KTO KARATAY ÜNİVERSİTESİ

**MÜHENDİSLİK ve DOĞA BİLİMLERİ FAKÜLTESİ**

**ELEKTRİK ELEKTRONİK**
**MÜHENDİSLİĞİ**

**BİTİRME PROJESİ**

# Hareket Ezberleyen 6 Eksen Robotik Kol

**2022-2023**

**BAŞAK YALÇINER || 21803016**

**EMRE ÖZKUL || 21803012**

**İçindekiler**

---

# 1st PART

## ÖZET

Tasarlamış olduğumuz robot kol 6 eksende serbest ve bağımsız bir şekilde hareket etmektedir. Projede hassasiyet odaklı bir çalışma izlediğimiz için temel eksen hareketleri için 5 adet step motor kullandık. Robot kolun uç kısmındaki kancayı kontrol etmek için ise 1 adet servo motor kullandık. Projede mikrokontroller olarak PIC18F46K22 kullanılmaktadır. 3 adet Joystick ile kontrol edilen 6 adet motor, robot kolun hareketlerini yapmasını sağlamaktadır. Devreye entegre 4 farklı buton ile sistemin Home konumuna gelmesi, hareket kaydetme moduna girilmesi, hareketlerin kaydedilmesi ve kayıtlı hareketlerin oynatılması işlemleri yapılmaktadır.

# 2nd PART

## PROJE HAKKINDA

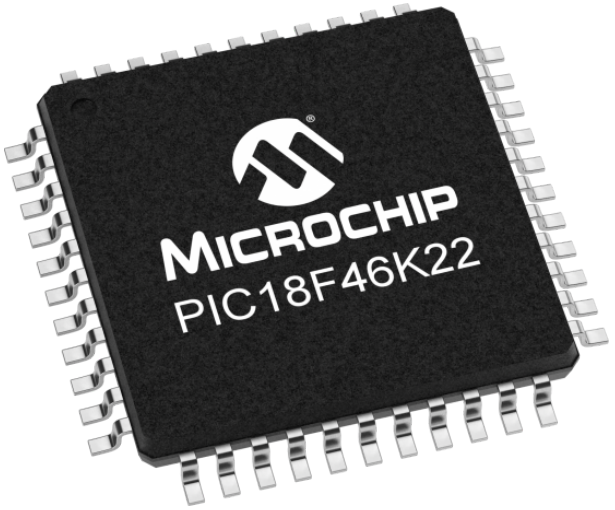
### 2.1 Hikaye ve Misyon

Projemizde misyon olarak her alanda kullanılabilecek bir robot kol tasarlamayı hedefledik. Bu alanlardan bazıları; Otomasyon, elektronik ve tıp alanları. Biz mühendislerin bildiği üzere teknoloji her zaman insanlardan daha güvenilirdir ve insan tarihi teknolojiye doğru evrimleşmek zorundadır. Bu nedenle robot kollar kullanılarak binlerce kilometre uzaktan birkaç hassas joystick ile ameliyat yapacak bir cerrah zannettiğimizden çok daha yakın bir tarihte hayatlarımızda olacak. Bu ve bunun gibi örnekler bizim bu projeyi yaparken kafamızda tasarladığımız temel hayaldir.

# 3rd PART

# KULLANILAN MALZEMELER

## 3.1 PIC Microcontroller




**Şekil 3.1**

PIC, uzun adıyla çevresel arayüz kontrol elemanı olarak bilinir. Yani birçok farklı elektronik cihaz için programlama imkanı sağlayan bir sistemdir. Bildiğiniz gibi birçok farklı elektronik cihazda mikroişlemciler var. Bu mikroişlemciler, cihazların ne için yapıldığını yapmasına izin verir. Ancak mikroişlemcilerin çalışacak şekilde programlanması gerekir. Bu programlamayı yapan sistem PIC olarak bilinir. Özellikle programlanabilir bir ayrıcalık sunması onu bir adım öne çıkarıyor. PIC, aktif yazılımla birlikte farklı ve kolay bir şekilde programlama yeteneği eşliğinde birçok elektronik sistemin çalışmasına izin verir. Özellikle C yazılım dilinin kullanımı ile hem amatör hem de profesyonel program programcıları tarafından kullanılan dillerde gelir. Bu sayede bilgisayardan çamaşır makinesine ve hatta dijital kol saatine kadar evinizdeki tüm elektronik cihazlar PIC sayesinde programlanabilir.

### 3.1.1 PIC'in Teknik Özellikleri

---

- ➔ C Derleyicisi, optimize edilmiş bir mimari / komut seti ile çalışır.
- ➔ 64 Kbyte adreslemeli doğrusal program belleği
- ➔ 4 Kbyte adreslemeli doğrusal veri belleği
- ➔ 16 bit genişliğinde talimatlar, 8 bit genişliğinde veri yolu
- ➔ Uyku modu: 100 nA
- ➔ Bekçi Köpeği Zamanlayıcısı: 500 nA
- ➔ Timer1 Osilatör: 500 nA, 32 kHz Esnek Osilatör Yapısı
- ➔ Yazılım seçilebilir frekans aralığı 31 kHz ila 16 MHz Aralık
- ➔ PLL kullanılarak 64 MHz performans mevcuttur
- ➔ Harici bileşen gerekmez
- ➔ 64 MHz'e kadar dört Kristal modu
- ➔ 64 MHz'e kadar iki harici Saat modu
- ➔ 32 khz'de Timer1 kullanan ikincil osilatör
- ➔ Çevre saati durursa güvenli kapanmaya izin verir
- ➔ İki hızlı osilatör Başlatma
- ➔ Tam 5.5 V çalışma
- ➔ 1.8V-3.6V çalışma için düşük voltaj seçeneği mevcuttur
- ➔ Yazılım kontrolü altında, kendi kendine yeniden programlanabilir
- ➔ Açılışta Sıfırlama (POR), açılış Zamanlayıcısı (PWRT) ve osilatör Başlatma Zamanlayıcısı (OST)
- ➔ Çip üzerinde osilatör ve yazılım aktivasyonu ile Genişletilmiş Watchdog Zamanlayıcısı (WDT)
- ➔ Programlanabilir kod koruması
- ➔ 10 bit çözünürlük
- ➔ 17 analog giriş kanalı
- ➔ 28 analog giriş kanalı
- ➔ Uyku dönüşümü sırasında kullanılabilir
- ➔ Programlanabilir Yüksek/Alçak Gerilim Algılama (PLVD) modülü
- ➔ Düğme, sensör veya kaydırıcı girişi için 28 kanala kadar
- ➔ İki raydan raya analog karşılaştırıcı
- ➔ Karşılaştırıcı giriş ve çıkışları dışarıdan erişilebilir ve yapılandırılabilir
- ➔ Seçilebilir çip üzerinde sabit voltaj referansı
- ➔ Yüksek akım alıcı/kaynak 25 mA/25 mA
- ➔ Bireysel olarak programlanabilen zayıf pull-up'lar
- ➔ Ayrı ayrı programlanabilir pin-on kesinti değişimi

- ➔ Üç harici kesme pimi
- ➔ Ön ölçekleyicili dört adede kadar 16 bit zamanlayıcı / sayaç
- ➔ En fazla üç adet 8 bit zamanlayıcı / sayaç
- ➔ Özel, düşük güçlü Timer1 osilatör
- ➔ İki adede kadar Yakalama / Karşılaştırma / PWM (ÇKP) modülü
- ➔ Bir, iki veya dört PWM çıkışı

### 3.1.2 PIC'in Tercih Edilme Sebepleri

---

**Performans**

32 kHz çalışma hızına sahip yüksek performanslı bir mikrodenetleyicilerdir.

**Güç Tüketimi**

Düşük voltajlarda çalışabilen bir mikrodenetleyicilerdir.

**Dahili Kristal**

Seçtiğimiz parçanın bir iç kristale sahip olması, tasarımımızı daha minimal boyutlarda alışmamızı sağladı.

**Hafıza**

Seçtiğimiz PIC, hem ROM hem de RAM kapasitesi olarak projenin gelişimine uygun altyapıyı sağlıyor.

## 3.2 NEMA17 & Nema23 Step Motor


**Şekil 3.2**

Step motorlar, dönme hareketini adım adım gerçekleştiren elektrikli motorlardır. Adımlı motorlar olarak da bilinen bu motorlar, elektrik darbeleriyle kontrol edilen manyetik alanlar sayesinde dönme hareketini gerçekleştirirler. Adımlı motorların başlıca özellikleri arasında yüksek çözünürlük, yüksek tork ve basit kontrol mekanizmaları bulunur.

### 3.2.1 NEMA17 & Nema23 Step Motor Teknik Özellikleri

NEMA17 ve NEMA23, adımlı motorların iki yaygın boyutu ve standartlarıdır. İşte her biri hakkında daha fazla bilgi:

**NEMA17 Step Motor:**

Adı: NEMA17, 1.7 inçlik (43,18 mm) kare montaj boyutuna atıfta bulunur.
Adım Açısı: Genellikle 1.8 derece adım açısıyla gelir. Bu, her adımda 1.8 derecelik bir dönüş hareketi anlamına gelir.
Tork: NEMA17 motorlarının torku, model ve yapılandırmaya bağlı olarak değişir. Genellikle 0.4 Nm (56 oz-in) ila 0.7 Nm (99 oz-in) arasında bir tork üretebilirler.
Boyut ve Ağırlık: NEMA17 motorlar küçük ve hafiftir. Yaklaşık olarak 42 mm x 42 mm x 48 mm (1.65 inç x 1.65 inç x 1.89 inç) boyutlarına sahiptir ve genellikle 250 gramdan az ağırlığa sahiptirler.

Güç Tüketimi: Tipik olarak 12V veya 24V DC güç kaynaklarıyla çalışırlar. Güç tüketimi motorun yapılandırmasına ve çalışma koşullarına bağlı olarak değişebilir.

**NEMA23 Step Motor:**

Adı: NEMA23, 2.3 inçlik (57,15 mm) kare montaj boyutuna atıfta bulunur.
Adım Açısı: Genellikle 1.8 derece adım açısıyla gelir. Bu da her adımda 1.8 derecelik bir dönüş hareketi anlamına gelir.
Tork: NEMA23 motorları, daha büyük boyutlarından dolayı NEMA17'ye göre genellikle daha yüksek tork üretebilir. Tork değerleri, model ve yapılandırmaya bağlı olarak değişir ve genellikle 0.9 Nm (128 oz-in) ila 3 Nm (425 oz-in) arasında değişebilir.
Boyut ve Ağırlık: NEMA23 motorlar daha büyük boyutlara sahiptir. Yaklaşık olarak 57 mm x 57 mm x 76 mm (2.24 inç x 2.24 inç x 2.99 inç) boyutlarına sahip olabilirler ve genellikle 1 kg'dan daha ağırdırlar.
Güç Tüketimi: Genellikle 24V veya 48V DC güç kaynaklarıyla çalışırlar. Güç tüketimi motorun yapılandırmasına ve çalışma koşullarına bağlı olarak değişir.

### 3.2.2 Step Motor Tercih Edilme Sebepleri

---

Robot kollarında step motorlar, birçok avantajı nedeniyle yaygın olarak tercih edilir. İşte step motorların robot kollarında kullanılmasını gerektiren bazı önemli faktörler:

**Hassas Kontrol:** Step motorlar, dönüş açısını küçük adımlar halinde değiştirme yeteneklerinden dolayı hassas kontrol sağlar. Robot kollarının hedeflenen konumlara tam olarak yerleştirilmesi ve hassas manipülasyon işlemleri için bu hassasiyet büyük önem taşır.

**Basit Kontrol Mekanizması:** Step motorlar, birkaç sinyal ve basit bir kontrol devresi ile kolayca çalıştırılabilir. Bu, robot kollarının kontrolünü basitleştirir ve sistem karmaşıklığını azaltır. Ayrıca, mikrodenetleyiciler veya diğer dijital cihazlarla entegrasyonu kolaydır.

**Yüksek Tork:** Step motorlar, sabit bir torkla çalıştıkları için robot kollarında yük taşıma kapasitesi için uygun seçimlerdir. Bu özellik, robot kollarının ağırlıkları taşıma, kaldırma veya manipülasyon gibi görevleri etkin bir şekilde yerine getirmesini sağlar.

**Durma ve Kilitlenme Özelliği:** Step motorlar, güç kaynağı kesildiğinde bile konumlarını koruyabilme yeteneğine sahiptir. Bu, robot kolunun istenmeyen bir şekilde hareket etmesini önleyerek güvenlik ve enerji tasarrufu sağlar.

**Maliyet-Etkinlik:** Step motorlar, diğer motor türlerine kıyasla genellikle daha ekonomiktir. Bu, robot kollarının maliyetini düşürmeye yardımcı olur ve uygulamaların daha erişilebilir hale gelmesini sağlar.

Sonuç olarak, step motorlar robot kollarında yaygın olarak tercih edilen bir motor tipidir. Hassas kontrol, basitlik, yüksek tork, kilitlenme özelliği ve maliyet-etkinlik gibi faktörler, robot kollarının doğru pozisyonlara yerleştirilmesi ve çeşitli görevlerin etkin bir şekilde yerine getirilmesi için step motorları ideal seçim yapar.

## 3.3 TB6600 Step Motor Sürücü


TB6600 Step Motor Sürücü, adımlı motorların kontrolünü ve sürülmesini sağlayan bir elektronik cihazdır. Bu sürücü, step motorları yüksek hassasiyetle ve güvenilir bir şekilde çalıştırmak için kullanılır. Özellikle robotik uygulamalarda ve CNC makinelerinde yaygın olarak tercih edilir.

### 3.3.1 TB6600 Step Motor Sürücü Teknik Özellikleri

---

➔ **Güç Kaynağı Gerilimi:** 9V ila 42V DC arası
- ◆ Sürücü, bu aralıktaki bir güç kaynağından çalışabilir. Genellikle 12V veya 24V DC güç kaynakları tercih edilir.

➔ **Çıkış Akımı:** 0.2A ila 5A arası (ayarlanabilir)
- ◆ Sürücü, adım motorlarının akımını ayarlamak için kullanıcıya esneklik sağlar. Akım ayarı, motorun torkunu ve ısınmasını kontrol etmek için önemlidir.

➔ **Adım Modları:** Tam adım, yarım adım, çeyrek adım, sekizinci adım, on altıncı adım ve 32'nci adım modları
- ◆ Sürücü, farklı adım modlarını destekleyerek motorun adım çözünürlüğünü ayarlamaya olanak tanır. Bu modlar arasında geçiş, sürücünün kontrol arayüzü aracılığıyla yapılabilmektedir.

➔ **Maksimum Hız:** 1000 RPM
- ◆ Sürücü, adım motorunun maksimum dönüş hızını 1000 RPM'ye kadar destekler. Hız, kullanılan motor ve uygulamaya bağlı olarak değişebilir.

➔ **Koruma Özellikleri:**
- ◆ **Aşırı Akım Koruması:** Sürücü, motorun aşırı akıma maruz kalması durumunda otomatik olarak koruma sağlar.
- ◆ **Aşırı Isınma Koruması:** Sürücü, aşırı ısınma durumunda otomatik olarak koruma sağlar.
- ◆ **Kısa Devre Koruması:** Sürücü, kısa devre durumunda otomatik olarak koruma sağlar.
- ◆ **Aşırı Gerilim Koruması:** Sürücü, aşırı gerilim durumunda otomatik olarak koruma sağlar.

➔ **Giriş Kontrol Sinyali**: Puls-Dir sinyali
- ◆ Sürücü, puls (darbe) ve yön (dir) sinyallerini giriş olarak kabul eder. Bu sinyaller, motorun dönüş yönünü ve adım sayısını belirlemek için kullanılır.

➔ **Boyutlar:** Genellikle 96mm x 56mm x 33mm

- Sürücünün boyutları, model ve üreticiye bağlı olarak değişebilir. Tipik olarak, TB6600 Step Motor Sürücüsü 96mm x 56mm x 33mm boyutlarında bir modüldür.

### 3.3.2 TB6600 Tercih Edilme Sebepleri

---

**Güç ve Gerilim Aralığı:** TB6600, genellikle 9V ile 42V arasındaki bir güç kaynağı gerilimine sahip motorları kontrol edebilir. Bu, farklı uygulamalarda çeşitli step motorlarla uyum sağlama esnekliği sunar.

**Akım Ayarı:** Sürücü, adım motorlarının akımını ayarlama imkanı sunar. Bu, motor performansını optimize etmek ve ısınma sorunlarını önlemek için önemlidir. Akım ayarı, sürücünün üzerinde bulunan potansiyometreler veya dijital arayüzler aracılığıyla yapılabilir.

**Adım Modları:** TB6600, yüksek adım çözünürlüğü sunan farklı adım modlarına (tam adım, yarım adım, çeyrek adım, vb.) sahiptir. Bu, motorun hareket hassasiyetini artırmak ve pürüzsüz bir hareket elde etmek için kullanılabilir.

**Koruma Özellikleri:** Sürücü, aşırı ısınma, kısa devre, aşırı akım ve aşırı gerilim gibi durumlarda otomatik olarak koruma sağlar. Bu, motor ve sürücünün güvenliğini ve dayanıklılığını artırır.

**Kontrol Arayüzü:** TB6600, adımlı motorların kontrolünü kolaylaştırmak için bir dizi kontrol arayüzü sunar. Bu arayüzler, mikrodenetleyiciler, PLC'ler veya diğer dijital cihazlarla entegrasyonu sağlar.

TB6600 Step Motor Sürücü, kullanımının kolaylığı ve sağladığı yüksek performans nedeniyle birçok proje ve uygulama için tercih edilmektedir. Adım motorlarının hassas kontrolünü sağlamak, hızlı ve doğru hareketleri yönetmek ve genel sistem performansını artırmak için ideal bir seçenektir.

## 3.4 BOM File

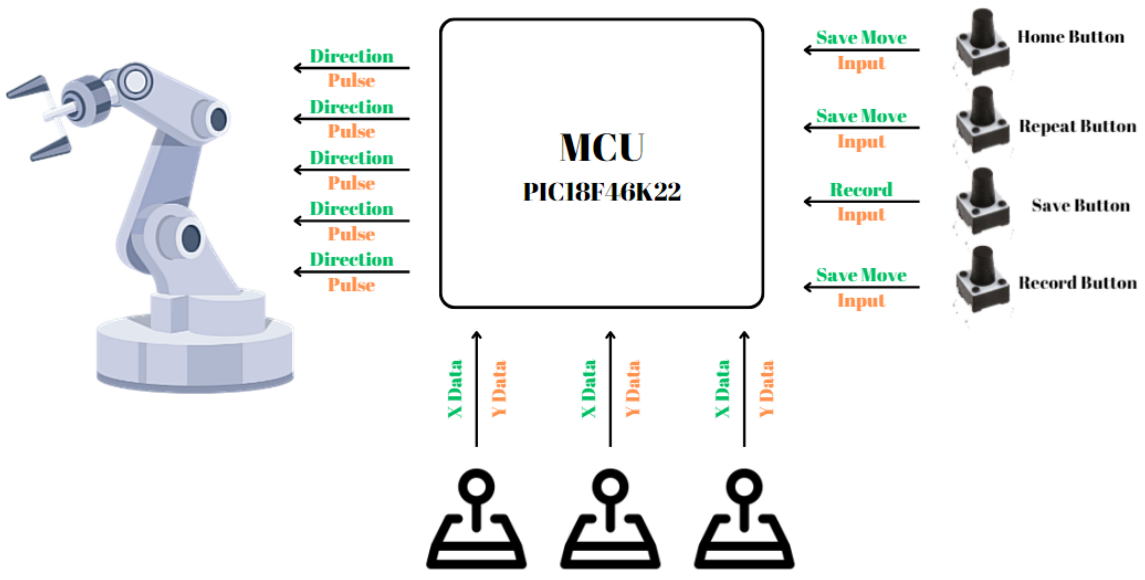
| | A | B | C | D |
|---|---|---|---|---|
| 1 | ID | Name | Designato | Quantity |
| 2 | 1 | 0.1u | C15,C1,C2 | 13 |
| 3 | 2 | Joystick2 | J9,J8 | 2 |
| 4 | 3 | 100R | R6,R7,R8,F | 9 |
| 5 | 4 | 10k | R1,R2,R3,F | 18 |
| 6 | 5 | FanLed | FANLED1 | 1 |
| 7 | 6 | 5VLED | 5VLED | 1 |
| 8 | 7 | 12VLED | 12VLED | 1 |
| 9 | 8 | 470uF/63\ | C6 | 1 |
| 10 | 9 | 470uF/35\ | C10 | 1 |
| 11 | 10 | 1000uF/6. | C7 | 1 |
| 12 | 11 | 30A Diyot | D1 | 1 |
| 13 | 12 | P4SMAJ15 | D2 | 1 |
| 14 | 13 | SS14 | D3 | 1 |
| 15 | 14 | Fuses | F1 | 1 |
| 16 | 15 | +12 V Hatt | H1 | 1 |
| 17 | 16 | ICSP | H2 | 1 |
| 18 | 17 | PM1 | J1,J2,J3,J4 | 5 |
| 19 | 18 | Servo | J6 | 1 |
| 20 | 19 | Joystick1 | J7 | 1 |
| 21 | 20 | K4-12×12_ | KEY1,KEY2 | 4 |
| 22 | 21 | 100uH | L1 | 1 |
| 23 | 22 | 1206B | LED4,LED5 | 8 |
| 24 | 23 | 12V Giriş | P1 | 1 |
| 25 | 24 | 1k | R14,R18,R | 8 |
| 26 | 25 | 220R | RLED5V | 1 |
| 27 | 26 | 560R | RLED12V,F | 2 |
| 28 | 27 | Switch | S100 | 1 |
| 29 | 28 | A4988 #1 | U1 | 1 |
| 30 | 29 | A4988 #2 | U2 | 1 |
| 31 | 30 | A4988 #3 | U3 | 1 |
| 32 | 31 | A4988 #4 | U4 | 1 |
| 33 | 32 | A4988 #5 | U5 | 1 |
| 34 | 33 | LM2576 | U7 | 1 |
| 35 | 34 | SMAJ28A | U8 | 1 |
| 36 | 35 | PIC18F46K | U9 | 1 |

# ŞEMATİK

Şekil 4.1

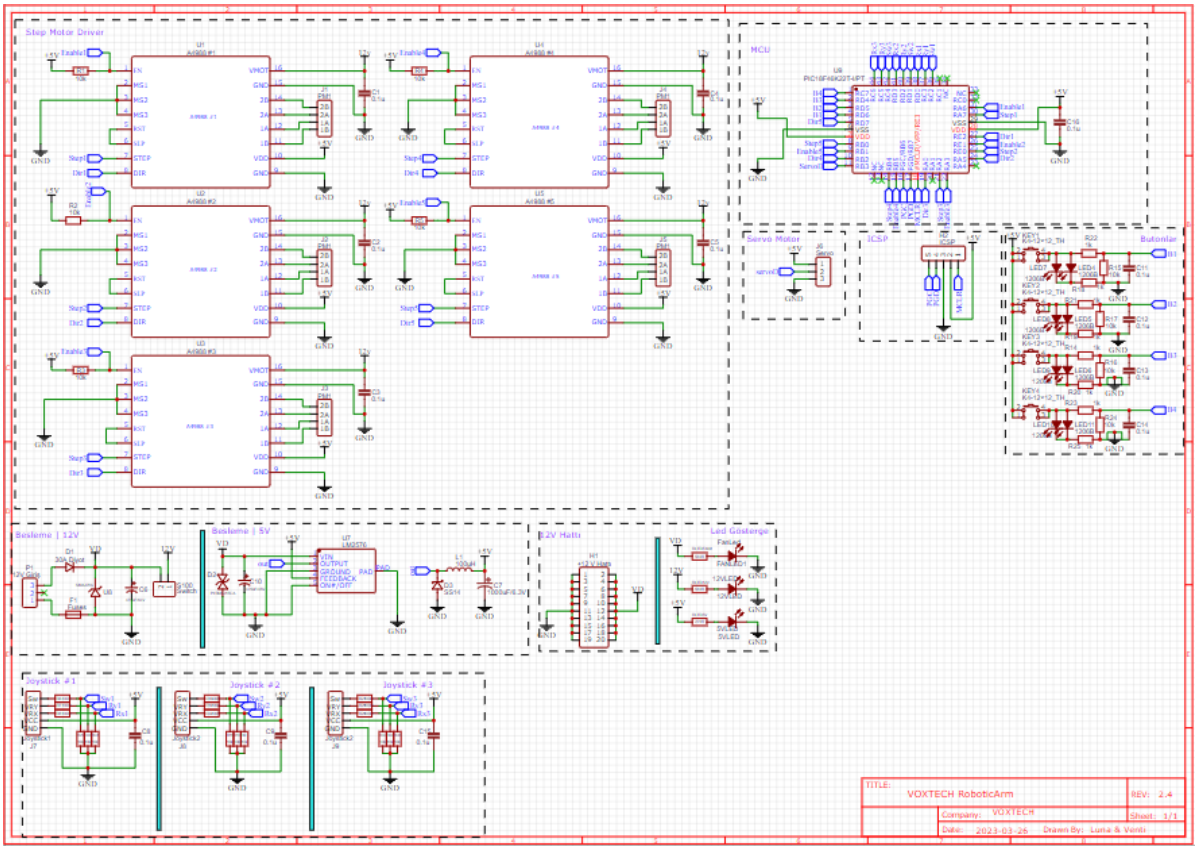
Tüm teknik çizimlerimizi EasyEDA designer programı ile gerçekleştirdik. Projemizin tasarım ve deneylerini simülasyon olarak EasyEDA programı ile gerçekleştirdik. EasyEDA programı, bir Elektrik ve Elektronik Mühendisi için gerekli olabilecek birçok özelliği içerir. Projede kullandıklarımızı bu özelliklerden özetlememiz gerekirse devre çizimi ve simülasyonu, şematik çıkarma, PCB tasarımı, 3D tasarım gibi bir çok özellik bulunmaktadır. Bu özelliklerle tasarladığımız Akıllı Sera Sistem devresimizin çıktılarını aşağıda görebilirsiniz.

## 4.1 Blok Diyagram

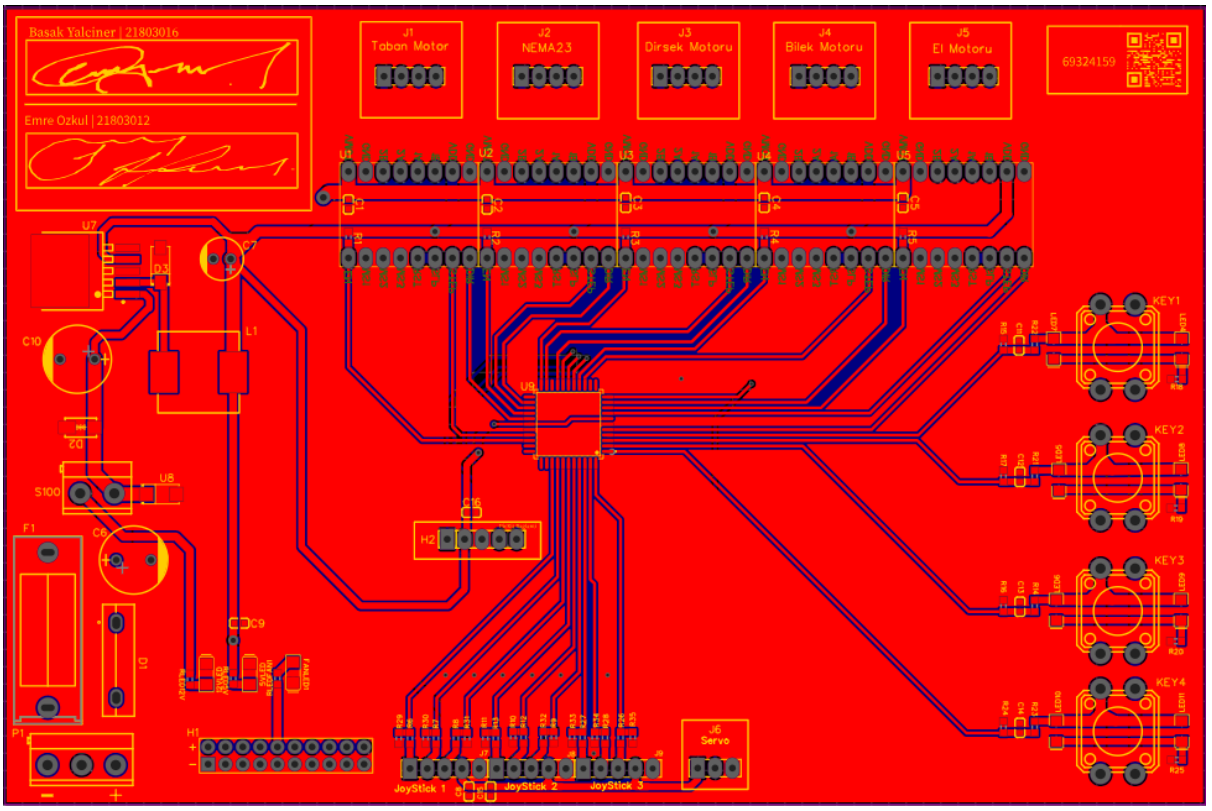


## 4.2 Devre Şematiği

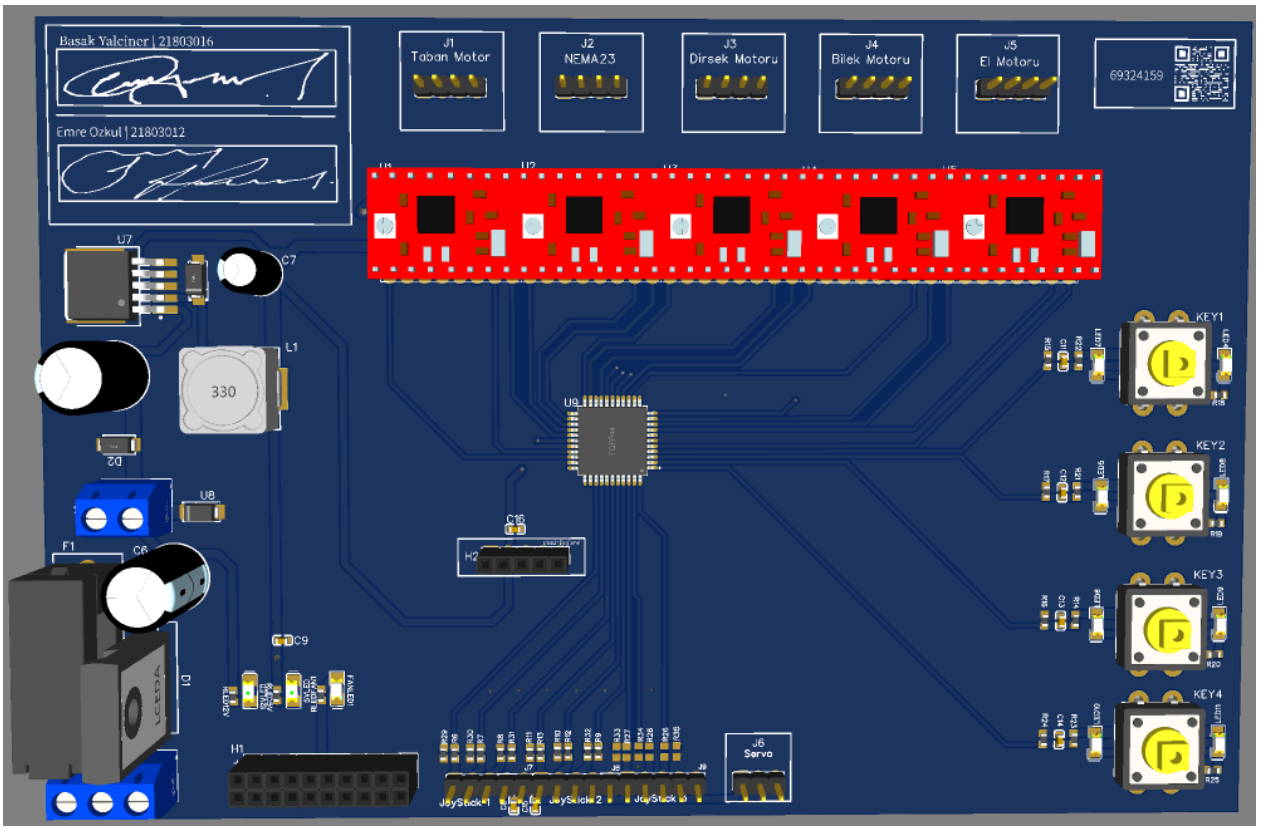
Devremizin tüm bağlantılarını şemada görebilirsiniz. Bu şemada PIC bağlantılarınızın ayakları görülebileceği gibi kullandığımız diğer tüm ürünlerin ayakları da şemada bağlantı yollarını görebilirsiniz. Daha anlaşılır ve basit olması için etiketleme yöntemi kullanılmıştır.



## 4.2 PCB Baskı Şematiği

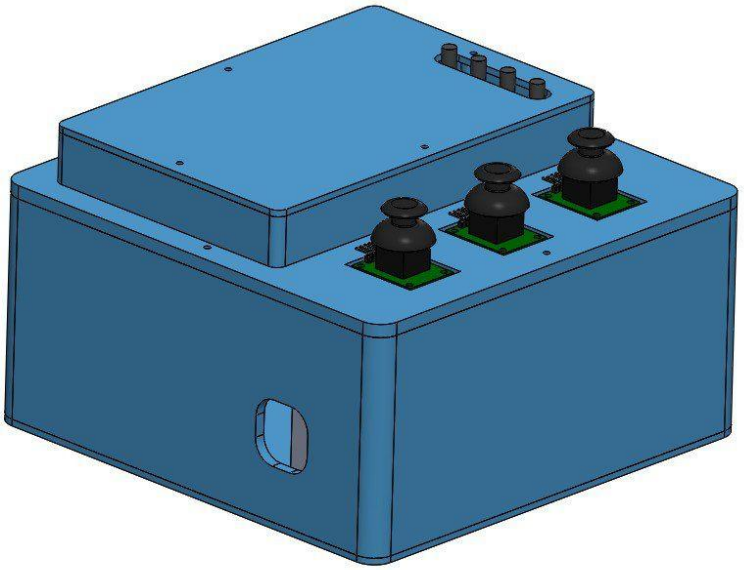


## 4.3 PCB Baskı 3D Görünümü



## 4.4 Robot Kol 3D Tasarımı

## 4.5 Kutu 3D Tasarımı



# KOD BLOĞU

```
#include <18f46k22.h>
#device *=16
#device adc=10

#include <stdint.h>

#fuses INTRC_IO                  //use internal osc (yes !!)
//#fuses PLLEN                     // Phase lock loop enable
#FUSES NOMCLR                    //Master Clear pin
disabled,pin 1( RE3 available !)
#FUSES NOWDT                     //No Watch Dog Timer
#FUSES NOBROWNOUT                // brownout reset
#FUSES NOPUT                     //No Power Up Timer
#FUSES NODEBUG                   //No Debug mode for ICD
#FUSES PROTECT                   //Code protected from reading
#FUSES CPD                       //EE protection
#FUSES WRT                       //Program memory write
protected
#FUSES NOWRTD                      //Data EEPROM write
protected
#FUSES NOWRTB                      //Boot block write
protected
#FUSES NOEBTR                      //Memory protected from
table reads
#FUSES NOEBTRB                     //Boot block protected from
table reads
#FUSES CPB                       //Boot Block code protection
#FUSES WRTC                      //configuration not registers
write protected
#FUSES NOFCMEN                   //Fail-safe clock monitor
enabled
#FUSES NOSTVREN                  //Stack full/underflow will
cause reset
#FUSES NOIESO                    //Internal External Switch
Over mode enabled
```

```
#FUSES NOXINST                  //Extended set extension and
Indexed Addressing mode disabled (Legacy mode)
#use delay(clock=64000000)

#use fast_io(b) // Port yönlendirme komutları b portu için
geçerli
#use fast_io(c) // Port yönlendirme komutları c portu için
geçerli
#use fast_io(d) // Port yönlendirme komutları d portu için
geçerli
#use fast_io(a) // Port yönlendirme komutları a portu için
geçerli
// Taban Motor sürücü pini tanımlamaları
#define ENABLE_TABAN PIN_E1
#define DIR_TABAN PIN_A5
#define STEP_TABAN PIN_E0

// Nema Motor sürücü pini tanımlamaları
#define ENABLE_NEMA PIN_A6
#define DIR_NEMA PIN_E2
#define STEP_NEMA PIN_A7

// Dirsek Motor sürücü pini tanımlamaları
#define ENABLE_DIRSEK PIN_A3
#define DIR_DIRSEK PIN_A0
#define STEP_DIRSEK PIN_A2

// Bilek Motor sürücü pini tanımlamaları
#define ENABLE_BILEK PIN_B5
#define DIR_BILEK PIN_B2
#define STEP_BILEK PIN_B4

// El Motor sürücü pini tanımlamaları
#define ENABLE_EL PIN_B1
#define DIR_EL PIN_D7
#define STEP_EL PIN_B0


// Hareket komutları
#define LEFT 0
#define RIGHT 1


// Hareketleri kaydetmek için dizi
int
```

```
rec_fl=0,i=0,bsk=0,ply_fl=0,clfl=0,home_fl=0,c7_fl=0,clfl_fl=0
,reset_fl=0;
signed int32 x=0, y1=0, y2=0, y3=0, z=0, dif_x=0,
a=0,b=0,dif_x_new=0,pulseX=0,pulseY1=0,dif_y1=0,dif_y1_new=0,p
ulseY2=0,dif_y2=0,dif_y2_new=0,pulseY3=0,dif_y3=0,dif_y3_new=0
,pulseZ=0,dif_z=0,dif_z_new=0;
signed int32
hafiza[10][1][5]={{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0
,0},{0,0,0,0,0}};
//sütun satır içerik


void moveMotorTaban(int direction)
{
   if(direction==LEFT){
      output_high(DIR_TABAN);  // Yön ayarlaması
      output_high(STEP_TABAN);           // Adım sinyalini
yüksek seviyeye ayarla
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_TABAN);
      x++;
      delay_us(300);
   }
   if(direction==RIGHT){
      output_low(DIR_TABAN);  // Yön ayarlaması
      output_high(STEP_TABAN);           // Adım sinyalini
yüksek seviyeye ayarla
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_TABAN);
      x--;
      delay_us(300);
   }
}

void moveMotorNema(int direction)
{
   if(direction==LEFT){
      output_high(DIR_NEMA);  // Yön ayarlaması
      output_high(STEP_NEMA);           // Adım sinyalini
yüksek seviyeye ayarla
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_NEMA);
```

```
      y1++;
      delay_us(300);
   }
   if(direction==RIGHT){
      output_low(DIR_NEMA);  // Yön ayarlaması
      output_high(STEP_NEMA);           // Adım sinyalini
yüksek seviyeye ayarla
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_NEMA);
      y1--;
      delay_us(300);
   }
}

void moveMotorDirsek(int direction)
{
   if(direction==LEFT){
      output_high(DIR_DIRSEK);  // Yön ayarlaması
      output_high(STEP_DIRSEK);           // Adım sinyalini
yüksek seviyeye ayarla
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_DIRSEK);
      y2++;
      delay_us(300);
   }
   if(direction==RIGHT){
      output_low(DIR_DIRSEK);  // Yön ayarlaması
      output_high(STEP_DIRSEK);           // Adım sinyalini
yüksek seviyeye ayarla
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_DIRSEK);
      y2--;
      delay_us(300);
   }
}

void moveMotorBilek(int direction)
{
   if(direction==LEFT){
      output_high(DIR_BILEK);  // Yön ayarlaması
      output_high(STEP_BILEK);           // Adım sinyalini
yüksek seviyeye ayarla
```

```
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_BILEK);
      y3++;
      delay_us(300);
   }
   if(direction==RIGHT){
      output_low(DIR_BILEK);  // Yön ayarlaması
      output_high(STEP_BILEK);           // Adım sinyalini
yüksek seviyeye ayarla
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_BILEK);
      y3--;
      delay_us(300);
   }
}

void moveMotorEl(int direction)
{
   if(direction==LEFT){
      output_high(DIR_EL);  // Yön ayarlaması
      output_high(STEP_EL);           // Adım sinyalini yüksek
seviyeye ayarla
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_EL);
      z++;
      delay_us(300);
   }
   if(direction==RIGHT){
      output_low(DIR_EL);  // Yön ayarlaması
      output_high(STEP_EL);           // Adım sinyalini yüksek
seviyeye ayarla
      delay_us(300);                    // Adım sinyalini 10ms
boyunca yüksek seviyede tut
      output_low(STEP_EL);
      z--;
      delay_us(300);
   }
}


void resetMoves()
{
```

```
pulseX=0-x;
pulseY1=0-y1;
pulseY2=0-y2;
pulseY3=0-y3;
pulseZ=0-z;

if(pulseX>0){
   moveMotorTaban(LEFT);
}
if(pulseX<0){
   moveMotorTaban(RIGHT);
}
if(pulseX==0){
   x=0;
}
if(x==0){
   if(pulseY1>0){
      moveMotorNema(LEFT);
   }
   if(pulseY1<0){
      moveMotorNema(RIGHT);
   }
   if(pulseY1==0){
      y1=0;
   }
}
if(x==0 && y1==0){
   if(pulseY2>0){
      moveMotorDirsek(LEFT);
   }
   if(pulseY2<0){
      moveMotorDirsek(RIGHT);
   }
   if(pulseY2==0){
      y2=0;
   }
}

if(x==0 && y1==0 && y2==0){
   if(pulseY3>0){
      moveMotorBilek(LEFT);
   }
   if(pulseY3<0){
      moveMotorBilek(RIGHT);
   }
```

```
         if(pulseY3==0){
            y3=0;
         }
      }
      if(x==0 && y1==0 && y2==0 && y3==0){
         if(pulseZ>0){
            moveMotorEl(LEFT);
         }
         if(pulseZ<0){
            moveMotorEl(RIGHT);
         }
         if(pulseZ==0){
            home_fl=0;
            z=0;
            reset_fl=1;
         }
      }
}

//!// Kaydedilen hareketleri tekrar etme fonksiyonu
void playMoves(){
   if(bsk==0){
      if(clfl==0 && clfl_fl==0){
         dif_x = 0-hafiza[0][0][0];
         dif_x_new = dif_x;

         dif_y1 = 0-hafiza[0][0][1];
         dif_y1_new = dif_y1;

         dif_y2 = 0-hafiza[0][0][2];
         dif_y2_new = dif_y2;

         dif_y3 = 0-hafiza[0][0][3];
         dif_y3_new = dif_y3;

         dif_z = 0-hafiza[0][0][4];
         dif_z_new = dif_z;
         clfl=1;

      }

      if(dif_x_new < 0){
         moveMotorTaban(LEFT);
         dif_x_new++;
      }
```

```
      if(dif_x_new > 0){
         moveMotorTaban(RIGHT);
         dif_x_new--;
      }

      if(dif_x_new==0){
         if(dif_y1_new < 0){
            moveMotorNema(LEFT);
            dif_y1_new++;
         }
         if(dif_y1_new > 0){
            moveMotorNema(RIGHT);
            dif_y1_new--;
         }
//!         if(dif_y1_new == 0){
//!            bsk++;
//!            clfl_fl = 1;
//!         }
      }

      if(dif_x_new==0 && dif_y1_new==0){
         if(dif_y2_new < 0){
            moveMotorDirsek(LEFT);
            dif_y2_new++;
         }
         if(dif_y2_new > 0){
            moveMotorDirsek(RIGHT);
            dif_y2_new--;
         }
//!         if(dif_y2_new == 0){
//!            bsk++;
//!            clfl_fl = 1;
//!         }
      }
      if(dif_x_new==0 && dif_y1_new==0 && dif_y2_new==0){
         if(dif_y3_new < 0){
            moveMotorBilek(LEFT);
            dif_y3_new++;
         }
         if(dif_y3_new > 0){
            moveMotorBilek(RIGHT);
            dif_y3_new--;
         }
         if(dif_y3_new == 0){
            clfl_fl = 1;
```

```
         }
      }
      if(dif_x_new==0 && dif_y1_new==0 && dif_y2_new==0 &&
dif_y3_new==0){
         if(dif_z_new < 0){
            moveMotorEl(LEFT);
            dif_z_new++;
         }
         if(dif_z_new > 0){
            moveMotorEl(RIGHT);
            dif_z_new--;
         }
         if(dif_z_new == 0){
            bsk++;
            clfl_fl = 1;
         }
      }
   }
//!
   if(bsk==1){
      if(clfl==1){
         dif_x = hafiza[0][0][0]-hafiza[1][0][0];
         dif_x_new = dif_x;

         dif_y1 = hafiza[0][0][1]-hafiza[1][0][1];
         dif_y1_new = dif_y1;

         dif_y2 = hafiza[0][0][2]-hafiza[1][0][2];
         dif_y2_new = dif_y2;

         dif_y3 = hafiza[0][0][3]-hafiza[1][0][3];
         dif_y3_new = dif_y3;

         dif_z = hafiza[0][0][4]-hafiza[1][0][4];
         dif_z_new = dif_z;
         clfl=2;
      }

      if(dif_x_new < 0){
         moveMotorTaban(LEFT);
         dif_x_new++;
      }
      if(dif_x_new > 0){
         moveMotorTaban(RIGHT);
         dif_x_new--;
```

```
      }
      if(dif_x_new == 0){

         if(dif_y1_new < 0){
            moveMotorNema(LEFT);
            dif_y1_new++;
         }
         if(dif_y1_new > 0){
            moveMotorNema(RIGHT);
            dif_y1_new--;
         }


      if(dif_x_new == 0 && dif_y1_new == 0){

         if(dif_y2_new < 0){
            moveMotorDirsek(LEFT);
            dif_y2_new++;
         }
         if(dif_y2_new > 0){
            moveMotorDirsek(RIGHT);
            dif_y2_new--;
         }

      if(dif_x_new == 0 && dif_y1_new == 0 && dif_y2_new ==
0){

         if(dif_y3_new < 0){
            moveMotorBilek(LEFT);
            dif_y3_new++;
         }
         if(dif_y3_new > 0){
            moveMotorBilek(RIGHT);
            dif_y3_new--;
         }
//!          if(dif_y3_new == 0){
//!             bsk=0;
//!             clfl_fl = 0;
//!             clfl = 0;
//!             ply_fl=0;
//!             reset_fl=0;
//!          }
      if(dif_x_new == 0 && dif_y1_new == 0 && dif_y2_new == 0
&& dif_y3_new == 0){
```

```
         if(dif_z_new < 0){
            moveMotorEl(LEFT);
            dif_z_new++;
         }
         if(dif_z_new > 0){
            moveMotorEl(RIGHT);
            dif_z_new--;
         }
         if(dif_z_new == 0){
            bsk=0;
            clfl_fl = 0;
            clfl = 0;
            ply_fl=0;
            reset_fl=0;
         }
      }
      }
      }
      }

   }
}


void main()
{
      set_tris_a(0x00);
      set_tris_b(0x00);
      set_tris_c(0b11111100);
      set_tris_d(0b01111111);
      // Motor sürücü pinlerini çıkış olarak ayarla
      output_low(ENABLE_TABAN);
      output_low(DIR_TABAN);
      output_low(STEP_TABAN);

      // Motor sürücü pinlerini çıkış olarak ayarla
      output_low(ENABLE_NEMA);
      output_low(DIR_NEMA);
      output_low(STEP_NEMA);

      // Motor sürücü pinlerini çıkış olarak ayarla
      output_low(ENABLE_DIRSEK);
      output_low(DIR_DIRSEK);
      output_low(STEP_DIRSEK);
```

```
      // Motor sürücü pinlerini çıkış olarak ayarla
      output_low(ENABLE_BILEK);
      output_low(DIR_BILEK);
      output_low(STEP_BILEK);

      // Motor sürücü pinlerini çıkış olarak ayarla
      output_low(ENABLE_EL);
      output_low(DIR_EL);
      output_low(STEP_EL);


while(TRUE) // sonsuz döngü
{
      output_high(STEP_NEMA);

      //ADC Portların ayarlanması.
      setup_adc_ports(sAN20|sAN15|sAN23|sAN22|sAN14);
      setup_adc(ADC_CLOCK_INTERNAL);


      set_adc_channel(20);
      delay_us(10); // Kanal geçişi için bir bekleme süresi
      int16_t joystick_taban = read_adc();
      delay_us(10);

      set_adc_channel(15); // AN15 pinine geç
      delay_us(10); // Kanal geçişi için bir bekleme süresi
      int16_t joystick_nema = read_adc(); // AN1 pininin
analog değerini oku
      delay_us(10);

      set_adc_channel(23); // AN15 pinine geç
      delay_us(10); // Kanal geçişi için bir bekleme süresi
      int16_t joystick_dirsek = read_adc(); // AN1 pininin
analog değerini oku
      delay_us(10);

      set_adc_channel(22); // AN15 pinine geç
      delay_us(10); // Kanal geçişi için bir bekleme süresi
      int16_t joystick_bilek = read_adc(); // AN1 pininin
analog değerini oku
      delay_us(10);

      set_adc_channel(14); // AN15 pinine geç
      delay_us(10); // Kanal geçişi için bir bekleme süresi
```

```
      int16_t joystick_el = read_adc(); // AN1 pininin analog
değerini oku

      // TABAN HAREKET
      if(joystick_taban > 700)
      {
         moveMotorTaban(LEFT);

      }
      // TABAN HAREKET
      else if(joystick_taban < 250)
      {
          moveMotorTaban(RIGHT);

      }

      // NEMA HAREKET
      if(joystick_nema > 700)
      {
         moveMotorNema(LEFT);

      }
      // NEMA HAREKET
      else if(joystick_nema < 250)
      {
          moveMotorNema(RIGHT);

      }

      // DİRSEK
      if(joystick_dirsek > 700)
      {
         moveMotorDirsek(LEFT);

      }
      // DİRSEK
      else if(joystick_dirsek < 250)
      {
          moveMotorDirsek(RIGHT);

      }
      // BİLEK
      if(joystick_bilek > 700)
      {
         moveMotorBilek(LEFT);
```

```
}
// BİLEK
else if(joystick_bilek < 250)
{
    moveMotorBilek(RIGHT);

}

// EL
if(joystick_el > 700)
{
   moveMotorEl(LEFT);

}
// EL
else if(joystick_el < 250)
{
    moveMotorEl(RIGHT);

}


// C7 pinine bağlı düğme kontrolü
if(input(PIN_C7) && rec_fl==0)
{
   rec_fl=1;
   delay_ms(500);

}
if(input(PIN_C7)){
   c7_fl = 1;
   delay_ms(200);
}

if(c7_fl==1 && rec_fl==1)
{
   resetMoves();
   if(x==0 && y1==0 && y2==0 && y3==0 && z==0){
      c7_fl=0;
      rec_fl=0;
      i=0;
   }

}
```

```
      if(rec_fl==1){
         if(input(PIN_D4) && i==0){
            hafiza[i][0][0]=x;
            hafiza[i][0][1]=y1;
            hafiza[i][0][2]=y2;
            hafiza[i][0][3]=y3;
            hafiza[i][0][4]=z;
            i=1;
            delay_ms(200);
         }

         if(input(PIN_D4) && i==1){
            hafiza[i][0][0]=x;
            hafiza[i][0][1]=y1;
            hafiza[i][0][2]=y2;
            hafiza[i][0][3]=y3;
            hafiza[i][0][4]=z;
            i=2;
            delay_ms(200);
         }
      }

   if(input(PIN_D5) && ply_fl==0)
   {
      ply_fl=1;
      delay_ms(200);
   }
   if(input(PIN_D5) && ply_fl==1)
   {
      clfl_fl = 0;
      clfl = 0;
      ply_fl=0;
      reset_fl=0;
      delay_ms(200);
   }
   if(ply_fl==1)
   {
      if(reset_fl==0){
         resetMoves();
      }
      if(reset_fl==1){
         playMoves();
      }
   }
//......... HOME BUTON
```

```
   if(input(PIN_D6) && home_fl==0)
   {
      home_fl=1;
      delay_ms(200);
   }
   if(input(PIN_D6) && home_fl==1)
   {
      home_fl=0;
      delay_ms(200);
   }
   if(home_fl==1)
   {
      resetMoves();
   }

//......... HOME BUTON
}
}
```