İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DOĞRUSAL (LİNEER) MOTORLU ENDÜSTRİYEL SİSTEMİN 3 BOYUTLU YAZICIYA DÖNÜŞTÜRÜLMESİ

YÜKSEK LİSANS TEZİ

Fatih TARAK

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Haziran 2021

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DOĞRUSAL (LİNEER) MOTORLU ENDÜSTRİYEL SİSTEMİN 3 BOYUTLU YAZICIYA DÖNÜŞTÜRÜLMESİ

YÜKSEK LİSANS TEZİ

Fatih TARAK

518171041

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Ali Fuat ERGENÇ

Haziran 2021

İTÜ, Lisansüstü Eğitim Enstitüsü'nün 518171041 numaralı Yüksek Lisans Öğrencisi Fatih TARAK, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "Doğrusal (Lineer) Motorlu Endüstriyel Sistemin 3 Boyutlu Yazıcıya Dönüştürülmesi" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Doç. Dr. Ali Fuat ERGENÇ** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Doç. Dr. Hülya Cebeci** İstanbul Teknik Üniversitesi	.....
	Doç. Dr. Türker Türker Yıldız Teknik Üniversitesi	.....

Teslim Tarihi : 11 Mayıs 2021

Savunma Tarihi : 23 Haziran 2021

ÖNSÖZ

Yüksek lisans öğrenimim süresince bilgi ve birikimlerini esirgemeyen tüm İstanbul Teknik Üniversitesi öğretim üyelerine ve özellikle danışmanlığımı üstlenen sayın Doç. Dr. Ali Fuat ERGENÇ hocama destekleri ve sabrı için teşekkür ederim.

Tez çalışmamda kullandığım sistemlerde teknik destek sağlayan Rockwell Otomasyon ailesine ayrıca teşekkür ederim.

Haziran 2021

Fatih TARAK

(Mekatronik Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

3D	: 3 Boyut
PLC	: Programlanabilir Lojik Kontrolcü
PLA	: Polilaktik Asit
ABS	: Akrilonitril Bütadien Stiren
IoT	: Nesnelerin İnterneti
IIoT	: Endüstriyel Nesnelerin İnterneti
CAD	: Bilgisayar Destekli Tasarım
CAM	: Bilgisayar Destekli İmalat
SLA	: Stereolitografi
SLS	: Seçici Lazer Sinterleme
FDM	: Eriyik Yığma Modelleme
UDP	: Kullanıcı Veri Bloğu Protokolü
TCP	: Geçiş Kontrol Protokolü

SEMBOLLER

G	: İyilik faktörü
τ	: Rotor sargısının kutup adımları
f	: Kaynak frekansı,
g	: Hava aralığı uzaklığı,
μ**0**	: Boşluğun geçirgenliği
vs	: Lineer senkron hız

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

DOĞRUSAL (LİNEER) MOTORLU ENDÜSTRİYEL SİSTEMİN 3 BOYUTLU YAZICIYA DÖNÜŞTÜRÜLMESİ

ÖZET

Üretim teknolojileri insanlığın isteklerine cevap verebilmek için sürekli gelişim göstermek zorundadır. Tarih boyunca temelini otomotiv, savunma, uzay ve havacılık gibi farklı sektörlerden alan yeni üretim tekniği ihtiyaçları gelişen teknoloji ve kontrol yöntemleri ile beraber yenilikçi imalat yöntemlerini doğurmuştur. 1980'lerden itibaren üzerinden çalışılan ancak son 20 senede büyük bir ivme kazanan eklemeli imalat (Additive Manufacturing) yöntemi yeni üretim tekniklerinden biridir.

Geleneksel imalat yöntemlerinde üretilen yarı mamul ya da parçadan freze, torna gibi malzeme eksiltme (talaş kaldırma) yöntemleri ile son ürün ortaya çıkmaktadır. Son dönemlerde geleneksel imalat teknikleri kullanan makine ve tezgâhlar ile çok hassas ve kaliteli yüzeye sahip parçalar üretebilmektedir. Bu durumun oluşmasında en büyük pay ileri kontrol yöntemleri, gelişmiş hareket sistemleri ve hassas eyleyicilerindir. Günümüzde neredeyse tüm ürünleri hem hafifletmek hem de dayanımı artırmak için çalışmalar yapılmaktadır. Çalışmalar sonucunda genellikle karmaşık geometrik yapıya sahip parçalar ortaya çıkmaktadır. Bu durum talaşlı imalat yöntemlerinin ve tezgâhlarının sınırlarını zorlamaktadır.

Eklemeli imalat yöntemi günümüz isteklerine en hızlı cevap verecek yöntem olarak karşımıza çıkmaktadır. Söz konusu yöntemde kabaca, malzeme katmanlar halinde üst üste eklenerek parça (ürün) oluşturulur. Bu neden ile katmalı imalat yöntemini kullanan sistemlere genel olarak 3D yazıcı (3 Boyutlu Yazıcı) denmektedir. 3D yazıcıların esnek yapısı birbiri ile alakası olmayan birçok sektör ve üreticinin ilgisini çekmiştir. Bugün farklı boyutta ve özellikteki 3D yazıcılar biyonik organ, motor parçası, elektronik kart ve hatta yaşanabilir ev üretimine kadar çeşitli konularda kullanılmaktadır. Bu çeşitliğin en büyük nedeni katmanlı imalat yönteminde çok farklı malzemelerin hammadde olarak kullanılmasıdır. Hobi amaçlı kullanılan 3D yazıcılarda genellikle termoplastik malzemeler kullanılmak ile beraber farklı sektörler için üretim yapan 3D yazıcı sistemlerinde metal, ahşap, seramik, çimento ve farklı bio-malzemeler hammadde olarak kullanılmaktadır. Eklemeli imalat yöntemi gelişen teknoloji ile son yıllarda çok hızlı gelişip yaygınlaşmasına rağmen bazı sektör ve uygulamalar için elzem olan yüksek yüzey kalitesini sağlayamamaktadır. Yüksek yüzey kalitesi ihtiyacı olan ürünler eklemeli imalat ile üretildikten sonra tekrar işleme (genellikle talaş kaldırma) sokularak istenilen kalite elde edilmektedir. Bu şekilde isterlere sahip olan ürünler özelinde farklı bir imalat yöntemi ortaya çıkmıştır. Hibrid (Melez) üretim yöntemi adı verilen bu yöntem ile talaşlı imalat ve eklemeli imalat

yetenekleri tek sistemde birleştirilmiştir. Bu sayede hali hazırda kullanılan iki yöntemin avantajları birleştirilmiştir.

Son on yılın gelişmekte olan üretim teknolojisi katmanlı imalat yöntemleri (3D yazıcı sistemleri) de bu değişimler ile beraber ele alınmıştır. Tez kapsamında laboratuvarda bulunan Rockwell Otomasyon doğrusal motor deney seti 3D yazıcıya dönüştürülmüştür. Deney setinde 3 eksende bulunan 4 adet doğrusal (lineer) motor ile kartezyen hareket sağlanmıştır. Sistemde X ve Y eksenlerinde hareket için birer adet, Z ekseninde hareket için ise iki adet doğrusal motor bulunmaktadır. Z ekseninde dikey hareketi sağlayan doğrusal motorlar Y eksenine paralel olarak konumlandırılmış ve oluşturulan mekanik düzenek ile dikey hareket sağlamaktadır. Bu mekanik sisteminde önceden bulunan tutucu (gripper) sökülüp yerine 3D yazıcı iticisi (Extruder) ve sıcak ucu (HotEnd) monte edilmiştir. 3D yazıcı baskı yatağı olarak metal 'L' tipi köşebentler ile sabitlenmiş cam kullanılmaktadır.

Eklemeli imalat yönteminde üretilecek parça önce bilgisayar destekli tasarım (Computer Aided Design) yazılımları ile oluşturulmaktadır. Tasarlanan katı modeller bir tür bilgisayar destekli üretim (Computer Aided Manufacturing) yazılımı olan dilimleyici (Slicer) yazılımlarına farklı dosya formatlarında aktarılır. Bu yazılımlar katı model tasarımlarını 3D yazıcı yazılımlarının anlayacağı G-Kodlarına çevirmektedir. G-Kodlar içerisinde sistemin hareket etmesi gereken X,Y ve Z ekseni konumları ve hızları, ısıtıcı uçtan akıtılacak malzeme miktarı, ısıtıcı uç ve varsa ısıtıcı yatağın sıcaklık değeri ve sisteme ait birçok özelliğin kullanımına ait bilgiler bulunmaktadır. 3D yazıcılardaki kontrolcülerde bulunan yazılımlar G-Kodları anlamlandırır ve gerekli eylemleri düzenler. Tez kapsamında G-Kod okuma işlemi bilgisayarda koşturulan ve Python dili ile hazırlanan yazılım ile gerçekleştirilmiştir. Sistem ve sistemde bulunan motorların kontrolü ise PLC içerisinde bulunan Studio5000 yazılımı ile kontrol edilmektedir. Harici bilgisayarda çalışan Python yazılımı ve PLC'de bulunan Studio5000 yazılımı arasındaki haberleşme Ethernet protokolü ile sağlanmıştır. Tez içeriğinde Endüstriyel Nesneleri İnterneti (IIoT) konusu çalışılmış ve bir endüstriyel kontrolcü olan PLC, bulut tabanlı MQTT protokolü ile kontrol edilmeye çalışılmıştır. Bu sayede haberleşme yöntemleri açısından iki farklı yazılım mimarisi oluşturulmuş oldu.

Sistem Allen-Bradley marka 1756-L73 PLC ile kontrol edilmekte ve her doğrusal motor için Kinetix 6000 servo motor sürücüsü bulunmaktadır. Kinetix 6000 servo motor sürücüler kontrol organı olan PLC ile Sercos protokolü ve 1756-M08SE Sercos modülü ile haberleşmektedir. Sistemde 3D yazıcının filament beslemesi için bir adet adım motoru kullanılmıştır ve bu adım motoru PLC'de bulunan 24V dijital çıkış birimleri tarafından TB6600 adım motoru sürücü ile beraber kontrol edilmektedir. 3D yazıcının ısıtıcısı (HotEnd) 12V ısıtıcı fişek rezistans ve 100K direnç değerine sahip NTC termistöründen oluşmaktadır. 12V ısıtıcı fişek rezistans PLA filamentin eriyebilmesi için gerekli olan 200 – 210 C^{o} sıcaklığa çıkarmaktadır. Sistem sıcaklığı 100K NTC ve NTCnin bağlı olduğu PLC analog giriş modülü ile kontrol altında tutulmaktadır.

TRANSFORMATION OF INDUSTRIAL LINEAR MOTION SYSTEM IN TO THE 3D PRINTER

SUMMARY

Production technologies must constantly evolve in order to respond to the demands of humanity. Throughout history, the needs of new production techniques, which are based on different sectors such as automotive, defence, space and aviation, have spawned innovative manufacturing methods together with developing technology and control methods. The Additive Manufacturing method, which has been worked on since the 1980s but has gained great momentum in the last 20 years, is one of the new production techniques.

The end product comes out with the material reduction (chip removal) methods such as milling and turning from the semi-finished product or part produced in traditional manufacturing methods. Recently, it has been able to produce parts with very sensitive and high quality surfaces with machines and benches using traditional manufacturing techniques. The biggest share in the occurrence of this situation is advanced control methods, advanced motion systems and precision actuators. Nowadays, studies are carried out to both lighten and increase the strength of almost all products. As a result of the studies, parts with complex geometric structures generally emerge. This situation pushes the limits of machining methods and benches.

The additive manufacturing method will emerge as a quick answer of new challenges of today's world. In manufacturing process, material is adding layer by layer to create a solid product. For this reason, systems that are using additive manufacturing method are generally called as a 3D printer. The types of 3D printers have attracted the attention of many sectors and they have application that are not related to each other. Today, 3D printers of different sizes and features are used in a variety of production ranging from bionic organ, engine part, electronic board and even liveable house production. Metal, wood, ceramic, cement and other different bio-materials are used as raw materials in 3D printer systems that produce for different sectors, however 3D printers which are used for hobby purposes are generally use thermoplastic materials in 3D printers. Although the add-on manufacturing method develops and becomes widespread rapidly, they cannot serve high quality surface applications that are essential for some sectors and applications. There is a new way to supply high quality surfaces for such applications. Products that require high surface quality are produced with additive manufacturing, and then re-processed (chip removal) to achieve the desired surface quality. With this method called hybrid production method, machining and additive manufacturing capabilities are combined in a single system. In this way, two methods that are currently used have been combined.

The developing production technology of the last decade, additive manufacturing methods (3D printer systems) are also discussed along with these changes in this work. Within the scope of the thesis, the Rockwell Automation linear motor experiment set in the laboratory was transformed into a 3D printer. Cartesian motion was achieved with 4 linear (linear) motors in 3 axes in the experiment set. In the system, there are one each for movement in X and Y axes and two linear motors for movement in Z axis. Linear motors that provide vertical movement in the Z axis are positioned parallel to the Y axis and provide vertical movement with the specific mechanical mechanism. In this mechanical system, the gripper previously found was removed and replaced with the 3D printer extruder and the hot end. A glass fixed with metal 'L' type brackets is used as 3D printer printing bed.

The part to be produced in the additive manufacturing method is first created with Computer Aided Design software. Designed solid models are transferred in different file formats (.stl or .step) to slicer software, which is a kind of Computer Aided Manufacturing software. These software translate solid model designs into G-Codes that 3D printer software can understand. Within the G-Codes, there are information about the X, Y and Z axis positions and speeds that the system should move, the amount of material to be flowed from the extruder, the temperature value of the hot end and the heater bed, if any, and the use of many features of the system. The software in the controllers of 3D printers read the G-Codes and arrange the necessary actions. Within the scope of the thesis, the G-Code reading process was carried out with software running on the computer and prepared with Python language. The control of the system and the motors in the thesis is controlled by Studio 5000 software in the PLC. The communication between the Python software running on the external computer and the Studio5000 software in the PLC is provided by the Ethernet protocol. Also in the thesis, Industrial Internet of Things (IIoT) topic was studied and PLC, an industrial controller, is tried to be controlled by cloud-based MQTT communication protocol. In this way, two different software architectures are created in terms of communication methods.

There are two different types of software algorithm built to control printing process. In the first algorithm, after computer sends G-Code data to PLC controller, computer wait for feedback to takes the next action (read new G-Code line). With this closed loop approach, system can be sure that previous motion has done. On the other hand, second algorithm is created with open loop method. Second algorithm not only reads G-Codes to take action but also computes the each process's motion time and software wait for that time duration to read next G-Code line.

The system is controlled by Allen-Bradley 1756-L73 PLC and there is a Kinetix 6000 servo motor driver for each linear motor. Kinetix 6000 servo motor drives communicate with PLC controller with Sercos protocol and 1756-M08SE Sercos module. A step motor is used in the system for feeding the filament to the 3D printer and this step motor is controlled by the 24V digital output units in the PLC together with the TB6600 step motor driver. The heater of the 3D printer (HotEnd) consists of a 12V heating cartridge resistor and a NTC thermistor with a resistance value of 100K. The 12V heating cartridge raises the resistance to the temperature of 200 - 210 C - required for the PLA filament to melt. System temperature is kept under control with PLC analogue input module to which 100K NTC and thermocouple are connected.

1 GİRİŞ

Değişen dünya ve insanlardaki ürün algısı, kullanıcılarda özelleştirilmiş (kişiselleştirilmiş) ürün talepleri oluşturmaktadır. Son yıllarda insanlar sadece kendisi için üretilmiş veya belli özelliklerini değiştirebildiği ürünleri tercih etmektedir. Bugün tüm dünyanın devasa ürün talebini karşılayan seri üretim hatlarının doğasında ise aynı ürünü aynı kalitede hızlı üretme felsefesi vardır. Bu durumun yanında küresel çapta yaşanan Covid-19 salgın hastalığı beraberinde ülkelerin sınırlarını süresiz kapatması, ithalat ve ihracata kısıtlama getirilmesi gibi olağan üstü kararlar getirdi. Bu durum başta sosyal hayata ve ticarete olduğu gibi küreselleşmeyi de sekteye uğrattı. Artık firmalar ve ülkeler alternatif tedarik zincirleri kurmaya ve kendileri için önemli olan ürünleri kendileri üretmeye çalışıyor.

Ürün talebindeki değişim ve Covid-19 salgın hastalığı insansız sistemlerin önemini tüm sektörler ve uygulamalarda bir kez daha ispatlamış oldu. İnsansız sistemlerin kaçınılmaz ve gerekli olduğunun anlaşılması IoT ve dolayısıyla Endüstri 4.0 ve Siber-Fiziksel sistemlerin ihtiyacını pekiştirmiş oldu. Söz konusu durumlar nesnelerin interneti (IoT) ve beraberinde getirdiği Endüstri 4.0 değişimi ile düşük maliyet ile çok sayıda üretim yapabilen seri üretim hatlarında bulunan talaşlı imalat tezgâhlarının esnekliklerini sorgulamaktadır.

Oluşan değişim ve tüm yeni değişimlere en uygun cevabı katmanlı üretim yöntemi verebilmektedir. Talaşlı imalatın aksine eklemeli imalatın doğası gereği parça üretimi için kalıp ya da fikstüre gerek yoktur. Bu neden ile farklı parçalar üretmek için makine ya da tezgâhta değişiklik yapmaya gerek yoktur, bir sonraki farklı parça hemen üretilebilir. Bu sayede birbirinden tamamen farklı şekil ve özelliklere sahip ürünler aynı cihazdan hiçbir değişiklik gerektirmeden arka arkaya üretilebilmektedir. Ayrıca eklemeli imalat yönteminin bir diğer avantajı ise talaşlı imalat yöntemleri gibi malzeme eksiltmediği için fire miktarı geleneksel yöntemler ile kıyaslandığında yok denecek kadar azdır ve bu sayede birim başına maliyet azalmaktadır

Bu çalışmada İTÜ Güç ve Hareket Kontrol laboratuvarında bulunan Rockwell otomasyon ürünü DriectDrive doğrusal motor sistemi termoplastik malzeme (PLA

filament) kullanan ve MQTT protokolü ile kontrolü sağlanan bir 3D yazıcıya dönüştürülmüştür.

2. 3 BOYUTLU YAZICILAR

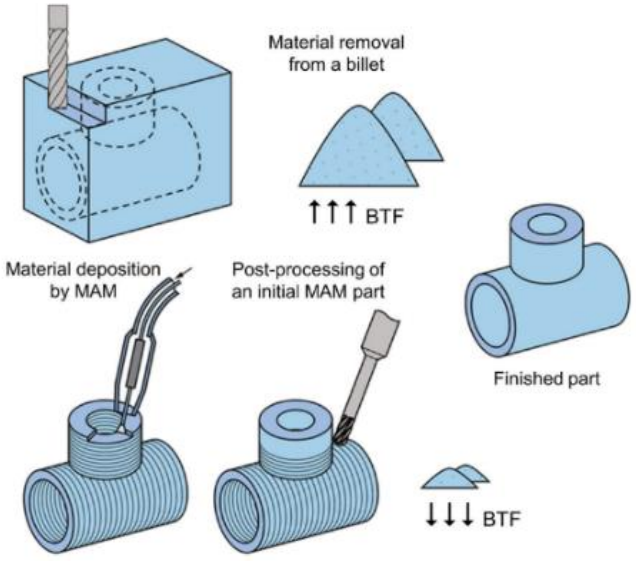
Geçmişte olduğu gibi, üretim teknolojisi, son yüzyılda teknoloji ve bilimdeki hızlı ilerlemeler sayesinde hem üreticiyi hem de tüketiciyi tatmin edecek şekilde gelişmeye devam etmektedir. Üretim sistemlerindeki en son yenilik Üç Boyutlu (3D) baskı teknolojisidir. 3D baskı makineleri (3D yazıcı) kullanılarak yapılan bu üretim türüne katmanlı üretim ya da eklemeli imalat denmektedir. Bunun nedeni parça üretiminin katman katman (layer-by-layer) yapılmasıdır. Eklemeli imalat geleneksel üretim teknikleri olan talaşsız ve talaşlı imalat usullerine göre çok farklı bir yapıdadır. İlk olarak Charles Hull tarafından 1984 yılında patentlenmiştir (Hull, 1984). 3D baskı, ışığın fotokimyasal süreçler kullanarak özel fotopolimer reçineyi katmanlar halinde katılaştırması (SLA, Stereolitografi) teknolojisi ile kullanılmaya başlanmıştır (*3D Systems*, 2021) . Günümüzde, geliştirilen diğer birçok eklemeli imalat yöntemleri ve malzemeler ile çok farklı sektörler ve uygulamalar için hızlı prototipleme ve üretim yapılmaktadır.

Eklemeli imalat yöntemi birçok sektör ve uygulama için kullanılabilir. Temelinde 3 boyutta hareket ve katma ekleme yöntemi kullanan bütün sistemler bu üretim yöntemine dâhildir. Uygulamalara ve ihtiyaca binaen farklı malzemeler hammadde olarak kullanılabilir. Çimento, seramik, ahşap, metal veya termoplastik polimer malzemeler yaygın olarak kullanılmaktadır. Diğer malzeme türlerine nazaran metal ve polimer malzemeler farklı çeşitleri ile geniş kullanım alanlarına yayılmıştır. Metal ürünler için toz yatağı füzyonu (örn.:LBM, EBM, DMLS) ve direkt enerji biriktirme (örn.: LDM, WAAM) teknikleri ile kabaca ikiye ayrılır.

Polimer malzemeler SLA (stereolithography), FDM (fused deposition modeling) ve SLS (selective laser sintering) yöntemleri ile kullanılmaktadır. Bu yöntemlerden en çok kullanılan yöntem ABS (acrylonitrile butadiene styrene) ve PLA (polylactic acid) polimer malzemelerinin kullanıldığı FDM yöntemidir (Paolini et al., 2019). Yazdırma işlemi sırasında sıcaklık PLA için 190ºC - 210ºC, ABS için 230ºC – 250ºC olmalıdır (*Artı Boyut*, 2021). 250ºC üstü yüksek sıcaklıkta kullanılan termoplastik polimerler

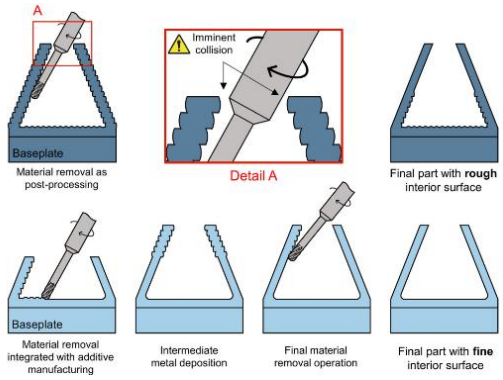
Yüksek Sıcaklık katmanlı imalat olarak adlandırılmıştır (Das et al., 2020) . 4D üretim tekniği temel sistem olarak 3D yazıcılar ile birebir olarak aynıdır. Bu sistemlerde dışarıdan bir etki ile önceden öğretilen şekle dönüşebilen akıllı malzemeler (Smart Materials) kullanılmaktadır. 4 boyutlu yazıcılarda zaman 4. boyut olarak isimlendirilmektedir. Söz konusu yöntem ile üretilen kendiliğinden dönüşen, şekil değiştiren ürünler uzay ve havacılık, medikal ve savunma sanayii gibi birçok farklı sektörde kullanılmaktadır (Mitchell et al., 2018).

İki veya daha fazla farklı üretim tekniğinin tek bir birleşik sistem ile birleştirilmesine hibrid üretim tekniği denilmektedir. Her ne kadar eklemeli metal üretim teknikleri geleneksel yöntemlerden birçok konuda daha iyi olsa da düşük üretkenlik ve yüzey kalitesi, metalürjik hatalar ve kısıtlı ölçü hassasiyeti gibi yönleri ile tüketiciye yeterli cevabı veremeyebilir. Bu durumda ise Hibrid metal üretim teknikleri elemeli imalat ve geleneksel talaşlı imalat yöntemlerinin avantajlarını birleştirerek yeni bir üretim tekniği sunmaktadır (Pragana et al., 2021). Şekil 2.1 ve şekil 2.2'de hibrid metal katmanlı imalat yöntemi ile geleneksel imalat yöntemleri iki farklı örnek üzerinde karşılaştırılmıştır.



Şekil 2.1 : Geleneksel / Hibrid üretim.

Hibrid üretim tekniği eklemeli üretim yöntemlerinin en büyük avantajı olan az miktarda fire vermeyi devam ettirmekte. Ayrıca karmaşık geometrilere sahip ürünleri geleneksel sistemlere göre daha hızlı ve ucuza üretebilmektedir (Lorenz et al., 2020).



Şekil 2.2 : Hibrid üretim tekniği

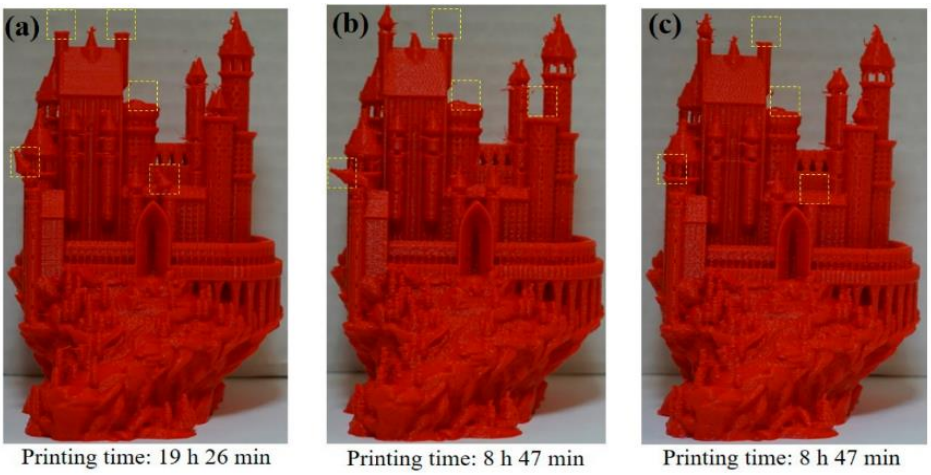
2.1 Çalışma İlkesi

Katmanlı üretim tekniği bilgisayar destekli modelleme ve bilgisayar destekli üretim yazılımları ile direkt ilişkilidir. 3D yazıcı için öncelikle üretilmek istenen parça bilgisayar destekli modelleme yazılımları (CAD) ile modellenmelidir. Dijital model ile üretilmek istenen parçanın nihai ölçüleri ve geometrisi belirlenmiş olur. Oluşturulan bilgisayar modeli .STL formatında kaydedilerek 3D yazıcıların bilgisayar destekli üretim yazılımı (CAM) olan dilimleyici (slicer) yazılımlarına aktarılır. Slicer yazılımı bilgisayar tasarımını 3D yazıcının kullanabileceği eksenlerin konum, hız, sıcaklık ve diğer birçok parametreyi kontrol eden .gcode uzantılı G-Kod dosyasına çevirir. G-kod dosyası kablolu ya da kablosuz farklı haberleşme protokolleri (Ethernet, MQTT, Bluetooth) ya da USB veya hafıza kartı ile 3D yazıcıya iletilir ve yazıcı ürünü basmaya başlar.

Piyasada bulunan ve genelde hobi amaçlı kullanılan 3D yazıcılarda çoğunlukla Arduino, RasperyPi gibi kontrolcüler kullanılmaktadır. Daha büyük çaptaki 3D yazıcılar çoğunlukla amaca özgü tasarlanan kendi kontrol birimleri ile ya da daha az sıklıkta kullanılan PLC'ler tarafından kontrol edilmektedir. PLC kullanımına örnek olarak özel olarak geliştirilen 3D inşaat yazıcısını gösterebiliriz. Sistemde Alt katman kontrolcüsü olarak kullanılan PLC ile özel olarak oluşturulan çimento karışımı katmanlar halinde dökülerek ev üretimi yapılmaktadır (Bazhanov et al., 2018).

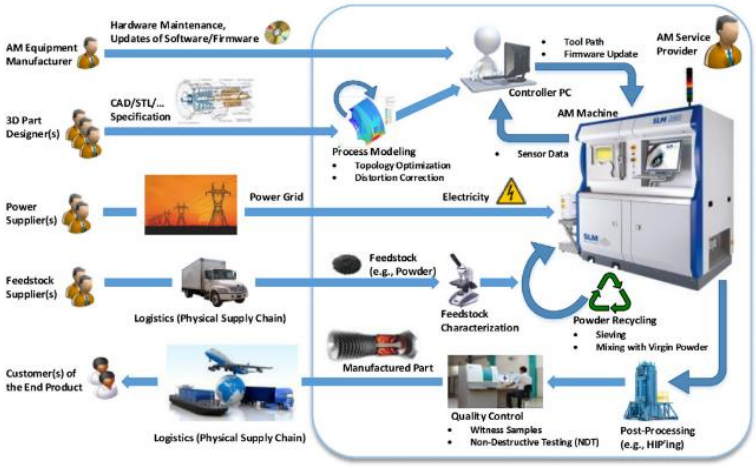
Tezdeki bulut temelli kontrol yapısının dışında ağ bağlantılı gecikmeleri düşüren çalışma sistematiği ile yapılan bir çalışmada, 3D yazıcı bulut ile kontrol edilmiş ve yazdırma işlemi için gerekli olan parametreler Google bulut sisteminden işlenmiştir. Bu yöntemde CAD dosyalarından dilimleyici (slicer) yazılımı ile elde edilen G-Kodlar bulut sistemi tarafında işlenmiş ve 3B yazıcıya TCP protokolü kullanılarak alt kademe

(low-lewel) komutlar (örn. adım motorlarının konumları) iletilmiştir. Çalışmada dâhili kontrolcü ile beraber Sydney / Avustralya ve Carolina / ABD de bulunan Google sanal bilgisayarları kullanılmış ve üç yapının yazdırma süreleri ve yüzey kalitesi benzer olmuştur (Şekil 2.3). Bu çalışmadan bulut tabanlı eklemeli imalat yöntemlerinin üzerinde daha çok çalışarak ve bu sistemler için geliştirilecek sanal bilgisayar ile 3D yazıcı teknolojisi daha da esnek ve verimli hale gelebilir (Okwudire et al., 2018).



Şekil 2.3 : Farklı konumlarda bulunan kontrolcülerin çıktıları

Katmalı üretim tekniği iş akış şemasında (şekil 2.4) açıkça görüldüğü üzere nerede ise tamamen ağ ve bilgisayar tabanlı bir üretim yöntemidir. Bu yönü kritik üretim tesisleri, üretim teknolojileri ve ürünler içeren firmalar için potansiyel bir tehdit kapısıdır (Yampolskiy et al., 2018). 3D yazıcılar endüstri 4.0 ile beraber seri üretim hatları gibi fabrikalardaki üretim sistemlerini değiştirecek potansiyele sahiptir.



Şekil 2.4 : Katmalı üretim iş akış şeması

Geleneksel merkezi ve hiyerarşik kontrol yöntemleri kullanılan seri üretim sistemleri yapısı gereği ürün çeşitliliğine ve kişiselleştirmeye uygun değildir. Ancak akıllı ve dağıtılmış kontrol tekniklerinin rahatça uygulanabileceği geleneksel ve eklemeli üretim teknikleri ile daha esnek üretim sistemleri kurulabilir. Yapılan örnek bir çalışmada seri üretim hücresi ve 3D yazıcılar akıllı kontrol yöntemleri ile bütünleştirilerek edilerek bu sağlanmıştır (Srinivasan et al., 2018).

2.2 Üstünlükleri

Eklemeli imalat yöntemi ile geleneksel üretim yöntemleri için zor ve maliyetli olan karmaşık geometriler ucuz ve hızlı şekilde üretilmektedir. Bu sayede tasarımcıların fikirleri neredeyse tamamen ürün olarak aktarılmaktadır. 3D yazıların yaygınlaşması ve kullanım kolaylığı sayesinde 3D yazıcı ürünleri kişiselleştirilebilir ve istenilen değişiklikler kolay şekilde yapılabilir.

Günümüzde birçok firma ürünlerini CAD ya da CAM dosyası olarak satıp son tüketiciye en yakın 3D yazıcıda ürünü üretmeyi ve teslim etmeyi planlıyor. Bu ve benzeri gelişmeler gelecekteki iş modellerinde değişikliğe yol açacaktır. Söz konusu gelişmeler 3D yazıcıların daha da yaygınlaşması, ucuzlaması ve malzeme teknolojisindeki ilerlemelere bağlıdır. Bugün 3D yazıcılar ile farklı özelliklerdeki plastikler, ahşap, birçok metal, çeşitli gıda ürünleri, çimento, seramik ve bio-materyaller ile neredeyse tüm sektörlere üretim yapılabilmektedir.

Eklemeli imalat yöntemi katmanlar oluşturarak üretmesi nedeniyle bir yönden biyolojik süreçleri taklit etmektedir. Geleneksel talaşlı imalat yöntemleri ile kıyaslandığında bu yöntem yapısı gereği büyük miktarlara çıkabilen atığı, fire miktarını azaltmaktadır. Ayrıca hali hazırda geleneksel imalat yöntemleri kullanılarak üretilen ürünler ve üretim süreçleri katmanlı imalat için yeniden tasarlanmasıyla hem üretim hem de ürün kullanımında iyileşmeler olacak ve kaynaklar çok daha verimli kullanılacaktır. Eklemeli üretim yöntemi yenileme, yeniden üretim ve üretici müşteri yakınlaşması sebebiyle ürün ömrü uzayacaktır. Daha kısa ve basit tedarik zinciri, bölgesel üretim ve yenilikçi dağıtım modelleri ile bu teknoloji bilindik iş modellerini değiştirecektir (Ford & Despeisse, 2016).

Metal katmanlı üretim teknikleri daha az atık çıkarmasına rağmen metal tozu (metal powder) ve yüksek enerji çubuğu (high-energy beams) kullanılan yöntemler genel

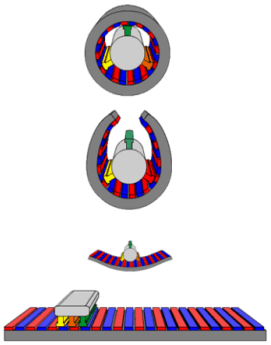
olarak geleneksel üretim yöntemlerine göre birim başına daha fazla elektrik harcamaktadır ve bu durum ürün maliyetine direkt etki etmektedir. Ancak geleneksel yöntemlerde kullanılmak zorunda olan bazı özel aletler, fikstürler ve verilen fire miktarı göz önüne alındığında az miktardaki üretimler için katmanlı imalat daha verimlidir (Rejeski et al., 2018).

3. DOĞRUSAL (LİNEER) MOTOR

3.1 Çalışma İlkesi

Robotik, taşıma sistemleri, takım tezgâhları ve birçok hassas üretim sistemleri yüksek hız ve hassasiyetli doğrusal harekete ihtiyaç duymaktadır. Neredeyse tüm sektör ve uygulamalarda ihtiyaç olan doğrusal hareket, döner motorlardan geleneksel yöntemler (dişli, vidalı mil, kasnak vb. mekanizmalar) ile elde edilmektedir. Bu tür mekanizmalar sistemlere doğrusal hareket hızı ve konumunda ciddi hassasiyet kayıpları oluşturduğu gibi mekanik sistemler olmasından dolayı sistem fazladan yük ve sürtünmeye maruz kalmaktadır. Ayrıca söz konusu mekanizmalar bakım, yağlama gibi ek giderler doğurmaktadır (Shi et al., 2006).

Tüm elektrik motorları hareketli ve hareketsiz parçalardan oluşmaktadır. Hareketli parçalara rotor ya da primer (mover, armature), hareketsiz parçaya ise stator ya da sekonder (seconder, platten) denilir. Doğrusal (Lineer) motorlar basitçe döner motorların kesilip açılması elde edilmiş olarak düşünülebilir (Şekil 3.1).



Şekil 3.1 : Doğrusal (Lineer) motor.

Döner elektrik motorlarda stator ve rotor silindirik yapıda olurken lineer motorlarda farklı şekil ve büyüklükte olabilir. Doğrusal motorlarda ki bu farklılığı sebebi motorların esnek yapıda olması ve ihtiyaca özgün tasarlanıp kullanılabilmesidir.

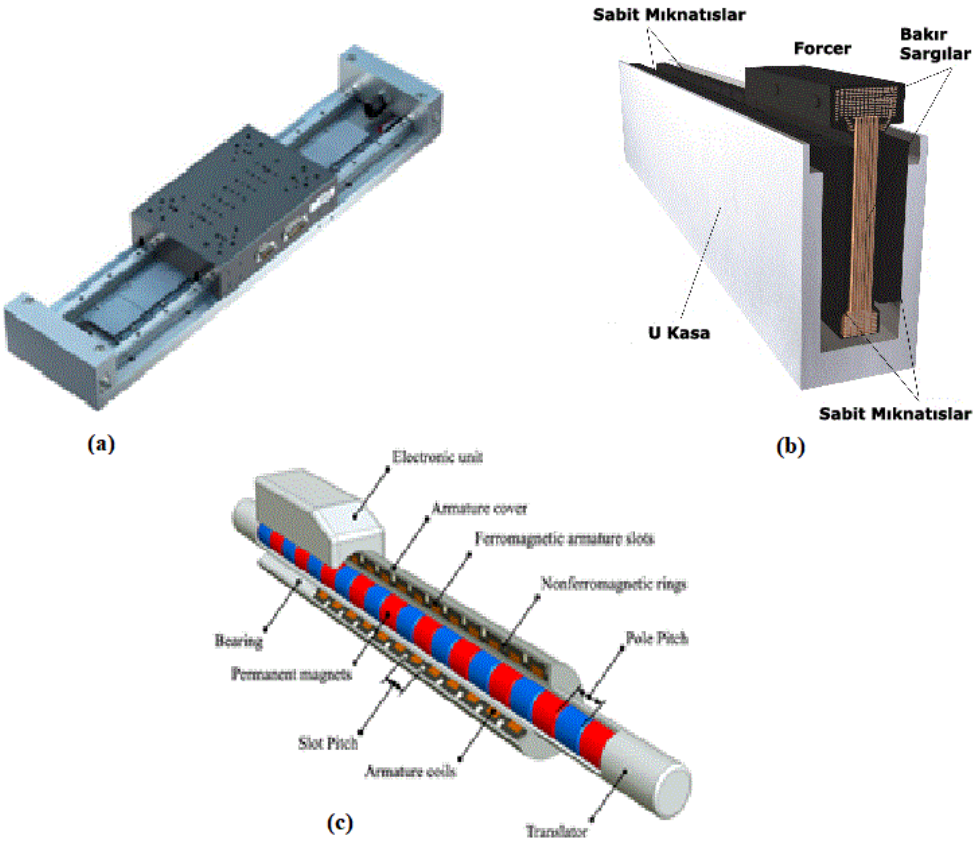
Diğer tüm elektrik motorlarında olduğu gibi doğrusal motorlardaki hareket stator ve rotorlara yerleştirilen bobin veya sabit mıknatısların ürettiği elektromanyetik kuvvetler ile sağlanır. Motorda bulunan enkoderden gelen kesin pozisyon bilgisine göre bobinler enerjilendirilir ve oluşan manyetik kuvvetler ile rotorun hareket ettirilir (Çepni, 2010).

Tüm elektrik motorları gibi lineer elektrik motorları da farklı özelliklerine (Besleme, Fırça durumu, Şekline ve Çekirdek tipine vb.) göre listelenebilir.

Besleme türlerine göre doğrusal elektrik motorlar AC (alternatif akım) ve DC (doğru akım) olmak üzere iki guruba ayrılır. Lineer olmayan yapısı nedeniyle, alternatif akım (AC) ile çalışan motorlar doğru akım (DC) ile çalışan elektrik motorlarına göre daha ileri kontrol teknikleri, karmaşık elektronik kontrol devreleri ve mikroişlemciler gerektirmektedir (Demİrcİ, 2013). DC motorlar genelde kısa mesafeli hareket ihtiyaçları için kullanılır iken AC doğrusal motorlar daha uzun sistemler için kullanılmaktadır. DC doğrusal motorlar kabaca üçe (konveksiyonel, relüktans ve recorder), AC motorlar ise ikiye (indüksiyon ve relüktans) ayrılır (Çepni, 2010).

Fırça durumuna göre döner motorlar gibi fırçalı ve fırçasız olarak ikiye ayrılır. Fırçalı doğrusal motorlarda fırçalar genelde statorda bulunur. Fırçalar hareketi sağlayan kutuplanmayı değiştirmek için motordaki sargıların akım yönlerini değiştirir. Fırçalar hareket sırasında sürekli temas halinde olduğu için bu tip motorlar genelde tercih edilmez. Fırçasız tasarımda temas eden parça yoktur bu nedenle daha uzun ömürlüdür ve bakımı kolaydır ancak kontrolü fırçasız motorlara göre biraz daha karmaşıktır.

Doğrusal motorlar şekillerine göre yassı, U tipi ve tüp olmak üzere genel olarak üçe ayrılır. Ancak doğrusal motorlar yapısı gereği esnek tasarım olanağına sahip olduğu için farklı ihtiyaç ve uygulamalar için farklı şekillerde tasarlanabilir. Yassı motorlar sektörde en çok kullanılan motor tipidir. Birbirine paralel stator ve rotordan oluşmaktadır (Şekil 3.2.a). U (kanal) tipi motorlar ise genelde U şeklindeki statordaki sabit mıknatıslar arasında hareket eden bakır sargılı rotordan oluşmaktadır (Şekil 3.2.b). Tüp şeklindeki doğrusal motorlar silindirik yapıya sahiptir (Şekil 3.2.c). Bazı özel uygulamalar için rotorun enine yönde dönme hareketi de sağlanabilmektedir (Gürdal, 2015).



Şekil 3.2 : Doğrusal (Lineer) motor tipleri.

3.2 Tasarımı

Doğrusal motor tasarımında birçok etken bulunmaktadır. Tez kapsamı doğrusal motor tasarımı olmadığı için sadece çok önemli tasarım parametrelerine değinildi.

Hava aralığı elektrik motorunun hareketli ve sabit parçasının arasındaki mesafeyi tanımlar. Motorlar tasarımında verimi etkileyen önemli parametrelerden biridir. Hava aralığı ne kadar fazla olursa motordaki elektromanyetik (mıknatıslanma) akımı daha büyük olacaktır bu neden ile motorun çıkış gücü ve verim düşecektir (Karaden et al., 2012). Bunun nedeni havanın geçirgenliğinin motor için kullanılan malzemelerin geçirgenliğinden daha az olmasıdır. Motor parçaları (stator ve rotor) üzerindeki manyetik alan çizgileri mecburen iki parça arsındaki hava boşluğundan geçmektedir. Hava boşluğunun neden olduğu yüksek direnç nedeniyle ihtiyaç duyulan akı değeri için daha fazla mıknatıslanma akımı gerekmektedir.

Kaçak akıları azaltmak için yapılan çalışmada bir asenkron motorun dişli şekli ve nüvesi yeniden tasarlanmış. Yeni tasarımda düşürülen hava aralığı sayesinde motorun statorunda üretilen akı %2.041 oranında artış elde edilmiştir. Ayrıca aynı çalışmada rotor ve stator kayıpları toplam %2,46 azaltılmış bu fark ile motorun tork değerinde %4 artış gözlemlenmiştir (Yetgin et al., 2012).

İyilik faktörü genel olarak bir makine ya da mekanizmanın bir enerji türünü farklı bir enerji türüne çevirmesine denilir. Elektrik motorlarında elektrik enerjisinin manyetik enerjiye ve nihayetinde kinetik enerjiye (dönme ya da doğrusal hareket) dönüşümüdür (Karaden et al., 2012).

$$G = \frac{2\mu_0 f \tau^2}{\pi \rho_s g} = \frac{\mu_0}{\pi \rho_s} . v_s . \frac{\tau}{g} \tag{3.1}$$

Denklemde, G iyilik faktörü, τ rotor sargısının kutup adımları, f kaynak frekansı, g hava aralığı uzaklığı, μ_0 boşluğun geçirgenliği ve v_s doğrusal senkron hızı göstermektedir (Gürdal, 2015).

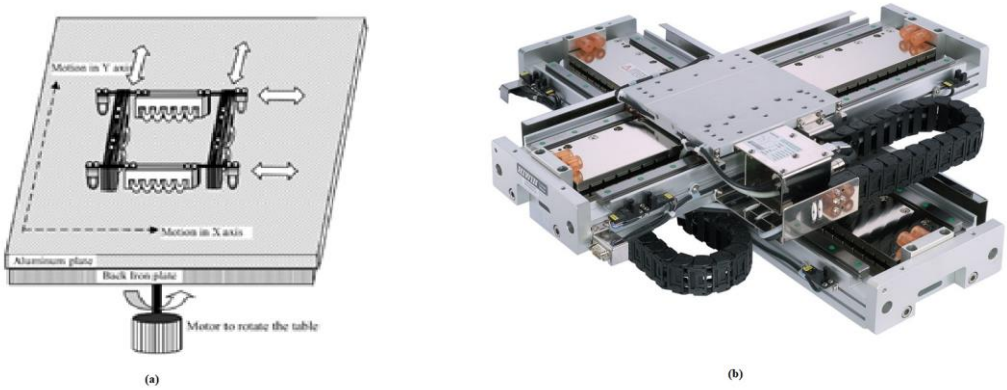
Doğrusal motorlarda t/g oranının daha büyük olduğu durumlarda daha verimli çalıştığı gözlemlenebilir. Bunun nedeni büyük t/g oranına sahip motorlarda sabit parça (primer) kaçak akısı daha düşük seviyelerde olur. Bu sayede sabit parça (primer) ve hareketli parça (sekonder) devreleri arasında yüksek manyetik kuplaj oluşur ve motor performansı artar. Hava aralığı büyüklüğünü gösteren g, elektrik motorları için genellikle mekanik sınırlandırmalar ve mevcut üretim teknolojisi ile belirlenir. Bu neden ile t/g oranı tasarımda daha esnek olunabilecek sabit parça (primer) sargısının kutup adımı olan t ye bağlı olur (Gürdal, 2015).

Kutup sayısı ile doğrusal motorlarda oluşan kayıplardan biri uç etkileridir. Bu kayıp kutup sayısı artırılarak kutuplar arasında paylaştırılıp azaltılabilir (Karaden et al., 2012).

3.3 Kullanım Yerleri

Doğrusal motorların taşıma sistemlerinde kullanılmasının en büyük nedenlerinden biri tekerlek ray ya da yol bağlantısının olmayışıdır. Bu sayede tekerlek yer temasının doğurduğu adezyon kuvveti, sürtünme gibi etkilerden bağımsız hareket elde edilebiliyor. Ayrıca dairesel hareketi doğrusal harekete dönüştürme sorunu olmadığı için herhangi bir çevrime ihtiyaç olmadan diret hareket sağlanabiliyor. Diğer yandan

sistemin normal kuvvetleri kontrol edilebilir (Hellinger & Mnich, 2009). Tez konusu olan 3D yazıcı gibi üç eksende hareket gerektiren sistemler için de doğrusal motorlar çok uygundur. Alexandria üniversitesinde yapılan çalışmada X-Y düzlemlerinde hareket sağlayan doğrusal motorlu sistem ve kontrolü geliştirilmiştir. Bu sistemde tezde bulunan sistemin aksine X ve Y eksenleri için ikişer doğrusal motor bulunmaktadır (Şekil 3.3.a). Geliştirilen sistemde doğrusal motorların hareketli bölümleri olan rotorlar (primer) dört köşeden tekerlekler ile tek bir alüminyum stator (sekonder) üstünde kullanılmıştır. Bu yapı sayesinde X-Y düzlemlerinde yüksek hassasiyet ve hızda hareket elde edilmiştir. Statorun altına konulan bir dairesel motor ile sisteme dönme yetisi kazandırılmıştır (Abd El-Halim et al., 2008). İki farklı düzlemde hareket gerektiren uygulamalar için farklı şekil ve özelliklerde birçok doğrusal motor tasarımı yapılmıştır (Şekil 3.3.b)



Şekil 3.3 : İki eksende çalışan doğrusal motor örnekleri.

Rusya'da yapılan bir çalışmada madenlerden çıkarılan cevheri maden dışına iletmek için kullanılan taşıma sistemlerinde doğrusal motorlu sistemler kullanılmıştır (Sarapulov et al., 2018).

Doğrusal motorların hızı, yüksek gücü ve verimli olması bu sistemlerin savunma sanayiinde de tercih edilmektedir. Uçak gemilerinde kullanılan geleneksel fırlatma sistemlerinden (katapult) daha ucuz, bakımının kolay, basit yapıda olması, gemide daha az yer kaplaması ve daha yüksek güç kapasitesine sahip olmasından dolayı elektromanyetik katapult kullanımı öne çıkmaktadır. Şuan aktif olarak ABD donanmasına ait USS Gerald R. Ford uçak gemisinde elektromanyetik uçak kalkış sistemi kullanılmaktadır. Savunma sanayiinde elektromanyetik fırlatma sistemlerinin kullanıldığı bir diğer alan ise elektromanyetik top sistemleridir. Yeni nesil top sistemleri barut kullanılmadan elektromanyetik kuvvet ile top mühimmatını ses hızının

6 katı ve üstüne çıkarıp hipersonik atış yapabilmektedir. Bu sistemler barutlu sistemlere göre daha ucuza ve hızlı hipersonik atışlar yapabilmesi, lojistik kolaylığı ve muadil sistemlere göre güvenli olması avantajlarıdır (*Urban Savunma*, 2021). Ülkemizde bu konuda Urban Savunma (Şahi 209), Aselsan (Tufan) ve Tübitak (Sapan) çalışmaktadır.

3.4 Üstünlükleri

Doğrusal motorların dairesel hareketli motorlara göre bir takım avantaj ve dezavantajları vardır.

Doğrusal motorlarda dişli, kasnak veya vidalı mil gibi mekanik düzenekler olmadan direkt doğrusal hareket üretilir. Motor ile eyleyici arasında bir mekanizma olmamasından dolayı sistem yüksek hassasiyet ve hız ile kontrol edilebilir. Ayrıca tasarımsal olarak dairesel hareketli motorlara göre daha az limit olduğu için ihtiyaca ve uyulamaya uygun olarak tasarlanıp imal edilebilir (Pooria NOROUZI, 2015). Doğrusal motorlara tasarlanabilecek nispeten basit kontrol sistemleri ile yüksek hızda ve hassasiyette hareket elde edilebilir. Nitekim Shinshu Üniversitesinde yapılan bir çalışmada doğrusal doğru akım motoru ile 200 gr ağırlığındaki yükü 27 ms de 10mm hareket ettiren bir kontrolcü tasarlanmıştır (Yajima et al., 2000).

Doğrusal motorlar gelişen kontrol sistemleri ve azalan maliyetleri nedeniyle kullanımı geçmişe göre artsa da hareket şekli dezavantaj doğura bilmektedir. Dairesel hareketli motorların başlangıcı ve sonu yoktur. Motor çember içinde bir döngüde hareket eder. Doğrusal motorların ise başlangıç ve bitiş uçları vardır (doğrusaldır). Bu uç etkisi motorların performansını olumsuz yönde etkilemektedir. Elektrik motorlar elektro manyetik kuvvetlerin etkisi ile çalışmaktadır. Motorların hareket eden (rotor) ve sabit (stator) parçalarının arasındaki hava boşluğu elektromanyetik enerjiyi tutmasından dolayı motorların verimini olumsuz yönde etkilemektedir. Doğrusal motorlarda söz konusu hava aralığı dairesel hareketli motorlara göre daha fazla olduğundan dolayı bu konuda kayıpları daha fazladır.

Doğrusal motorların sabit (stator) genişliklerinin hareketli (rotor) parçanın genişliğinden büyük olması durumunda kenar etkileri doğacak ve motor verimini olumsuz yönde etkileyecektir (Karaden et al., 2012).

4. NESNELERİN İNTERNETİ

IoT bilgi teknolojileri endüstrisinde yaklaşık 20 yıl önce internetin yaptığı gibi yeni bir çağ açmaktadır. Yapay zekâ, büyük veri, sensör teknolojileri, bilgi güvenliği, otomasyon ve haberleşme teknolojileri gibi neredeyse düşünülebilecek tüm altyapı ve sistemleri ihtiyacına uygun şeklinde birleştirebilmektedir. Haberleşme teknolojisi kablosuz ve kablolu olmak üzere ikiye ayrılır. Ethernet, Profinet ve Bus sistemleri kablolu wi-fi, NFC, RFID ve bluetooth gibi teknolojiler ise kablosuz sistemlere örnek olarak verilebilir. Hangi sistemin nerede kullanacağı tamamen imkân ve ihtiyaçlara bağlıdır. Tüm haberleşme sistemleri mesaj, gönderici, alıcı, iletim ortamı ve protokolden oluşan 5 ana birimden oluşmaktadır. Bu öğelerden birinin olamaması durumda sağlıklı bir iletilişim gerçekleşemez. Dünya genelinde birçok kurum, üniversite ve özel girişim IoT standartlarını belirlemek ve düzenlemek için çalışmaktadır. Bu kurumlardan bazıları Elektrik ve Elektronik Mühendisleri enstitüsü (IEEE), uluslararası telekomünikasyon birliği (ITU) ve çalışmalarını diğer yapılar ile beraber sürdüren OneM2M'dir. IoT cihazları TCP/IP protokolünü sağlayan yazılım ve donanım kullanabilen Zengin Kaynak (Resource-Rich) ve bu protokolü sağlamayan Kısıtlı Kaynak (Resource-Constraint) cihazları olmak üzere kabaca iki ana gruba ayrılır (Singh et al., 2019).

Endüstriyel nesneler (PLC, Scada, sensörler, endüstriyel kameralar, eyleyiciler) arasında yaygınlaşmış makineler arası haberleşme (M2M) kavramı iki cihaz arasındaki önceden programlanmış tek yollu bir iletişim türüdür. Sadece iki cihazı kapsayan bu haberleşme sistemi IoT ile son kullanıcının cihazları (akıllı telefon, akıllı televizyon, tablet vs.) dâhil tüm nesneler ile kullanılabilir hale gelmektedir. Söz konusu bu cihazları birbirine bağlayabilecek birçok iletişim protokolü bulunmaktadır. Bu protokollerden bir ya da birkaçı ihtiyaç ve altyapıya göre kullanılabilinir. Endüstriyel nesnelerin IoT felsefesi ile kullanılması ile Endüstriyel Nesnelerin İnterneti (IIoT) kavramı doğmuştur. IIoT ağı ile tüm endüstrilerin bağlanabilirliği artmış ve sonuç olarak verim artmış, zaman ve maliyet tasarrufu yapılmıştır. Oluşan alt yapı ile müşteri ile direkt iletişim kurulabildiği gibi yöneticilerin daha hızlı ve sağlıklı karar

verebilmesi için üretim ile alakalı tüm bilgileri anlık ve net olarak elde edebilmektedir (Gavlas et al., 2018).

IoT için geliştirilen diğer birçok uygulama katmanı protokolü de endüstriyel nesneler için de kullanılabilir. Bu haberleşme sistemlerinden CoAP (Constraint Application Protocol) UDP üzerinden çalışır. Sistem alıcı ve gönderici arasındaki istek – cevap ilişkisine göre oluşturulmuştur ve daha çok kısıtlı bent genişliğine sahip altyapılarda kullanılır (El Ouadghiri et al., 2020). MQTT (Message Queuing Telemetry Transport) Andy Stanford Clark ve Arlen Nipper tarafında 1999 yılında geliştirilen uygulama katmanı haberleşme protokolüdür. MQTT özellikle Makine-Makine haberleşmeleri için tasarlanmış TCP/IP protokolü temelli mesaj tabanlı haberleşme sistemidir. Düşük bant genişliği ve muhtemel öngörülemeyen gecikmelere sahip ağlar için özel olarak geliştirilmiştir. Söz konusu protokol mesaj iletimi için Yayın – Abone (Publish-Subscribe) ilkesini kullanmaktadır. Yayıncılar (Publisher) ve aboneler (Subscriber) arasındaki iletişim bir yayıncıdan bir aboneye, bir yayıncıdan çoklu aboneye ya da çoklu yayıncıdan çoklu aboneye şeklinde olabilir. İki taraflı iletişime izin veren haberleşme metodu ile bir cihaz hem yayıncı hem de abone olabilmektedir. Bu iletişim yöntemi altyapı ve ihtiyaca uygun olarak tasarlanmadır (Singh et al., 2019). CoAP UDP altyapısı yüzünden paket kayıpları yaşayabilirken TCP/IP protokolü kullanan MQTT sisteminde hem paket kaybı yaşanmamaktadır hem de gecikme daha azdır. Ayrıca Abone – Yayıncı ilişkisi MQTT haberleşmesini IoT uygulamaları için daha uygun kılmaktadır (El Ouadghiri et al., 2020).

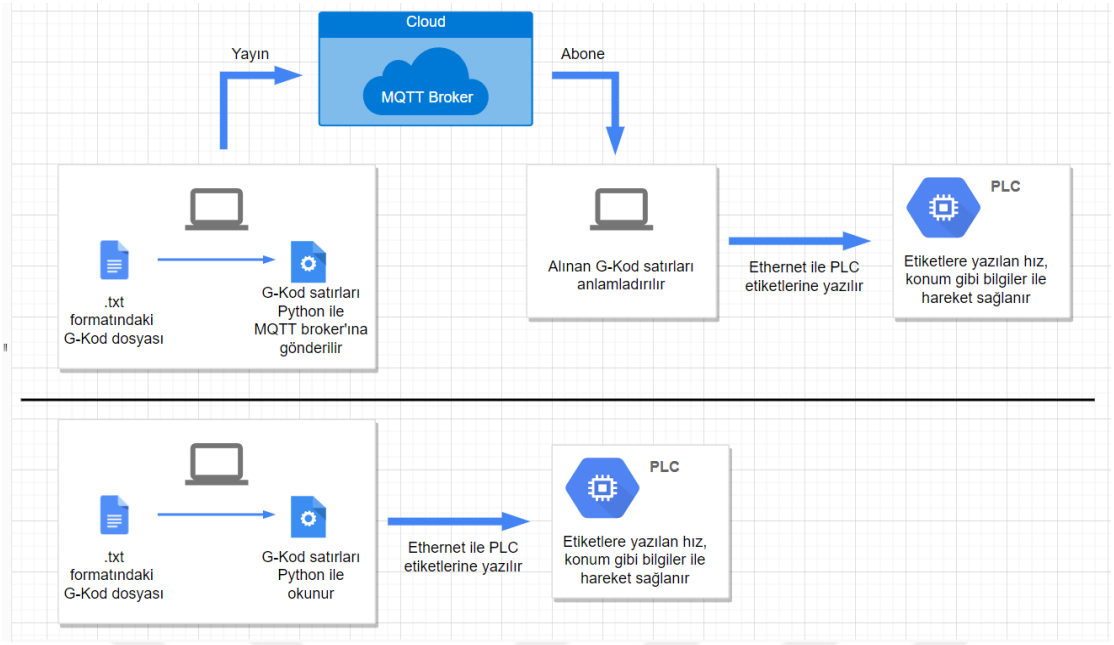
Son dönemlerde yaşanan IoT, bulut tabanlı yüksek kapasiteli bilgisayarlar ve yapay zekâ gibi dijital teknoloji yenilikleri ileri düzey üretim teknikleri (katmanlı imalat) ve yeni yüksek hızlı ve güvenli haberleşme olanakları (5G) ile birleştiğinde ortaya sadece hayal gücünün sınırlandırabileceği bir üretim sistemi çıkarmaktadır. IIoT, endüstriyel nesneler, yazılımlar ve insanlar arasında şuana kadar görülmemiş bir bağlantı sunuyor. IIoT alt yapısının yapay zekâ ve yenilikçi üretim sistemleri ile birleşmesi ile üretim çevikliği, verimi ve kalitesi gibi birçok konuda büyük ilerlemeler olacaktır. Data analizindeki hızlı ilerleme ile büyük veri ve yapay zekâ araçları artık bulut ortamında çok geniş alanlara yayıldı. Bu ilerlemeler ile IIoT teknolojisinin birlikte kullanımı 4. Endüstri devrimini ayni akıllı üretim sistemlerini getirmektedir. Akıllı üretim sistemine sahip akıllı fabrikalar çok yüksek kaliteli ürünleri her müşteriye özel olacak kadar esneklik ve verimlilik ile üretme kapasitesine sahiptir (Lu et al., 2020).

5. SİSTEM

Tezde, Rockwell Otomasyon ürünü Allen-Bradley marka lineer hareket deney seti kullanılmıştır (Şekil 5.1). Daha önceden lazer sensörleri ile yüzey çıkarımı için kullanılan deney seti 3 eksende lineer bağımsız hareket edebilmektedir.

Şekil 5.1 : Tez kapsamında kullanılan deney seti.

Sistem Allen-Bradley marka 1756-L73 ControlLogix PLC ile kontrol edilmekte ve her doğrusal motor için Kinetix 6000 servo motor sürücüsü bulunmaktadır. Kinetix 6000 servo motor sürücüler kontrol organı olan PLC ile Sercos protokolü ve 1756-M08SE SECOS modülü ile haberleşmektedir. Sistemde 3D yazıcının filament beslemesi için bir adet Nema 17 adım (step) motor kullanılmıştır ve bu adım motor PLC tarafından TB6600 adım (step) motor sürücü ile kontrol edilmektedir. 3D yazıcının ısıtıcısı (HotEnd) 12V ısıtıcı fişek ve 100K dirence sahip NTC'den oluşmaktadır. 12V ısıtıcı fişek PLA filamentin eriyebilmesi için gerekli olan 200º C – 210º C sıcaklığa çıkarmaktadır. Sistem sıcaklığı 100K NTC ve PLC ile kontrol altında tutmaktadır. Çalışmada ileriki bölümlerde belirtilecek olan Python ve Studio 5000 yazılımları kullanılmıştır. Python MQTT protokolünü kullanarak haberleşme ve G-Kod dosyalarını okumak, Studio 5000 ise PLC ile doğrusal motorları ve giriş / çıkış birimlerini kontrol etmek için kullanılmıştır. Sistemde iki farklı yöntem kullanılmıştır. Yöntemler ve sistemin genel işleyişi şekil 5.2'de gösterilmiştir.



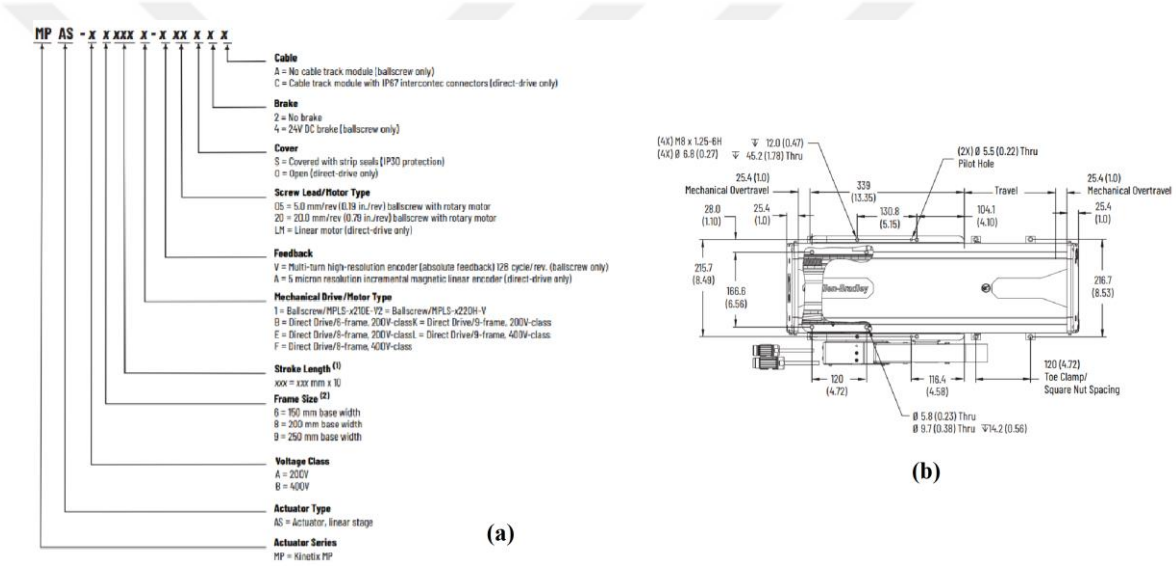
Şekil 5.2 : Sistemin genel akış şeması.

5.1 Mekanik Yapı

Sistemde X ve Y eksenlerindeki hareket için birer adet doğrusal motor bulunmaktadır. Z ekseninde hareket için şekil 5.3'de görüldüğü gibi mekanizma ile sağlanmıştır. Yükselme ve alçalama hareketi Y eksenine paralel duran birbirine eş iki doğrusal motorun zıt yönde hareket etmesi ile sağlanmaktadır. Motorların aynı yöndeki hareketleri ile Y ekseninde hareket edilebilir. Deney setinin Z ekseninde hareketini sağlayan mekanizma alüminyumdan imal edilmiştir. Mekanizmanın ucunda deney seti tasarlanan tutucu (gripper) tez kapsamında kullanılmayacağı için sökülmüştür.

Şekil 5.3 : Z ekseninde hareket sağlayan lineer motorlar ve mekanizma.

X ve Z eksenlerinde hareketi Allen-Bradley marka MP serisi MPAS-A8xxxE-ALMO2x katalog numaralı entegre doğrusal motorlar kullanılmıştır. Z ekseninde aynı stator üzerinde iki adet rotor kullanılmış ve mekanik mekanizma sayesinde doğrusal yükselme ve alçalma hareketi sağlanmıştır. Y ekseni ise MPAS-A8xxxE-ALMS2x katalog numaralı motor ile hareket ettirilmektedir. Doğrusal motorlarda 5 mikron hassasiyete sahip manyetik enkoderler bulunmaktadır. Bu nedenle MPAS serisi doğrusal motorlar yüksek güç ve doğruluk isteyen uygulamalarda kullanılmaktadır. Ayrıca motor içinde mekanik parça olmadığı için (dişli, vidalı mil vs.) uzun ömürlü ve bakım gerektirmemektedir. Şekil 5.4' deki teknik resim ve katalog numarası açıklamalarından da anlaşılacağı gibi motorlar 200mm taban genişliğine ve 5 mikron artırımlı tip manyetik enkodere sahip ve frensizdir (Automation, 2021).



Şekil 5.4 : Doğrusal (Linner) motor katalog bilgisi ve teknik resmi.

Sistemde kullanılan doğrusal motorlar farklı motor sürücüler ile kullanılmış. Ancak X ve Y ekseninde bulunan doğrusal motorlar birbirine bağlı olduğu için tek bir Çoklu Eksen (Multi Axis) motor sürücü ile beraber kullanılabilirdi.

Deney setinin 3D yazıcıya dönüştürülmesi için daha önceden tutucu (Gripper) olan yere filament besleyici (itici) mekanizma (Extruder) (Şekil 5.5.a) ve sıcak uç (Hot End) (Şekil 5.5.b) eklenmiştir.

(a) (b)

Şekil 5.5 : İtici mekanizma (Extruder) ve Isıtıcı uç (HotEnd).

İtici mekanizma Nema 17 adım (step) motor ve filametntin iletimi sağlayan dişli – rulman ikilisinden oluşmaktadır. Adım motoru sayesinde filament şekil 5.5'deki sıcak uca iletilir. Burada 200 ºC de ısıtılan filament baskı için akıtılır. Sistemde baskı yüzeyi olarak 25cm x 25cm ölçülerinde 4mm kalınlığında cam kullanılmıştır. Cam baskı yüzeyi sisteme L köşebentler ile monte edilmiştir.

5.2 Elektronik Yapı

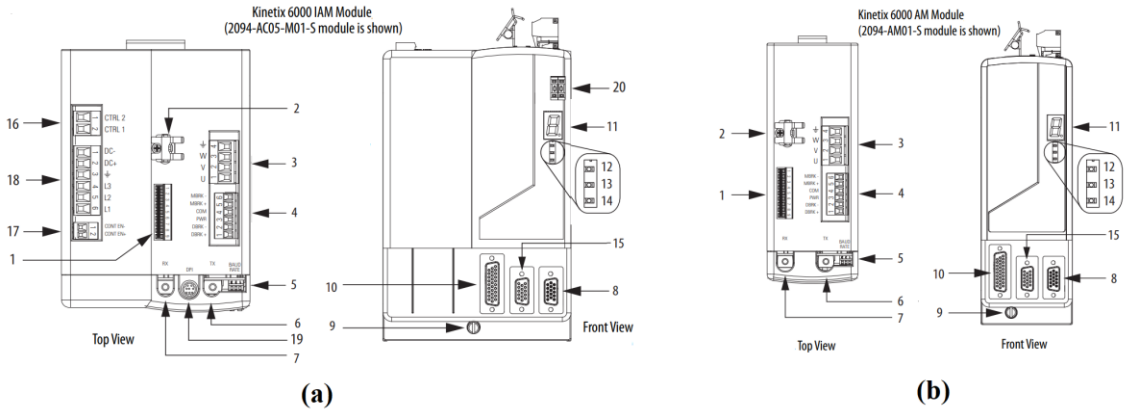
Sistemde 4 bölümlük (slot) 1756-A4 ControlLogix şasisi üzerinde sırasıyla 1756-L73 ControlLogix PLC, 1756-M08SE SERCOS modülü, 1756-EN2TR Enthernet/IP modülü ve 1756-OB16I DC dijital çıkış bulunmaktadır. Ayrıca Ethernet ile 1794-AENT Flex I/O modülü üzerinden 1794-IE8/B Analog giriş ve 1794-OB16D/A 24V DC çıkış ve IRT8 termokupl modülü kullanılmıştır.

1756-A4 ControlLogix şasisi 220V ile beslenmektedir ve kendisine bağlı modüller için 24V besleme sağlamaktadır. Modüller arasındaki iletişim şasi üzerindeki Bus sistemi ile sağlanmaktadır. 1756-L73 PLC kontrolcüsü içerisinde 1 GB büyüklüğünde hafıza kartı ve kapasitör tabanlı enerji depolama birimi vardır. Kontrolcü ile üzerindeki USB girişi ile iletişim sağlanabildiği gibi Ethernet modülü üzerinden de bağlanılabilinir.

PLC şasisindeki ikinci bölümde yer alan 1756-EN2TR Ethernet/IP modülünde 2 adet 100 Mbps hıza sahip ethernet ve 1 adet 12 Mbps hızda USB girişi vardır ve standart TCP/UDP/IP protokollerini desteklemektedir. Sistemde ethernet modülü sayesinde kişisel bilgisayar ve Flex I/O modülleri ile iletişim kurulmuştur.

Flex I/O modüllerinde ısıtıcı ucun (Hot End) sıcaklığını kontrol etmek için kullanılan sıcaklık sensörü için analog giriş ve fan ve ısıtıcı rezistans (Fişek rezistans) için 24 V DC çıkış kullanılmıştır. PLC şasisindeki SERCOS (Serial Real Time Communication System) modülü ile Kinetix 6000 lineer servo motor sürücüleri ile haberleşme kurulmuştur.

Sistemde kullanılan doğrusal motorların kontrolü için yüksek voltaj çıkışlı 2094-AC09-M02-S ve 2094-AM01 katalog numaralı Allen-Bradley Kinetix 6000 motor sürücüler kullanılmıştır. Entegre Servo Sürücüler 200 V ile 230 V arasında sırasıyla 15A ve 9A çıkış verebilmektir (Automation, 2006). Kinetix 6000 doğrusal motor sürücüler birbirleri ve kontrol sistemi olan PLC ile fiber optik kablolar sayesinde SERCOS haberleşme arayüzü ile haberleşmektedir. Haberleşme bağlantısı dışında Kinetix motor sürücülerinin motor, geri besleme diğer bağlantıları yapılmıştır. Şekil 5.6'da gerekli bağlantı noktaları ve çizelge 5.1'de açıklamaları gösterilmiştir.



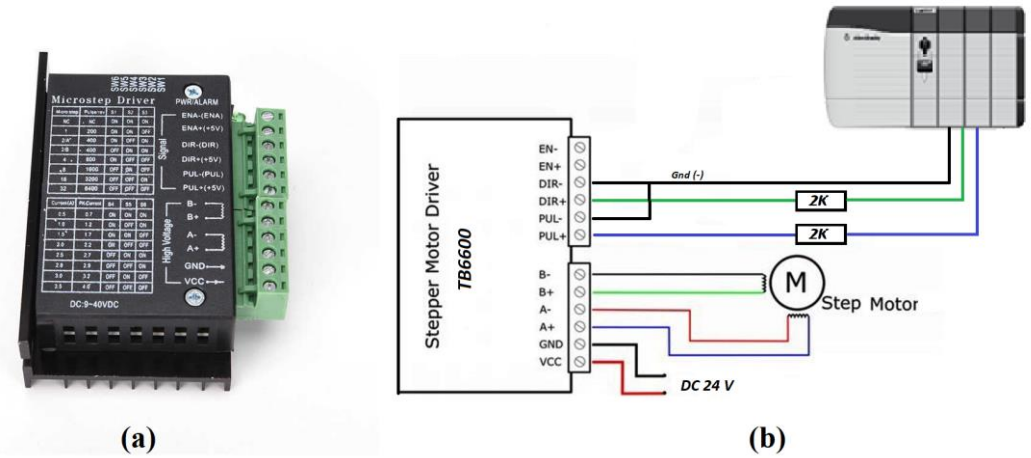
Şekil 5.6 : 2094-AC09-M01-S ve 2094-AM-01 sürücüsü bağlantı noktaları.

Çizelge 5.1 : Kinetix 6000 bağlantıları.

2094-AC-M01-S / 2094-AM01-S			
1	STO bağlantısı	**11**	Seven-Segment
2	Kablo tutucu	**12**	Drive durumu
3	MP bağlantısı	**13**	COMM durumu
4	BC bağlantısı	**14**	Bus durumu
5	Sercos iletişim/güç butonları	**15**	MF bağlantısı
6	Tx bağlantısı	**16**	CPD bağlantısı
7	Rx bağlantısı	**17**	CED bağlantısı
8	AF bağlantısı	**18**	IPD bağlantısı
9	Bağlantı vidası	**19**	DPI bağlantısı
10	I/O bağlantısı	**20**	Sercos adres butonları

Filament itici mekanizmada kullanılan Nema 17 adım motoru için iki fazlı motorlar için kullanılan ve 4 ampere kadar çıkış akımı verebilen TB6600 Stepper motor sürücü kullanılmıştır (Şekil 5.7.a). Adım motor sürücü üzerindeki anahtarlar sayesinde 8 farklı akım çıkışı ve 7 farklı mikro adım ayarlanabilir. Sistemde kullanılan Nema 17 adım motor için 2A ve 1600 adım/tur seçilmiştir. TB6600 sürücü üzerinde motor kontrolü için 3 farklı lojik giriş, step motor çıkışları ve sürücü besleme girişleri bulunmaktadır. 9-42V arasında besleme yapılabilen sürücü sistemde 24V ile beslenmiştir.

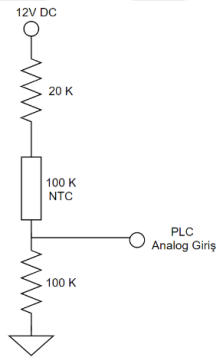
Motor kontrolü için gerekli olan konutların verildiği Dır+/Dır-, Pul+/Pul- ve En+/En- girişleri lojik 5V ile kumanda edilmelidir. Dır (Direction) girişi ile adım motorun yönü, En (Enable) girişi ile motorun kullanıma sunulması ve Pul (Pulse) girişi ile de adım motoru için gerekli olan pulse sinyali verilmektedir. Bu girişlerde bulunan yüksek hızlı optokuplör izolasyonu sayesinde sinyallerde oluşabilecek parazitlenme ve dalgalanmalar engellenmiştir. Sistem PLC ile kontrol edildiğinde dolayı adım motoru sürücüsüne giden kontrol sinyalleri 24V dur. Bu neden ile PLC'den gelen dijital sinyaller seri bağlı 2 KΩ değerindeki dirençler ile kullanılmıştır. TB6600 step motor sürücü bağlantıları şekil 5.7.b'da gösterilmiştir.



Şekil 5.7 : TB6600 Step motor sürücü ve bağlantısı.

Step motor sürücü Pul girişinden aldığı her pulse sinyali ile step motoru bir adım hareket ettirmektedir. Dolayısı ile iki adım arasında geçen süre adım motorun hızını belirlemektedir. Bu nedenle sistemden TB6600 motor sürücüsüne Pul sinyali en az gecikme ile gitmelidir. Sistemde kullanılan dâhili 1756-OB16I 24V DC dijital çıkış modülü düşük gecikmeye sahiptir. 16 adet izole DC çıkışı bulunan modül açık konuma 1ms, kapalı konuma 2ms hızda gelebilmektedir.

Sistemde kullanılan ısıtıcı ucun sıcaklığını ölçmek için 100 KΩ büyüklüğünde NTC termistör kullanılmıştır. NTC termistöründen gelen analog veri 1794-IE8/B Analog giriş modülü ölçülmüştür. Modülde 8 adet bağımsız akım ve voltaj girişi bulunmaktadır. Girişlerde 4/20mA arası akım ve -10/10V, 0-10V arası gerilim ölçülebilmektedir. Cihazda 10V güç kaynağı olmadığı için fan ve ısıtıcı fişek rezistansta kullanılan 12V güç kaynağı NTC sensörü için de kullanılmıştır. En yüksek gerilimi 10V olan analog girişe zarar vermemek için gerilim bölücü devre (Şekil 5.8) tasarlanmıştır. Oda sıcaklığında (25ºC) 100 KΩ direnç değeri bulunan NTC PLA tipi filamentin erime sıcaklığı olan 200ºC de yaklaşık 0,5 Ω değerine kadar düşmektedir. -10/10 V arasında ölçüm yapabilen analog giriş modülü için oluşturulan devre sayesinde analog modülün girişinde 25ºC de yaklaşık 5,4 V, 200ºC de ise yaklaşık 10V gerilim oluşmaktadır. 1794-IE8/B modülünün gerilim girişindeki 200k Ω direnç değeri ile 200ºC sıcaklık PLC'de yaklaşık analog 29000 değerlerinde okunmaktadır.



Şekil 5.8 : NTC / PLC analog giriş gerilim bölücü devre.

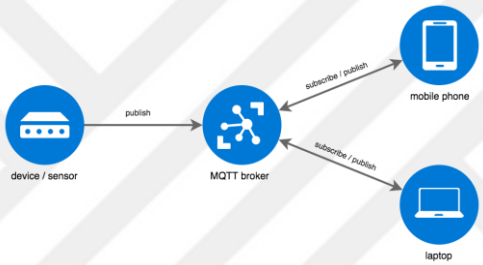
5.3 Haberleşme

5.3.1 Ethernet

Ethernet iletişim protokü 1973 yılında Xerox Parc'da makineleri birbirine bağlamak ve veri transferi yapmak amacıyla icat edildi. Bugün kolay ve anlaşılır yapısı, güvenilirliği ve kapasitesi sayesinde ağ iletişiminin temeli durumundadır (Karel, 2021). Sistemde bulunan 1756-AEN2TR Ethernet/IP modülü ile PLC hem kişisel bilgisayar hem de FlexI/O giriş/çıkış birimleri ile senkronize olarak çalışabilmektedir. Ethernet ağı için Rockwell otomasyonun RSLinx Classic yazılımı kullanılmaktadır.

5.3.2 MQTT

Makine-Makine haberleşme ve IoT uygulamalarında kullanılan MQTT (Message Queuning Telemetry Transport) protokolü mesaj tabanlı iletim altyapısı kullanmaktadır. İstek-Yanıt (Request-Response) prensibine dayanan HTTP protokolünden farklı olarak Yayın-Abone (Publish-Subscriber) ilkeli iletişim modelini benimsemiştir (Şekil 5.9). Yayın-Abone iletişimi TCP/IP altyapısı kullanması nedeniyle Windows, Linux, MacOs gibi çok farklı platformlarda güven ile kullanılmaktadır. MQTT protokolü bir veriyi abonesine iletirken yayıncıyı bekletmez veya bir cihaz hem yayıncı hem de abone olabilmektedir. Ayrıca haberleşme sırasında abone ve yayıncı aynı anda bağlı olmak zorunda değildir. Abone ağa bağlandığı zaman son gönderiyi alabilmektedir.



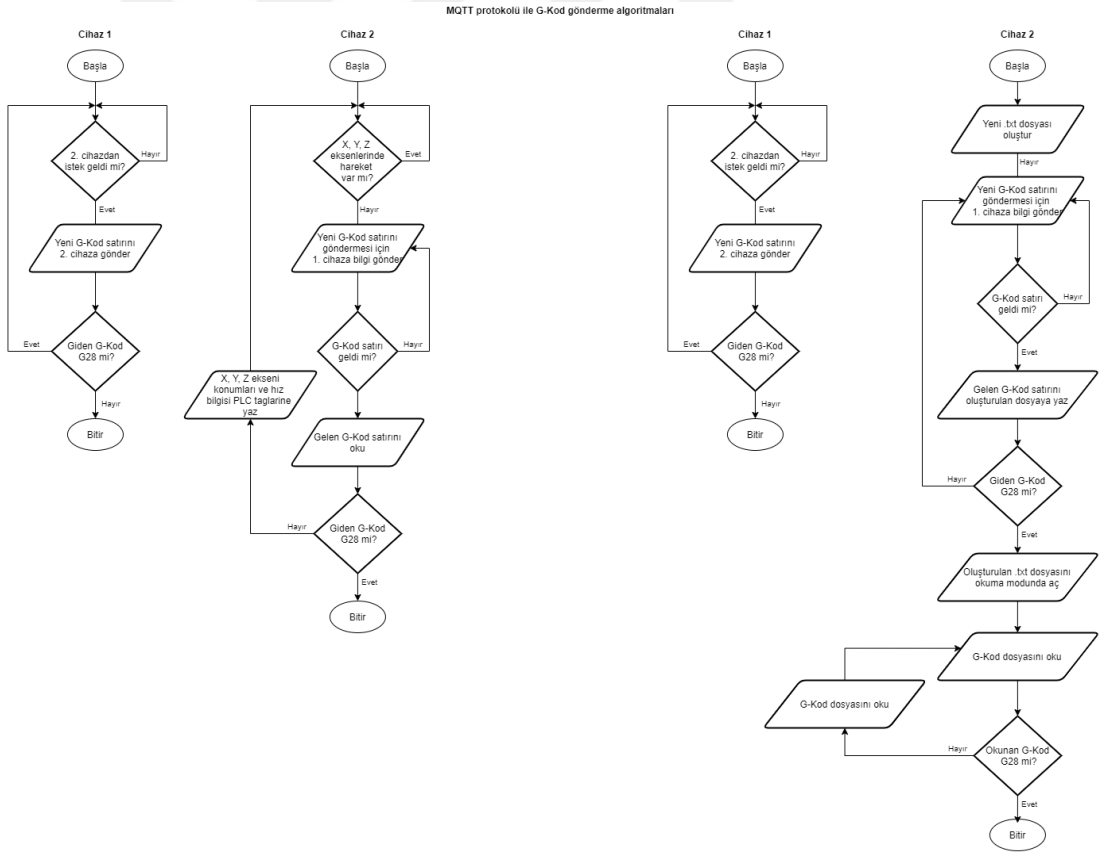
Şekil 5.9 : MQTT haberleşme protokolü yapısı.

Bu özellikleri MQTT haberleşme sistemini asenkron protokol yapmaktadır. Haberleşme sırasında veri sağlayan yayıncı (Publisher) veriyi MQTT sunucularına (Broker / Server) iletir ve sunucu aldığı veriyi abonelere iletir. Söz konusu iletişimde karmaşa olmaması için her yayıncı ve abone bir konuya (Topic) abone olur ve bu konuya veri gönderip alınır. MQTT sunucusu tarafından yönetilen konular (Topic) yayıncılar ile aboneler arasında sanal iletişim kanalı kurmaktadır. Bu sayede yayıncı ve aboneler arasında direkt bir bağlantı yoktur. Aboneler veri almak için bir adrese ihtiyacı yoktur, sunucuda bulunan bir konuya (Topic) abone olması veri alması için yeterlidir (Devnot, 2021; Eczacıbaşı, 2021; MQTT, 2021). Mosquitto, HiveMQ, Azure, CloudMQTT ve Eclipse gibi ücretli ve ücretsiz birçok MQTT sunucusu kullanılabilmektedir.

Tezde MQTT haberleşme protokolü iki farklı yapıda kullanılmıştır. İlk olarak dilimleyici yazılımlarının oluşturduğu G-kodları satır satır iletilmiş ve sistem kontrol edilmeye çalışılmıştır. Her ne kadar MQTT haberleşmesi ms mertebelerinde

gecikmeye sahip olsa da G-kod satırının MQTT abonesi tarafından anlamlandırılması ve motorlara gerekli pozisyon ve hız komutlarının gitmesi zaman almaktadır.

Söz konusu olacak herhangi bir gecikme 3D yazıcılar için fazla ya da eksik malzeme yığılmasına neden olmaktadır. Bu gecikme ve hata nedeni ile parçanın yapısı, ölçüsü ve/veya şekli değişebilmektedir. Bir diğer MQTT uygulaması ise yayıncının tüm G-kod dosyasını aboneye göndermesi, abonenin kendi içinde MQTT ile gelen G-kod satırları ile G-kod dosyasını tekrar oluşturması ve sonrasında yeni dâhili dosyadan G-kodları okuyup motorlara bilgi vermesinden oluşmaktadır. Bu yöntem ile baskı sırasında oluşabilecek ağ ve işlem kaynaklı gecikmelerin önüne geçilmiştir ancak IIoT felsefesine uzak kalmıştır. Her iki yöntemin de sistem akış şeması şekil 5.10'de verilmiştir. Akış şemasında 1. cihaz dilimleyici (slicer) yazılımından çıkan G-kod dosyasını gönderen (Publisher), 2. cihaz ise MQTT ile veriyi alıp anlamlandıran ve PLC'ye ileten cihazdır.



Şekil 5.10 : Satır ve dosya gönderimi yapan akış şeması.

5.3.3 SERCOS

SERCOS (Serial Real Time Communation System) 1991 yılında dijital hareket sürücüleri ile kontrolcüler arasında iletişim sağlamak için geliştirilen bir haberleşme arayüzüdür. SERCOS protokolünde kullanılan fiber optik kablo sayesinde kullanıldığı sistem elektriksel gürültü ve parazitlenmeye karşı dayanıklı, hızlı ve nispeten sade yapısı nedeniyle hareket kontrol uygulamalarına hızlı şekilde kurulup kullanılmaktadır. Diğer haberleşme sistemlerinden farklı olarak tek fiber kablo ile yaklaşık 255 motor sürücüsü kontrol edilebilmektedir. Birçok avantajı nedeniyle farklı marka kontrolcü ve sürücüler SERCOS ile kullanılabilmektedir. Ayrıca üçüncü nesil SERCOS (III) IP tabanlı iletişim ile ethernet protokolünü destekleyecektir (*Botek,* 2021). Tezde kullanılan Kinetix 6000 lineer servo motor sürücüler kontrolör ile 1756-M08SE SERCOS modülü üzerinden haberleşmektedir. SERCOS ağı Rockwell otomasyon ürünü RSLinx Classic yazlımı üzerinden kurulmuştur.

5.4 Yazılım

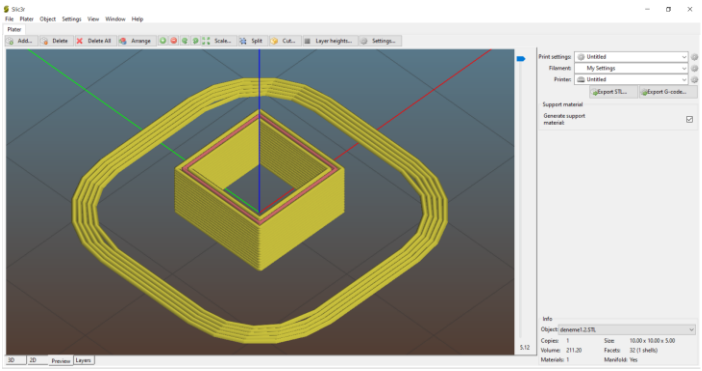
Tez kapsamında kullanılan yazılımlar kullanım sırasında göre açıklanmıştır.

5.4.1 Dilimleyici yazılımı

3D yazıcıda basılmak istenen parça 3D katı modelleme (CAM) yazılımlarında tasarlandıktan sonra katmanlara bölünmesi gerekiyor. Katmanlara ayırma işlemi dilimleyici (Slicer) denilen yazılımlar ile yapılıyor. Bu yazılımlar sayesinde 3D katı model G-Kod dosyasına dönüştürülüyor. G-Kod aslında nümerik kontrol için geleneksel ve modern üretim tekniklerinde (CNC, lazer kesme, su jeti vs.) kullanılan bir programlama dilidir. Günümüzde katmanlı üretim yöntemlerinde de kullanılmaktadır.

Eklemeli üretim yönteminde G-Kodlar basitçe 3D yazıcı baskı ucunun gitmesi gereken konumları ve hızı, ısıtıcı uç (HotEnd) ve yatak sıcaklığı, fan ve 3D yazıcılarda bulunan birçok özelliği barındıran .gcode uzantılı özel kodlarıdır. Piyasada ücretli ve ücretsiz birçok dilimleme yazılımı bulunmaktadır.

Bu yazılımlara örnek olarak Ultimaker Cura, Slic3r, Simplify3D ve ideaMaker örnek olarak verilebilir. Tez kapsamında yapılan baskılarda Slic3r yazılımı tercih edilmiştir (Şekil 5.11). Şekil 5.11’ de test için üretilen bir parçaya ait G-Kod satırlarından bir örnek verilmiştir.



```
M107
M104 S200 ; set temperature
G28 ; home all axes
G1 Z5 F5000 ; lift nozzle

; Filament gcode

M109 S200 ; set temperature and wait for it to be reached
G21 ; set units to millimeters
G90 ; use absolute coordinates
M82 ; use absolute distances for extrusion
G92 E0
G1 Z0.350 F1200.000
G1 E-2.00000 F2400.00000
G92 E0
G1 X5.000 Y-13.124 F1200.000
G1 E2.00000 F2400.00000
G1 F900
G1 X7.510 Y-12.727 E2.15714
G1 X9.775 Y-11.573 E2.31427
G1 X11.572 Y-9.776 E2.47141
G1 X12.726 Y-7.512 E2.62855
```

Şekil 5.11 : Slic3r yazılımı ve G-Kod örneği.

5.4.2 G-Kod okuma ve haberleşme yazılımı

Tez kapsamında, oluşturulan G-Kodları okuma ve haberleşme protokolleri ile PLC kontrolcüsüne iletim Python ile yapılmıştır. Python 1990 yılında geliştirilmeye başlanmış ve 1994 yılında (Python 1.0) yayınlanmış yüksek seviye bir programlama dilidir. Günümüzde Google, CERN, NASA ve YouTube gibi büyük firma ve kurumlar tarafından kullanılmaktadır. Ayrıca oluşturulan kütüphaneleri sayesinde BlockChain, yapay zekâ, derin öğrenme ve makine öğrenmesi uygulamalarında sıklıkla tercih edilmektedir. Python dili basit ve modüler yapısı, girintilere dayalı söz dizilimi ve neredeyse tüm platformlarda verimli çalışması nedeni ile dünya üzerinde kullanılan en yaygın dillerden biridir (Bilginç, 2021). PyCharm Python dili için JetBrains firması tarafından geliştirilmiş ücretsiz entegre geliştirme (IDE), kod düzenleme ortamıdır. Otomatik kod tamamlama, akıllı kod editörü, hata tespiti gibi özellikleri ve tüm işletim sistemleri ile çalışabilmesinden ötürü çok sayıda kullanıcısı vardır (Bilginç PyCharm, 2021). Tez kapsamında dilimleyici (Slicer) yazılımlarından elde edilen G-Kodları PyCharm derleyicisinde Python dili ile okunmuş, anlamlandırılmış ve Studio 5000 yazılımına gönderilmiştir.

Dilimleyici yazılımın ürettiği G-Kod dosyasının Python ile açılması için .gcode olan uzantı .txt olarak değiştirilmelidir. Bu sayede Python ile sıradan bir metin dosyası açar gibi G-Kod dosyası açılabilir. Tezde kullanılmak üzere üç farklı kod sistemi ve iki farklı kod yapısı kullanılmıştır. Kod sistemlerinin farklılığı kullanılan haberleşme altyapılarından kaynaklanmaktadır. Kullanılan MQTT ve Ethernet protokolleri ile G-Kodlarını satır ve dosya olarak Studio 5000'e bağlı cihaza MQTT protokolü ile aktaran iki farklı yapı kurulmuştur (Şekil 5.10). Üçüncü kod sisteminde ise MQTT protokolü kullanılmamış, G-Kod okuyan Python programı ethernet arayüzü ile direkt olarak Studio 5000 yazılımına bağlanmıştır.

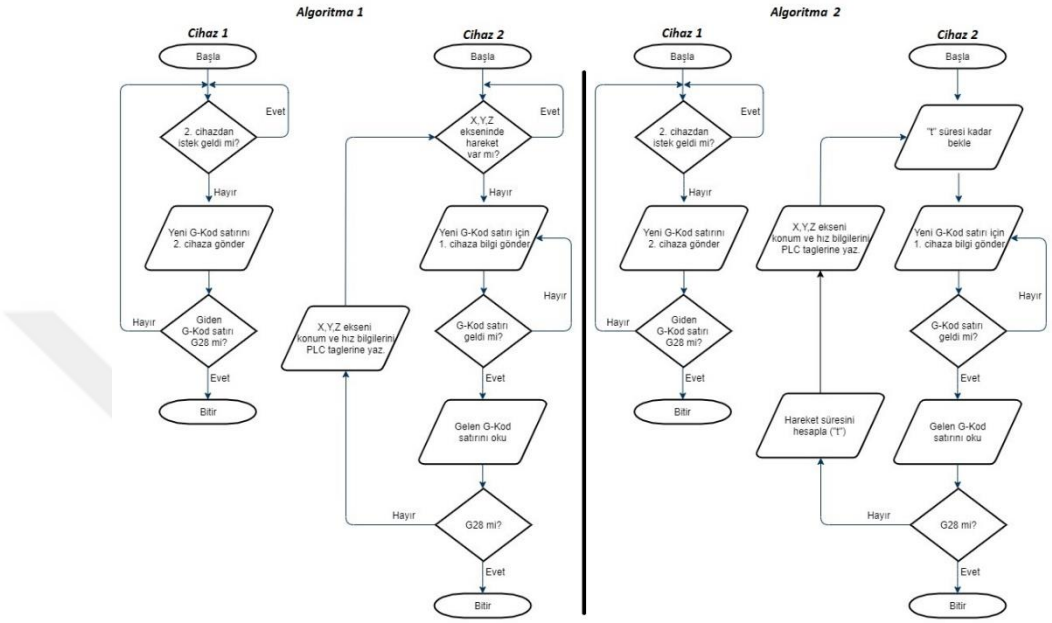
Program oluşturulurken matematik işlemleri için “math”, bazı işlemlerde gecikme sağlamak için “time” ve Studio 5000 yazılımı ile haberleşme sağlamak için “pycom3” kütüphaneleri kullanılmıştır. Pycom3 kütüphanesi ControlLogix kontrolcüleri ile ethernet arayüzü ile IP tabanlı iletişim kurmaktadır. Kütüphane sayesinde iletişim kurmak için PLC kontrolcüsüne bağlı olan Ethernet/IP modülünün IP adresi bilinmelidir. Pycom3 kütüphanesi Studio 5000 kontrolcü etiketlerine (Controller Tags) ‘.write’ komutu ile değer yazılabiliyor ve ‘.read’ komutu ile etiketteki değeri gerçek zamanlı olarak okuyabiliyor. Programda ‘plc’ olarak tanımlanan kontrolcü için komutlar ‘plc.write’ ve ‘plc.read’ olarak kullanılmıştır. Tez kapsamında oluşturulan farklı kod sistemlerinde MQTT ile haberleşme içeren kodlarda daha önce bahsedildiği gibi G-kodlar satır veya dosya olarak PLC ye bağlı olan cihaza aktarılmıştır.

MQTT haberleşmesi için “Paho” kütüphanesi ve HiveMQ (broker.hivemq.com) sunucusu kullanılmıştır. Sunucu üzerinden yayıncı (Publisher) ile haberleşme kurabilmek için gerekli olan başlık (Topic) ‘satir’ ve ‘durum’ olarak belirlenmiştir. MQTT ile mesaj gönderiminden önce şekil 5.10’da görüldüğü üzere G-Kod satırını alacak olan cihaz satırı gönderecek cihaza MQTT üzerinden durum bilgisi göndermektedir. Eğer durum onayı alınırsa G-Kod satırı gönderen cihaz mesaj iletimini yapar ve diğer satır için ‘durum’ başlığını dinler. Söz konusu yapıda iki cihaz hem yayıncı hem de abone olarak kullanılmıştır. MQTT haberleşmesi için gerekli kodlar hariç MQTT kullanılan ve kullanılmayan programlarda algoritma yapısı aşağıda belirtildiği gibidir.

G-Kod okuyan programı başlatıldığında ilk önce ısıtıcı fişek rezistörünü açan anahtarı tetikler ve PLC analog girişine bağlı NTC termistörünün değeri kontrol edilir. Analog değer 200ºC ye karşılık gelen 29100 değerine gelinceye kadar “isitici” fonksiyonu içindeki döngüde beklenir ve değere ulaşılınca G-Kod satırları okunmaya başlanır.

Daha önce bahsedilen 2 farklı algoritma yapısı G-Kod satırlarının okunması ile ilgilidir (Şekil 5.12). İlk yapıda satır okunmadan önce X, Y ve Z eksenindeki motorların hareket durumuna ‘plc.read’ komutu ile bakılmaktadır. Alınan geri birimde lineer motorların hiçbirinde hareket yoksa G-Kod satırı okunur. Bu yöntem PLC’den Python programına gelecek geri bildirime bağlı kapalı çevrim sistemdir. Diğer yöntem ise geri bildirim olmadan oluşturulan açık çevrim ile bir sonraki satırın okunmasını gerçekleştirir. Bu yöntemde ilk değeri ‘0’ olan bir zaman değişkeni vardır. Satır okuma

kodundan önce bulunan bekleme komutu (time.sleep) bu zaman değişkenine bağlıdır. Her satırda yapılacak olan hareketin ne kadar süreceği hesaplanır ve bu zaman değişkenine (“t”) yazılır. Bu sayede program önceki satırdaki hareket bitene kadar beklemede kalır. İkinci yöntem PLC’den geri bildirim olmaması nedeni ile daha hızlı çalışmaktadır.



Şekil 5.12 : Sistemde kullanılan iki farklı algoritma yapısı.

G-Kod dosyasındaki her satır önce G ve M kodlarına göre ayrılır. ‘G’ harfi ile başlayan satır “GCode”, ‘M’ harfi ile başlayan satır “MCode” fonksiyonlarına gönderilir. GCode fonksiyonuna gelen satır hangi G-Kodu olduğu anlamak maksadı ile satırda bulunan boşluklar (space) ile bölünür. Bu işlemden sonra satırın programda kullanılan G0, G1 ve G28 komutlarından hangisi olduğu kontrol edilir. G0 ve G1 komutlarının olduğu satırda konum bilgisi ifade eden X, Y, Z, hareket hızını mm/dk olarak ifade eden F ve akıtılması gereken filament uzunluğunu mm olarak gösteren E harfleri ve değerleri ayıklanır. F değeri mm/dk dan Studio 5000 yazılımında kullanılan mm/sn birimine çevrilir. Mutlak koordinat sistemi ile oluşturulan G-Kodlar ile her bir eksenin hızı ayrı ayrı bir önceki konum ve F hız değeri kullanılarak hesaplanır. X ve Y ekseninde hareket için G-Kodundaki konum bilgisi direkt kullanılırken Z ekseninde hareketi sağlayan mekanizma nedeniyle lineer motorlara verilecek konum bilgisi tekrar hesaplanmaktadır. Hesaplanan konum bilgisi lineer motorlardan birine direkt diğerine ise -1 ile çarpılarak yazılır.

Oluşturulan G-Kodlarda E değeri mutlak değer olarak tanımlanmıştır. Step motor sürücüde kullanılan pulse sinyalinin frekansı E değerinden hesaplanmaktadır. Bir önceki E değeri ile arasındaki değişim, satırdaki lineer hareket süresi, step motor dişli çevre uzunluğu ve 1 tur için adım sayısı ile G-Kodundaki akıtma miktarını verecek pulse sinyal frekansı hesaplanır. Frekans, sinyali üretecek olan Studio 5000 zamanlayıcılarına (Timer) açma / kapama süresi olarak ms olarak yazılır. G28 kodu 3D yazıcının "Home" sıfır konumuna gitme komutudur. G28 komutunda X, Y ve Z eksenindeki lineer motorlar '0' konumlarına 10 mm/sn hız ile gitmeleri sağlanır. 3D yazıcı G-Kod dosyalarında bulunan G98, G91, G92 ve diğer G komutları tez kapsamında gereksiz olduğu için kullanılmamıştır.

G-kod dosyasında bulunan 'M' kodlar ise 3D yazıya ait diğer özellikler ve hareket haricinde kalan ayarlamalar için kullanılmaktadır. Programda tezde inşa edilen 3D yazıcıda kullanılan M84, M106, M107 ve M104 komutları kullanılmıştır. M84 komutu G-Kodların bittiğini ifade eden dosyadaki son koddur. Bu kod okunduğunda sistem baskısı yapılan parçanın kolayca alınabilmesi için parçadan uzaklaşır, lineer servo motorlarını ve okumak için açtığı G-kod dosyasını kapatır ve Python kodunu sonlandırır. M106 fan açma, M107 ise fan kapama komutudur. Bu komutlar direkt olarak fan çıkışını tetikleyen anahtar etiketine bağlıdır. M104 ise ısıtıcı uç (HotEnd) sıcaklık ayarını belirtmektedir. Sistemde sadece PLA tipi filament kullanıldığı için sıcaklık değerini değiştirmeye gerek kalmamıştır.

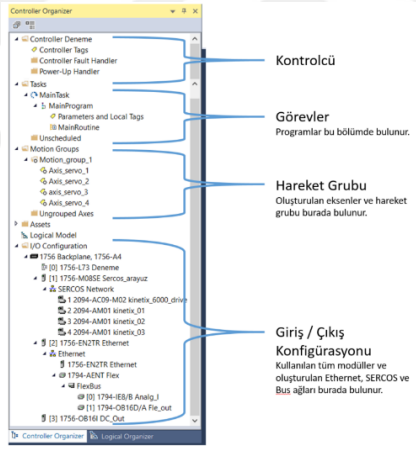
5.4.3 Hareket ve kontrol Yazılımı

Allen-Bradley marka CompactLogix ve tezde kullanılan ControlLogix PLC'leri Rockwell otomasyona ait Studio 5000 yazılım ile programlanmaktadır. Studio 5000 yazılımı SCADA uygulamalarında kullanılan Factory Talk, ağ ve haberleşme uygulaması RSLinx Classic ve RSLogix 5000 gibi yazılımları ihtiva etmektedir. Diğer birçok PLC yazılımda olduğu gibi Studio 5000 merdiven (Ladder LAD), sıralı fonksiyon şeması (Sequental Function Chart SFC) ve fonksiyon blok diyagramı (Function Blok Diagram FBD) programlama tekniklerini kullanmaktadır (Automationtr, 2021). Studio 5000 yazılım sürümü (versiyon) kullanılacak kontrolcünün sürümü ile aynı olmak zorundadır. Üst veya alt sürüm olan cihaz ve yazılım çalışmamaktadır. Eğer Studio 5000 yazılımı kullanılacak kontrolcüden daha yüksek bir sürüme sahipse Rockwell otomasyon ürünü olan ControlFlash yazılımı ile kontrolcünün sürümü yükseltilebilir.

Ancak kontrolcünün sürümü yazılımdan daha yüksek ise sürüm düşürme yapılamaz ve söz konusu yazılım ve kontrolcü birlikte kullanılamaz.

Sistemde kullanılan tüm modüller Studio 5000 yazılımda tanımlanmalıdır. Tanımlama işlemi için modüllerin gerekli özellikleri bilinmeli ve sistemde kullanılacak olan kontrolcü ile aynı revisionda olmasına dikkat edilmelidir. Aynı revisionda olmayan modüller kontrolcü ve birbirleri ile beraber çalışmayacaktır. Modüller arasındaki sürüm değişikliği yazılım ile kontrolcü arasındaki gibi zor değildir. ControlFlash yazılımı bağlı olan modüllerin revision yükseltmeleri yapılabilir. Kullanılan tüm modüllerin kontrolcü ile revisionda olması gerekmektedir. Revision yükseltmesi yapılan modüller önceki revisionları kaybetmez, ihtiyaç halinde revision düşürülebilir. Bu değişiklik modül özellik penceresinden yapılabilmektedir.

Studio 5000 kontrolcü düzenleyicisinde (Controller Organizer) 5 ana başlık (Controler, Tasks, Motion Groups, Assets ve I/O Configuration) bulunmaktadır (Şekil 5.13).

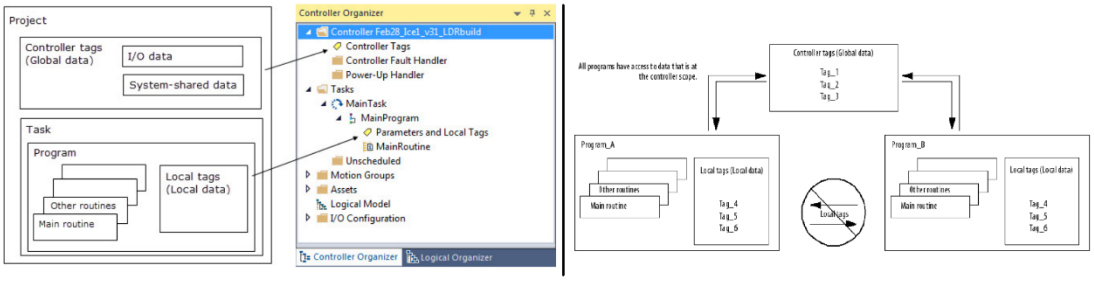


Şekil 5.13 : Studio 5000 Controller Organizer.

Kontrolcü (Controller) bölünde kullanılan kontrolcü ve kontrolcü içinde tanımlı olan etiketler (Tags) bulunmaktadır. Görevler (Tasks) bölümünde yapılan ana ve alt programlar ile bu programlarda kullanılan yerel etiketler (Local Tags) ve parametreler (Parameters) bulunaktadır. Hareket grubu (Motion Groups) bölümde ise hareket eksenleri (Axis) tanımlanarak oluşturulan hareket grubu (Motion Group) bulunulur. Hareket eksenleri oluşturmak için kullanılan motorların özellikleri, taşıyacakları yükleri ve katalog numaraları gibi birçok bilgi girilmelidir. Varlıklar (Assets) başlığında programlarda kullanılan kütüphane, veri tipleri ve diğer değişkenler bulunmaktadır. Giriş / Çıkış konfigürasyonu bölümü altında PLC şasisi içinde ve

ethernet yada SERCOS gibi farklı ağlar ile kontrol edilen modül, sürücü ve diğer cihazlar bulunmaktadır. Sistemde bulunan tüm cihazların kullanımı için burada yapılan tanımlamaların doğru yapılması elzemdir. Tezde ilk olarak 1756-L73 kontrolcü, 1756-M08SE SERCOS, 1756-EN2TR Ethernet/IP ve 1756-OB16I DC çıkış modülleri bu bölüme eklenmiştir. Sonrasında ethernet modülü altında açılan ağ ile kullanılan 1794-AENT Flex I/O giriş çıkış modülü tanımlanmıştır. Ethernet ile bağlanan Flex I/O modülü kendisine bağlı olan diğer birimler ile FlexBus olarak adlandırılan bus sistemi ile haberleşmektedir. Tezde kullanılan 1794-IE8 analog giriş ve 1794-OB16D 24V DC çıkış modülleri bahsedilen bus sistemi ile kontrol edilmektedir. Tezde kullanılan bir diğer ağ yapısı ise doğrusal motor sürücülerin kontrolü için kurulan SERCOS iletişim ağıdır. Bu ağ 1756-M08SE SERCOS modülü altında hareket sağlayan her bir doğrusal servo motoru kontrol etmek için kullanılan Kinetix 6000 motor sürücülerden oluşmaktadır. SERCOS ağına eklenen motor sürücüler hareket grubu (Motion Group) altında oluşturulan eksenler (Axis) ile ilişkilendirilmelidir. Ayrıca diğer modüllerde olduğu gibi kullanılan motor sürücülerinin sürümleri (Revision) aynı olmak zorundadır.

Allen-Bradley PLCleri programlarda kullanılan tüm değişken ve veriler için etiket (Tag) kullanmaktadır ve tüm kontrol yapısı bu etiketler ile sağlanmaktadır. Studio 5000 yazılımında etiketler iki farklı şekilde ele alınmaktadır. Bunlar dışarıdan okuma ve yazmaya izin verilen aynı anda farklı alt programlar tarafından kullanılabilen kontrolcü etiketleri (Controller Tags) ve sadece kendi ana ve alt programlarında kullanılan yerel etiketlerdir (Local Tags). Etiketlerin ana program ve alt programlar tarafından kullanımı şekil 5.14'de detaylı olarak gösterilmiştir. Studio 5000'de çizelge 5.2'da belirtiği gibi temel olarak dört farklı etiket tipi bulunmaktadır.



Şekil 5.14 : Kontrolcü ve yerel etiketlerin kullanımı.

Çizelge 5.2 : Studio 5000 Etiket Tipleri.

Etiket tipi	**İşlevi**
Base	Programlarda kullanılan değerleri saklar
Alias	Başka bir etiketi temsil eder
Produced	Başka bir kontrolcüye gönderilen veri saklanır
Consumed	Başka bir kontrolcüden alınan veri saklanır

Tez kapsamında oluşturulan programda çoğunlukla Base etiket tipi, hareket kontrolü için oluşturulan eksenlerde ise Alias tipi etiketler kullanılmıştır. Kullanılan tüm etiketler Ek C'de verilmiştir. Etiketlerde saklanan veriler de ihtiyaç ve kullanıma göre çeşitlendirilmiştir (Çizelge 5.3).

Çizelge 5.3 : Studio 5000 Veri tipleri.

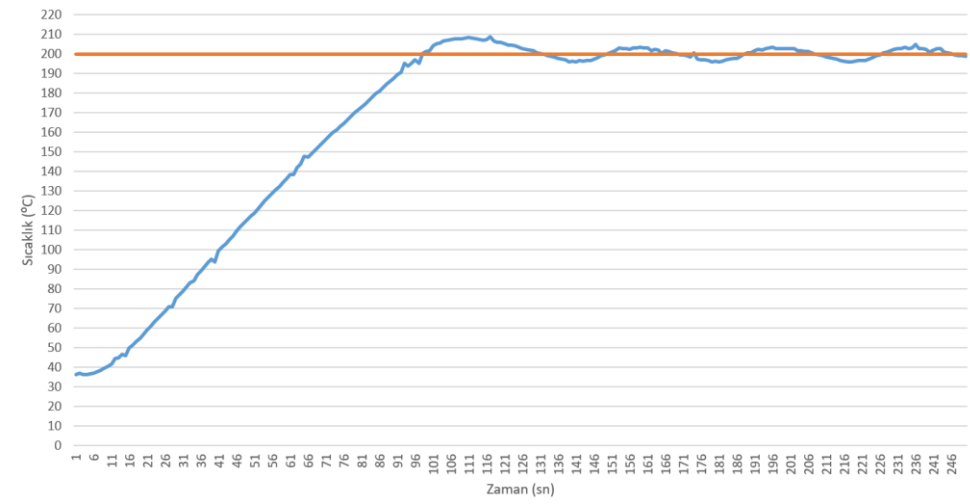
Veri tipi	**İşlevi**
REAL	Reel sayılar (-3.40282347E^{38} / 3.40282347E^{38} arasında)
INT	Tam sayılar (-32,767 / +32,767 arasında çok hızlı yapılar için)
String	ASCII karakterleri
BOOL	Bit (Dijital giriş/çıkış bitleri)
COUNTER	Sayıcı
DINT	Tam sayılar (-2,147,483,648 / +2,147,483,647 arasında)
TIMER	Zamanlayıcı
CONTROL	Sıralayıcı

Tezde sistem merdiven (Ladder) diyagramları ile programlandı. Program içinde giriş / çıkış, hesaplama, hareket ve kontrol blokları kullanıldı. Sistemde kişisel bilgisayarda G-Kodların okunması ile edilen lineer motor konum ve hızları, step motor pulse süresi, soğutucu fan ve ısıtıcı fişek rezistör durumu gibi oluşturulan kontrolcü (Controller Tags) etiketlerine yazılmaktadır. Doğrusal motor hareketi için gerekli olan bilgiler MAM (Motion Axis Move) hareket komutlarına ait etiketlere yazılmıştır. X ve Y eksenleri için direkt yazılan konum değerleri Z ekseninde bulunan iki doğrusal motorda farklı uygulanmıştır. Z eseninde hareket iki doğrusal motorun zıt hareketleri ile sağlandığı için konum bilgisi bir motor için aynen yazılırken diğer motor için -1 ile çarpılarak yazılmıştır. Ayrıca doğrusal servo motorların açılması ve kapanması için MSO (Motion Servo On) ve MSF (Motion Servo Off) komutlarını tetikleyen anahtarlar BOOL tipi veri içeren etiketler ile kontrol edilmiştir.

Benzer bir program satırı (Ladder Rung) lineer motorların “0” (Home) pozisyonlarını belirlemek için kullanıldı. MAH (Motion Axis Home) komutu tetiklendiği zaman doğrusal motorların o anki konumlarını “0” (Home) olarak değiştirir. Programda bu tetikleme Home isimli etiket ve anahtar ile yapılmıştır.

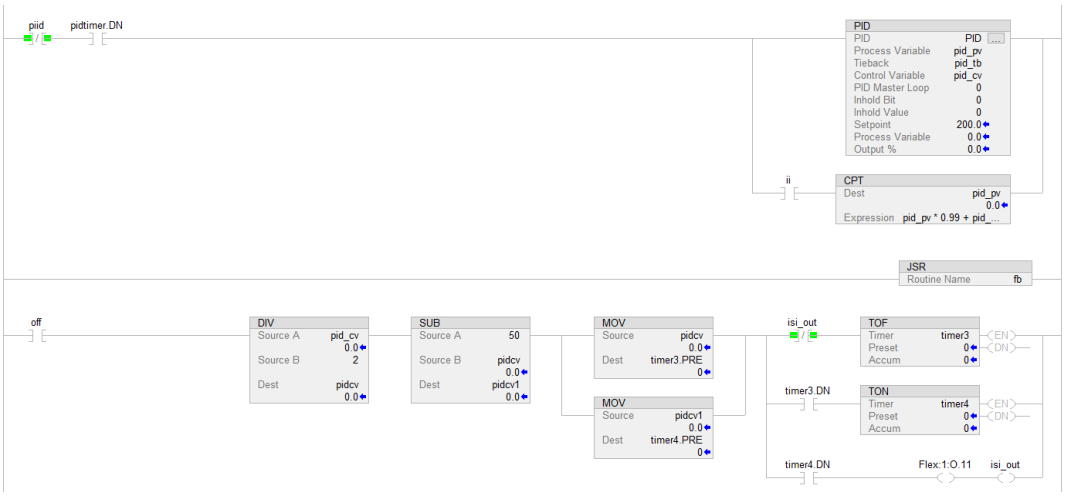
3D yazıcılarda ürün ve baskı kalitesi tüm değişkenlerin yanında en çok sıcaklığa bağlıdır. Bu neden ile sıcaklık kontrolü sistem için çok önemlidir. Tez kapsamında sıcaklık kontrolü iki farklı yöntem ile yapılmıştır. Sıcaklık bilgisi NTC termistöründen alınan analog voltaj değeri ve kullanılan harici termokupldan (Thermocouple) alınan analog sıcaklık değeri ile ölçülmektedir. NTC’nin bağlı olduğu 1794-IE8 analog giriş modülünde yaklaşık 10V’a (≈200ºC) denk gelen analog değer 29100 olarak ölçülmüştür. Termokupl ise Allen-Bradley sistemlerinde sıcaklık ölçümü için hazır kullanılan Flex I/O 1794-IRT8 modülü ile kullanılmıştır. Modül sayesinde sıcaklık değeri direkt olarak ölçülebilmektedir. Bu iki ölçüm aracı ile sıcaklık değeri daha hassas ölçülmektedir.

Birinci yöntemde programda ısıtıcı fişek rezistansını tetikleyen anahtar G-kodları okunmaya başlamadan önce açılmakta ve NTC analog değer 29180 ve termokupl sıcaklık değeri 200ºC olana kadar açık kalmaktadır. Değer geçildiği anda ilgili röle tetiklenmesi bırakılarak ısıtıcı kapatılmaktadır. Isıtıcı uçta bulunan fan ise hem G-Kodlar ile hem de ısıtıcıdan gelen analog değerin 29300’ü veya termokupl sıcaklık değerinin 207ºC geçmesi ile açılmaktadır. Bu yöntem ile yapılan kontrolde sıcaklık şekil 5.15’deki gibi ölçülmüştür.



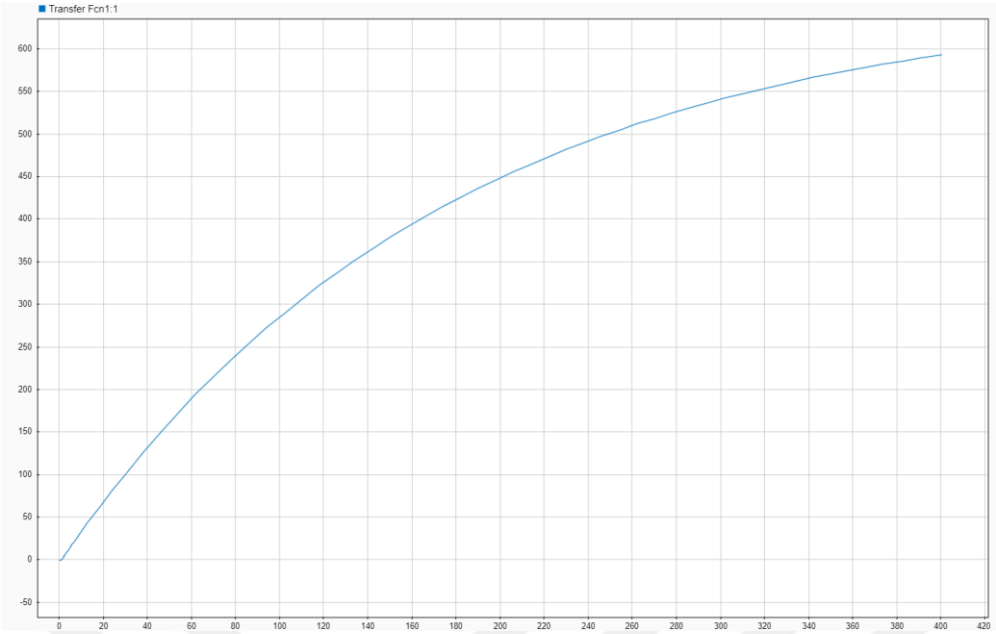
Şekil 5.15 : Aç / Kapa sıcaklık kontrolü.

Diğer sıcaklık kontrol yönteminde PID kontrolcü kullanılmıştır. Studio 5000 yazılımında bulanan PID bloğuna ayarlanan sıcaklık değeri (200ºC) ile termokupldan gelen sıcak değerinin farkı giriş sinyali olarak verilmiştir. Şekil 5.16'de gösterilen PID bloğundan çıkış olarak alınan sinyal ile ısıtıcı rezistans kontrol edilmiştir. Sıcaklık değerinin 207ºC'yi geçmesi durumunda ise soğutucu fan devreye girmektedir. PID çıkış sinyali % olarak alınmakta ve bu değer ile röleyi tetiklemek için görev döngüsü (Duty Cycle) oluşturulmaktadır. PWM sinyaline benzer olarak elde edilen bu sinyalde örnek olarak PID çıkışı %100 ise röle sürekli açık, %50 ise tanımlı olan sürenin yarısında açık diğer yarısında kapalı konumdadır.



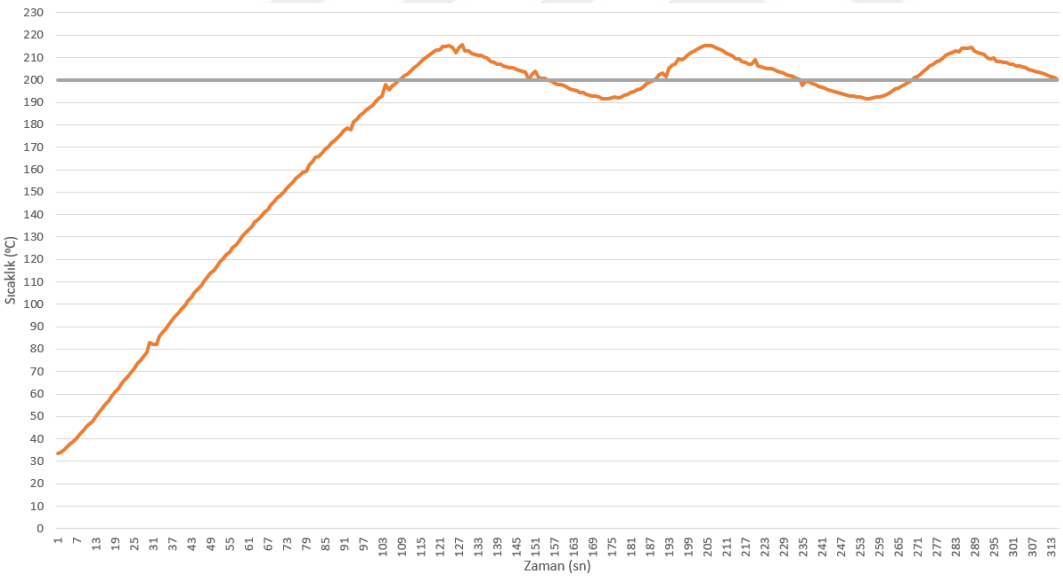
Şekil 5.16 : Studio 5000 PID yapısı.

Sistemin transfer fonksiyonu bilinmediği için PID kontrolcü için önemli olan P,I ve D değerleri denem ve kestirim yolu ile oluşturulmuştur. Isıtıcı sistemin transfer fonksiyonunu elde etmek için sistem birçok kez çalıştırılmış ve sistemin ölçülebilen alandaki davranışı yaklaşık olarak elde edilmiştir (Şekil 5.17). Isıtıcı fişeğin yüksek sıcaklıklarda işlevini kaybetmesi, metal parçaların yüksek sıcaklığa çıkamaması ve ölçüm cihazlarının limitleri nedeni ile ısıtıcı rezistans son değerlerine kadar çalıştırılamamıştır.



Şekil 5.17 : Isıtıcı sistem davranışı.

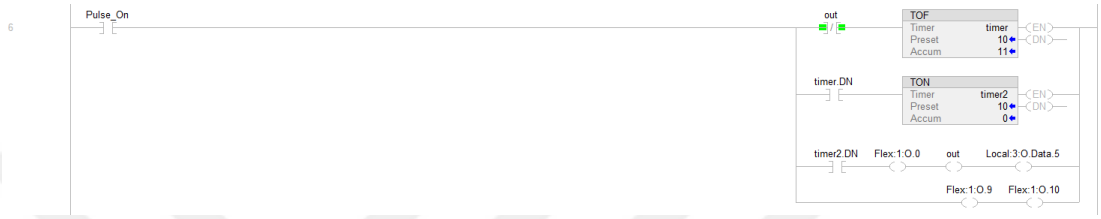
PID ile yapılan sıcaklık kontrolü şekil 5.18'deki gibi sonuç vermektedir. Daha fazla denem ile oluşturulacak P ve I değerleri ile daha iyi kontrol yapısı elde edilebilir.



Şekil 5.18 : PID ile sıcaklık kontrolü**.**

Filamentin itilmesini sağlayan Nema 17 adım motor için gerekli olan pulse sinyalleri Studio 5000 yazılımda sinyal üretici blok olmadığı için zamanlayıcı (Timer) blokları ile oluşturulmuştur. G-Kodlarda belirtilen, çekilmesi gereken filament uzunluğundan Python dilinde oluşturulan yazılım ile step motora verilmesi gereken pulse sinyalinin periyod süresi hesaplanmıştır. Her satır için hesaplanan süre zamanlayıcılara yazılmış ve step motor için gerekli olan pulse (kare dalga) sinyali oluşturulmuştur.

Bunun için iki adet zamanlayıcı blok kullanılmıştır. Bunlardan birisi ayarlanan zaman dolunca çıkışını enerjilendirilen TON diğeri ise ters mantıkta çalışan ve ayarlı süreye ulaşınca çıkış enerjisini kesen TOF bloklarıdır. Şekil 5.19'daki yapıda Pulse_On anahtarı açıldıktan sonra ayarlanan (Preset) değerlerine göre TON ve TOF zamanlayıcıları step motor sürücüsünün Pul girişine bağlı Local:3:OData.5 çıkışında kare dalga üretmektedir. Ancak zamanlayıcılar sinyal üretmek için tasarlanmadığı için en az 1ms ye ayarlanabilmektedir. Bu neden ile step motor yüksek hızlara çıkamamaktadır.

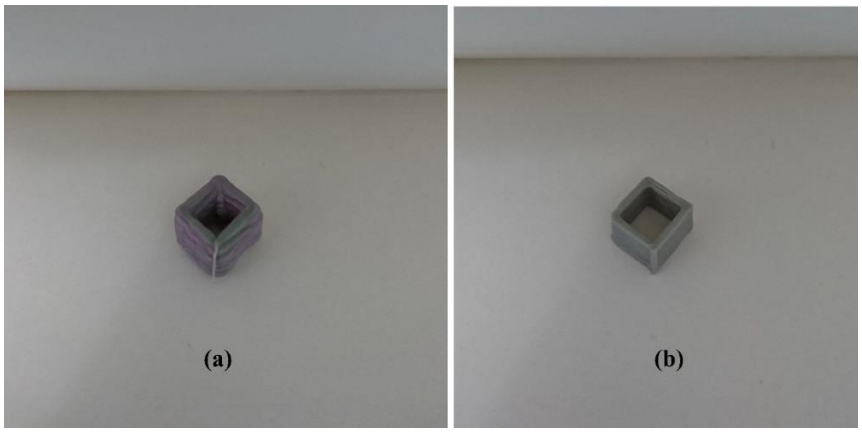


Şekil 5.19 : Zamanlayıcılar ile kare sinyal oluşturma yapısı.

6. SONUÇ VE ÖNERİLER

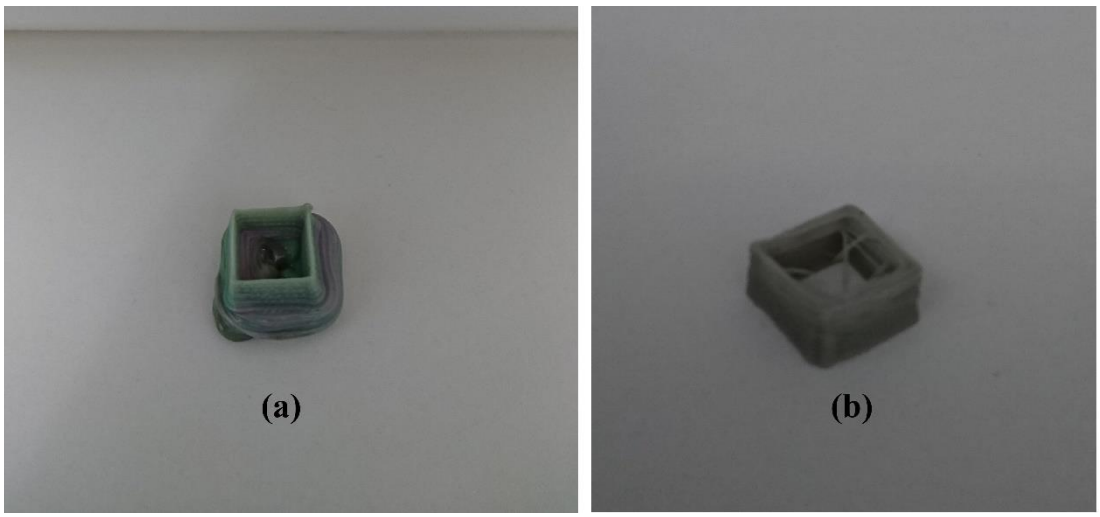
Tez çalışmasında İTÜ Güç ve hareket kontrol laboratuvarında bulunan Rockwell Direct Drive doğrusal motor deney seti 3D yazıcıya başarıyla dönüştürülmüştür. Tez kapsamında iki farklı algoritma yapısı kurulmuştur. Söz konusu algoritmaların birinde endüstriyel kontrolcü Allen-Bradley marka PLC endüstriyel nesnelerin interneti (IIoT) dönüşümü ile MQTT protokolü sayesinde uzaktan kontrol edilmiştir. Diğerinde ise PLC ethernet haberleşme protokolü ile kablolu kontrol edilmiştir. MQTT protokolü ile kontrol edilen sistemde MQTT ve ağ kaynaklı gecikmeler yaşanmıştır. Oluşan gecikmeler 3D baskı sırasında malzeme birikmesine neden olduğu için baskı kalitesini düşürmüştür. Bu neden ile çalışmalara ethernet protokolü ile devam edilmiştir.

3D baskı sırasında eksenlerdeki hareketi ve filament itişini sağlayan motorların konum ve hız bilgileri gibi birçok veri dilimleyici yazılımlar tarafından oluşturulan G-Kodları ile elde edilmektedir. Dilimleyici programların oluşturduğu veriler baskı için kritik öneme sahiptir. Tez sırasında ilk olarak 3D baskı dilimleyici internet servis yazılımı kullanılmıştır. Daha sonra ise açık kaynak olarak sunulan Slic3r yazılımı kullanılmıştır. Şekil 6.1.a'da söz konusu internet tabanlı yazılım kullanılarak, şekil 6.1.b'de ise slic3r yazılımı ile oluşturulan G-Kodların kullanıldığı ürünler görüntülenmektedir.



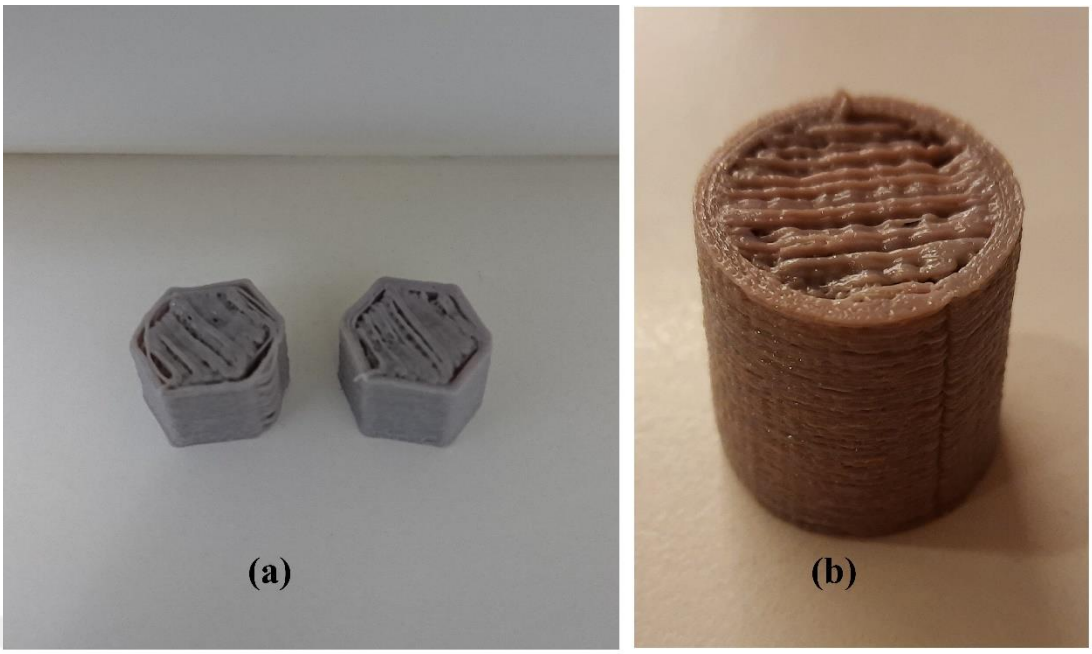
Şekil 6.1 : Web tabanlı ve Slic3r dilimleyici deneme baskıları.

Rockwell otomasyon deney düzeneği ve kullanılan diğer ürünler ile alakalı yeterli doküman olmamasından dolayı birçok parametre deneme yolu ile oluşturulmuştur. Deneme yolu ile oluşturulan parametrelerin başında kabul edilebilir kalitede yüzeye sahip ürünler için önemli olan sıcaklık ve baskı hızı hesaplamalar sonucunda çıkan yaklaşık değerler üzerinde testler yapılarak uygun değer elde edilmiştir. Şekil 6.2'de 1cm x1cmx1cm ölçülerinde 0,4mm kalınlığa sahip parçada baskı sırasında değiştirilen sıcaklık parametresinin etkisi net olarak görülmektedir. Şekil 6.2.b'deki parçada ise şekil 6.2.a'da ayarlanan sıcaklık değeri sabit tutulmuştur. Ancak ısıtıcı ucun uzun süre soğutucu fan etkisinde kalmasından dolayı filament yeteri kadar ısınmamış ve baskı hataları oluşmuştur.



Şekil 6.2 : Deneme baskıları.

Sıcaklık gibi baskı hızı da baskı kalitesini etkileyen değişkenlerden biridir. Tezde kullanılan Slic3r ve diğer birçok dilimleyici yazılımında baskı hızı kontrol edilebiliyor olsa dahi denemeler sırasında yüksek hızlı eksen hareketlerinin baskıyı olumsuz etkilediği gözlemlenmiştir. Şekil 6.3'de tamamen aynı parçaya ait iki farklı baskı görülmektedir. Soldaki üründe iç doldurma baskısı sırasında yüksek hızdan dolayı filament bir miktar savrulmuş ve parçanı dışına taşmıştır. Bu durum üzerine G-Kod anlamlandırma işlemlerinin yapıldığı Python kodunda en yüksek baskı hızı 25 mm/s olarak sınırlandırılmıştır. Hız kısıtlaması sonrasında şekil 6.3'de sağ taraftaki parça basılmıştır. Yapılan denemeler sonucunda bir çok ürün basılmış ve tatmin edici düzeyde yüzey kalitesi başarı elde edilmiştir (Şekil 6.3).



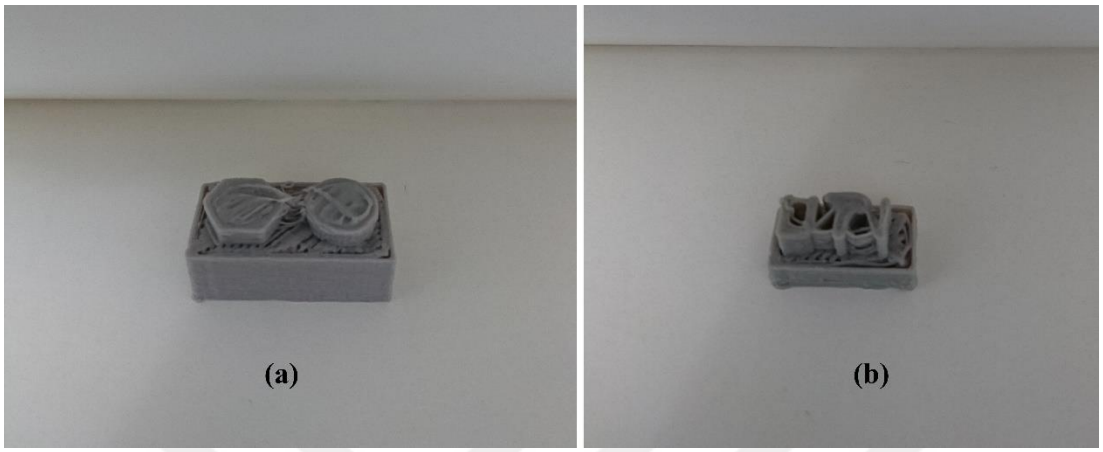
Şekil 6.3 : Baskı hızı limitleme deneme baskısı ve silindir deneme baskısı.

Daha iyi baskı kalitesi elde edilmesi için sistemde farklı dilimleyici yazılımlar denenmelidir. Dilimleyici yazılımlarda genellikle hali hazırda piyasada yaygın olarak 3d yazıcıları ve ayarları bulunmaktadır. Yeni denenecek dilimleyici yazılımda sisteme en yakın yazıcı seçilerek ya da mümkün ise yazılıma yeni yazıcı tanımlayarak oluşturulan G-Kodlar denenmelidir. Piyasada iç doldurma performansı iyi olmadığı bilinen Slic3r yazılımının bu eksikliği parça baskıları sırasında gözlemlenmiştir. Bu durum şekil 6.3'ün üst yüzeylerinde açıkça görülmektedir.

Eklemeli imalat doğası gereği katmanlardan oluşmaktadır. Üretim sırasında devamlı olmasa bile aynı seviyedeki katmanlar birlikte oluşturulmak zorundadır. Bu zorunluluk nedeni ile farklı baskı noktaları arasında hareket edilirken filament itişi olmasa bile malzeme akışı olmaktadır. İstenmeyen malzeme akışı iki baskı noktası arasında olmaması gereken bir bağlantı kurar ve deyim yerinde ise bir köprü oluşturmaktadır. Bu istenmeyen durum günümüz 3D yazıcılarında geri çekme (Retraction) ile engellenmiş durumdadır. Bu özelliğin sisteme eklenmesi ile şekil 6.4'deki gibi istenmeyen baskıların oluşmasını engellenebilir.

Tezde kullanılan Allen-Bradley marka PLC, sürücü ve doğrusal motorlarında sistem enerjilendirildiğinde doğrusal motorların o anki konumu mutlak '0' (Home) pozisyonu olarak ayarlanıyor.

Bu durum 3D baskı için büyük sakıncalar içermektedir. Oluşabilecek kazaları engellemek için sistem açıldığında baskı başlamadan önce doğrusal motorların limit noktalarına koyulacak mekanik anahtarlar (limit switch) ile eksenlerin olması gereken '0' noktaları ayarlanmalıdır.



Şekil 6.4 : Baskı denemeleri.

Baskı yüzeyi olarak kullanılan cam tabla X ve Y eksenlerinde motorlara metal 'L' tipi köşebent ile sabitlenmiştir. Bu rijit yapı, XY düzleminde yapılan dairesel hareketlerde titremektedir ve söz konusu titreşim baskı kalitesini etkilemektedir (Şekil 6.3). Titreşimi önlemek amacı ile sistem ve baskı tablası arasında titreşim emici yapı oluşturulmalıdır.

KAYNAKLAR

Automation, R. (2006). *Kinetix 6000 Multi-axis Servo Drive*. *July*, 1–100.

Automation, R. (2021). *Kinetix Linear Motion Specifications*. 1–174. https://literature.rockwellautomation.com/idc/groups/literature/documents/td/knx-td002_-en-p.pdf

Bazhanov, A., Yudin, D., & Porkhalo, V. (2018). Development of modular control software for construction 3D-printer. *IOP Conference Series: Materials Science and Engineering*, *327*(2). https://doi.org/10.1088/1757-899X/327/2/022011

Çepni, M. E. (2010). *Haziran 2010*. Istanbul Technical University.

Das, A., Chatham, C. A., Fallon, J. J., Zawaski, C. E., Gilmer, E. L., Williams, C. B., & Bortner, M. J. (2020). Current understanding and challenges in high temperature additive manufacturing of engineering thermoplastic polymers. *Additive Manufacturing*, *34*(December 2019), 101218. https://doi.org/10.1016/j.addma.2020.101218

Demirci, R. (2013). *Çift Yanlı Doğru Akım Lineer Motor Tasarımı 2 . Ç ift Yanlı Doğru Akım Lineer Motorunun Tasarımı ve Çalışma Prensibi*. 42–54.

El Ouadghiri, M., Aghoutane, B., & El Farissi, N. (2020). Communication model in the internet of things. *Procedia Computer Science*, *177*, 72–77. https://doi.org/10.1016/j.procs.2020.10.013

Ford, S., & Despeisse, M. (2016). Additive manufacturing and sustainability: an exploratory study of the advantages and challenges. *Journal of Cleaner Production*, *137*, 1573–1587. https://doi.org/10.1016/j.jclepro.2016.04.150

Gavlas, A., Zwierzyna, J., & Koziorek, J. (2018). Possibilities of transfer process data from PLC to Cloud platforms based on IoT. *IFAC-PapersOnLine*, *51*(6), 156–161. https://doi.org/10.1016/j.ifacol.2018.07.146

Gürdal, O. (2015). *Elektrik Makinalarının Tasarımı* (Issue September 2015). Bursa Orhangazi University.

Hellinger, R., & Mnich, P. (2009). *Linear Motor-Powered Transportation : History , Present Status , and Future Outlook*. *97*(11), 1892–1900.

Hull, C. W. (1984). Apparatus for Production of Three-Dmensonal Objects By Stereo Thography. *Patent*, *19*, 16. https://patents.google.com/patent/US4575330

Karaden, C., Prof, M. A. N., & Akpinar, A. S. (2012). *Lineer motor ile levha hareketi*.

Lorenz, K. A., Jones, J. B., Wimpenny, D. I., & Jackson, M. R. (2020). A review of hybrid manufacturing. *Proceedings - 26th Annual International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference, SFF 2015*, 96–108.

Lu, Y., Witherell, P., & Jones, A. (2020). Standard connections for IIoT empowered smart manufacturing. *Manufacturing Letters*, *26*, 17–20. https://doi.org/10.1016/j.mfglet.2020.08.006

Mitchell, A., Lafont, U., Hołyńska, M., & Semprimoschnig, C. (2018). Additive manufacturing — A review of 4D printing and future applications. *Additive Manufacturing*, *24*(October), 606–626. https://doi.org/10.1016/j.addma.2018.10.038

MQTT. (2021). *MQTT*. https://mqtt.org/software/

Okwudire, C. E., Huggi, S., Supe, S., Huang, C., & Zeng, B. (2018). Low-level control of 3d printers from the cloud: A step toward 3d printer control as a service. *Inventions*, *3*(3). https://doi.org/10.3390/inventions3030056

Paolini, A., Kollmannsberger, S., & Rank, E. (2019). Additive manufacturing in construction: A review on processes, applications, and digital planning methods. *Additive Manufacturing*, *30*(September), 100894. https://doi.org/10.1016/j.addma.2019.100894

Pooria NOROUZI. (2015). *HIGH PERFORMANCE POSITION CONTROL OF LINEAR BRUSHLESS DC MOTOR*. *August*, 2015. http://weekly.cnbnews.com/news/article.html?no=124000

Pragana, J. P. M., Sampaio, R. F. V., Bragança, I. M. F., Silva, C. M. A., & Martins, P. A. F. (2021). Hybrid metal additive manufacturing: A state-of-the-art review. *Advances in Industrial and Manufacturing Engineering*, *2*(October 2020), 100032. https://doi.org/10.1016/j.aime.2021.100032

Rejeski, D., Zhao, F., & Huang, Y. (2018). Research needs and recommendations on environmental implications of additive manufacturing. *Additive Manufacturing*, *19*, 21–28. https://doi.org/10.1016/j.addma.2017.10.019

Sarapulov, F. N., Smolyanov, I. A., & Rodionov, I. E. (2018). Investigation of traction linear induction motor for conveyor train. *Proceedings - 2018 17th International Ural Conference on AC Electric Drives, ACED 2018*, *2018-April*, 1–4. https://doi.org/10.1109/ACED.2018.8341685

Shi, J. L., Liu, T. H., & Chang, Y. C. (2006). Adaptive controller design for a sensorless IPMSM position control system. *IECON Proceedings (Industrial Electronics Conference)*, *40*(2), 1326–1331. https://doi.org/10.1109/IECON.2006.347221

Singh, A., Payal, A., & Bharti, S. (2019). A walkthrough of the emerging IoT paradigm: Visualizing inside functionalities, key features, and open issues. In *Journal of Network and Computer Applications* (Vol. 143, pp. 111–151). Academic Press. https://doi.org/10.1016/j.jnca.2019.06.013

Srinivasan, R., Giannikas, V., McFarlane, D., & Thorne, A. (2018). Customising with 3D printing: The role of intelligent control. *Computers in Industry*, *103*, 38–46. https://doi.org/10.1016/j.compind.2018.09.003

Yajima, H., Wakiwaka, H., Minegishi, K., Fujiwara, N., & Tamura, K. (2000). Design of linear DC motor for high-speed positioning. *Sensors and Actuators, A: Physical*, *81*(1), 281–284. https://doi.org/10.1016/S0924-4247(99)00175-2

Yampolskiy, M., King, W. E., Gatlin, J., Belikovetsky, S., Brown, A., Skjellum, A., & Elovici, Y. (2018). Security of additive manufacturing: Attack taxonomy and survey. *Additive Manufacturing*, *21*(February), 431–457.

https://doi.org/10.1016/j.addma.2018.03.015

Yetgin, A. G., Turan, M., & Çanakoğlu, A. G. (2012). A novel slitted tooth core design to decrease leakage flux in induction motor. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *27*(3), 607–614.

Url-1 3D systems. (2021). <https://www.3dsystems.com/our-story> , erişim tarihi 08.04.2021.

Url-2 Artı Boyut. (2021). <https://www.artiboyut.com/index.php/tr/bilgi-bankasi/39-3d-yazici-filament-ozellikleri> , erişim tarihi 06.04.2021.

Url-3 Automationtr. (2021). Automationtr. <http://www.automationtr.com/rockwell-automationdan-studio-5000-application-code-manager-yazilimi.html>, erişim tarihi 08.04.2021.

Url-4 Devnot. (2021). Devnot. <https://devnot.com/2017/mqtt-nedir-nasil-bir-mimaride-calisir/>, erişim tarihi 08.04.2021.

Url-5 Eczacıbaşı. (2021). Eczacıbaşı. <https://www.ebi.com.tr/blog/mqtt-nedir-iot-ile-baglantisi-nedir/>, erişim tarihi 08.04.2021.

Url-6 Botek. (2021). < https://botekotomasyon.com/blog/>

Url-7 Karel. (2021). <Karel. https://www.karel.com.tr/blog/ethernet-nedir-nasil-calisir#:~:text=Metcalfe daha sonra Intel ve,Enstitüsü%2C Ethernet 802.3 standardını onayladı.>, erişim tarihi 08.04.2021.

Url-8 Urban Savunma. (2021).< https://urbansavunma.com.tr/sahi-209/>, erişim tarihi 10.04.2021.

EKLER

Ek A: G-Kod okuma ve Haberleşme Yazılımı (Python)

Ek B: Hareket ve Kontrol yazılımı (Studio500)

Ek C: Hareket ve kontrol Yazılımında Kullanılan Kontrolcü Etiketleri

EK A

Doğrusal motorları kendi “0” noktalarına götüren ve sonrasında doğrusal servo motorları kapatan Python kodu;

```
#Tüm Motorların "0" konumuna götürür
import time

#PLCye bağlanma
from pycomm3 import LogixDriver

with LogixDriver('192.168.1.14', init_program_tags=True,
init_tags=True) as plc:
    print(plc)

    #PLC Taglerine yaza
    plc.write(('Servo_On', True))
    time.sleep(0.1)
    plc.write(('X_Move', 0), ('Y_Move', 0), ('X_hiz', 10), ('Y_hiz',
10), ('Z_Move', 0))
    plc.write(('X_MAM', True), ('Y_MAM', True), ('Z_MAM', True))  #
X move içine bnları koy
    time.sleep(0.1)  # X move içine bnları koy
    plc.write(('X_MAM', False), ('Y_MAM', False), ('Z_MAM', False))
# X move içine bnları koy
    time.sleep(12)
    plc.write(('Servo_On', False),('Servo_Off', True))
    time.sleep(0.1)
    plc.write(('Servo_Off', False))
```

Doğrusal motorların bulunduğu konumu kullanıcıya bildiren ve istenmesi halinde bulunduğu konumu “0” olarak ayarlayan Python kodu;

```
#Motorların bulunduğu konumu Home "0" yapar
import time
from pycomm3 import LogixDriver
#PLC ye bağlanma
with LogixDriver('192.168.1.14', init_program_tags=True,
init_tags=True) as plc:
    print(plc)
    #Dongu
    while True:
        # motor konumlarını alır
        X_Poz = plc.read('Axis_servo_1.ActualPosition')[1]
        Y_Poz = plc.read('Axis_servo_2.ActualPosition')[1]
        Z_Poz = plc.read('axis_servo_3.ActualPosition')[1]
        print("Actual X Axis Position= ", X_Poz, "Actual Y Axis
Position = ", Y_Poz, "Actual Z Axis Position= ", Z_Poz)
        #komuta gre mevcut konumu home yapar
        kom = int(input("Press 1 to Homing\nPress 2 to Exit"))
        if kom == 1:
            plc.write(('Home', True))
            time.sleep(0.1)
            plc.write(('Home', False))
        elif kom == 2:
            exit()
```

```
        else:
            print("## Wrong Input ##")
```

Geri bildirimli ethernet bağlantı yöntemli Python yazılımı;

```
#Geri bildirimli Ethernet kodu
import math
import time

# Global Degiskenler
X1 = 0
Y1 = 0
X2 = 0
Y2 = 0
v = 20
z = 0
z2 = 0
z1 = 0
e1 = 0
e2 = 0
adim = 0

#Başlangıc HotEnd sıcaklık bekleme fonk
def isitici():
    bekle = True
    plc.write(('isi', True))
    while bekle == True:
        analog = plc.read('Flex:0:I.Ch0Data')[1]
        if analog > 29060:
            bekle = False
        time.sleep(1)

#Gkod satırını okuma fonk
def GCode(satir):
    global X1, X2, Y1, Y2, z, z2, z1, v, e1, e2, adim
    b = satir.split(" ")

    for bas in b:

        if bas.startswith("G"):

            deger = int(bas[1:])  # G kodu içindeki değeri int
olarak çektik


            if deger == 0 or deger ==1: #G0 ya da G1
                print("G kodu değeri 0 yada 1")
                i = 1

                while i < int(len(b)): # X,Y,Z,F,E değerleri
okunuyor
                    gez = b[i]
                    i += 1
                    if gez.startswith("F"):
                        fdegeri = float(gez[1:])
                        print("f=", fdegeri)
                        u = fdegeri / 60
                    elif gez.startswith("X"):
                        xdegeri = float(gez[1:])
```

```
                        print("x=", xdegeri)
                        X2 = xdegeri
                    elif gez.startswith("Y"):
                        ydegeri = float(gez[1:])
                        print("y=", ydegeri)
                        Y2 = ydegeri
                    elif gez.startswith("E"):
                        edegeri = float(gez[1:])
                        print("e=", edegeri)
                        # step motor hesaplaması
                        e2 = edegeri
                        e = e2-e1
                        adim = (1600 * e) / (8 * math.pi) # (1600
adım = 10*pi mm ise)
                    elif gez.startswith("Z"):
                        zdegeri = float(gez[1:])
                        print("z=", zdegeri)
                        z2 = zdegeri

                ##### Feedback alma ######
                print("G0-1 ok")
                durx = plc.read('A_1.IP')[1]
                dury = plc.read('A_2.IP')[1]
                durz = plc.read('A_3.IP')[1]
                Pulse_On = bool(durx and dury and durz)
                ##### Hız hesaplaması ######
                Xf = math.fabs(X2 - X1)
                Yf = math.fabs(Y2 - Y1)
                Xff = Xf * Xf
                Yff = Yf * Yf
                V2 = v * v
                if Xf == 0:
                    Vy = v
                    Vx = 5 # herhangi bir deger
                elif Yf == 0:
                    Vx = v
                    Vy = 5 # herhangi bir deger
                else:
                    Vy = math.sqrt((V2 * Yff) / (Xff + Yff))  #
Y_Mam a yazılacak hız degeri
                    Vx = (Xf * Vy) / Yf  # X_Mam a yazılacak hız
degeri
##############################################
                z = z2 - z1
                m = 241.86 - z
                m2 = m * m
                f = math.sqrt(84100 - m2)
                f = f - 160.01168832307218
##############################################
                print(X1, X2, Y1, Y2, z, v)
                print(Vx, Vy)
                if not Yf == 0:
                    t = Yf / Vy
                elif not Xf == 0:
                    t = Xf / Vx
                else:
                    t = 0
                if adim == 0:
                    pulse = 0
                else :
                    pulse = t / adim
```

```
                # PLC yazma

                plc.write(('X_Move', X2), ('Y_Move', Y2), ('X_hiz',
Vx), ('Y_hiz', Vy),('Z_Move', f), ('Pulse', pulse))
                plc.write(('X_MAM', True), ('Y_MAM', True),('Z_MAM',
True), ('Pulse_On', Pulse_On))
                time.sleep(0.01)
                plc.write(('X_MAM', False), ('Y_MAM',
False),('Z_MAM', False))

                e1 = e2
                X1 = X2
                Y1 = Y2
                print(X1, X2, Y1, Y2, z, v)

            elif deger == 90:
                print("G kodu değeri 90")
                # Set absolute Positoining
            elif deger == 91:
                print("G kodu değeri 91")
                # Set Realtive Positioning
            elif deger == 92:
                print("G kodu değeri 92")
                # Set Position
            elif deger == 28:
                print("G kodu değeri 28")
                # Move to Origin
                plc.write(('X_Move', 0), ('Y_Move', 0), ('X_hiz',
10), ('Y_hiz', 10), ('Z_Move', 0))
                plc.write(('X_MAM', True), ('Y_MAM', True),
('Z_MAM', True))  # X move içine bnları koy
                time.sleep(0.1)  # X move içine bnları koy
                plc.write(('X_MAM', False), ('Y_MAM', False),
('Z_MAM', False))  # X move içine bnları koy
                time.sleep(5)
            else:
                print("G kodda sorun var")
        else:
            break

# M kodlarını okuma fonk
def MCode(Msatiri):

    c = Msatiri.split(" ")
    for ilk in c:
        if ilk.startswith("M"):
            deger = int(ilk[1:])  # M kodu içindeki değeri int
olarak çektik

            if deger == 82:
                print("M kodu değeri 82")
                # Use absolute distances for extrusion
            elif deger == 84:
                print("M kodu degeri 84")  # durdurma komutu
                plc.write(('Servo_Off', True))
                time.sleep(0.1)
                plc.write(('Servo_Off', False))
                tecrube.close()
                exit()
            elif deger == 106:
```

```
                print("M kodu değeri 106")
                # Fan On
                plc.write(('Fan', True))
            elif deger == 107:
                print("M kodu değeri 107")
                # Fan Off
                plc.write(('Fan', False))
            elif deger == 204:
                print("M kodu değeri 204")
            elif deger == 205:
                print("M kodu değeri 205")
            elif deger == 140:
                print("M kodu değeri 140")
            elif deger == 186:
                print("M kodu değeri 186")
            elif deger == 104:                  #HotEnd sıcalık
                print("M kodu değeri 104")
                i = 1
                while i < int(len(c)):
                    gez = c[i]
                    i += 1
                    if gez.startswith("S"):
                        sdegeri = int(gez[1:]) # Sıcaklık değeri
                        print("sicaklik =", sdegeri)
                    else:
                        break
            elif deger == 109:                  #yatak sıcaklk
                print("M kodu değeri 109")
            else:
                print("M kodda sorun var")

########################################################################
########################################
########################################   Main Loop
#######################################################
########################################################################
########################################


#PLCye bağlanma
from pycomm3 import LogixDriver

with LogixDriver('192.168.1.14', init_program_tags=True,
init_tags=True) as plc:
    print(plc)

    tecrube = open("bos_kup10-i.gcode", "r")

    ### Servo On ###
    plc.write(('Servo_On', True))
    time.sleep(0.1)
    plc.write(('Servo_On', False))

    ###  Isıtıcı aç ###
    isitici()

    #Dongü
    while 1:

        ##### Feedback alma ######
        durumx = not plc.read('Axis_servo_1.MoveStatus')[1]
```

```
        durumy = not plc.read('Axis_servo_2.MoveStatus')[1]
        durumz = not plc.read('axis_servo_3.MoveStatus')[1]
        durum = durumx and durumy and durumz

        if durum == 1:
            #time.sleep(0.5)
            satir = tecrube.readline()
            print(satir)

            if not satir.startswith(";"):

                if satir.startswith("G"):
                    GCode(satir)

                elif satir.startswith("M"):
                    MCode(satir)

                else:
                    print("satirda sorun var")
```

Geri bildirimsiz ethernet bağlantı yöntemli Python yazılımı;

```
#Geri bildirimsiz Ethernet kodu
import math
import time

# Global Degiskenler
X1 = 0
Y1 = 0
X2 = 0
Y2 = 0
v = 0
z = 0
z2 = 0
z1 = 0
e1 = 0
e2 = 0
adim = 0
t = 0
Pulse_On = False

#Başlangıc HotEnd sıcaklık bekleme fonk
def isitici():
    bekle = True
    plc.write(('isi', True))
    while bekle == True:
        analog = plc.read('Flex:0:I.Ch0Data')[1]
        if analog > 29060:
            bekle = False
        time.sleep(1)

#Gkod satırını okuma fonk
def GCode(satir):
    global X1, X2, Y1, Y2, z, z2, z1, v, e1, e2, adim, t, Pulse_On
    b = satir.split(" ")

    for bas in b:

        if bas.startswith("G"):
```

```
            deger = int(bas[1:])  # G kodu içindeki değeri int
olarak çektik


            if deger == 0 or deger ==1: #G0 ya da G1

                i = 1

                while i < int(len(b)): # X,Y,Z,F,E değerleri
okunuyor
                    gez = b[i]
                    i += 1
                    if gez.startswith("F"):
                        fdegeri = float(gez[1:])
                        print("f=", fdegeri)
                        v = fdegeri / 60
                        if v > 20:  # max hiz 20 mm/sn
                            v = 20
                    elif gez.startswith("X"):
                        xdegeri = float(gez[1:])
                        print("x=", xdegeri)
                        X2 = xdegeri
                    elif gez.startswith("Y"):
                        ydegeri = float(gez[1:])
                        print("y=", ydegeri)
                        Y2 = ydegeri
                    elif gez.startswith("E"):
                        edegeri = float(gez[1:])
                        print("e=", edegeri)
                        # step motor pulse hesaplaması
                        e2 = edegeri
                        e = e2 - e1
                        if e == 0:
                            Pulse_On = False
                        elif e > 0:
                            Pulse_On = True
                        else:
                            print("e negatif")
                            Pulse_On = False
                            e1 = 0
                            e2 = 0
                        adim = (1600 * e) / (8 * math.pi) # (1600
adım = 0.8*pi mm ise)
                    elif gez.startswith("Z"):
                        zdegeri = (float(gez[1:]))
                        print("z=", zdegeri)
                        z2 = zdegeri
                ##### Hız hesaplaması ######
                #print("G0-1 ok")
                Xf = math.fabs(X2 - X1)
                Yf = math.fabs(Y2 - Y1)
                Xff = Xf * Xf
                Yff = Yf * Yf
                V2 = v * v
                if Xf == 0:
                    Vy = v
                    Vx = 5 # herhangi bir deger
                elif Yf == 0:
                    Vx = v
                    Vy = 5 # herhangi bir deger
```

```
                else:
                    Vy = math.sqrt((V2 * Yff) / (Xff + Yff))  #
Y_Mam a yazılacak hız degeri
                    Vx = (Xf * Vy) / Yf  # X_Mam a yazılacak hız
degeri
################## Z hesaplaması ################
                z = z2 - z1
                m = 243.918 - z
                m2 = m * m
                f = math.sqrt(83521 - m2)
                f = f - 155
##########################################
                #print(X1, X2, Y1, Y2, z, v)
                #print(Vx, Vy)
                if not Yf == 0:
                    t = Yf / Vy
                elif not Xf == 0:
                    t = Xf / Vx
                else:
                    t = 0
                if adim <= 0:
                    pulse = 0
                else:
                    pulse = ((t / adim) / 2) * 1000 # ms
                    if pulse < 10:
                        pulse = 10
                    else:
                        pulse = pulse

                # PLC tag yazma
                plc.write(('X_Move', X2), ('Y_Move', Y2), ('X_hiz',
Vx), ('Y_hiz', Vy),('Z_Move', f), ('Pulse', pulse))
                plc.write(('X_MAM', True), ('Y_MAM', True),('Z_MAM',
True), ('Pulse_On', Pulse_On))
                plc.write(('X_MAM', False), ('Y_MAM', False),
('Z_MAM', False))

                e1 = e2
                X1 = X2
                Y1 = Y2
                print(X1, X2, Y1, Y2, z, f)

            elif deger == 90:
                print("G kodu değeri 90")
                # Mutlak koordinant
            elif deger == 91:
                print("G kodu değeri 91")
                # Artırımlı koordinant
            elif deger == 28:
                print("G kodu değeri 28")
                # Home yapma
                plc.write(('X_Move', 0), ('Y_Move', 0), ('X_hiz',
10), ('Y_hiz', 10), ('Z_Move', 0))
                plc.write(('X_MAM', True), ('Y_MAM', True),
('Z_MAM', True))  # X move içine bnları koy
                time.sleep(0.1)
                plc.write(('X_MAM', False), ('Y_MAM', False),
('Z_MAM', False))  # X move içine bnları koy
                time.sleep(10)
            else:
                print("G kodda sorun var")
```

```
        else:
            break
# M kodlarını okuma fonk
def MCode(Msatiri):

    c = Msatiri.split(" ")
    for ilk in c:
        if ilk.startswith("M"):
            deger = int(ilk[1:])  # M kodu içindeki değeri int
olarak çektik

            if deger == 82:
                print("M kodu değeri 82")
                # extrusion için mutlak
            elif deger == 84:
                print("M kodu degeri 84")  # Durdurma komutu
                plc.write(('X_Move', 20), ('Y_Move', 20), ('X_hiz',
10), ('Y_hiz', 10), ('Z_Move', 40))
                plc.write(('X_MAM', True), ('Y_MAM', True),
('Z_MAM', True), ('Pulse_On', False))
                time.sleep(0.1)
                plc.write(('X_MAM', False), ('Y_MAM', False),
('Z_MAM', False))
                time.sleep(10)
                plc.write(('Servo_Off', True))
                time.sleep(0.1)
                plc.write(('Servo_Off', False))
                tecrube.close()
                exit()
            elif deger == 106:
                print("M kodu değeri 106")
                # Fan Aç
                plc.write(('Fan', True))
            elif deger == 107:
                print("M kodu değeri 107")
                # Fan Kapat
                plc.write(('Fan', False))
            elif deger == 104:   #sıcaklık değeri
                print("M kodu değeri 104")
                i = 1
                while i < int(len(c)):
                    gez = c[i]
                    i += 1
                    if gez.startswith("S"):
                        sdegeri = int(gez[1:]) # Sıcaklık değeri
                        print("sicaklik =", sdegeri)
                    else:
                        break
            elif deger == 109:              #yatak sıcaklığı
                print("M kodu değeri 109")
            else:
                print("M kodda sorun var")

##########################################################################
########################################
########################################   Main Loop
#########################################################
#########################################################################
########################################

#PLC ye bağlanma
```

```
from pycomm3 import LogixDriver

with LogixDriver('192.168.1.14', init_program_tags=True,
init_tags=True) as plc:
    print(plc)

    ### .txt dosyası ###
    tecrube = open("sil.gcode", "r")

    ### Servo On ###
    plc.write(('Servo_On', True))
    time.sleep(0.1)
    plc.write(('Servo_On', False))

    ### Isıtıcı ###
    isitici()
    ## Döngü ##
    while 1:

        time.sleep(t)
        satir = tecrube.readline()
        print(satir)
        if not satir.startswith(";"):

            if satir.startswith("G"):
                GCode(satir)
            elif satir.startswith("M"):
                MCode(satir)
            else:
                print("satirda sorun var")
```

MQTT haberleşme protokolü kullanan geri bildirimli algoritmada PLC'ye bağlanan cihazın Python yazılımı:

```
#MQTT Kodları / 2. Cihaz

#MQTT için Paho ktüphanesi tanımlanıyor
import paho.mqtt.client as mqtt
from time import sleep
import math

# Global Degiskenler
X1 = 0
Y1 = 0
X2 = 0
Y2 = 0
v = 0
z = 0
z2 = 0
z1 = 0
e1 = 0
e2 = 0
adim = 0
t = 0
Pulse_On = False

#Başlangıc HotEnd sıcaklık bekleme fonk
```

```
def isitici():
    bekle = True
    plc.write(('isi', True))
    while bekle == True:
        analog = plc.read('Flex:0:I.Ch0Data')[1]
        if analog > 29060:
            bekle = False
        sleep(1)

#Mkod satırını okuma fonk
def MCode(Msatiri):

    c = Msatiri.split(" ")
    for ilk in c:
        if ilk.startswith("M"):
            #print(Msatiri[1:])
            deger = int(ilk[1:])  # M kodu içindeki değeri int
olarak çektik
            if deger == 82:
                print("M kodu değeri 82")
                # extrusion için mutlak kooord
            elif deger == 84:
                print("M kodu degeri 84")  # durdurma komutu
                plc.write(('Servo_Off', True))
                sleep(0.1)
                plc.write(('Servo_Off', False))
                #tecrube.close()
                exit()
            elif deger == 106:
                print("M kodu değeri 106")
                # Fan On
                plc.write(('Fan', True))
            elif deger == 107:
                print("M kodu değeri 107")
                # Fan Off
                plc.write(('Fan', False))
            elif deger == 104:              #set temperature
                print("M kodu değeri 104")
                i = 1
                while i < int(len(c)):
                    gez = c[i]
                    i += 1
                    if gez.startswith("S"):
                        sdegeri = int(gez[1:]) # Sıcaklık değeri
                        print("sicaklik =", sdegeri)
                    else:
                        break
            elif deger == 109:              #set bed temperature
                print("M kodu değeri 109")
            else:
                print("M kodda sorun var")

#Gkod satırını okuma fonk
def GCode(satir):
    global X1, X2, Y1, Y2, z, z2, z1, v, e1, e2, adim, t, Pulse_On
    b = satir.split(" ")

    for bas in b:
        #print("bas=", bas)

        if bas.startswith("G"):
```

```
            #print("g")
            deger = int(bas[1:])  # G kodu içindeki değeri int
olarak çektik
            #print(deger)

            if deger == 0 or deger == 1:
                print("G kodu değeri 0 yada 1")
                i = 1

                while i < int(len(b)):
                    gez = b[i]
                    i += 1
                    if gez.startswith("F"):
                        fdegeri = float(gez[1:])
                        print("f=", fdegeri)
                        v = fdegeri / 60
                        if v > 20:  # max hiz 20 mm/sn
                            v = 20
                    elif gez.startswith("X"):
                        xdegeri = float(gez[1:])
                        print("x=", xdegeri)
                        X2 = xdegeri
                    elif gez.startswith("Y"):
                        ydegeri = float(gez[1:])
                        print("y=", ydegeri)
                        Y2 = ydegeri
                    elif gez.startswith("E"):
                        edegeri = float(gez[1:])
                        print("e=", edegeri)
                        # step motor hesaplaması
                        e2 = edegeri
                        e = e2 - e1
                        if e == 0:
                            Pulse_On = False
                        elif e > 0:
                            Pulse_On = True
                        else:
                            print("e negatif")
                            Pulse_On = False
                            e1 = 0
                            e2 = 0
                        adim = (1600 * e) / (8 * math.pi)  # (1600
adım = 0.8*pi mm ise)
                    elif gez.startswith("Z"):
                        zdegeri = (float(gez[1:]))
                        print("z=", zdegeri)
                        z2 = zdegeri

                # Hız hesaplama
                Xf = math.fabs(X2 - X1)
                Yf = math.fabs(Y2 - Y1)
                Xff = Xf * Xf
                Yff = Yf * Yf
                V2 = v * v
                if Xf == 0:
                    Vy = v
                    Vx = 5  # herhangi bir deger
                elif Yf == 0:
                    Vx = v
                    Vy = 5  # herhangi bir deger
                else:
```

```
                    Vy = math.sqrt((V2 * Yff) / (Xff + Yff))  #
Y_Mam a yazılacak hız degeri
                    Vx = (Xf * Vy) / Yf  # X_Mam a yazılacak hız
degeri
                ################## Z hesaplaması ###############
                z = z2 - z1
                m = 243.918 - z
                m2 = m * m
                f = math.sqrt(83521 - m2)
                f = f - 155
                ##########################################
                # print(X1, X2, Y1, Y2, z, v)
                # print(Vx, Vy)
                if not Yf == 0:
                    t = Yf / Vy
                elif not Xf == 0:
                    t = Xf / Vx
                else:
                    t = 0
                if adim <= 0:
                    pulse = 0
                else:
                    pulse = ((t / adim) / 2) * 1000  # ms
                    if pulse < 10:
                        pulse = 10
                    else:
                        pulse = pulse

                # PLC yazma
                plc.write(('X_Move', X2), ('Y_Move', Y2), ('X_hiz',
Vx), ('Y_hiz', Vy), ('Z_Move', f), ('Pulse', pulse))
                plc.write(('X_MAM', True), ('Y_MAM', True),
('Z_MAM', True), ('Pulse_On', Pulse_On))
                plc.write(('X_MAM', False), ('Y_MAM', False),
('Z_MAM', False))

                e1 = e2
                X1 = X2
                Y1 = Y2
                print(X1, X2, Y1, Y2, z, f)

            elif deger == 90:
                print("G kodu değeri 90")
                # Mutlak koordinant
            elif deger == 91:
                print("G kodu değeri 91")
                # Artırımlı koordinant
            elif deger == 28:
                print("G kodu değeri 28")
                # Home yapma
                plc.write(('X_Move', 0), ('Y_Move', 0), ('X_hiz',
10), ('Y_hiz', 10), ('Z_Move', 0))
                plc.write(('X_MAM', True), ('Y_MAM', True),
('Z_MAM', True))  # X move içine bnları koy
                sleep(0.1)
                plc.write(('X_MAM', False), ('Y_MAM', False),
('Z_MAM', False))  # X move içine bnları koy
                sleep(10)
            else:
                print("G kodda sorun var")
        else:
```

```
            break

###################### Okuma Fonksiyonu ##############

def okuu (gelen):
    gel = gelen.replace("'"," ").replace("b","
").replace(","",".").strip()
    print(gel)
    if not gel.startswith(";"):
        if gel.startswith("G"):
            print("G if")
            gel = gel[:-2]
            GCode(gel)
        elif gel.startswith("M"):
            gel = gel[:-2]
            MCode(gel)
        else:
            print("Satirda sorun var.!")

###################### Mqtt fonksiyonları ##############

def on_message(client, userdata, msg):
    #print( "MQTT Broker'dan Gelen Mesajın Konusu: " + msg.topic + "
" + "MQTT Broker'dan Gelen Mesaj: " + str(msg.payload))
    okuu(str(msg.payload))
    #print(msg.payload)

#mqtt de abone olunan konu
def on_connect(client, userdata, flags, rc):
    if (rc == 0):
        client.subscribe(topic="Satir", qos=2)
        print("\nİstemci Satir konusuna abone olmuştur.")

    else:
        client.loop.stop()

def on_disconnect(client, userdata, flags, rc):
    if rc != 0:
        client.connect(host="broker.hivemq.com", port=1883,
keepalive=60, bind_address="")

###################### Mqtt ile durum gönderme ########

def Gonderme_fo (msj):
    mesaj=msj
    client = mqtt.Client(client_id="", userdata=None,
transport="tcp")
    client.loop_start()
    client.on_disconnect = on_disconnect  # Callback (Geri Çağrı)
fonksiyonu
    client.connect(host="broker.hivemq.com", port=1883,
keepalive=60, bind_address="")
    client.publish(topic="Durum", payload=mesaj, qos=2)
    print("Mesaj iletimi gerçekleştirildi.")
    # client.loop_forever
    sleep(1)

#########################  Main Loop  #################

from pycomm3 import LogixDriver
```

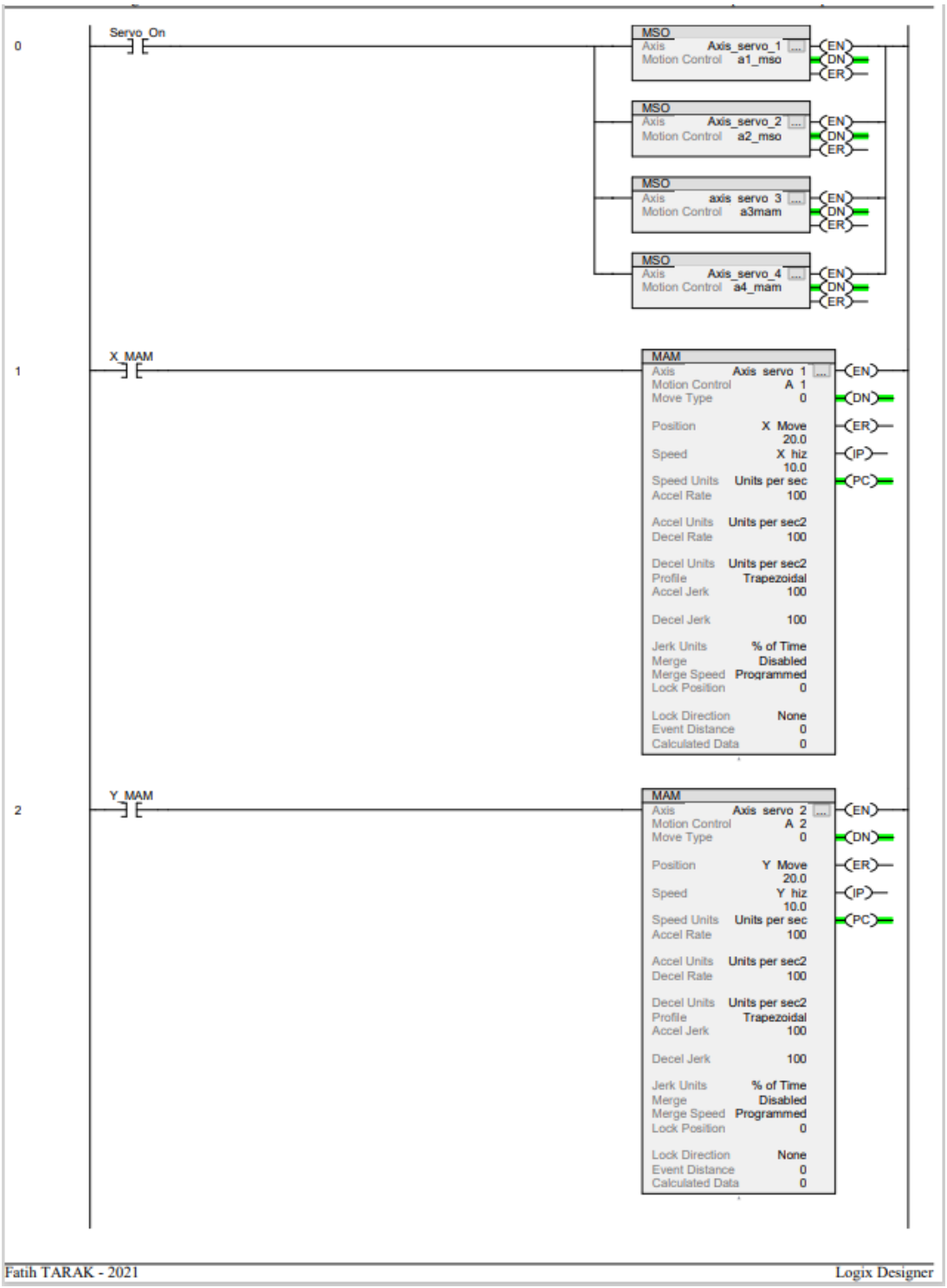
```
with LogixDriver('192.168.1.14', init_program_tags=True,
init_tags=True) as plc:
    print(plc)

    tecrube = open("ilk_deneme.txt", "r")

    ### Servo On ###
    plc.write(('Servo On', True))
    sleep(0.1)
    plc.write(('Servo_On', False))

    ###  Isıtıcı aç ###
    isitici()

    client = mqtt.Client(client_id="", userdata=None,
transport="tcp")
    # client.loop_start()
    client.on_disconnect = on_disconnect
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(host="broker.hivemq.com", port=1883,
keepalive=60, bind_address="")

    #Dongu
    while True:
        client.loop_start()
        durumx = not plc.read('A_1.IP')[1]  # geri gnderilecek
        durumy = not plc.read('A_2.IP')[1]  # geri gnderilecek
        durumz = not plc.read('A_3.IP')[1]  # geri gnderilecek
        durum = durumx and durumy and durumz
        # durum = (input("\ndurum:"))
        sleep(1)
        print(durum)
        Gonderme_fo(durum)
        sleep(1)

        # client.loop_forever()
```

MQTT haberleşme protokolü kullanan geri bildirimli algoritmada kullanıcıya ait cihazın Python yazılımı:

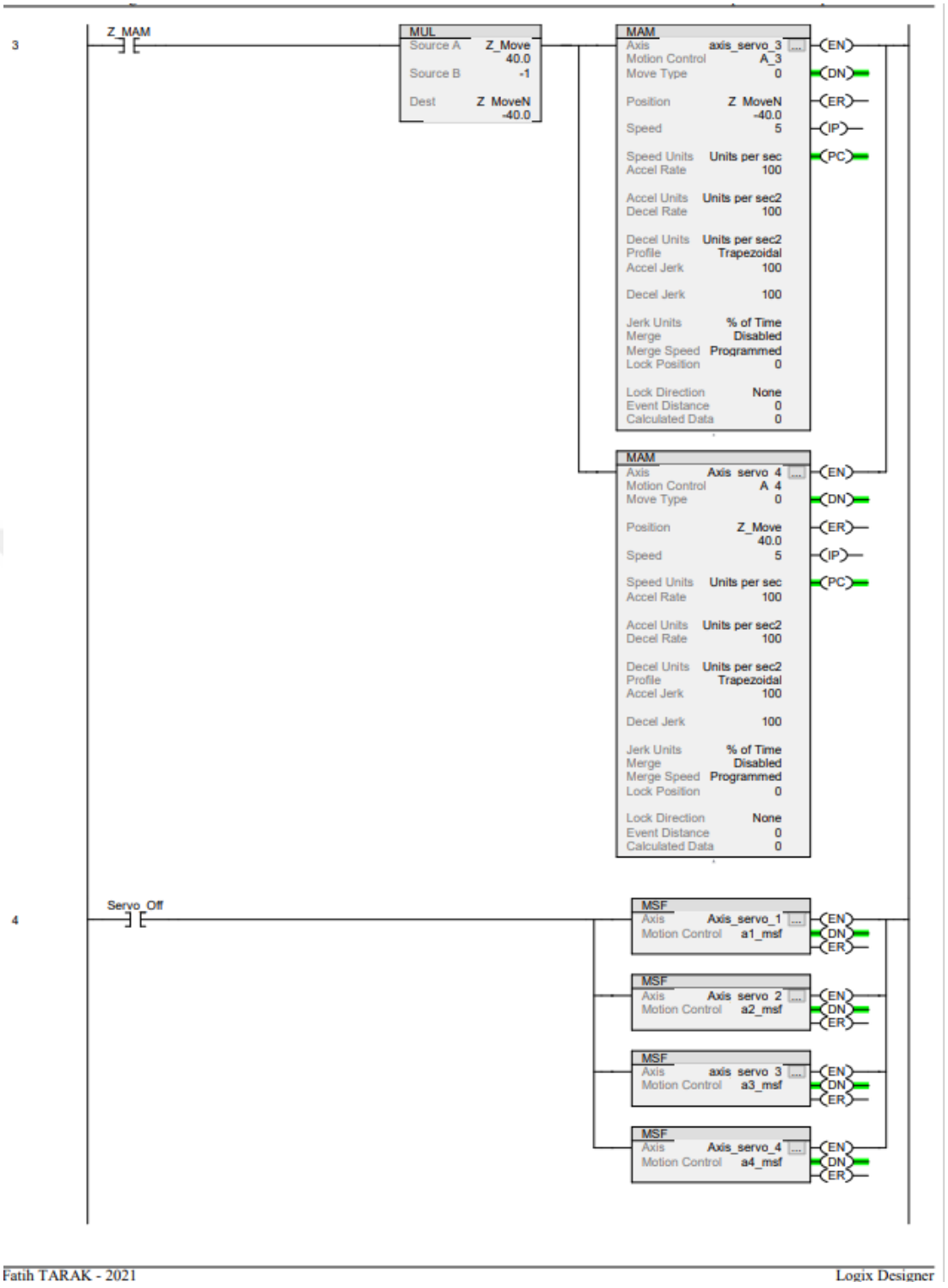
```
#MQTT Kodları / 1. Cihaz

#MQTT icin Paho ktüphanesi tanımlanıyor
import paho.mqtt.client as mqtt
from time import sleep
#Global değişken
durum = False

################ Mqtt Fonksiyonları ######
def on_disconnect(client, userdata, flags, rc):
    if rc != 0:
        client.connect(host="mqtt.eclipse.org", port=1883,
keepalive=60, bind_address="")

#2. cihazdan gelen durum bilgisi
def on_message(client, userdata, msg):
    global durum
```

```
    print(
        "MQTT Broker'dan Gelen Mesajın Konusu: " + msg.topic + " " +
"MQTT Broker'dan Gelen Mesaj: " + str(msg.payload))

    dur = str(msg.payload)
    duru = int(dur.replace("'"," ").replace("b","
").replace(","",".").strip())
    durum = bool(duru)
    print(type(durum), durum)

#mqtt de abone olunan konu
def on_connect(client, userdata, flags, rc):
    if (rc == 0):

        client.subscribe(topic="Durum", qos=2)
        print("İstemci Durum konusuna abone olmuştur.")
        sleep(5)

################# Main Loop #########

tecrube = open("yarim-ilk-mm-normal.gcode.txt", "r") #acılan g kod
dosyası
client = mqtt.Client(client_id="", userdata=None, transport="tcp")

# Callbacks (Geri Çağrılar) fonksiyonları
client.loop_start()
client.on_disconnect = on_disconnect
client.on_connect = on_connect
client.on_message = on_message
#kullanılan mqtt brokerı
client.connect(host="broker.hivemq.com", port=1883, keepalive=60,
bind_address="")

#Dongu
while True:

    print(durum)
    if durum ==True:
        client.loop_start()

        satir = tecrube.readline()
        client = mqtt.Client(client_id="", userdata=None,
transport="tcp")
        client.on_disconnect = on_disconnect  # Callback (Geri
Çağrı) fonksiyonu
        client.connect(host="broker.hivemq.com", port=1883,
keepalive=60, bind_address="")
        client.publish(topic="Satir", payload=satir, qos=2)
        print("Mesaj iletimi gerçekleştirildi.")
        sleep(1)
```

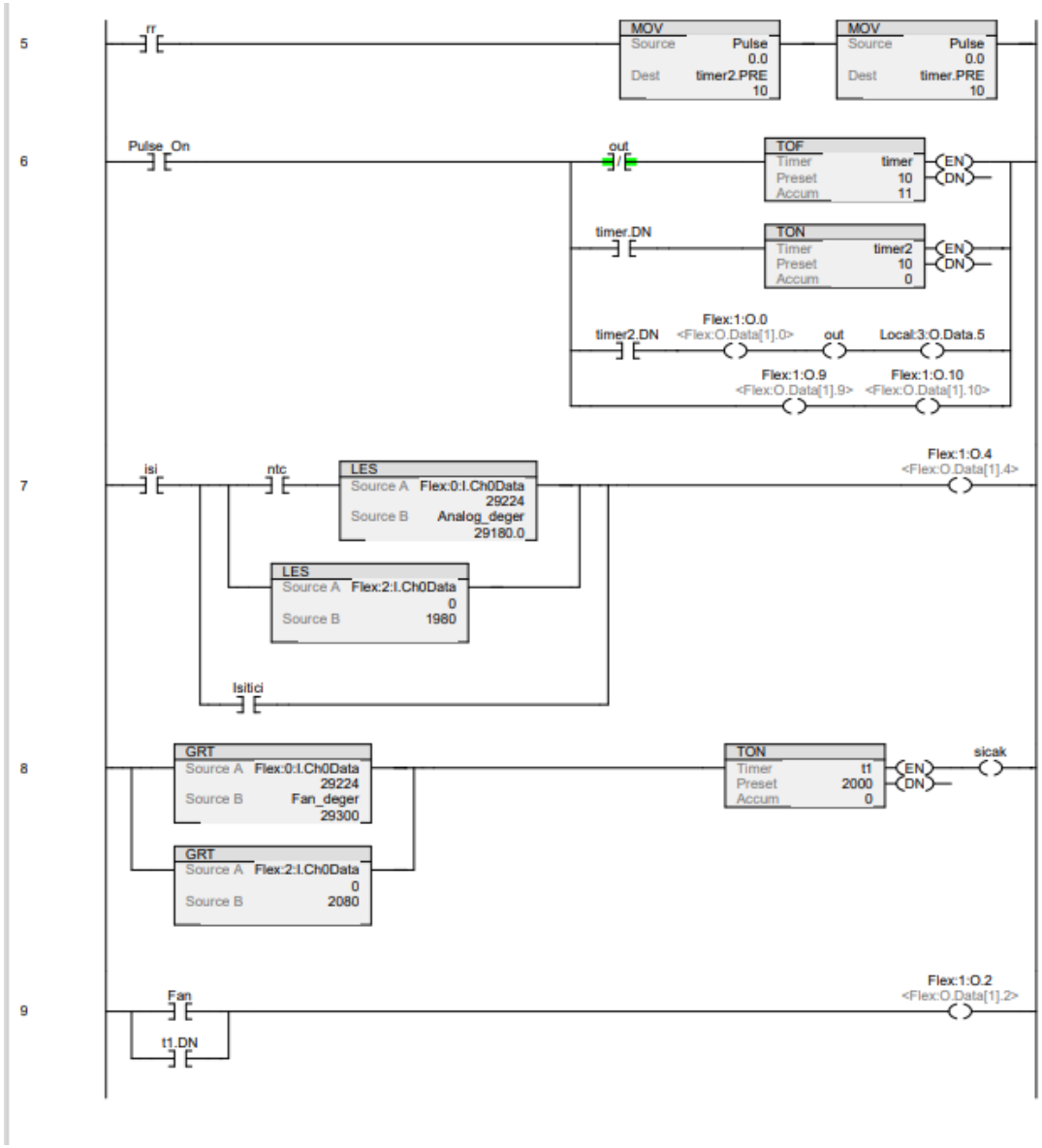
EK B



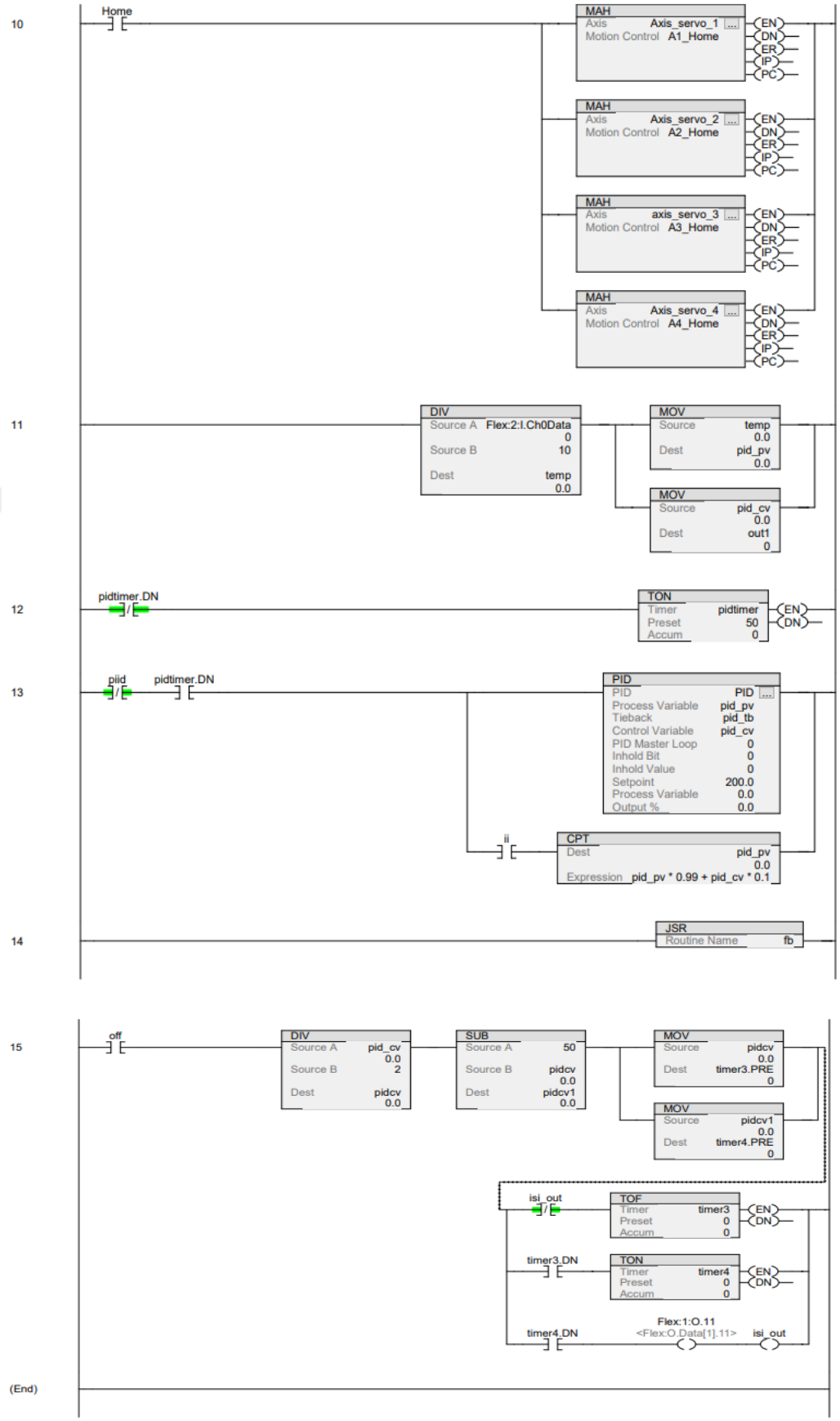
Şekil B 1 : Hareket ve kontrol yazılımı



Şekil B.1 (devam) : Hareket ve kontrol yazılımı.



Şekil B.1 (devam) : Hareket ve kontrol yazılımı.



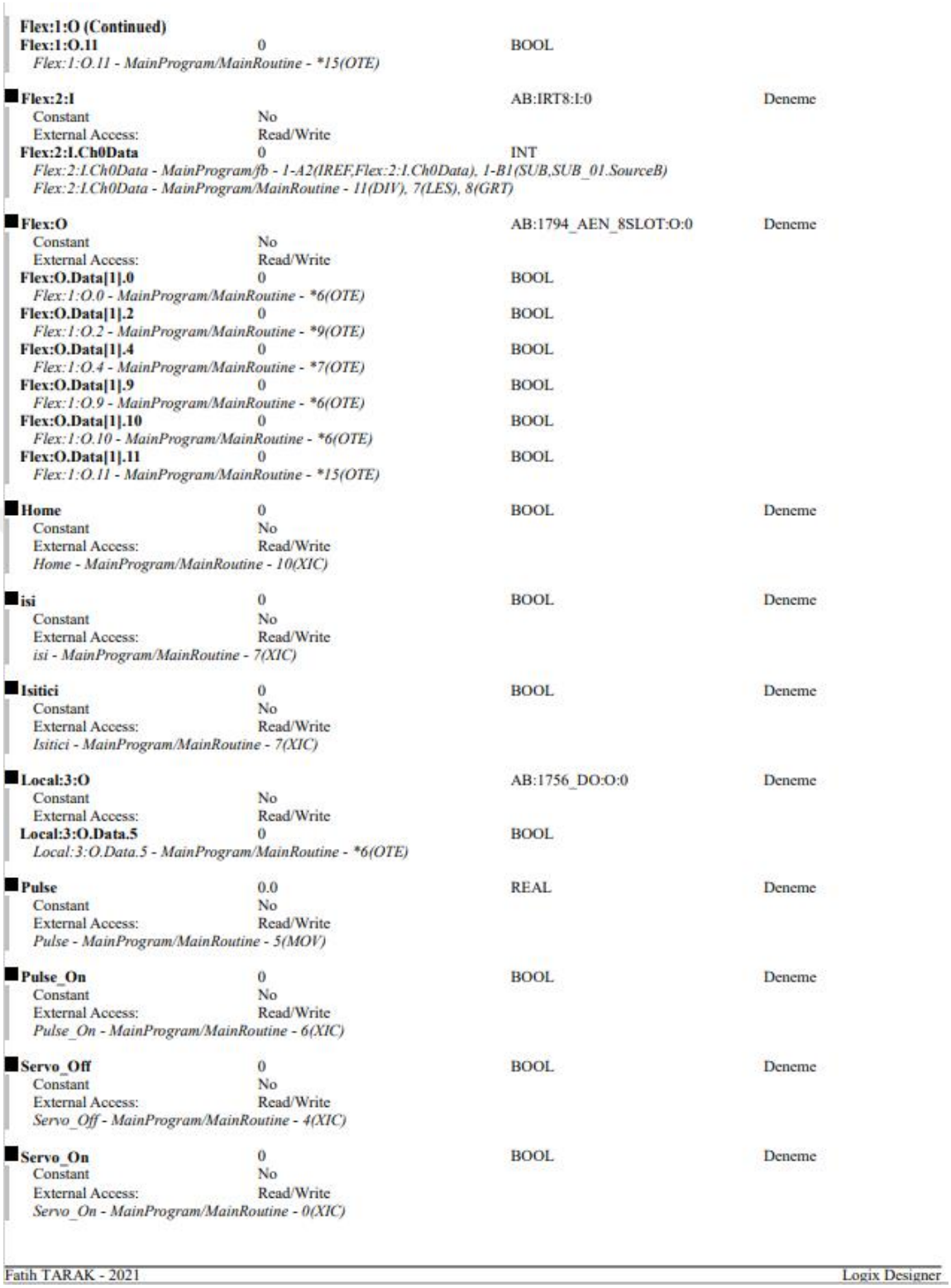
Şekil B.1 (devam) : Hareket ve kontrol yazılımı.

EK C

Name	Value	Data Type	Scope
A_1		MOTION_INSTRUCTION	Deneme
Constant	No		
External Access:	Read/Write		
*A_1 - MainProgram/MainRoutine - *1(MAM)*			
A_2		MOTION_INSTRUCTION	Deneme
Constant	No		
External Access:	Read/Write		
*A_2 - MainProgram/MainRoutine - *2(MAM)*			
A_3		MOTION_INSTRUCTION	Deneme
Constant	No		
External Access:	Read/Write		
*A_3 - MainProgram/MainRoutine - *3(MAM)*			
A_4		MOTION_INSTRUCTION	Deneme
Constant	No		
External Access:	Read/Write		
*A_4 - MainProgram/MainRoutine - *3(MAM)*			
Analog_deger	29180.0	REAL	Deneme
Constant	Yes		
External Access:	Read/Write		
Analog_deger - MainProgram/MainRoutine - 7(LES)			
Axis_servo_1		AXIS_SERVO_DRIVE	Deneme
External Access:	Read/Write		
Axis_servo_1 - MainProgram/MainRoutine - 0(MSO), 1(MAM), 10(MAH), 4(MSF)			
Axis_servo_2		AXIS_SERVO_DRIVE	Deneme
External Access:	Read/Write		
Axis_servo_2 - MainProgram/MainRoutine - 0(MSO), 10(MAH), 2(MAM), 4(MSF)			
axis_servo_3		AXIS_SERVO_DRIVE	Deneme
External Access:	Read/Write		
axis_servo_3 - MainProgram/MainRoutine - 0(MSO), 10(MAH), 3(MAM), 4(MSF)			
Axis_servo_4		AXIS_SERVO_DRIVE	Deneme
External Access:	Read/Write		
Axis_servo_4 - MainProgram/MainRoutine - 0(MSO), 10(MAH), 3(MAM), 4(MSF)			
Fan	0	BOOL	Deneme
Constant	No		
External Access:	Read/Write		
Fan - MainProgram/MainRoutine - 9(XIC)			
Flex:0:I		AB:1794_IE8:I:1	Deneme
Constant	No		
External Access:	Read/Write		
Flex:0:I.Ch0Data	29224	INT	
Flex:0:I.Ch0Data - MainProgram/MainRoutine - 7(LES), 8(GRT)			
Flex:1:O	2#0000_0000_0000_0000	INT	Deneme
AliasFor:	Flex:O.Data[1]		
Base Tag:	Flex:O.Data[1]		
Constant	No		
External Access:	Read/Write		
Flex:1:O.0	0	BOOL	
*Flex:1:O.0 - MainProgram/MainRoutine - *6(OTE)*			
Flex:1:O.2	0	BOOL	
*Flex:1:O.2 - MainProgram/MainRoutine - *9(OTE)*			
Flex:1:O.4	0	BOOL	
*Flex:1:O.4 - MainProgram/MainRoutine - *7(OTE)*			
Flex:1:O.9	0	BOOL	
*Flex:1:O.9 - MainProgram/MainRoutine - *6(OTE)*			
Flex:1:O.10	0	BOOL	
*Flex:1:O.10 - MainProgram/MainRoutine - *6(OTE)*			

Fatih TARAK - 2021 Logix Designer

Şekil C. 1 : Hareket ve kontrol yazılımında kullanılan etiketler.



Flex:1:O (Continued)
Flex:1:O.11 0 BOOL
*Flex:1:O.11 - MainProgram/MainRoutine - *15(OTE)*

Flex:2:I AB:IRT8:I:0 Deneme
Constant No
External Access: Read/Write
Flex:2:I.Ch0Data 0 INT
Flex:2:I.Ch0Data - MainProgram/fb - 1-A2(IREF,Flex:2:I.Ch0Data), 1-B1(SUB,SUB_01.SourceB)
Flex:2:I.Ch0Data - MainProgram/MainRoutine - 11(DIV), 7(LES), 8(GRT)

Flex:O AB:1794_AEN_8SLOT:O:0 Deneme
Constant No
External Access: Read/Write
Flex:O.Data[1].0 0 BOOL
*Flex:1:O.0 - MainProgram/MainRoutine - *6(OTE)*
Flex:O.Data[1].2 0 BOOL
*Flex:1:O.2 - MainProgram/MainRoutine - *9(OTE)*
Flex:O.Data[1].4 0 BOOL
*Flex:1:O.4 - MainProgram/MainRoutine - *7(OTE)*
Flex:O.Data[1].9 0 BOOL
*Flex:1:O.9 - MainProgram/MainRoutine - *6(OTE)*
Flex:O.Data[1].10 0 BOOL
*Flex:1:O.10 - MainProgram/MainRoutine - *6(OTE)*
Flex:O.Data[1].11 0 BOOL
*Flex:1:O.11 - MainProgram/MainRoutine - *15(OTE)*

Home 0 BOOL Deneme
Constant No
External Access: Read/Write
Home - MainProgram/MainRoutine - 10(XIC)

isi 0 BOOL Deneme
Constant No
External Access: Read/Write
isi - MainProgram/MainRoutine - 7(XIC)

Isitici 0 BOOL Deneme
Constant No
External Access: Read/Write
Isitici - MainProgram/MainRoutine - 7(XIC)

Local:3:O AB:1756_DO:O:0 Deneme
Constant No
External Access: Read/Write
Local:3:O.Data.5 0 BOOL
*Local:3:O.Data.5 - MainProgram/MainRoutine - *6(OTE)*

Pulse 0.0 REAL Deneme
Constant No
External Access: Read/Write
Pulse - MainProgram/MainRoutine - 5(MOV)

Pulse_On 0 BOOL Deneme
Constant No
External Access: Read/Write
Pulse_On - MainProgram/MainRoutine - 6(XIC)

Servo_Off 0 BOOL Deneme
Constant No
External Access: Read/Write
Servo_Off - MainProgram/MainRoutine - 4(XIC)

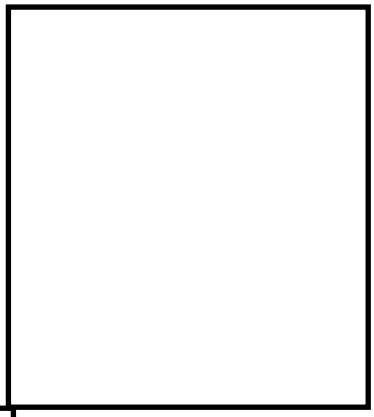
Servo_On 0 BOOL Deneme
Constant No
External Access: Read/Write
Servo_On - MainProgram/MainRoutine - 0(XIC)

Şekil C.1 (devam) : Hareket ve kontrol yazılımında kullanılan kontrolcü etiketleri.

sicak	0	BOOL	Deneme
Constant	No		
External Access:	Read/Write		
*sicak - MainProgram/MainRoutine - *8(OTE)*			
X_hiz	10.0	REAL	Deneme
Constant	No		
External Access:	Read/Write		
X_hiz - MainProgram/MainRoutine - 1(MAM)			
X_MAM	0	BOOL	Deneme
Constant	No		
External Access:	Read/Write		
X_MAM - MainProgram/MainRoutine - 1(XIC)			
X_Move	20.0	REAL	Deneme
Constant	No		
External Access:	Read/Write		
X_Move - MainProgram/MainRoutine - 1(MAM)			
Y_hiz	10.0	REAL	Deneme
Constant	No		
External Access:	Read/Write		
Y_hiz - MainProgram/MainRoutine - 2(MAM)			
Y_MAM	0	BOOL	Deneme
Constant	No		
External Access:	Read/Write		
Y_MAM - MainProgram/MainRoutine - 2(XIC)			
Y_Move	20.0	REAL	Deneme
Constant	No		
External Access:	Read/Write		
Y_Move - MainProgram/MainRoutine - 2(MAM)			
Z_MAM	0	BOOL	Deneme
Constant	No		
External Access:	Read/Write		
Z_MAM - MainProgram/MainRoutine - 3(XIC)			
Z_Move	40.0	REAL	Deneme
Constant	No		
External Access:	Read/Write		
Z_Move - MainProgram/MainRoutine - 3(MAM), 3(MUL)			
Z_MoveN	-40.0	REAL	Deneme
Constant	No		
External Access:	Read/Write		
*Z_MoveN - MainProgram/MainRoutine - *3(MUL), 3(MAM)*			

Şekil C.1 (devam) : Hareket ve kontrol yazılımında kullanılan kontrolcü etiketleri.

ÖZGEÇMİŞ



Ad-Soyad **:** Fatih TARAK

ÖĞRENİM DURUMU:

- **Lisans** **:** 2017, Erciyes Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği
- **Yüksek Lisans** **:** 2021, İstanbul Teknik Üniversitesi, Mekatronik Mühendisliği Anabilim Dalı, Mekatronik Mühendsliği Programı

MESLEKİ DENEYİM:

- 2019 - , Araştırma Görevlisi, Bahçeşehir Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Mekatronik Mühendisliği Bölümü