APİS LİSELER ARASI MODEL UYDU YARIŞMASI

TEMEL ELEKTRONİK VE PROGRAMLAMA

EĞİTİM DÖKÜMANI

HAZIRLAYAN: ALTUĞ ERTAN

Mikroişlemci Nedir?

- Mikroişlemci, üzerinde herhangi bir giriş/çıkış pini veya bellek bulundurmayan yalnızca CPU(Central Processing Unit)'dan oluşan çok küçük entegrelerdir. CPU verilen her türlü talimatın gerçekleştirilebilmesi için gerekli birimlere sinyal gönderen, gelen verilerin kısa süreliğine depolanıp, işlendiği ve matematiksel hesaplamaların yapıldığı birimdir. Özetle, bağlı oldukları sistemin beyni olarak görev yaparlar.

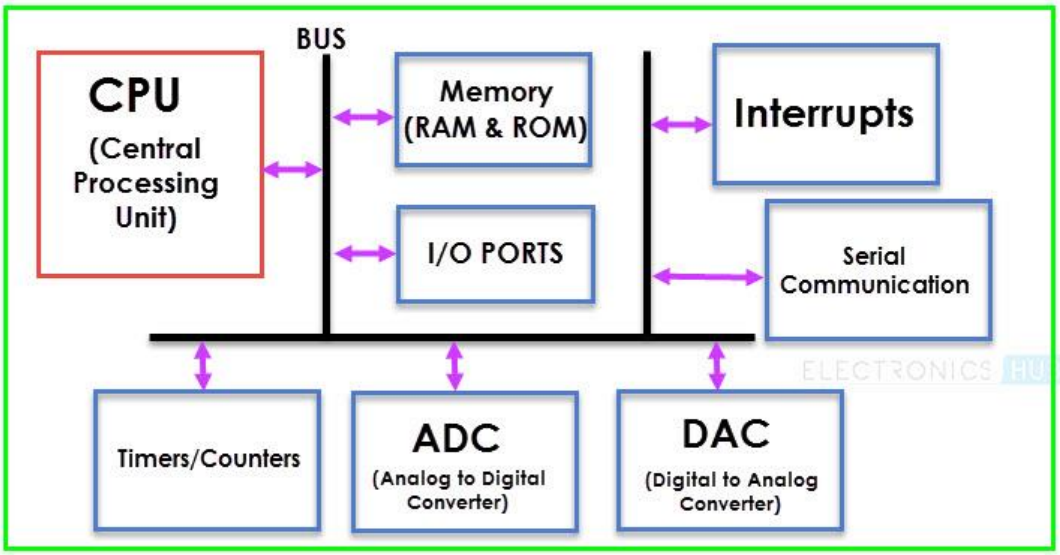
Figür 1: Intel Core i7 mikroişlemci

Mikrokontrolcü Nedir?

- Mikrokontrolcü, mikroişlemcinin aksine üzerinde CPU, I/O (giriş/çıkış) pinleri ve bellek bulunduran küçük elektronik devrelerdir. Yani her miktrokontrolcü üzerinde bir mikroişlemci bulundurur ve küçük bir bilgisayar görevi görür.
- Mikrokontrolcüler genel olarak gömülü sistemler olarak kullanılırlar, yani kendileri için önceden belirlenmiş spesifik bir amacı yerine getirirler, kişisel bilgisayarlar ise özel bir amaca hizmet etmeyen istediğimizde müzik dinleyebildiğimiz, oyun oynayabildiğimiz ve birçok farklı şeyi yerine getirebildiğimiz sistemlerdir.
- Mikrokontrolcüler üzerlerinde bulunan I/O pinleri sayesinde çevre birimlerle veri alışverişini gerçekleştirir ve üzerlerinde bulunan bellekler sayesinde ise istenilen bazı verilerin depolanmasını sağlarlar.
- Mikrokontrolcülerde genel olarak RAM(Random Access Memory) ve ROM(Read Only Memory) olmak üzere iki bellek vardır.
- RAM geçici hafızadır, mikrokontrolcünün gücü kesildiğinde içindeki veriler silinecektir. ROM ise kalıcı hafızadır ve mikrokontrolcüye güç aktarımı kesilse bile içindeki veriler kayıtlı kalacaklardır.
- Model uydumuzun üzerindeki mikrokontrolcünün RAM'i uçuş boyunca sensörlerden elde ettiğimiz verileri sürekli güncel bir şekilde hafızasında tutup gerekli işlemlere sokacaktır. Ayrıca model uydumuzun üzerindeki zil, uydu yere indikten sonra ötmeye başlayacaktır. Biz uyduyu bulup devreyi kapatıp tekrar açtıktan sonra zilin ötmediğini görürüz çünkü zilin ötmesini sağlayan şey kodumuz içerisindeki bazı değişkenlerin

uçuş boyu elde ettiği değerlerdi. Biz uyduyu kapatıp açıktan sonra RAM kendini sıfırladığından ötürü zilin ötmesi için gerekli değerleri kaybettik ve zil yine yerde olmasına rağmen ötmedi.

- ROM dediğimiz hafıza mikrokontrolcüler de genellikle EEPROM (Electronically Erasable Programmable Read Only Memory) olarak bulunur. EEPROM içerisine veri kaydedilebilen ve bu veriyi mikrokontrolcü kapansa da aklında tutabilen ve de istendiği zaman içeriği okunabilen bir bellektir. Örneğin, model uydunuzla uçuş esnasında yer istasyonuna veri aktarıyorsunuz ve mikrokontrolcünüz herhangi bir nedenden dolayı kendine reset attı ve tekrar telemetriyi göndermeye devam etti. Eğer bütün verilerinizi sadece RAM'de depolarsanız böyle bir durum yaşandıktan sonra bazı telemetri verileriniz uydunuz hakkında yanlış bilgi gönderilmesine neden olabilir. Telemetri cümlemizdeki <Paket Numarası> verisi, sağlıklı iletişim devam ettiği sürece her saniyede bir artan bir değerdir. Mesela siz uçuşunuzun 90'ıncı saniyesindesiniz ve paket numaranız 90. Tam bu anda mikrokontrolcünüz reset atar ve yeniden başlarsa, aslında yer istasyonuna 91'inci paketi göndermiş olur ancak telemetri cümlesinde paket numaranız 1 olarak yazacaktır çünkü RAM kendini sıfırlamış olacaktır. Bu tip olaylara karşın bazı veriler için EEPROM belleğini kullanarak sürekli sensörlerden elde ettiğimz güncel değerleri kaydetmeliyiz ve mikrokontrolcümüz her kendini başlattığında ona bu değerleri EEPROM'dan okuması gerektiği komutunu vermeliyiz, böylece değerlerimiz kaldığı yerden mantıklı bir şekilde devam edecektir.

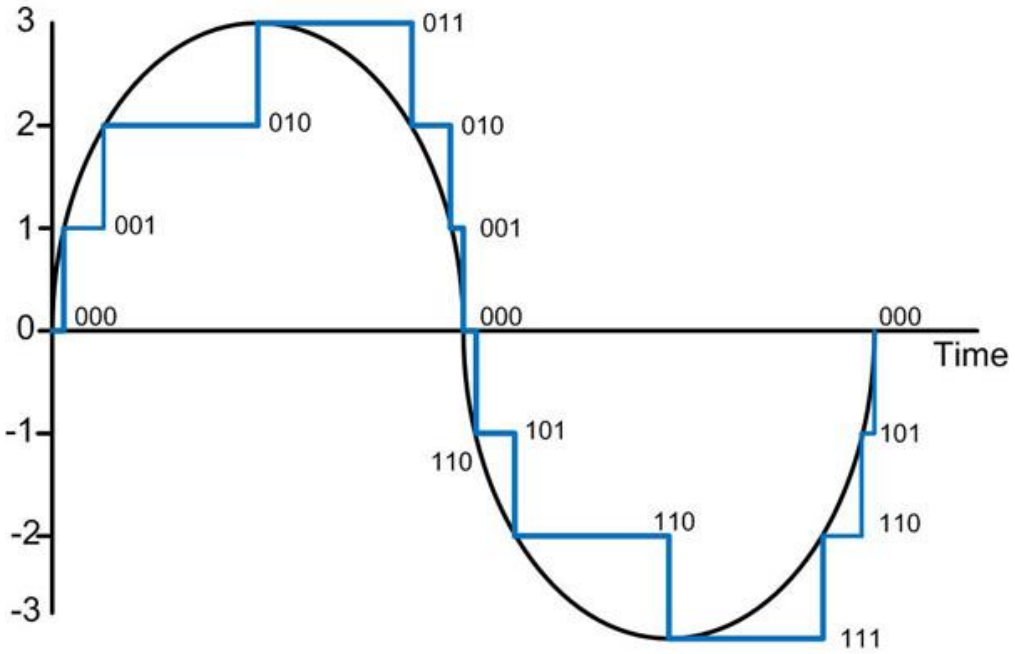


Figür 2: Mikrokontrolcünün Yapısı

- Yukarıdaki şekil bir mikrokontrolcünün iç yapısını temsil etmektedir, şekildeki terimleri kısaca açıklayalım.
- **CPU:** Mikrokontrolcümüzün beynidir. Çevre birimlere yapılması gereken işlerin talimatını verir, çevre birimlerden alınan dönütleri değerlendiri ve her türlü aritmetik işlemi gerçekleştirir.
- **BUS:** Mikrokontrolcü içerisinde bulunan bütün birimlerin birbirleriyle haberleşmesini sağladığı yollardır, veriler bu kanallar üzerinden iletilirler.

- **Memory (RAM & ROM):** Mikrokontrolcünün belleğidir. Gereken her türlü veriyi, talimatı hafızasında saklar. Geçici ve kalıcı olmak üzere ikiye ayrılır.
- **Serial Communication:** Seri haberleşme protokollerini kullanarak sensör gibi cihazlardan veri almamızı ve onlara veri iletmemizi sağlayan giriş/çıkış noktalarıdır.
- **Interrupts:** Bu kelime "kesme" anlamına gelir. Mikrokontrolcü üzerinde bulunan sadece belli başlı giriş/çıkışlar kesme işlevini yerine getirebilir. Kesme pinlerimizden birine bir cihaz takarız ve kodumuzun içine bu cihazla ilgili özel durumlar için bir fonksiyon yazarız. Bu özel durum gerçekleştiğinde kesme pininden CPU'ya sinyal gider ve CPU o an çalışan koda kısa bir ara vererek yazdığımız özel fonksiyonu çalıştırır ve sonra tekrar ana koda kaldığı yerden devam eder.
- **I / O Ports:** Giriş/çıkış pinleridir. Mikrokontrolcüye bu pinler üzerinden çevre birimlerini bağlarız ve aralarındaki dijital sinyal alışverişini sağlarız.
- **Timer / Counters:** Bu kelimeler sırasıyla "zamanlayıcı" ve "sayaç" anlamına gelirler. Zamanlayıcı bizim verdiğimiz ayara göre belli bir sayıdan geriye bir süreyi sayabilir veya sıfırdan bir kronometre gibi ileriye süreyi tutabilir. Sürenin çözünürlüğünü mikrokontrolcümüzün kapasiteleri dahilinde istediğimiz gibi ayarlayabiliriz. Sayaç ise koşul olarak belirlenmiş bir olay gerçekleştiği zaman artan veya azalan parametrelerdir. Örneğin, mikrokontrolcümüzün zamanlayıcısını sıfırdan başlatarak ve onu bir saniye çözünürlüğüne sahip bir kronometre olarak kullanarak, telemetrimizi her saniyede bir kere gönderebilmek için gereken süre verisini elde ederiz. Bu her bir saniye geçtiğinde ise yeni telemetri cümlesini gönderirken paket numaramızı bir arttırırız, bu da sayaca örnektir.
- **ADC (Analog to Digital Converter):** Analog sinyali dijital sinyale çeviren dönüştürücülerdir. Gerçek hayatta var olan her türlü fiziksel nicelik süreklidir ve herhangi bir değeri alabilir, işte biz bu tip tanımlamaya uyan sinyallere analog sinyaller diyoruz. Ancak elektronik cihazlarımız kapasiteleri gereği herhangi bir veriyi sürekli yani kesintisiz olarak elde edemezler, eğer böyle bir şey olsaydı cihazlarımız çökerdi. Bunun yerine belli başlı çok kısa periyodik zaman aralıklarıyla veri alışverişini gerçekleştirirler. Aynı zamanda elektronik cihazlarımız bir işlem veya bir veri alışverişini yaparken herhangi bir değeri direk olarak elde edemezler bu işleri 1 ve 0 sayılarının kombinasyonlarını kullanarak gerçekleştirirler, biz bu 1 ve 0 sayılarına **bit (binary digit)** diyoruz. Yani mikroişlemcilerimiz, mikrokontrolcülerimiz ve bilgisayarlarımız bütün işlerini bitleri kullanarak gerçekleştirirler, bitlerden oluşan sinyallere dijital sinyaller diyoruz. İşte bu ADC'lerde dışarıdan ölçülmüş bir analog sinyali mikrokontrolcümüz anlayabilsin diye dijital sinyale çevirir. Örneğin, mikrokontrolcümüz model uydumuzun devresini besleyen pilin voltajını ölçebilmek için ADC'lerden faydalanacaktır.
- **DAC (Digital to Analog Converter):** Biz 1 ve 0 sayı kombinasyonlarından oluşan verileri kolaylıkla okuyamayız. Bu yüzden DAC'lar mikrokontrolcülerimizin içinde bulunan dijital sinyalleri bizim anlayabileceğimiz analog sinyallere yani gerçel sayılara çevirirler. Örneğin, bir arabanın üzerindeki hız sensörü ölçtüğü hızı mikrokontrolcüye bir dijital sinyal olarak şu şekilde "11110" gönderir ve mikrokontrolcüyle gösterge ekranı arasına takılı bir 5 bitlik DAC bu dijital sinyalin

karşılığı olan “30” sayısını analog sinyale çevirip ekrana yazdırır. Böylece biz hızı göstergede 30 km/h olarak görürürüz.

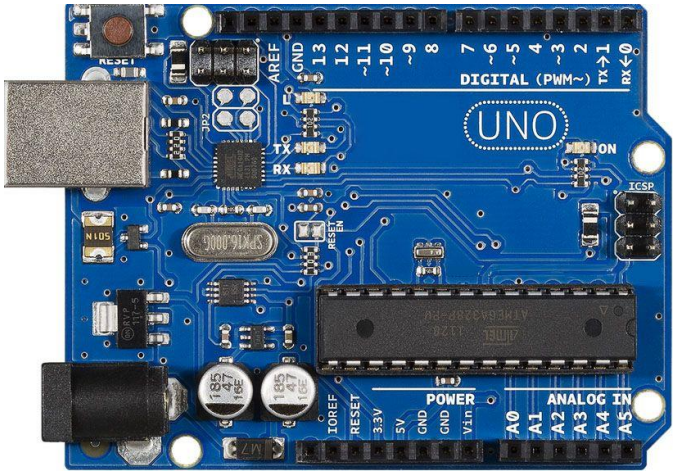


Figür 3: Analog vs. 3 bitlik Dijital Sinyal

Popüler Mikrokontrolcü Devre Kartları

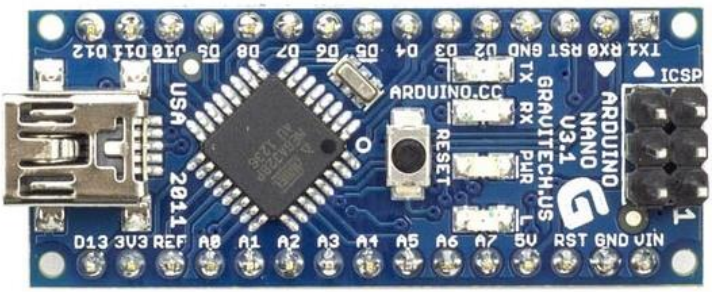
- Bu başlık altında bazı popüler mikrokontrolcü sağlayıcılarının devre kartları hakkında bilgi verilecektir. Hepsinin tek tek özellikleri anlatılmayacaktır sadece Arduino serisinin özellikleri örnek olarak verilecektir.

Arduino



Figür 4: Arduino Uno

Mikrokontrolcü	ATmega328
Çalışma Gerilimi	5 V
Dijital Giriş-Çıkış Pinleri	14 adet
Analog Giriş Pinleri	6 adet
Her Giriş-Çıkış Pini için Akım	40 mA
Flash Hafıza	32 KB
SRAM	2 KB
EEPROM	1 KB
Uzunluk	68.6 mm
Genişlik	53.4 mm
Ağırlık	25 g



Figür 5: Arduino Nano

Mikrokontrolcü	ATmega328
Çalışma Gerilimi	5 V
Dijital Giriş-Çıkış Pinleri	16 adet
Analog Giriş Pinleri	8 adet
Her Giriş-Çıkış Pini için Akım	40 mA
Flash Hafıza	32 KB
SRAM	2 KB
EEPROM	1 KB
Uzunluk	43 mm
Genişlik	19 mm
Ağırlık	5 g

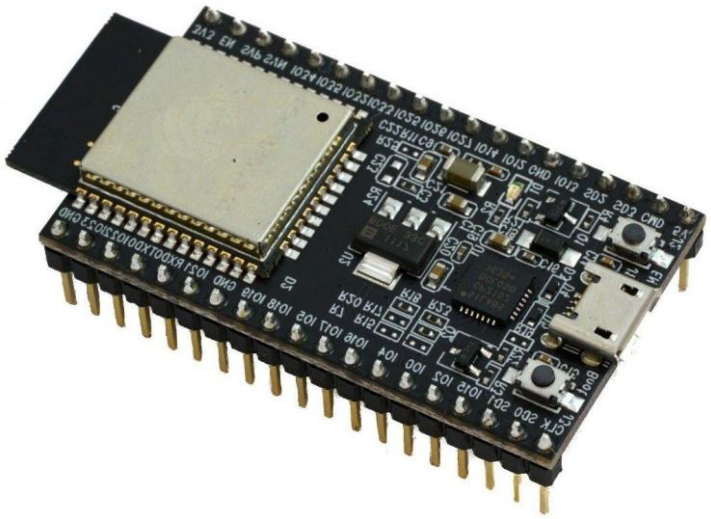
Figür 6: Atmega328 Mikrokontrolcü

- Arduino, Atmel marka mikrodenetleyicilere devre kartı üreten İtalyan bir şirkettir. Burada dediğimiz gibi Arduino bir mikrokontrolcü değildir, Arduino anlaşmalı olduğu mikrokontrolcü üreticisinin mikrokontrolcülerini kullanarak hobi elektroniğine merak sarmış kişilere bu mikrokontrolcü sistemlerini daha rahat kullanabilmeleri için devreler dizayn eder.
- Arduino Uno ve Nano'nun özelliklerine baktığımız zaman boyutları dışında bir çok özelliğinin aynı olduğunu görüyoruz. Bizim durumumuzda özellikleri aynı olan iki devre kartını karşılaştırdığımız zaman boyutları küçük ve ağırlığı az olanı kullanmak daha mantıklı olacaktır.
- Mikrokontrolcülerimizi ve sensörlerimizi beslerken kullandığımız güç kaynağının voltajına dikkat etmeliyiz. Mesela burada tabloda gördüğünüz üzere Arduinoların çalışma gerilimi 5 volttur yani devremizi 5 voltla beslemeliyiz. Ancak piyasada 5 voltluk pil bulamayabiliriz bu gibi durumlarda biz voltaj regülatörlerine başvururuz. Voltaj regülatörü türüne göre düşük bir voltajı yükseltebilir veya yüksek bir voltajı düşürebilir. Biz de ihtiyaçlarımız doğrultusunda Arduino'ya düzgün bir şekilde güç aktarabilmek için bu regülatörleri kullanmalıyız. Eğer az voltaj verirsek Arduino beklenen performansı sergilemeyebilir veya çok voltaj verirsekte zarar görüp bozular. Bu konu da dikkatli olmalıyız.
- Dijital, analog giriş/çıkış pin sayıları ve seri haberleşme $(I^2C, SPI\ ve\ UART)$ pin sayıları yaptığımız projede ne kadar çevre cihazı kullanacağımıza bağlı olarak önem arz eder. Mesela bizim yarışmamız baz alındığında görevileri gerçekleştirebilmek için gerekli sensörleri seçip bu sensörlerin hangi pinlerden kaç tane kullandığını hesaba katmamız gerekir. Eğer bu donanımsal konuyu projemizin başlarında dikkatlice değerlendirmezsek ve ihtiyacımız olandan daha az bağlantıya sahip bir mikrokontrolcü seçersek ileride tasarımımız da büyük değişiklikler yapmamız gerekebilir ve bu da projeye ciddi zarar verecektir.
- **Flash Hafıza:** Flash hafıza, mikrokontrolcümüzün içine gömdüğümüz yazılımın tutulduğu bellektir. Model uydu projesini baz aldığımız zaman bellek tipleri arasında en çok göz önüne almamız gerekendir. Model uydu projemizi yaparken hazırladığımız ana kod içerisine sensörleri kullanabilmek için birçok kütüphane ekleyeceğiz ve

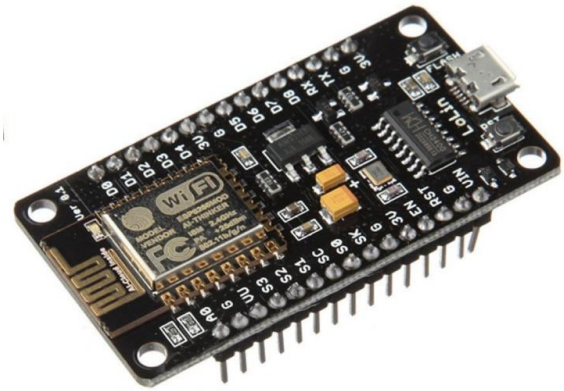
onların içlerindeki fonksiyonları kullanacağız. Kütüphaneler nispeten büyük dosyalardır ve bazı kütüphaneler flash hafızada ciddi yer kaplayabilirler. Bu yüzden model uydumuzun devresini tasarlamadan önce, seçtiğimiz sensörlerin kütüphanelerini ana kodumuza ekleyip, bu kodu derleyip ne kadarlık bir hafıza kullandıklarını kontrol etmemiz gerekir. Eğer bu süreci uygulamazsak, ilerleyen zamanlarda mikrokontrolcümüze kodumuzu yüklerken yetersiz alan hatası alabiliriz ve mikrokontrolcü değişikliğine gitmek zorunda kalırız. Mesela kodunuzun boyutu 32 KB'yi geçiyorsa Arduino Nano veya Uno kullanamazsınız.

- **SRAM (Static RAM):** Mikrokontrolcülerde kullanılan RAM tipidir, detayına girilmeyecektir. Model uydu projemiz çok yüksek işlem kapasitesi gerektirmediği için herhangi bir mikrokontrolcünün sahip olduğu SRAM kapasitesi bizim için yeterli olacaktır. Örneğin, Arduino'nun 2 KB'lık SRAM'i bizim için gayet yeterlidir.
- **EEPROM:** Model uydu projemiz dahilinde EEPROM hafızamıza en fazla birkaç önemli değişkeni kaydedeceğimizi düşünürsek çok küçük belleğe sahip EEPROM'lar bile bize yeterli olacaktır. 4 byte'lık 5 tane float değişkenini EEPROM'a kaydetmek istediğimizi düşünelim bunlar 20 byte'lık bir yer işgal edeceklerdir. Arduino'yu baz aldığımız zaman 1 KB = 1000 Byte'lık bir hafızası olduğu düşünüldüğünde 4 byte'lık 250 tane float değişkenini EEPROM'da kayıtlı tutabiliriz. Bu yüzden mikrokontrolcülerin EEPROM değerleri konusunda da tedirgin olmamıza gerek yok.
- Uzunluk, genişlik ve ağırlık değerleri mikrokontrolcü ve sensör seçimleri yaparken önemlidir. Model uydumuzun boyutları göz önüne alındığında en optimim boyutlardaki devre kartını inşa etmeliyiz. Bunun için de olabildiğinde iyi performanslı ve küçük boyutlu kartlar seçmeliyiz.

Espressif



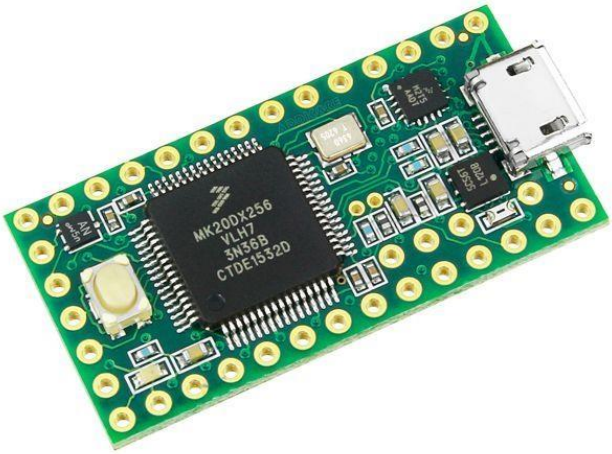
Figür 7: ESP32



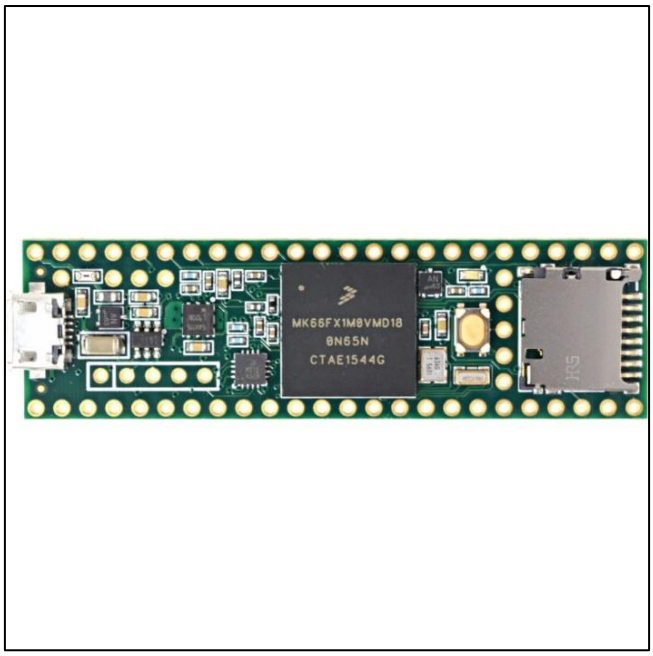
Figür 8: ESP8266

- Espressif ucuz mikrokontrolcü kartları üreten Çin menşeli bir şirkettir ve hobi elektroniğiyle ilgilenenler arasında çok popülerdir.
- Yukarıda görmüş olduğunuz üzere ESP8266 modelinin üzerinde Wi-Fi modülü vardır. Tweet atma üzerine kurulu bonus görev dikkate alındığında, bu mikrokontrolcü devreye ekstra bir modül koymadan sizin internete bağlanma ihtiyacınızı karşılama potansiyeline sahiptir.

PJRC Teensy



Figür 9: Teensy 3.2



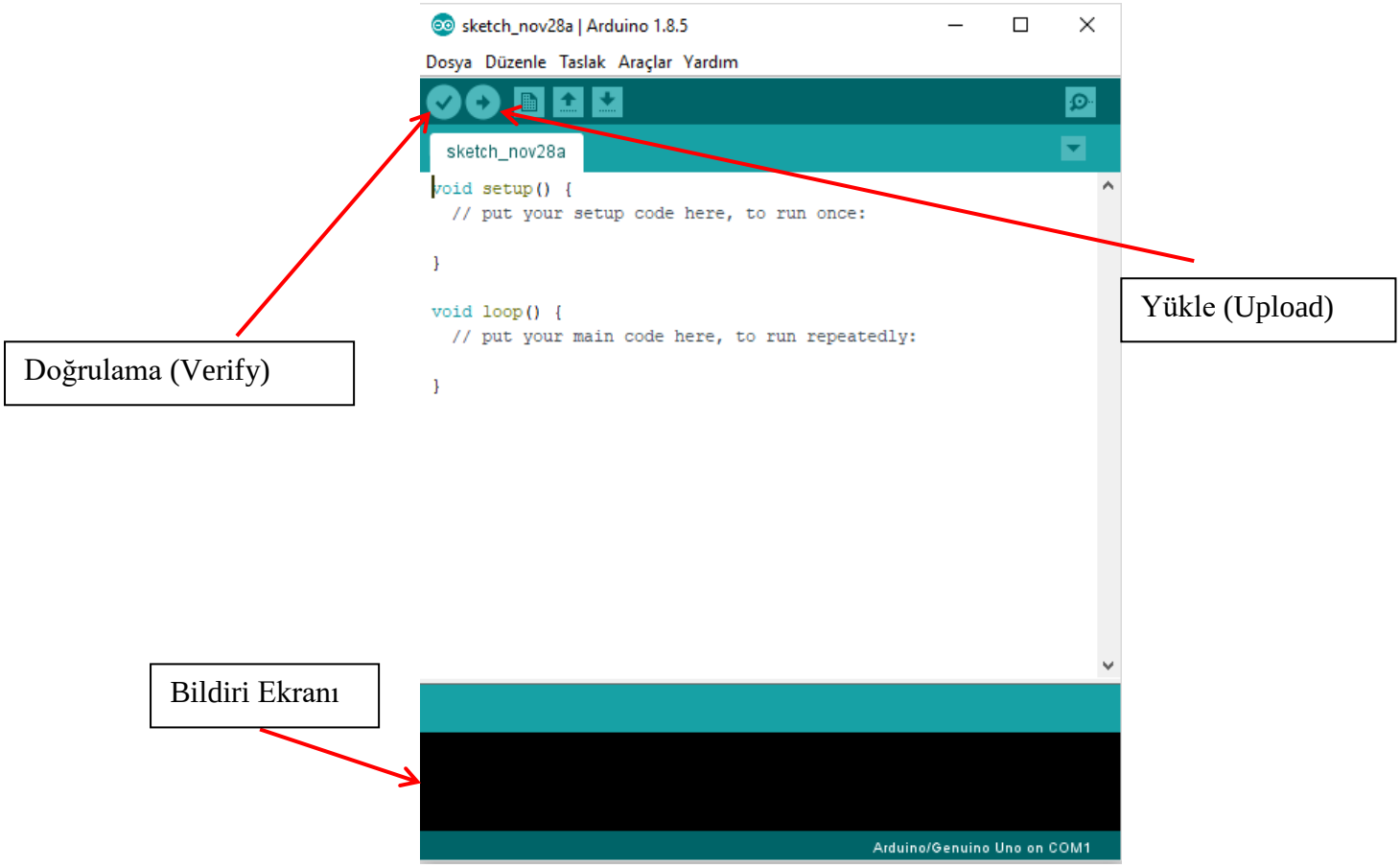
Figür 10: Teensy 3.6

- PJRC orta performansta mikrokontrolcü kartları üreten Amerikan bir şirkettir.
- Teensy 3.6 modelinin üzerinde bir SD kart soketinin bulunduğunu görüyoruz. Verileri uçuş boyunca bir SD karta kaydedeceğimiz göz önüne alındığında, bu soket özelliği sayesinde ana devremize herhangi bir harici SD kart soketi takmamıza gerek kalmaz ve yerden kazanç sağlayabiliriz.

IDE (Integrated Development Environment)

- Mikrokontrolcüleri bilgisayar üzerinden programlayabilmemiz için bazı programlara ihtiyacımız vardır. Bunlar editör, derleyici ve hata ayıklıcıdır.
- **Editör:** Mikrokontrolcümüz için tasarladığımız programı kullanacağımız yazılım dilinde satır satır yazabileceğimiz programlardır. En basitinden "Not Defteri" uygulamasını buna bir örnek olarak gösterebiliriz.
- **Derleyici:** Yazdığımız kodu bilgisayar diline çeviren programdır. Derleyiciler olmadan yazdığımız yazılımlar bilgisayar ve mikrokontrolcülere hiçbir anlam ifade etmez.
- **Hata Ayıklayıcı:** Yazdığımız programdaki söz dizilimiyle ilgili (syntax) veya komutsal olan hataları bize söyler. Bunları düzeltmeden derleme işlevini yerine getiremeyiz.
- Öncellikle IDE'nin türkçe karşılığı "Bütünleşik Geliştirme Ortamı"dır. IDE'ler içlerinde hem editör, hem derleyici hem de hata ayıklayıcı bulunduran programlardır, bu yüzden onlara bütünleşik diyoruz. Bir IDE kullanarak yazdığımız bir programı doğrudan mikrokontrolcü içine yükleyebiliriz.

- Arduino IDE, Keil Microvision ve Atollic popüler IDE'lere birer örnektir. Biz model uydu projemizi yaparken yukarıda bahsettiğimiz hangi mikrokontrolcüyü kullanırsak kullanalım onu Arduino IDE üzerinden rahatça programlayabiliriz.
- Şimdi Arduino IDE'yi kullandığımızı varsayarak programın arayüzünü biraz tanıtalım.

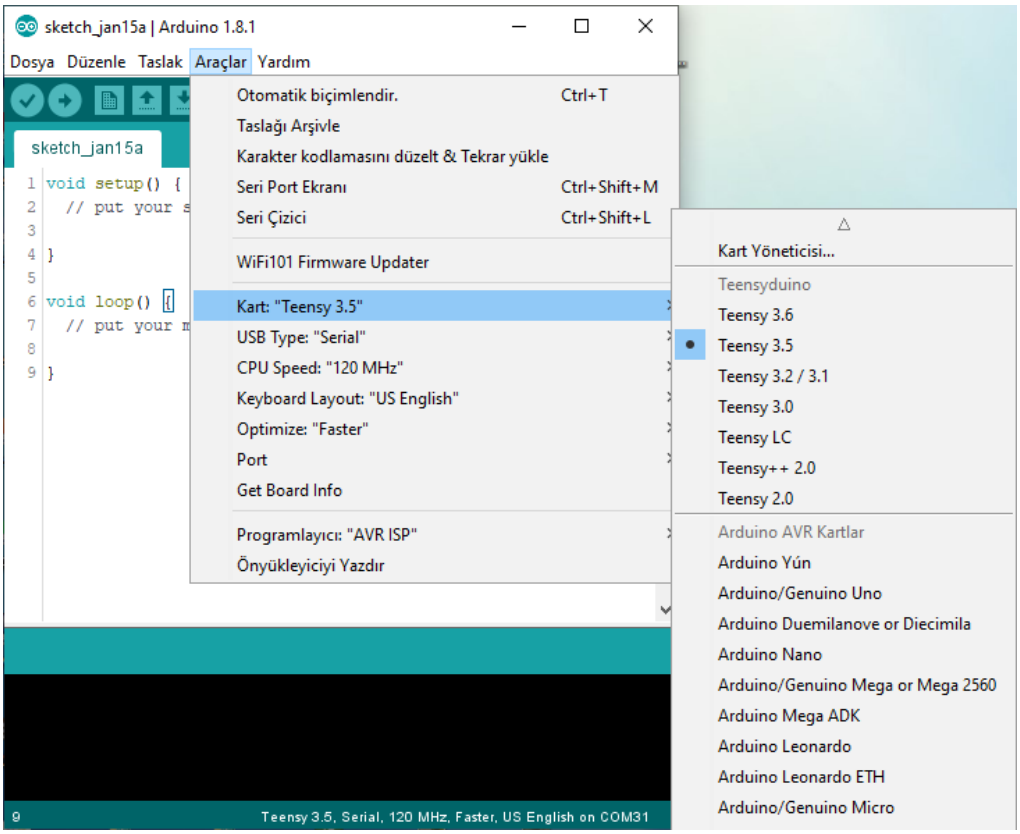


Figür 11: Arduino IDE Arayüzü

- Ana kodumuzu yukarıda gördüğümüz arayüzün içine yazarız. Sonra doğrulama komutuna basarız ve hata ayıklayıcı ile derleyici çalışır. Eğer bir hata varsa bu bize bildiri ekranında gösterilir ve derleme işlemi yarıda kesilir. Hata olmadığı takdirde ise kodumuzun başarıyla derlendiği bildiri ekranında ifade edilir. Kodumuzda bir hata yoksa yükle komutunu kullanarak onu mikrokontrolcümüzün içine gömebiliriz.

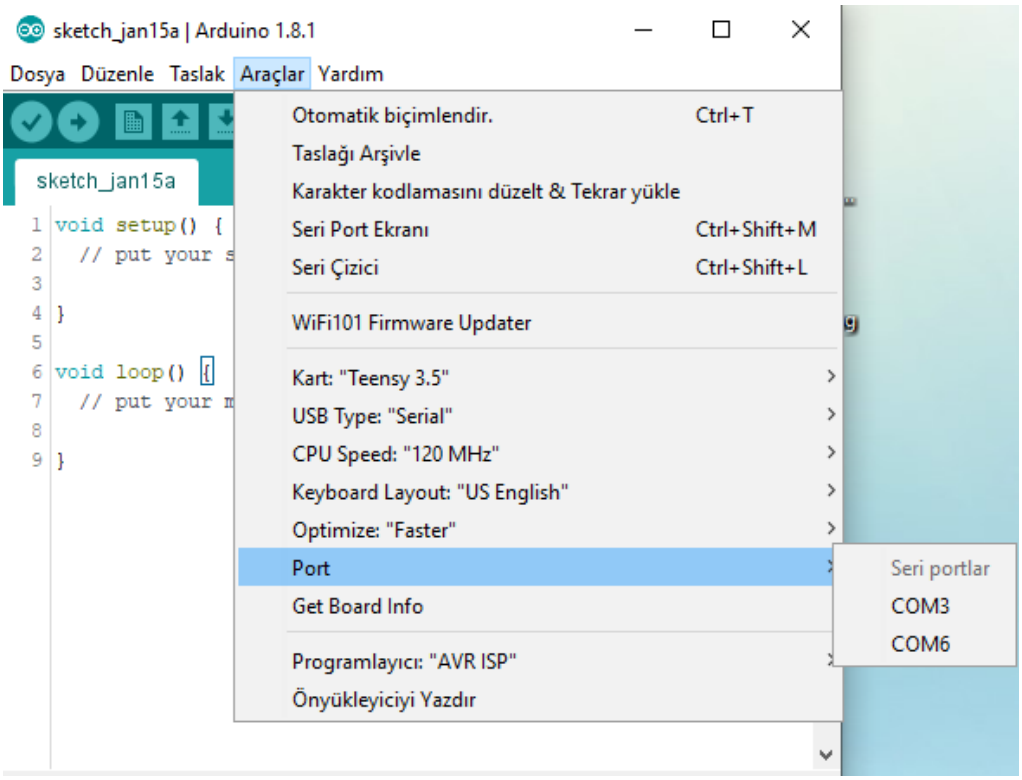
Temel Arduino IDE Ayarları

- **Kart seçimi:** Projemizde kullanacağımız mikrokontrolcü modelinin seçildiği kısımdır.



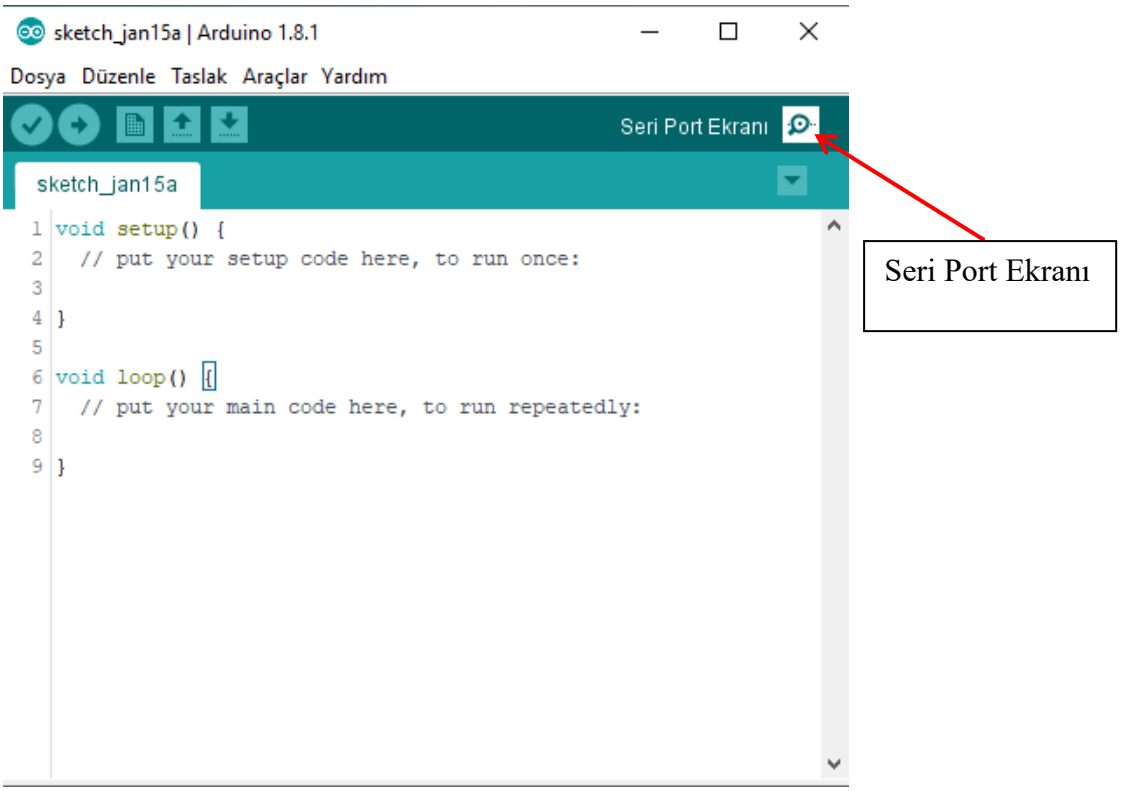
Figür 12: Kart Seçimi

- Araçlar (Tools) menüsüne gelerek Kart (Board) kısmının üzerine geldiğimizde birçok mikrokontrolcü modeli ekranda gözükecektir. Biz bu modeller arasından kendi projemizde kullanacağımızı seçmeliyiz. Yanlış kart seçimi yapıldığı takdirde, kodlarımız mikrokontrolcüye yüklenmeyip hata verecektir. Normalde Arduino IDE'nin kartlar kısmında sadece Arduino modelleri gözükecektir, biz diğer bahsettiğimiz ESP ve Teensy kartlarını kullanabilmek için internetten bunların kütüphanelerini indirmeliyiz. Örneğin, Teensy kartlarını Arduino IDE'de kullanabilmek için Google'a Teensyduino yazarak gerekli adımları takip ederek bu imkana erişebiliriz.
- **Port seçimi:** Mikrokontrolcünün bilgisayara bağlı olduğu ve haberleşeceği portun seçilmesi. Eğer aynı anda birden fazla cihaz bilgisayara takılıysa, mikrokontrolcünün bağlı olduğu port aygıt yöneticisinden kontrol edilmelidir. Portlar COMXX olarak isimlendirilirler. XX kısmı bir sayıyı temsil etmektedir.



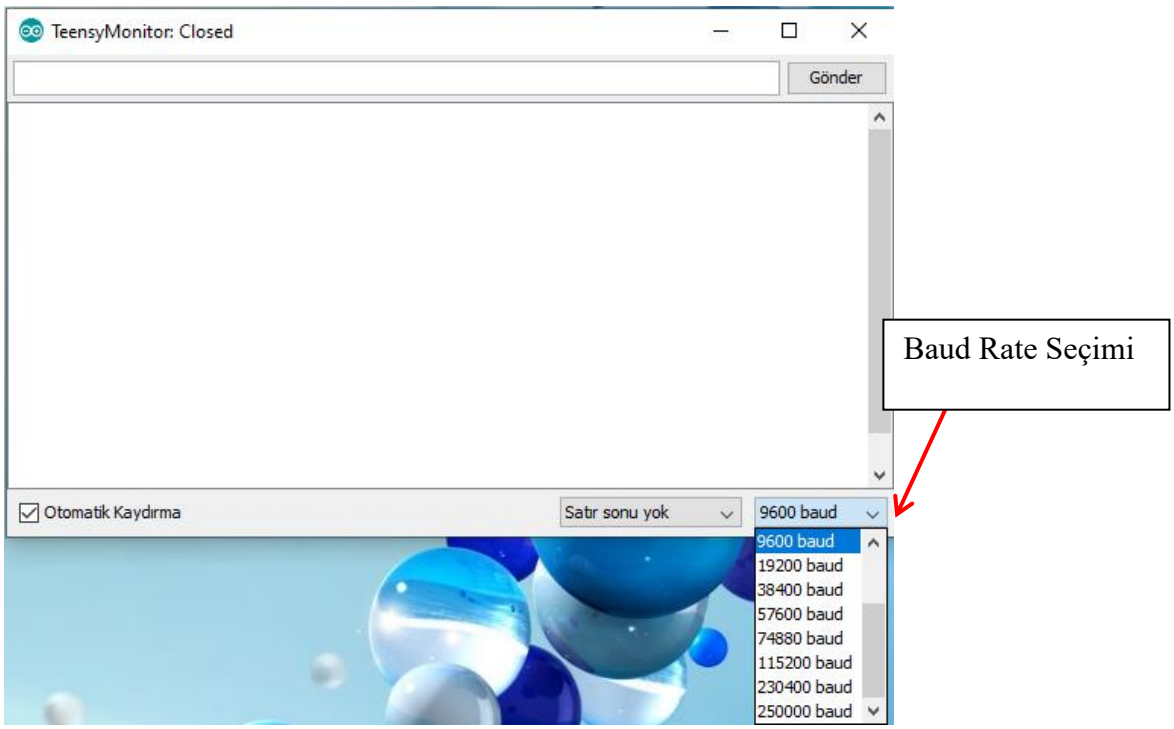
Figür 13: Port Seçimi

- Araçlar menüsünden Port kısmına gelerek mikrokontrolcümüzün bağlı bulunduğu portu seçmeliyiz. Mesela yukarıdaki görsele göre mikrokontrolcümüz COM6'ya bağlı ama biz COM3'ü seçtik ve kodumuzu mikrokontrolcümüze yüklemeye çalıştık. Program hata verecektir ve kodu yükleyemeyecektir.
- **Baud rate seçimi:** Mikrokontrolcü ile bilgisayar arasındaki veri alışverişi kapasitesinin ayarıdır, birimi bps(bit per second)'dır. Genellikle değiştirilmemiş ayarı 9600 bps'dir. Örneğin; 9600 bps, saniyede mikrokontrolcüden bilgisayar 9600 bitlik veri gideceğini ifade eder. Bu bizim için yeterli ve idealdir. GPS ve haberleşme modüllerini kullanırken bu performans yetersiz kalırsa baud rate'imizi 115200 bps'e çıkarabiliriz.



Figür 14: Seri Port Ekranının Açılması

- Seri Port Ekranı (Serial Monitor), mikrokontrolcümüzden gönderdiğimiz "ekrana yazdır" komutlarının çıktısının gerçekleştiği yerdir. Yani sensörlerden aldığımız verilerin değişimini ekrana yazdırmak ve onları kontrol etmek istersek, seri port ekranını açmalıyız.

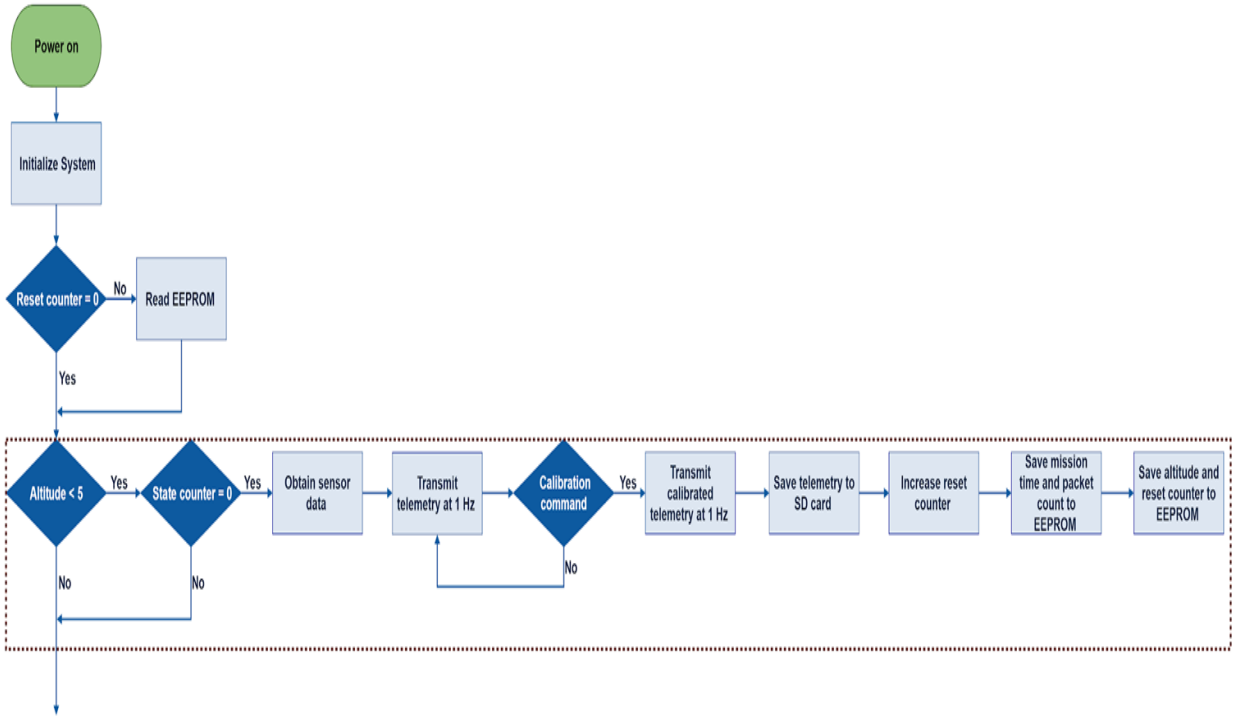


Figür 15: Baud rate seçimi

- Seri port ekranını açtığımız zaman yukarıdaki gibi bir ekran açılacaktır ve yazdırılan şeyler oraya gelecektir. Kodumuzun içinde belirttiğimiz bilgisayarla mikrokontrolcünün haberleşeceği Baud Rate'i açtığımız seri porttan da seçmemiz gerekmektedir. Örneğin kodumuzda baud rate'i 9600 bps olarak yazıp, seri portta 19200 bps seçersek, ekrana gelmesi gereken karakterler dışında anlamsız karakterler geldiğini görebiliriz. Buna mani olmak için seri porttaki baud rate ayarımızı da koddakiyle aynı olacak şekilde düzenlemeliyiz.

Algoritma Nedir?

- Algoritma bir problemi çözebilmek veya elde edilenden daha performanslı bir sonuca ulaşabilmek için takip edilen işlem basamaklarıdır.
- Algoritmaların ilk adımları ve son adımları kesinlikle bellidir, oluşturduğumuz algoritma ilk adımdan son adıma gidene kadar aradaki basamaklarda kararlar vererek en mantıklı çözüme ulaşır.
- Elimizdeki problem bazı ana kısımlara ayrılır. Öncelikle her bir kısım için ayrı algoritmalar oluşturulur. Ardından bu her kısım birleştirilir ve ana sorunun çözümü için gereken algoritma akışı oluşturulmuş olur.
- Biz projemizi hazırlarken birçok algoritma geliştireceğiz. Örneğin aldığımız yükseklik verilerini belli bir algoritmaya sokarak doğru uçuş statülerini elde etmeye çalışacağız. Algoritmalarımızı doğru kuramazsak, uydumuzun işleyişinde ve verilerinde hata elde ederiz.



Figür 16: Blok Diyagram

- Yukarıda gördüğünüz blok diyagram 2019 senesinde CanSat Competition'a katılırken takımımızın hazırladığı ve uçuş yazılımımızın algoritma akışını gösteren blok diyagramın küçük bir parçasıdır. Uçuş yazılımınızı yazmadan önce bu tip blok diyagramlar oluşturarak algoritmanızın akışını modellemeniz, yazacağınız kodu kolaylaştıracaktır.
- Bu tip akış diyagramlarında algoritmanın başlangıcını ve bitişini yuvarlak veya oval köşeli bir dikdörtgenler gösterir. Normal dikdörtgenler yapılacak bir işi, komutu ifade ederler. Eşkenar dörtgenler ise bazı parametler göz önüne alınarak karar verilmesi gereken noktaları gösterirler.
- Yukarıdaki blok diyagramında sistem başlatılıyor,herhangi bir reset durumu var mı diye kontrol ediliyor. Eğer varsa EEPROM okunuyor, yoksa sensörlerden alınan yükseklik verisine göre uçuş statüsüne karar veriliyor. Kahverengi kesikli çizgilerle taranmış olan komuta zinciri, uydu drone ile yükseltilmeden önce yerde durduğu andaki statüyü ifade ediyor. Bu statünün koşulu yüksekliğin 5 metreden küçük olması ve eğer bu doğruysa diğer adımlar gerçekleşmeye başlıyor. Adımlar ise sensörlerden güncel verilerin alınması, bu verilerin 1 Hz'de telemerti olarak yer istasyonuna iletilmesi, telemetri verisinin SD karta kaydedilmesi ve de gerekli verilerin EEPROM'a kaydedilmesidir. Daha sonra tekrar güncel yükseklik verisine göre uçuş statüsü sorgulanacaktır ve yükseklik 5 metreden büyükse başka bir uçuş statüsüne girilecektir.
- Bu akış diyagramlarını hem yazılım geliştirme süreciniz kolay olsun diye yapmanız gerekiyor, hem de biz sizden bunları PDR ve CDR raporlarında istiyoruz.

Temel Kodlama Bilgisi

- Algoritmaların büyük bir çoğunluğu karar verme mekanizmalarından oluşmaktadır ve algoritmamızı koda dökerken bu mekanizmaları oluşturabilmek için birtakım yazılımsal fonksiyonlar kullanırız.
- Fonksiyonlara gelmeden önce biraz değişkenlerden bahsedelim. Değişken (Variable) dediğimiz şeyler yazılım geliştirirken çok fazlaca kullandığımız, belli sayısal değerleri veya sözel ifadeleri içlerinde tutan küçük depolardır. Biz bu depolara istediğimiz gibi isim verebiliriz, ve deponun ismini yazılımımızda kullandığımızda aslında içindeki değeri kullanmış oluruz. Aynı zamanda da bu depoların içindeki değerleri istediğimiz zaman değiştirebiliri, bu yüzden onlara değişken diyoruz.
- Değişkenler içlerinde tuttukları değerlerin türlerine göre sınıflandırılırlar. Bu sınıfları inceleyelim.
- **int (integer):** Integer "tamsayı" anlamına gelmektedir. Bir değişken tanımlarken biraz sonra göstereceğimiz gibi öncelikle değişkenin tipini sonra ona koyacağımız adı en son da değişkenin değerini yazarız. Tamsayı değişkenleri **int** adı başa getirilerek tanımlanırlar.

- **float (floating point numbers):** Float “Kayan noktalı sayılar” demektir, yani ondalık kısım içeren sayılardır. Örneğin, 3.14 yani pi sayısı. Değikenin başına **float** eki getirilerek tanımlanırlar.
- **char (character):** Character “Karakter” anlamına gelir ve bu tip değişkenlere klavyemiz üzerinde bulunan herhangi bir karakteri atayabiliriz. Örneğin bu bir sayı da olabilir ‘34’, bu bir işaret de olabilir ‘!’ veya bu bir harfte olabilir ‘a’.
- **String:** String “Karakter Dizisi” anlamına gelir. Stringler normalde C programlama dilinde **char** yapısı kullanılarak tanımlanırlar. Ancak Arduino C/C++ tabanlı bir programlama dili olmasına rağmen birtakım başka fonksiyonlarda içermektedir ve siz bir karakter dizisini **String** ön ekiyle de tanımlayabilirsiniz. **Char** değişkeniyle tek bir karakter depolayabilirken, **String** değişkeni ile yan yana yazılmış bir sürü karakteri depolayabiliriz. Örneğin, “Merhaba Dünya!”.
- Şimdi bu değişkenleri nasıl tanımlayacağınıza dair birkaç örnek verelim. Bu yazdıklarımın aynısını Arduino IDE içerisinde yapabilirsiniz.

$$\boldsymbol{int}\ yaş = 23$$
$$\boldsymbol{float}\ boy = 1.75$$

$$\boldsymbol{char}\ kan_grubu = 'A'$$

$$\boldsymbol{String}\ isim = "Ahmet"$$

- Öncelikle yukarıda kafanız karışmasın diye birinci değişkenin adını yaş olarak tanımladım ancak değişken isimlerinde Türkçe karakter kullanmak programda hataya sebep olacaktır. Bu yüzden değişkenin adını yaş değil yas olarak tanımlamalıyız. **Char** tipi değişkenler tanımlanırken **‘’** tek tırnak işaretleri içerisine alınırlar. **String** tanımlanırken ise karakter dizisi **“”** çift tırnak işaretleri içerisine alınır.
- Eğer bu değişkenleri kullanmazsak bir çok veriyi sürekli elle girmemiz gerekir ve bu verilerden birinde bile hata olduğu zaman, tek tek o veriyi kullandığımız yerleri bulup elimizle düzeltmemiz gerekir. Bu hem çok yorucu hem de daha çok hataya yol açabilecek bir yöntem. Bunun yerine biz bir değeri değişken olarak tanımlayıp, o değerin gelmesi gereken yerlere ismini yazarsak, o değişkenin değerinde hata yapmış olsak bile bu sorunu kolayca düzeltebiliriz. Değişkeni tanımladığımız ilk satıra gidip = simgesinin karşısındaki değeri değiştiririz ve değişken adının geçtiği bütün satırlarda böylece değişkenin değeri değişmiş olur.
- Fonksiyonlar (function) ise aslında matematik dersinde gördüğümüz fonksiyonlarla aynı işlemi yaparlar. Matematikte bir fonksiyon işleme girecek bir veya birkaç parametreden ve bu parametrelerin gerçekleştireceği operasyonda oluşurlar. Bir yazılım içerisinde de fonksiyonlar belli başlı görevleri yerine getirmek için kullanılan kod bloklarıdırlar. Görevi yerine getirecek parametreleri ve bir dizi komutu barındırırlar. Bunu örnek vererek anlatalım.
- Matematiksel bir $f(x)$ fonksiyonu tanımlayalım.

$$f(x) = x^2 + 5x + 3$$

- Yukarıdaki fonksiyonda x fonksiyonun parametresidir, $x^2 + 5x + 3$ ise yapılacak işlemdir, bu fonksiyonda x parametresi yerine bir değer koyarak sayısal bir çıktı elde ederiz.

$$f(2) = 2^2 + 5 \times 2 + 3 = 17$$

- Şimdi bu fonksiyonun aynısı C programlama diliyle yazalım. Yukarıda değişkenlerin tiplerinden bahsetmiştik. Tamsayı, ondalık sayılar veya karakterler gibi. Programlama dilinde bir fonksiyon tanımlarken, aynı değişkenlerde yaptığımız gibi fonksiyonun türünü ve ismini belirtmeliyiz. Ardından bu fonksiyonda kullanılacak parametreleri belirtip, fonksiyonun içeriğini yazmalıyız. Bir fonksiyonun türü onun sonuç olarak hangi tipte bir değer vereceğini belli eder. Yani eğer fonksiyonumuzun bir tamsayı çıktı vermesi gerekiyorsa onu **int** ön adıyla tanımlamalıyız. Mesela yukarıdaki fonksiyonumuzun sadece tamsayılar için tanımlı olduğunu düşünelim bu şartlar altında çıktlarımız da hep tamsayı olacaktır. Şimdi kod bloğunu inceleyelim.

```
int f_fonksiyonu(x) {
return (x * x + 5 * x + 3);
}
```

- Yukarıda $f(x)$ matematiksel fonksiyonumuzun programlama dili karşılığını tanımladık. **Int** ön eki programımızdan çıkan sonucun tamsayı olacağını belirtiyor. f_fonksiyonu tanımladığımız fonksiyonun kodumuz içerisindeki adı, fonksiyonumuzu kullanmak istediğimiz zaman bu adı yazacağız. Parentez içersindeki x de fonksiyonumuzun parametresini belirtiyor. Sonra { kullanarak kod bloğumuz açtık. İki süslü parantez arası olan kodlar bir bloğu temsil eder ve her fonksiyon bir kod bloğudur. İkinci satırda return yazdık ve yanına da fonksiyonumuzdan beklediğimiz matematiksel işlemi yazdık. Return (Döndürmek) komutu fonksiyonumuzun artık sonuna geldiğimizi ifade eder ve fonksiyon çıktısı olarak yanında yazan işlemin sonucu verir. Son olarak da } ile bloğumuzu kapatırız.
- Yukarıda tanımladığımız fonksiyon ise kodumuzda şöyle kullanabiliriz.

```
int sonuc = 0;
sonuc = f_fonksiyonu(2)
```

- Yukarıdaki komut dizisinde tamsayı olan ve ismi sonuc olan bir değişken tanımlayıp değerini 0 yaptık. Daha sonra bu sonuc değişkeninin değerini fonksiyonda x = 2 koyulduğunda çıkacak sonuca eşitledik. Aynı yukarıdaki matematiksel fonksiyonumuzda olduğu gibi fonksiyonun çıktısı 17 olacaktır ve sonuc değişkenine 17 sayısı atanacaktır.
- Şimdi Arduino'da ön tanımlı olarak bulunan ve sık sık kullanacağımız **if(), while()** ve **for()** fonksiyonlarından bahsedeceğiz.
- **if():** If "Eğer" anlamına gelmektedir ve tek kullanılabildiği gibi **else if()** ve **else()** fonksiyonlarıyla seri olarakta kullanılabilmektedir. If ve serisi olan fonksiyonlara

parametre olarak koşul ifadeleri gelmektedir. Bu alt alta dizilen koşullardan o an hangisi doğruysa o fonksiyonun kod bloğuna girilir ve kod bloğu içinde yazılan işlemler yapılır. Hemen bir örnek verelim:

```
int sınav_puanım = 91;
if ( sınav_puanım > 90) {
Serial.println("A");
}
else if (sınav_puanım > 70) {
Serial.println("B");
}
else if (sınav_puanım > 50) {
Serial.println("C");
}
else {
Serial.println("F");
}
```

- Yukarıdaki kodda bir sınavdan aldığımız puanı 91 olarak tanımladık. Sonra aldığımız sınav puanının hangi harf notuna denk geldiğini test etmek için bir **if-else** serisi yazdık. Burada sınav puanımız tek tek 90,70 ve 50 notlarıyla karşılaştırıldı. Eğer 90 puandan büyükse A aldık, eğer 90 puandan küçük ve 70 puandan büyükse B aldık, eğer 70 puandan küçük ve 50 puandan büyükse ise C aldık. Sınav notumuz 50'den bile küçükse bu durumda F alıyoruz.
- Koşul serimize her zaman **if()** fonksiyonuyla başlarız ve eğer **if()** içindeki durum doğru değilse, ardından gelen **else if()** blokları incelenir. Eğer bunlardan hiçbiri de doğru değilse, en son bir **else** bloğu tanımlanır ve yukarıdaki şartlardan hiçbiri sağlanmıyorsa bunu yap diye bir komut dizisi verilir.
- **Serial.println()** komutu ile harf notumuzu ekrana yazdırıyoruz yani seri port ekranına. Bu fonksiyon Arduino'ya özgüdür.
- **if()** fonksiyonlarını, yükseklikleri karşılaştırarak uçuş statülerini elde etmek için kullanacağız.
- **while():** While "-iken" anlamına gelmektedir. Yani bir koşul doğru iken, doğru olduğu sürece yazdığımız işlemler sürekli tekrarlanacaktır. Ne zaman bu koşul yanlış olursa operasyonlar tekrar edilmeyi kesecek ve bir alt satırdaki kodlarla yazılımımız akmaya devam edecektir. Hemen bir örnek yapalım.

```
int carpim = 1;
while (carpim < 100) {
carpim = carpim * 5;
}
```

- Yukarıdaki kod bloğunda bir çarpma işleminin sonucu olan çarpım değişkenini tanımladık ve ona 1 değerini atadık. Sonra **while()** fonksiyonumuza koşul olarak

dedik ki; eğer çarpım değeri 100'den küçükse sürekli kendi içinde bulunan işlemleri yap ta ki 100 çarpım değeri 100'den büyük olana kadar.

- carpim = carpim *5 yazan satırdaki işlem şunu yapmaktadır. Öncelikle eşittir işaretinin her zaman sağındaki işlem yapılır. Yani burada ilk çarpım değerimiz 5 ile çarpılır ve tekrar güncel haliyle carpim isimli değişkenimize atanır.

$$carpim = 1$$
$$carpim = carpim \times 5 = 5$$
$$carpim = carpim \times 5 = 25$$
$$carpim = carpim \times 5 = 125$$

- İşlem yukarıda gördüğünüz üzere tekrar tekrar yapılır. En altta değerimiz 125'e varır ve 125 > 100 olduğu için kodumuz bir daha **while()**'ın içine girmez ve altındaki yeni komutu gerçekleştirir. Tabii burada biz bir komut yazmadık eğer olsaydı onu gerçekleştirecekti.
- **for():** Aynı **while()** gibidir, belli bir koşul doğru olduğu sürece içerisinde bulunan işlemleri gerçekleştirir. Tek farkı parametrelerinin tanımlanış şeklidir. Biz **while()** ve **for()** tipi fonksiyonlara döngü (loop) fonksiyonları diyoruz.
- **for()** fonksiyonun döngü sayısını ayarlamak için çok yukarılarda öğrendiğimiz üzere bir sayaç (counter) değişkeni tanımlamamız gerekir. Her turun sonunda bu sayaç bellik bir miktar arttırılır ve gerekli koşulu sağladığı sürece de döngü devam ettirilir.
- **for(**sayacın tanımı; döngü için gerekli koşul; sayacın arttırılma miktarı). Hemen bir örnek yapalım:

```
int toplam = 0;
for(int sayac = 1; sayac <= 20; sayac +=2) {
 toplam += sayac;
}
```

- Yukarıdaki kod bloğumuzda yeni bir operator tanıdık +=. Bu işaret sağında bulunan sayıyı veya değişkeni simgenin solunda bulunana ekler. Kısacası aşağıda paylaşacağımız iki işlem birbirine denktir:

$$toplam = toplam + sayac$$
$$toplam += sayac$$

- Bu kod bloğu için bir sayaç tanımladık ve birinci tur için ona 1 değerini atadık. Sonra blok içindeki toplam değişkenine sayacımızı ekledik ve toplam değişkenimizin değeri 1 oldu. Bu turun sonunda da sayacımızı 2 arttırdık ve böylece sayaç değerimiz 3'e çıktı. Ardından sayacımız 20'ye küçük eşit mi diye kontrol ettik ve küçük olduğu için aynı işlemleri yapmaya devam ettik. Ne zaman sayaç 20'den büyük olursa o zaman bu döngüden çıkacağız.