865, 695),  cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,0),   1)


if (ret_Magnet=="On"):
      cv2.putText(img_Platform, "   On  ",      (865,695),        cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,255,0), 1)
else: cv2.putText(img_Platform, "   Off " ,     (865,695),        cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,0,255), 1)
```

```
        #Hitit Logo Basımı ---------------------------------------

        # Region of Image (ROI) : Logo eklenecek Image Bolgesi
        roi               = img_Platform[-Logo_Size-644:-644,-Logo_Size-875:-875]                  # x: 875-100:875-100

        # Maskeninm Indeksi
        roi[np.where(img_Logo_Mask)]   = 0
        roi                           += img_Logo

# Referans XY Öğren ----------------------------------------------------------------------
def Referans_XY(img_Platform):
    # Referans Noktası : RGBY

    x,y, xr,yr,  xg,yg,  xb,yb,  xy,yy   = 0,0,  0,0,  0,0,  0,0,  0,0

    # blur İşlemi yapılır
    blurred = cv2.GaussianBlur(img_Platform, (11,11), 0)

    # HSV formatına dönüştürülür (BRG >> HSV)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    # Renkler için maske oluşturulur
    red_mask    = cv2.inRange(hsv, HSV_redLower,   HSV_redUpper)
    green_mask  = cv2.inRange(hsv, HSV_greenLower, HSV_greenUpper)
    blue_mask   = cv2.inRange(hsv, HSV_blueLower,  HSV_blueUpper)
    yellow_mask = cv2.inRange(hsv, HSV_yellowLower,HSV_yellowUpper)

    # Maskelerin etraflarındaki gürüntüler silinir
    red_mask    = cv2.erode(red_mask,    None, iterations = 2)
    red_mask    = cv2.dilate(red_mask,   None, iterations = 2)

    green_mask  = cv2.erode(green_mask,  None, iterations = 2)
    green_mask  = cv2.dilate(green_mask, None, iterations = 2)

    blue_mask   = cv2.erode(blue_mask,   None, iterations = 2)
    blue_mask   = cv2.dilate(blue_mask,  None, iterations = 2)

    yellow_mask = cv2.erode(yellow_mask, None, iterations = 2)
    yellow_mask = cv2.dilate(yellow_mask,None, iterations = 2)
```

```
# Kırmızı Renk Tespiti ------------------------------------------------------------------------

x,y  = 0,0

# kontur işlemi yapılır
(contours,_)= cv2.findContours(red_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

center = None

for i, c in enumerate(contours):
    area = cv2.contourArea(c)

    if  area > 1000:

        rect = cv2.minAreaRect(c)                  # dikdörtgene çevir
        ((x,y), (width,height), rotation) = rect

xr = round(x-xo)
yr = round(y-yo)



# Yeşil Renk Tespiti --------------------------------------------------------------------------

x,y  = 0,0

# kontur işlemi yapılır
(contours,_)= cv2.findContours(green_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

center = None

for i, c in enumerate(contours):
    area = cv2.contourArea(c)

    if  area > 1000:

        rect = cv2.minAreaRect(c)                 # dikdörtgene çevir
        ((x,y), (width,height), rotation) = rect

xg = round(x-xo)
yg = round(y-yo)
```

```
# Mavi Renk Tespiti ------------------------------------------------------------------------

x,y  = 0,0

# kontur
(contours,_)= cv2.findContours(blue_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

center = None

for i, c in enumerate(contours):
    area = cv2.contourArea(c)

    if  area > 1000:

        rect = cv2.minAreaRect(c)                 # dikdörtgene çevir
        ((x,y), (width,height), rotation) = rect

xb = round(x-xo)
yb = round(y-yo)


# Sarı Renk Tespiti ------------------------------------------------------------------------

x,y  = 0,0

# kontur işlemi yapılır
(contours,_)= cv2.findContours(yellow_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

center = None

for i, c in enumerate(contours):
    area = cv2.contourArea(c)

    if  area > 1000:

        rect = cv2.minAreaRect(c)                  # dikdörtgene çevir
        ((x,y), (width,height), rotation) = rect

xy = round(x-xo)
yy = round(y-yo)

return xr,yr, xg,yg, xb,yb, xy,yy
```

```
# Robot XY Öğren ------------------------------------------------------------------
def Robot_XY(img_Platform):           # Robot Color : YELLOW (SARI)

    ret_Robot = "None"
    x,y,  x_rob,y_rob     = 0,0,  0,0

    # blur işlemi yapılır
    blurred = cv2.GaussianBlur(img_Platform, (11,11), 0)

    # HSV formatına dönüştürülür (BRG >> HSV)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    # SARI için maske oluştur
    yellow_mask = cv2.inRange(hsv, HSV_yellowLower, HSV_yellowUpper)

    # Maske etrafındaki gürüntü silinir
    yellow_mask = cv2.erode(yellow_mask,  None, iterations = 2)
    yellow_mask = cv2.dilate(yellow_mask, None, iterations = 2)

   # SARI Renk Tespiti ---------------------------------------------------------------

    # kontur işlemi yapılır
    (contours,_)= cv2.findContours(yellow_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    center = None

    for i, c in enumerate(contours):
        area = cv2.contourArea(c)

        if  area > 1000:

            rect = cv2.minAreaRect(c)        # dikdörtgene çevir
            ((x_rob,y_rob), (width,height), rotation) = rect

            if (y_rob> 200):
               ret_Robot = "Ok"
               x = x_rob
               y = y_rob

    return ret_Robot, round(x-xo), round(y-yo)
```

```
# Bad Object XY Öğren ------------------------------------------------------------------
def Bad_XY(img_Platform):                   # BAD Object Color : RED (KIRMIZI)

    ret_Bad = "None"
    Cx = 0       #   Nesne Yok, Renk (Cx) = 0
    x,y  = 0,0

    # blur işlemi yapılır
    blurred = cv2.GaussianBlur(img_Platform, (11,11), 0)

    # HSV formatına dönüştürülür (BRG >> HSV)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    # KIRMIZI için maske oluştur
    red_mask = cv2.inRange(hsv, HSV_redLower, HSV_redUpper)

    # Maske etrafındaki gürüntü silinir
    red_mask = cv2.erode (red_mask, None, iterations = 2)
    red_mask = cv2.dilate(red_mask, None, iterations = 2)

    # Kırmızı Renk Tespiti ------------------------------------------------------------------------

    # kontur işlemi yapılır
    (contours,_)= cv2.findContours(red_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    center = None

    for i, c in enumerate(contours):
        area = cv2.contourArea(c)

        if  area > 1000 :

            rect = cv2.minAreaRect(c)             # dikdörtgene çevir
            ((x_obj,y_obj), (width,height), rotation) = rect

            if ((y_obj > 180) and (x_obj<700)):   # Çalışma Alanı İçi Kontrolü
               ret_Bad = "Ok"
               x = x_obj
               y = y_obj

    return ret_Bad, round(x-xo), round(y-yo)
```

```
# Good Object XY Öğren ------------------------------------------------------------------
def Good_XY(img_Platform):                  # GOOD Object Color : GREEN (YEŞİL)

    ret_Good = "None"
    Cx = 0       #   Nesne Yok, Renk (Cx) = 0
    x,y  = 0,0

    # blur işlemi yapılır
    blurred = cv2.GaussianBlur(img_Platform, (11,11), 0)

    # HSV formatına dönüştürülür (BRG >> HSV)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    # YEŞİL için maske oluştur
    green_mask = cv2.inRange(hsv, HSV_greenLower, HSV_greenUpper)

    # Maske etrafındaki gürüntü silinir
    green_mask = cv2.erode (green_mask, None, iterations = 2)
    green_mask = cv2.dilate(green_mask, None, iterations = 2)

    # YEŞİL Renk Tespiti ---------------------------------------------------------------------

    # kontur işlemi yapılır
    (contours,_)= cv2.findContours(green_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    center = None

    for i, c in enumerate(contours):
        area = cv2.contourArea(c)

        if  area > 1000 :

            rect = cv2.minAreaRect(c)                # dikdörtgene çevir
            ((x_obj,y_obj), (width,height), rotation) = rect

            if ((y_obj > 180) and (x_obj<700)):      # Çalışma Alanı İçi Kontrolü
               ret_Good = "Ok"
               x = x_obj
               y = y_obj

    return ret_Good, round(x-xo), round(y-yo)
```

```
# Other Object XY Öğren ----------------------------------------------------------------

def Blue_XY(img_Platform):                    # OTHER Object Color : BLUE (MAVİ)

    ret_Blue = "None"
    Cx = 0       #   Nesne Yok, Renk (Cx) = 0
    x,y  = 0,0

    # blur işlemi yapılır
    blurred = cv2.GaussianBlur(img_Platform, (11,11), 0)

    # HSV formatına dönüştürülür (BRG >> HSV)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    # MAVİ için maske oluştur
    blue_mask = cv2.inRange(hsv, HSV_blueLower, HSV_blueUpper)

    # Maske etrafındaki gürüntü silinir
    blue_mask = cv2.erode (blue_mask, None, iterations = 2)
    blue_mask = cv2.dilate(blue_mask, None, iterations = 2)

    # kontur işlemi yapılır
    (contours,_)= cv2.findContours(blue_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    center = None

    for i, c in enumerate(contours):
        area = cv2.contourArea(c)

        if  area > 1000 :

            rect = cv2.minAreaRect(c)                  # dikdörtgene çevir
            ((x_obj,y_obj), (width,height), rotation) = rect

            if ((y_obj > 180) and (x_obj<700)):        # Çalışma Alanı İçi Kontrolü
               ret_Blue = "Ok"
               x = x_obj
               y = y_obj

    return ret_Blue, round(x-xo), round(y-yo)
```

```
# Function (Platform Taraması : Robot ve Nesne Koordinatları belirlenir) ----------------------
def Platform_Tara():

    success, img_Platform = cap.read()

    ret_Robot,  Rx, Ry    = Robot_XY (img_Platform)
    ret_Bad, Ox, Oy       = Bad_XY(img_Platform)

    return ret_Robot, Rx, Ry, ret_Bad, Ox, Oy


# Function (Classify Object Operation) ------------------------------------------------------------
def Classify_Object():

    success, img_Platform  = cap.read()
    ret_Robot, Rx, Ry      = Robot_XY (img_Platform)
    ret_Bad, Ox, Oy        = Bad_XY(img_Platform)

    while(ret_Bad=="Ok"):

           if (ret_Robot=="Ok"):

               # Object'e Git-----------------
               Go_XY( Rx, Ry, Ox, Oy)

           while( ( abs(Ox-Rx)>20 ) or ( abs(Oy-Ry)>20) ):     # Object'e gidilene kadar BEKLE ve pozisyon bilgisi ver

               success, img_Platform  = cap.read()
               Platform_Hazirla(img_Platform, ret_Robot, Rx, Ry, ret_Bad, Ox, Oy, 0, 0, 0, 0, 0, 0, 0, 0, 0)

               cv2.imshow("ROBOTIC PLATFORM",img_Platform)

               cv2.waitKey(1)

               ret_Robot, Rx, Ry     =  Robot_XY(img_Platform)

           # Nesne Al -------
           Get_Object()

           # Atık Alanına Git -------
           Go_WasteArea()

           while( ( abs(xw-Rx)>30 ) or ( abs(yw-Ry)>30) ):       # Atık Alanına gidilene kadar BEKLE ve Pozisyon bilgisi ver
```

```
            success, img_Platform  = cap.read()
            Platform_Hazirla(img_Platform, ret_Robot, Rx, Ry, ret_Bad, Ox, Oy, 0, 0, 0, 0, 0, 0, 0, 0, 0)
            cv2.imshow("ROBOTIC PLATFORM",img_Platform)
            cv2.waitKey(1)

            ret_Robot, Rx, Ry     =  Robot_XY(img_Platform)

            print (" Wait for (Çöpe Gidiliyor)  : Robot ( {}   {} ) ".format( Ox, Rx, Oy, Ry) )

        # Nesne Bırak ------
        Put_Object()

        # Home pozisyonuna Git
        Go_Home()


        while( ( abs(-200-Rx)>50 ) or ( abs(200-Ry)>50) ):     # Home'a gidilene Kadar BEKLE ve Pozisyon bilgisi ver

            success, img_Platform  = cap.read()
            Platform_Hazirla(img_Platform, ret_Robot, Rx, Ry, ret_Bad, Ox, Oy, 0, 0, 0, 0, 0, 0, 0, 0, 0)
            cv2.imshow("ROBOTIC PLATFORM",img_Platform)
            cv2.waitKey(1)

            ret_Robot, Rx, Ry     =  Robot_XY(img_Platform)

            print (" Wait for (Going to Home) : Robot ( {}   {} ) ".format( Ox, Rx, Oy, Ry) )

        # Classify İşleminden Çıkış Yap
        break


# Function (G-Code Go XYZ Position) ------------------------------------------------------------
def Go_XY( Rx, Ry, Ox, Oy):

   Dx, Dy    =  Ox-Rx, Oy-Ry

   Dx, Dy =  round(Dx/6), round(Dy/6)

   Dx = -Dx                            #  x için  (-) ters yon değişimi Platform için

   if (Dx<0): str_Dx = str(Dx)         # Pozitif için + ilave et
   else     : str_Dx = "+" + str(Dx)
```

```
   if (Dy<0): str_Dy = str(Dy)          # Pozitif için + ilave et
   else     : str_Dy = "+" + str(Dy)

   str_Dz = "+0"

   sp_cnc.write(bytes("G91\n", 'utf-8') );
   sp_cnc.write(bytes("G01 X"+str_Dx+" Y"+str_Dy+" Z"+str_Dz+" F260\n",'utf-8') )


# Function (G-Code Go XYZ Position - Absolute ) ------------------------------------------------------------
def Go_XY_Absolute( Tx, Ty):

   Tx, Ty =  round(Tx/6), round(Ty/6)

   Tx = -Tx                             # x için  (-) ters yon değişimi Platform için

   if (Tx<0): str_Tx = str(Tx)          # Pozitif için + ilave et
   else     : str_Tx = "+" + str(Tx)

   if (Ty<0): str_Ty = str(Ty)          # Pozitif için + ilave et
   else     : str_Ty = "+" + str(Ty)

   str_Tz = "+0"

   sp_cnc.write(bytes("G90\n", 'utf-8') );
   sp_cnc.write(bytes("G01 X"+str_Tx+" Y"+str_Ty+" Z"+str_Tz+" F260\n",'utf-8') )

# Function (G-Code Go PARK Position) -----------------------------------------------------------
def Go_Home():

    success, img_Platform  = cap.read()
    ret_Robot, Rx, Ry     = Robot_XY(img_Platform)

    Go_XY( Rx, Ry, xh, yh)

# Function (G-Code Go COP Position) ------------------------------------------------------------
def Go_WasteArea():

    success, img_Platform  = cap.read()
    ret_Robot, Rx, Ry     = Robot_XY(img_Platform)

    Go_XY( Rx, Ry, xw, yw)
```

```
# Function (Robotik Proses LOOP) ------------------------------------------------------------
def Robotik_Proses():

 ret_Magnet = "Passive"

 ret_Power, ret_Ref_R,ret_Ref_G,ret_Ref_B,ret_Ref_Y,Duty,_Rxr,_Ryr,_Rxg,_Ryg,_Rxb,_Ryb,_Rxy,_Ryy = Lamba_Guc_Kontrolu()

 while(True):
        B_x, B_y,   G_x, G_y = 0,0,   0,0

        success, img_Platform = cap.read()

        ret_Robot, Rx, Ry, ret_Bad, Ox, Oy =  Platform_Tara()
        Dx, Dy                              =  Ox-Rx, Oy-Ry

       Platform_Hazirla(img_Platform,ret_Robot,Rx,Ry,ret_Bad,Ox,Oy,ret_Power,ret_Ref_R,ret_Ref_G,ret_Ref_B,ret_Ref_Y,Duty, 0,0,0)

        if success:

           # blur işlemi yapılır
           blurred     = cv2.GaussianBlur(img_Platform, (11,11), 0)

           # HSV formatına dönüştürülür (BRG >> HSV)
           hsv         = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

           # Maskeler oluşturulur
           red_mask     = cv2.inRange(hsv, HSV_redLower,     HSV_redUpper)
           green_mask   = cv2.inRange(hsv, HSV_greenLower,   HSV_greenUpper)
           yellow_mask  = cv2.inRange(hsv, HSV_yellowLower,  HSV_yellowUpper)
           blue_mask    = cv2.inRange(hsv, HSV_blueLower,    HSV_blueUpper)

           # Maskelerin etrafında kalan gürültüler silinir
           red_mask    = cv2.erode (red_mask,      None, iterations = 2)
           red_mask    = cv2.dilate(red_mask,      None, iterations = 2)
           green_mask  = cv2.erode (green_mask,    None, iterations = 2)
           green_mask  = cv2.dilate(green_mask,    None, iterations = 2)
           yellow_mask = cv2.erode (yellow_mask,   None, iterations = 2)
           yellow_mask = cv2.dilate(yellow_mask,   None, iterations = 2)
           blue_mask   = cv2.erode (blue_mask,     None, iterations = 2)
           blue_mask   = cv2.dilate(blue_mask,     None, iterations = 2)
```

```
# KIRMIZI (BAD Object) Renk Tespiti ------------------------------------------------------------------------

# kontur işlemi yapılır
(contours,_)= cv2.findContours(red_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

center = None

for i, g in enumerate(contours):
     area = cv2.contourArea(g)
     #if area in range(2500,8500):

     if area > 1000 :

         rect = cv2.minAreaRect(g)    # dikdörtgene çevir
         ((x,y), (w,h), r) = rect

         R_x = x
         R_y = y

         if(R_y>160)and(R_x<700):

                # Kutu
                box = cv2.boxPoints(rect)
                box = np.int64(box)

                # moment bulunur
                M = cv2.moments(g)
                center = (int(M["m10"]/M["m00"]), int(M["m01"]/M["m00"]))

                # Kontur çizilir: Siyah
                cv2.drawContours(img_Platform, [box], 0, (0,0,0),1)

                # Merkeze nokta konur: Siyah
                cv2.circle(img_Platform, center, 5, (0,0,0),-1)

                # Nesne adı yazılır : Bad
                cv2.putText(img_Platform,"Bad",(round(x),round(y)),cv2.FONT_HERSHEY_COMPLEX_SMALL,1, (38,250,250), 2)
```

```
# YEŞİL (GOOD Object) Renk Tespiti ------------------------------------------------------------------------

# kontur işlemi yapılır
(contours,_)= cv2.findContours(green_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

center = None

for i, g in enumerate(contours):
     area = cv2.contourArea(g)

     if area > 1000 :

         rect = cv2.minAreaRect(g)            # dikdörtgene çevir
         ((x,y), (w,h), r) = rect

         G_x = x
         G_y = y

         if(G_y>160):

                # Kutu
                box = cv2.boxPoints(rect)
                box = np.int64(box)

                # moment bulunur
                M = cv2.moments(g)
                center = (int(M["m10"]/M["m00"]), int(M["m01"]/M["m00"]))

                # Konturu çizilir: Siyah
                cv2.drawContours(img_Platform, [box], 0, (0,0,0),1)

                # Merkeze nokta konur: Siyah
                cv2.circle(img_Platform, center, 5, (0,0,0),-1)

                # Nesne adı yazılır : Good
                cv2.putText(img_Platform,"Good",(round(x),round(y)),cv2.FONT_HERSHEY_COMPLEX_SMALL,1,(38,250,250),2)
```

```
# SARI (ROBOT) - Renk Tespiti ------------------------------------------------------------------------

# kontur işlemi yapılır
(contours,_)= cv2.findContours(yellow_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

center = None

for i, c in enumerate(contours):
     area = cv2.contourArea(c)

     if   area > 1000 :

         rect = cv2.minAreaRect(c)              # dikdörtgene çevir
         ((x,y), (width,height), rotation) = rect

         Y_x =  x
         Y_y =  y

         if(Y_y>160):

               # kutu
               box = cv2.boxPoints(rect)
               box = np.int64(box)

               # moment bulunur
               M = cv2.moments(c)
               center = (int(M["m10"]/M["m00"]), int(M["m01"]/M["m00"]))

               # Kontur çizilir: Siyah
               cv2.drawContours(img_Platform, [box], 0, (0,0,0),1)

               # Merkeze nokta konur: Siyah
               cv2.circle(img_Platform, center, 5, (0,0,0),-1)

               # Nesne adı yazılır : Robot
              cv2.putText(img_Platform,"Robot",(round(x),round(y)),cv2.FONT_HERSHEY_COMPLEX_SMALL,1,(38,250,250),2)
```

```
    # MAVİ (OTHER Object)Renk Tespiti ------------------------------------------------------------------------

    # kontur işlemi yapılır
    (contours,_)= cv2.findContours(blue_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    #(contours,_)= cv2.findContours(red_mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    center = None

    #if len(contours) > 10000:

    for i, c in enumerate(contours):
         area = cv2.contourArea(c)

         if   area > 1000 :

             rect = cv2.minAreaRect(c)               # dikdörtgene çevir

             ((x,y), (width,height), rotation) = rect

             Y_x =  x
             Y_y =  y

             if(Y_y>160):

                   # kutu
                   box = cv2.boxPoints(rect)
                   box = np.int64(box)

                   # moment bulunur
                   M = cv2.moments(c)
                   center = (int(M["m10"]/M["m00"]), int(M["m01"]/M["m00"]))

                   # Kontur çizilir: Siyah
                   cv2.drawContours(img_Platform, [box], 0, (0,0,0),1)

                   # Merkeze nokta konur: Siyah
                   cv2.circle(img_Platform, center, 5, (0,0,0),-1)

                   # Nesne adı yazılır : <Boş>
                   cv2.putText(img_Platform, "", (round(x),round(y)), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (38,250,250), 2)


cv2.imshow("ROBOTIC PLATFORM",img_Platform)
```

```
        #-------------------------------------------------------------------------
        # SISTEM ANA MENUSU ------------------------------------------------------
        #-------------------------------------------------------------------------

        key = cv2.waitKey(1) & 0xFF

        if   chr(key).upper() == "C":  Classify_Object()              # Classify Object
        elif chr(key).upper() == "P":  ret_Magnet = Put_Object()      # Put Object
        elif chr(key).upper() == "G":  ret_Magnet = Get_Object()      # Get Object
        elif chr(key).upper() == "B":  Go_XY( Rx, Ry, Ox, Oy)         # Go Bad Object
        elif chr(key).upper() == "W":  Go_WasteArea()                 # Go Waste Area
        elif chr(key).upper() == "H":  Go_Home()                      # Go Home
        elif chr(key).upper() == "Q":                                 # Quit
                                       Lamba_Yak(0)
                                       break
        #-------------------------------------------------------------------------
        # SISTEM ANA MENUSU ------------------------------------------------------
        #-------------------------------------------------------------------------

# -------------------------------------
# MAIN ROBOTIC PROCESS BLOCK ----------
# -------------------------------------

Robotik_Proses()


# Belleği Boşalt ----------------------
cap.release()
cv2.destroyAllWindows()
# -------------------------------------


# Program Sonu ------------------------------------------------------------------------
# -------------------------------------------------------------------------------------
```