METE HOCA PANDA

EĞLENDİREREK ARDUINO KODLAMA ÖĞRETEN MİNİ OYUN KONSOLU

EĞİTİM ve KULLANIM KİTABI (ARDUINO IDE)

(Son güncelleme: 29 Ağustos 2022)

www.metehoca.com

METE HOCA

DİĞER METE HOCA ÜRÜNLERİ

ARDUINO ÖNCESİ TEMEL ELEKTRONİK EĞİTİM ve DENEY SETİ

ARDUINO'da iyi olmak için öncelikle ELEKTRONİK'te iyi olmak gerekir.

Arduino Öncesi Temel Elektronik Eğitim ve Deney Seti, Arduino'ya doğru ve güçlü bir şekilde başlamak için gereken temel elektronik bilgisini önemli detayları atlamadan, gereksiz teknik bilgilerden arındırarak ve olabildiğince eğlenceli bir şekilde vermeyi hedefleyen bir settir.

Oldukça kapsamlı hazırlanmış bu set ile elektrik ve elektroniğin temellerini **METE HOCA** farkıyla hızlı ve basit bir şekilde öğrenecek, Arduino eğitimlerinin çok önemli olmasına rağmen çoğu zaman göz ardı edilen elektronik devre bileşenleri konusunda çok daha bilinçli olabileceksiniz.

ARDUINO'YA GÜÇLÜ BAŞLANGIÇ EĞİTİM ve PROJE SETİ

ÖZGÜN ve KULLANIŞLI projeler yapabilmek için ARDUINO'yu doğru öğrenmek gerekir.

Arduino'ya Güçlü Başlangıç Eğitim ve Proje Seti, bayrağı öncül set olan Arduino Öncesi Temel Elektronik Eğitim ve Deney Seti'nin bıraktığı yerden devralıyor ve geleceğin tasarımcılarını Arduino ile tanıştırıyor.

METE HOCA'nın "gereksiz bilgilerden arındırılmış basit ve eğlenceli anlatım" felsefesini devam ettiren bu set ile Arduino'yu en temelden, ilginç kuruluş hikâyesinden başlayarak keşfedeceksiniz.

ARDUINO HEADER ETİKETLERİ

Klon Arduino kullanırken yapılan yanlış bağlantılar projemizin çalışmamasından, olası bir kısa devre nedeniyle Arduino board'umuzun, proje modüllerinin, hatta bilgisayarımızın bozulmasıyla sonuçlanabiliyor.

Orijinal Arduino'larda bulunan baskılar yüksek pin header yapılarının getirdiği hatalı bağlantıları asgariye indiriyor. Klon Arduino board'lar ise ucuzluk adına bu özelliğe sahip değiller ve hatalara davetiye çıkarıyorlar.

METE HOCA ARDUINO HEADER ETİKETİ serisi özellikle yeni öğrenen küçüklerin kolaylıkla yapabilecekleri bu hataların önüne geçebilmenizi sağlıyor. Header etiketleri Arduino Uno, Arduino Mega ve Protoshield için hazırlandı.

METE HOCA LOGICSHIELD

LogicShield, dijital elektroniğin mantıksal işleyişini ve kodlama yapısını anlamak için hayati önemde olan **lojik kapı**ların öğrenciler tarafından deneyimlenmesini sağlayan bir Arduino shield'idir. Arduino Uno ve Mega'da kullanılabilir.

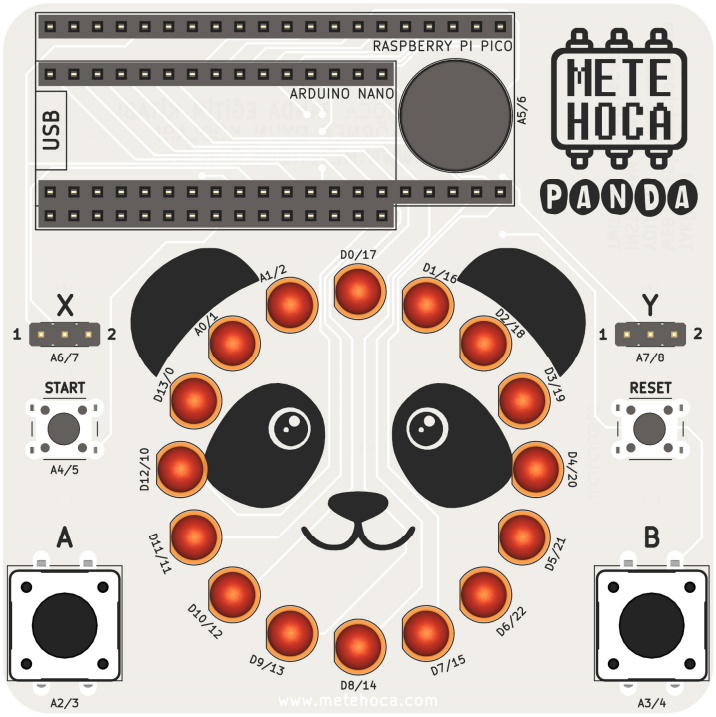
Ağustos 2022, Mete K. Atay

www.metehoca.com

İÇİNDEKİLER

METE HOCA PANDA NEDİR?

Mete Hoca Panda, Arduino Nano ve Raspberry Pi Pico ile kendi basit oyunlarınızı kodlayabileceğiniz mini bir oyun konsoludur.



Panda Arduino Nano kullanılarak Arduino IDE ile ve Raspberry Pi Pico kullanılarak Arduino IDE, MicroPython veya CircuitPython ile kodlanabilir.

Panda üzerinde halka şeklinde yerleştirilmiş 16 LED, kolayca programlanabilir 3 buton, 2 anahtar ve istenen frekansta ses üretilebilecek pasif buzzer bulunur.

LED ve butonlar doğrudan geliştirme kartı pinlerine bağlı olduklarından dolayı kullanılmaları için herhangi bir kütüphane veya özel yönteme ihtiyaç yoktur.

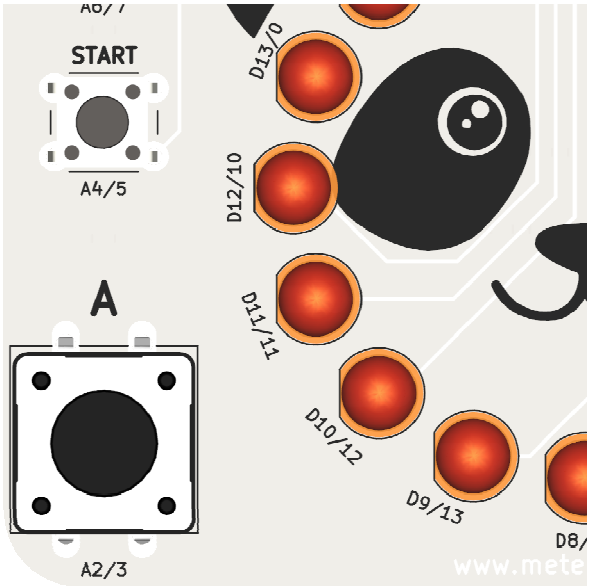
Panda ile bir veya iki kişilik oyunlar geliştirmek mümkün, her şey sizin yaratıcılığınıza bağlı!

NASIL KULLANILIR?

Panda'yı kullanabilmek için **Arduino Nano** veya **Raspberry Pi Pico** geliştirme kartlarından birine ihtiyacımız var. Bu kartların klon sürümleri de aynı şekilde kullanılabilir. Panda üzerine takılı geliştirme kartı'nın USB girişi üzerinden güç alır.

Panda üzerinde yer alan bileşenlerin hangi geliştirme kartının hangi pinine bağlı oldukları etraflarında **Nano/Pi Pico** biçiminde yazılıdır. Örneğin START butonunun altında yazan **A4/5**, bu butonun Nano'nun **A4** ve Pi Pico'nun **GP5** pinine bağlı olduğunu ifade eder. **D0/17** yazılı LED de Nano'nun **0** numaralı dijital pinine ve Pi Pico'nun **GP17** kodlu pinine bağlıdır.

Panda üzerinde yer alan 16 LED halka şeklinde dizili olduğu için sürekli değişen veya birbirini takip eden animasyonlar üzerine kurulu oyunlar veya görsel şovlar kodlanabilir. LED'lerin her biri koruma direnci ile GND'ye bağlıdır ve ışık vermeleri için pinlerine HIGH (1) göndermek yeterlidir.



A, **B** ve **START** butonları ise pull-down dirençleri ile bağlıdır ve basıldıklarında ilgili pine HIGH (1) gönderirler.

RESET butonu geliştirme kartlarının ilgili pinlerine (RST/RUN) bağlıdır ve çalışan kodu en baştan başlatır.

Pasif buzzer, tone() komutu veya PWM sinyali ile kontrol edilebilir ve istenen frekansta ses çıkarılabilir.

Panda üzerindeki **X** ve **Y** kodlu anahtarlar Nano ve Pi Pico'da farklı yöntemlerle kontrol edilirler. Pi Pico'da sıradan bir dijital giriş gibi okunabilen bu anahtarlar Nano'nun A6 ve A7 analog pinlerine bağlı oldukları için hangi konumda oldukları bu pinlerden okunan analog değerin miktarına göre belirlenir.

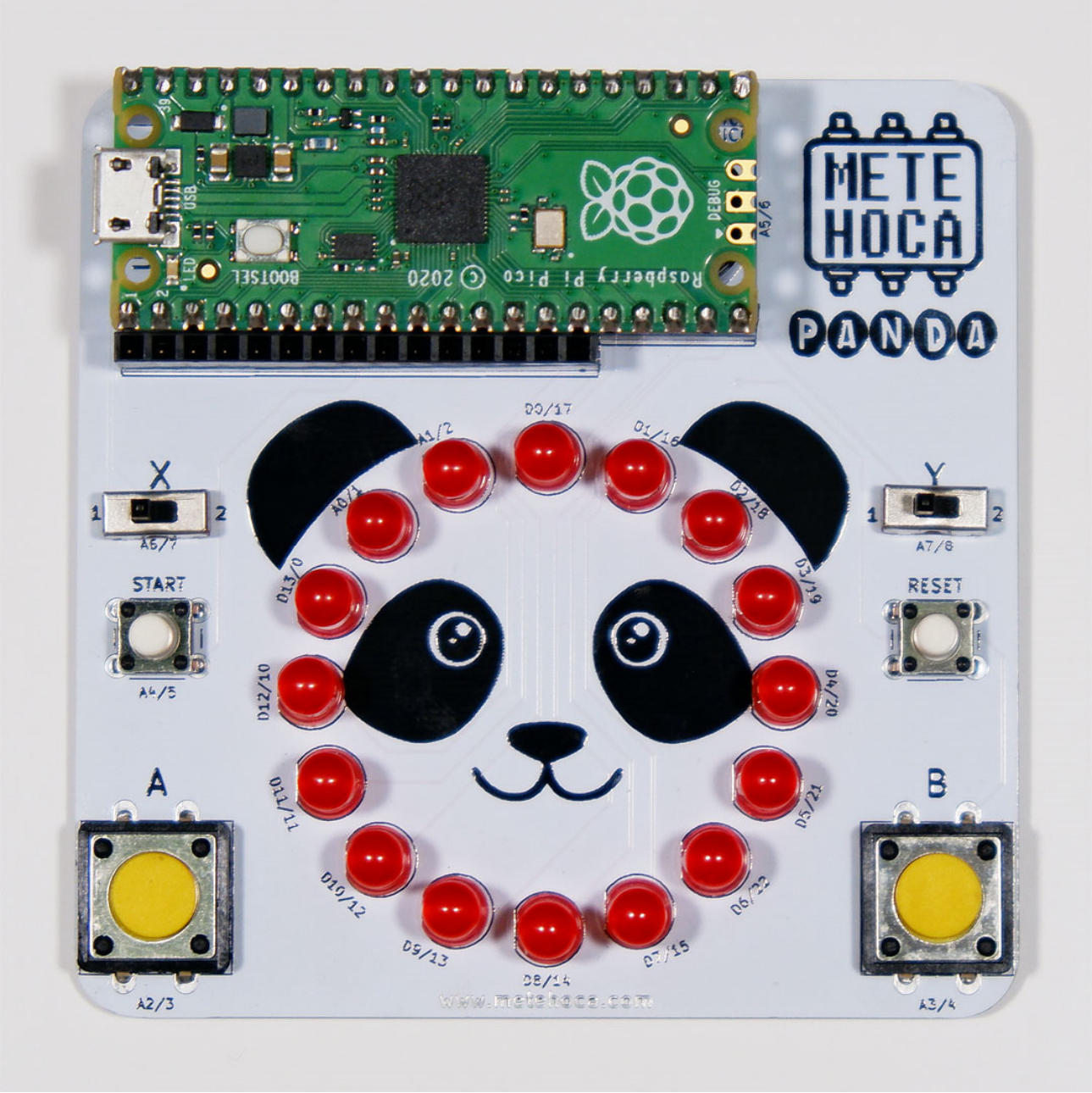
KULLANIRKEN NELERE DİKKAT EDİLMELİDİR?

Mete Hoca Panda, her elektronik cihaz gibi kısa devrelere karşı hassastır ve çalışırken iletken bir yüzeye konulması, üzerine iletken bir şeyler düşürülmesi veya bir sıvı teması durumunda bağlı olduğu Arduino ile birlikte bozulabilir.

Geliştirme kartları ve Panda asla metal yüzeyler üzerinde kullanılmamalı, üzerine iletken herhangi bir şey değdirilmemelidir.

Panda üzerinde Arduino Nano ve Raspberry Pi Pico için ayrı bağlantı header'ları yer alır. 15'er pinlik iki kısa header Nano'nun bağlantı yuvasıyken, 20 pinlik uzun header'lar ise Pi Pico içindir. Bu geliştirme kartları USB girişi sola bakacak şekilde takılmalıdır.

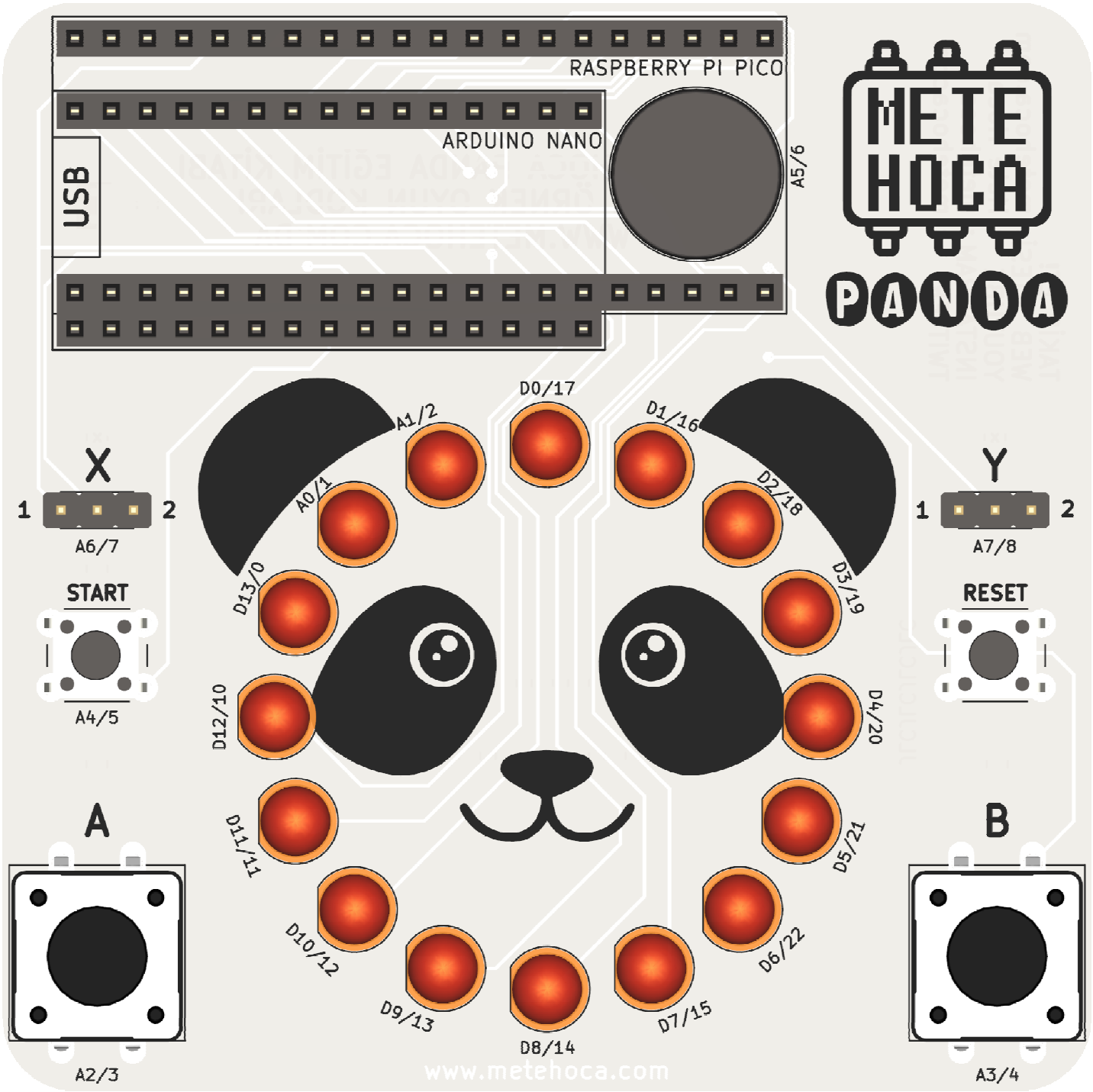
Nano ve Pi Pico'yu Panda üzerine takarken çok dikkatli olunmalı ve pinlerinin eğilmemesine özen gösterilmelidir. Ayrıca tüm pinlerin header üzerine girdiğinden emin olunmalıdır. Yanlışlıkla kaydırarak takmak geliştirme kartı veya Panda'nın bozulmasıyla sonuçlanabilir.



Panda kullanılmadığı zamanlarda geliştirme kartları üzerinden sökülmeli ve zarar görmemesi için paketinde saklanmalıdır. Geliştirme kartını Panda üzerinde takılı bırakmak header soketlerinin zaman içinde gevşeyip temassızlık yapmasına neden olabilir.

PANDA'YI DAHA YAKINDAN TANIYALIM

Merhaba, ben PANDA!



Panda'nın temeli halka şeklinde dizilmiş 16 adet kırmızı LED'den oluşuyor. Bu LED'ler ile döngülü oyunlar veya çeşitli görsel şovlar hazırlamamız mümkün. LED'lerin istediğimiz kadarını (veya hepsini!) aynı anda yakabiliyoruz.

Halka LED'lerimize etkileşim katan ise A ve B olarak isimlendirilmiş ana butonlar. Panda ile yazdığımız oyunları bu butonlar ile oynayabiliriz. Panda'nın iki alt köşesine yerleştirilmiş bu iki buton ile bir veya iki kişilik oyunlar kodlayabiliriz.

Kodladığımız oyunumuz içinde oyun başlatıcı olarak ayarlamamız için kullanabileceğimiz START butonumuz da A ve B butonları gibi istediğimiz gibi programlanabilir.

RESET butonu ise doğrudan geliştirme kartlarının RESET/RUN pinlerine bağlı olduğu için programlanamıyor.

X ve Y harfleri ile kodlanmış 2 anahtar ise oyunumuzun Kolay / Zor veya 1 kişilik / 2 kişilik gibi oyuncu tarafından seçilebilen farklı modlara sahip olabilmesine imkân veriyor. Bu anahtarlar sola çekildiklerinde bağlı oldukları pine HIGH (1) gönderirken, sağa çekildiklerinde ise LOW (0) gönderirler.

Anahtarlar Arduino Nano'da dijital giriş ve çıkış olarak kullanılamayan A6 ve A7 pinlerine bağlıdır. Dolayısıyla bu pinleri HIGH veya LOW olarak okumamız mümkün olmuyor. Nano'da anahtar konumlarını analog değer okuyarak belirleyebiliyoruz.

Analog girişler 0-5 Volt arasını 0-1023 değerleri arasında bir değer olarak okuyup gösterirler. Anahtarlarımız girişe ya 0 Volt (GND), ya da 5 Volt gönderdikleri için okuyacağımız değer ya 0, ya da 1023 (veya 1000 üzeri herhangi bir sayı) olacaktır. Okuduğumuz bu değerlere göre anahtarın hangi konumda olduğunu anlayabiliriz.

Son olarak Panda üzerinde A5 (Nano) veya GP6 (Pi Pico) pinleri ile kontrol edebildiğimiz pasif buzzer'a gelelim. Pasif buzzer, bağlı olduğu pinden gönderilen frekansa bağlı olarak farklı tonlarda sesler çıkarabilen minik bir hoparlördür. Basit yapısı nedeniyle genelde bip şeklindeki uyarı tonları çıkarmaktan çok da öteye gidemez.

Buzzer'ı Arduino IDE ile kodlarken **tone()** komutunu kullanabiliriz. Pi Pico kullanırken MicroPython veya CircuitPython ile programlamak istediğimizde dilin **pwmio** kütüphanesini kullanmamız gerekiyor.

Aşağıdaki tabloda Panda'nın üzerindeki bileşenlerin özelliklerini ve Nano ile Pi Pico üzerinde hangi pinlere bağlı olduklarını görebiliriz. Halka şeklinde dizilmiş LED'ler herhangi bir sıralamaya ihtiyaç duymasa da kod içinde hangi LED'i yakmak istediğimizi karıştırmamak ve dizi içinde kullanırken hangisi olduğu bilmek için en tepedekine 0 ismi verip saat yönünde 15'e kadar girmek hataların önüne geçecektir.

İsim	Kodu	Bağlantı Tipi	Nano Pini	Pi Pico Pini
A Butonu	A	Giriş	A2	GP3
B Butonu	B	Giriş	A3	GP4
START Butonu	START	Giriş	A4	GP5
RESET Butonu	RESET	Resetleme	RST	RUN
Buzzer	-	Çıkış	A5	GP6
X Anahtarı	X	Giriş	A6 (Analog Giriş)	GP7
Y Anahtarı	Y	Giriş	A7 (Analog Giriş)	GP8
LED 0	-	Çıkış	D0	GP17
LED 1	-	Çıkış	D1	GP16
LED 2	-	Çıkış	D2	GP18
LED 3	-	Çıkış	D3	GP19
LED 4	-	Çıkış	D4	GP20
LED 5	-	Çıkış	D5	GP21
LED 6	-	Çıkış	D6	GP22
LED 7	-	Çıkış	D7	GP15
LED 8	-	Çıkış	D8	GP14
LED 9	-	Çıkış	D9	GP13
LED 10	-	Çıkış	D10	GP12
LED 11	-	Çıkış	D11	GP11
LED 12	-	Çıkış	D12	GP10
LED 13	-	Çıkış	D13	GP0
LED 14	-	Çıkış	A0	GP1
LED 15	-	Çıkış	A1	GP2

PANDA'YI ARDUINO NANO İLE KODLAMAK

Panda'yı **Arduino Nano** kullanarak Arduino IDE ile kodlayabiliyoruz. Nano aslında abisi Arduino Uno'nun breadboard veya Panda gibi genişletme PCB'leri üzerine takılabilen **küçültülmüş** halinden başka bir şey değil ve tamamen aynı şekilde kodlanıyor.

Yapmamız gerekenler çok basit: Nano'muzu Panda'ya yerleştirmek, USB kablo ile bilgisayara bağlamak, Arduino IDE'nin Araçlar menüsünden kart olarak Arduino Nano ve Port menüsünden de Nano'muzun COM portunu seçmek…

Klon Nano'ların büyük çoğunluğu eski bootloader yüklenmiş olarak geldiği için Araçlar menüsünden İşlemci seçeneğini **ATmega328P (Old Bootloader)** seçmemiz gerekebilir. Aksi halde kod yüklememiz başarılı olmaz. Nano'muzda yeni bootloader varsa **ATmega328P** seçmeliyiz.

Nano'muzu kodlamaya hazırladıysak Panda üzerindeki LED'leri yakmayı denemekle başlayalım.

LED YAKMAK

Panda'nın halka halinde dizili 16 LED'ini kullanarak çok eğlenceli ışık oyunları yapabiliriz. O zaman ilk LED'imizi yakarak başlayalım.

Panda üzerindeki LED'ler Arduino Nano pinleri ile GND arasına bağlı ve koruma direnci olarak 1 KΩ kullanılıyor. Bir LED'i yakmak istediğimiz zaman bağlı olduğu pine 1, yani HIGH göndermemiz yeterli.

LED'lerin Nano'nun hangi pinlerine bağlı olduğu çevrelerinde yazılı. D0'dan D13'e kadar dijital pinler ile birlikte A0 ve A1 pinlerini kullanarak LED'leri yakabiliriz.

```
int LED1 = 8;                // LED pinini LED1 olarak tanımlıyoruz

void setup() {
  pinMode(LED1, OUTPUT);     // LED pinini çıkış olarak ayarlıyoruz
}

void loop() {
  digitalWrite(LED1, HIGH); // LED'i yakıyoruz
  delay(100);                // Biraz bekleyelim
  digitalWrite(LED1, LOW);  // LED'i söndürüyoruz
  delay(100);                // Biraz bekleyelim
}
```

Yukarıdaki sketch kodunu tanıyor olmalıyız; meşhur Blink! Nano'nun 8. pinine bağlı LED'i kısa aralıklarla yakıp söndürüyor.

Arduino kartlarda dijital pinlerin isimleri D1, D2, D3… şeklinde tanımlansa da kod içerisinde D harfi olmadan kullanılırlar. Analog pinleri dijital çıkış olarak kullanmak için ise A0, A1 biçiminde yazmamız gerekiyor.

Panda üzerinde yer alan 16 LED'i sırayla kontrol ederek çeşitli görsel etkiler elde edebiliriz. Ancak bu kadar fazla LED'i teker teker kontrol edecek kodları hazırlamak uzun, yorucu ve kafa karıştırıcı olabilir. Bu durumlarda işimizi kolaylaştırmak için farklı kodlama yöntemleri kullanabiliriz. Bu tür iş kolaylaştırıcı kod yapılarına ilerleyen bölümlerde değineceğiz.

Şimdi ise LED'leri bildiğimiz temel yöntemlerle yakarak küçük bir görsel animasyon yapalım. Aşağıdaki sketch kodunu deneyelim, bakalım neler olacak… Bu kodu daha da geliştirmek size kalmış.

```
int LED1 = 0;
int LED2 = 4;
int LED3 = 8;
int LED4 = 12;

void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
}

void loop() {
  digitalWrite(LED1, HIGH);
  delay(100);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, HIGH);
  delay(100);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, HIGH);
  delay(100);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, HIGH);
  delay(100);
  digitalWrite(LED4, LOW);
}
```

BUTONLARI KULLANMAK

Sıra geldi butonları kodlamaya. Panda üzerinde 4 adet buton bulunuyor ve bu butonlardan 3'ü kullanıcı tarafından programlanabiliyor. **RESET** butonu Arduino Nano'nun **RST** pinine bağlı ve bastığımızda sketch kodunu en baştan başlatıyor.

A, B ve START olarak isimlendirilen butonlar sırasıyla A2, A3 ve A4 pinlerine bağlı. Butonlar Panda üzerinde kolay ve kafa karıştırmayacak şekilde kullanılabilmesi için PULL-DOWN dirençleri ile kullanılıyor. Yani butona basıldığında bağlı olduğu pin 1 (HIGH) olurken, basılmadığında 0 (LOW) oluyor.

A ve B butonları oyuncular tarafından aktif kullanılacak aksiyon butonları. START ise adından anlaşılabileceği gibi hazırlayacağımız oyunları başlatan buton olarak kullanılabilir.

Aşağıdaki sketch kodu A butonuna basıldığında 8. pine bağlı LED'i yakıyor, çekildiğinde de söndürüyor.

```
int LED1 = 8;                // LED pinini tanımlıyoruz
int ButonA = A2;             // Buton pinini tanımlıyoruz

void setup() {
  pinMode(LED1, OUTPUT);     // LED pinini çıkış olarak ayarlıyoruz
  pinMode(ButonA, INPUT);    // Buton pinini GİRİŞ olarak ayarlıyoruz
}

void loop() {
  if (digitalRead(ButonA)) { // Butona basıldıysa 1 olur, if döngüsü harekete geçer
    // Butona basıldığında çalışacak kod
    digitalWrite(LED1, HIGH);
  }
  else {
    // Buton basılı değilse çalışacak kod
    digitalWrite(LED1, LOW);
  }
}
```

ANAHTARLARI KULLANMAK

Panda, yazdığımız oyunların farklı modlarını seçebilmemiz için iki adet anahtara sahip. Böylece oyunu Kolay/Zor, 1 Oyunculu/2 Oyunculu veya farklı bir kullanımla buzzer sesini kapatıp açabilecek şekilde kodlayabiliriz.

Bu anahtarlar **X** (soldaki) ve **Y** (sağdaki) olarak işaretlenmiş durumda ve anahtarların pozisyonları sağ ve sol taraflarında 1 ve 2 olarak belirtiliyor. Hazırladığımız oyun kodlarının oynanış metinlerini hazırlarken bu işaretlemeleri kullanmamız oyuncunun oyuna hızlıca uyum sağlayabilmesini ve doğru şekilde oynayabilmesini sağlayacaktır.

Panda üzerindeki LED'ler ve butonlar Arduino Nano üzerindeki tüm dijital ve analog pinleri kullanıyor. Anahtarları kullanmak için ise Arduino Uno'da olmayıp Nano'da olan ve sadece analog giriş olarak kullanılabilen A6 ve A7 pinlerine muhtaç kaldık.

Bu 2 pini `digitalRead()` komutuyla okumamız mümkün olmuyor. Bu pinlere buton veya anahtar bağladıysak değeri `analogRead()` komutu ile okuyup okunan değeri kod içinde değerlendirerek buton veya anahtarın konumuna karar vermeliyiz.

Panda üzerindeki 2 konumlu anahtarlar konumlarına göre bu pinlere GND veya 5 Volt veriyor. Anahtarları PCB üzerinde gösterilen 1 (sol) konumuna getirdiğimizde bağlı olduğu pine 5 Volt, 2 (sağ) konumuna getirdiğimizde pine 0 Volt (GND) veriyor. Bu değerleri de `analogRead()` komutu ile okuyup değerlendirmemiz gerekiyor.

Aşağıda örnek bir anahtar kullanımını görebiliriz.

```
int LED1 = 12;                    // LED1 pinini tanımlıyoruz
int LED2 = 4;                     // LED2 pinini tanımlıyoruz

void setup() {
  pinMode(LED1, OUTPUT);          // LED1 pinini çıkış olarak ayarlıyoruz
  pinMode(LED2, OUTPUT);          // LED2 pinini çıkış olarak ayarlıyoruz
}

void loop() {
  int AnahtarX = analogRead(A6); // X anahtarındaki değeri okuyoruz
  if (AnahtarX > 900) digitalWrite(LED1, HIGH);
    else if (AnahtarX < 50) digitalWrite(LED1, LOW);

  int AnahtarY = analogRead(A7); // Y anahtarındaki değeri okuyoruz
  if (AnahtarY > 900) digitalWrite(LED2, HIGH);
    else if (AnahtarY < 50) digitalWrite(LED2, LOW);
}
```

Anahtar 1 konumuna getirildiyse ilgili Arduino pinine 5 Volt gidecektir ve kullandığımız analogRead() komutu **ideal şartlarda** 1023 değerini okuyacaktır. Ancak elektronik dünyasında hiçbir şey mükemmel olmuyor ve ideal ortamı yakalamak düşündüğümüzden daha zor olabiliyor.

Bu yüzden kodlama esnasında biraz esnek davranmalı ve belirli toleranslar çerçevesinde hareket etmeliyiz.

Yukarıdaki sketch koduna dönecek olursak, pinden okunan değer 900'ün üzerindeyse (ideal değer 1023) anahtar 1 konumundadır. Aynı şekilde pinden okunan değer 50'nin altındaysa (ideal değer 0) anahtar 2 konumunda demektir.

BUZZER'I KULLANMAK

Panda'nın üzerindeki bir diğer eğlenceli bileşen buzzer. Panda'da **pasif buzzer** adı verilen ve Arduino'dan göndereceğimiz sinyallere göre istediğimiz frekansta ses çıkarabileceğimiz buzzer kullanılıyor. Pasif buzzer aslında sadece uyarı tonları gibi yüksek frekanstaki sesleri çıkarabilecek şekilde tasarlanmış ucuz ve küçük bir hoparlördür. Özetle çıkarabileceği sesler farklı tonlardaki biip biiiiiip'lerden ileri gidemez.

Bir oyun kodu yazıyorsak eğlencemize sesleri, özellikle farklı tondaki uyarı seslerini de dâhil etmek isteyebiliriz. Oyunu kazanınca duyulan iki veya üç kısa bip veya kaybedince bunu herkese duyuran biraz daha kalın tonda uzun bir biiiiiip! sesi oyun deneyimimizi artırır.

Panda üzerindeki buzzer Arduino'nun A5 pinine bağlanmış durumda. Arduino IDE bu tür buzzer'ları kullanabilmemiz için bize çok basit bir komut veriyor: **tone()**. Bu komut ile istediğimiz pine bağlı pasif buzzer'dan istediğimiz frekansta ses çıkarabiliyoruz.

O zaman lafı fazla uzatmadan örneğimize geçelim.

```
int Buzzer = A5;            // Buzzer pinini tanımlıyoruz
int ButonA = A2;            // A butonu pinini tanımlıyoruz

void setup() {
  pinMode(Buzzer, OUTPUT); // Buzzer pinini çıkış olarak ayarlıyoruz
  pinMode(ButonA, OUTPUT); // A butonunun pinini çıkış olarak ayarlıyoruz
}

void loop() {
  if (digitalRead(ButonA)) tone(Buzzer, 500); // A basılı: Buzzer'ı 500 Hz'de öttür!
    else noTone(Buzzer);                      // Buton basılı değil, Buzzer'ı sustur
}
```

Arduino'nun tone() komutunun kullanımını yukarıdaki sketch kodunda görüyoruz. Tone komutuna sadece buzzer'ın bağlı olduğu pin ve istenen ses frekansını giriyoruz.

```
tone(Buzzer'ın bağlı olduğu pin, ses frekansı);
```

Ses frekansı olarak 31 ile 8.000.000 arasında bir değer girebiliriz. İnsan kulağı 20.000 Hz'den fazlasını duyamadığı için üst değeri fazla zorlamaya gerek yok. Pratikte buzzer'dan duymak isteyeceğimiz ses frekansı aralığı 200-3000 arasında bir yerlerde olacaktır. 1000 Hz ise en yaygın kullanılan uyarı tonudur.

tone() komutu ile başlattığımız bipleme biz yeni bir kod ile durdurana kadar devam eder. Bunun için **noTone()** komutunu kullanmamız gerekiyor. Bu komut gösterilen pindeki buzzer'ı susturur.

```
noTone(Buzzer'ın bağlı olduğu pin);
```

Kod içerisinde kısa bir biplemeye ihtiyacımız varsa bu süreyi tone() ile noTone() komutları arasına **delay()** komutu ile bekletme koyarak yapabiliriz. Aşağıdaki kod parçacığında buzzer'dan 1 saniye (1000 milisaniye) aralıklarla 2000 Hz frekansında bir bip sesi elde ediyoruz.

```
void loop() {
  tone(Buzzer, 2000);
  delay(1000);
  noTone(Buzzer);
  delay(1000);
}
```

PANDA'YI KODLAMAK İÇİN PRATİK YÖNTEMLER

LED'LERİ DİZİ (ARRAY) İLE KONTROL ETMEK

Panda'nın halka şeklinde dizilmiş 16 LED'ini kullanarak oyunlar yapmak istediğimiz zaman LED'lere teker teker erişmenin kolay olmadığını fark edeceğiz. Bunu çözmek için programlama dillerinin temel taşlarından **dizi**lere başvurmalıyız.

Dizinin ne olduğu ile ilgili kısa bir bilgi ile başlayalım. En basit anlatımla dizi, içinde birden fazla değer tutabilen ve bu değerlere sırayla erişebilmemizi sağlayan geniş kapsamlı bir değişkendir.

Panda'da 16 adet LED pini değişkeni tutmak yerine tümünü tek bir dizi değişkeni içine yerleştirip istediğimiz LED'i kolayca kontrol edebiliriz.

```
const byte LED[16] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,A0,A1};
```

Panda oyun kodu örneklerinin çoğunda yukarıdaki yapıyla karşılaşabiliriz. Köşeli parantezlerinin arasına baktığımızda 16 adet değer tutacağını anladığımız **LED** dizisi içinde Panda üzerindeki LED'lerin pin numaraları yer alıyor.

Kod içinde içeriği hiç değişmeyeceği için başına **const** eklediğimiz bu değişken 255'ten büyük sayı içermediği için tipini genelde kullanılan **int** yerine **byte** yaptık. Böylece Arduino Nano'nun kısıtlı hafızasını tutumlu bir şekilde kullanıyoruz.

```
digitalWrite(LED[8], HIGH);
```

İstediğimiz LED'i kontrol etmemiz işte bu kadar kolay. Yukarıdaki kod dizinin 8. LED'ini yakıyor. Dizilerde içerik sayısı 0'dan başlayarak sayılır. Yani en son LED olan A1'e ulaşmak için LED[15] kodunu kullanmalıyız.

Elbette yine teker teker erişeceksek dizi kullanmamızın hiçbir faydası yok. Dizinin bize katacağı avantajı aşağıdaki kod parçacığında görebiliriz.

```
for (int x = 0; x < sizeof(LED); x++) pinMode(LED[x], OUTPUT);
```

Bu tek satırlık kod ile LED pinlerinin hepsini tek seferde çıkış olarak ayarlayabiliyoruz. Muhteşem, değil mi?

Yukarıdaki kodda `sizeof(LED)` komutunu kullandık. Bu komut dizinin sahip olduğu değişken sayısını bize döndürür. Bu durumda bu komut 16 değerini döndürecektir. Bu komut yerine Panda'da doğrudan 16 da kullanabilirdik, ancak bu kez `sizeof(LED)` komutunu öğrenemezdik **;)**

Dizilerin içerik sayısının 0'dan başlaması sizeof() komutu ile okuduğumuz değer'i doğrudan kullanamayacağımız anlamına gelir. Yani LED[16] diye bir şey olamaz. Bu tür kullanımlarda dikkatli olmak zorundayız.

```
for (int x = 0; x < sizeof(LED); x++) digitalWrite(LED[x], HIGH);
```

Pinleri toplu halde çıkış yaptıktan sonra sıra tüm LED'leri yakmaya geldi. Yukarıdaki kod satırı tek satırda 16 LED'i birden yakıyor. digitalWrite() komutu içindeki HIGH'ı LOW yaparsak aynı şekilde söndürebilir de!

Gördüğümüz gibi diziler aynı amaca sahip olan ve sırayla ihtiyacımız olacak değişkenlerimizi saklamamız ve kolaylıkla kullanmamız için bize çok yardımcı oluyor. Panda üzerinde bulunan LED'ler tam da bu fonksiyona ihtiyaç duyuyorlar. Panda ile başlayarak kodlamada sıklıkla ihtiyaç duyulan dizilerin nasıl kullanıldığına aşina olmak gelecekteki projelerimizde çok daha başarılı olmamızı sağlayacak.

O zaman diziler hakkında tüm bu öğrendiklerimizi bir araya getirerek görsel bir etki yaratan bir kod yazalım.

```
const byte LED[16] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,A0,A1};

void setup() {
  for (int x = 0; x < sizeof(LED); x++) pinMode(LED[x], OUTPUT);
}

void loop() {
  for (int x = 0; x < sizeof(LED); x++) { // LED'leri sırayla yakalım
    digitalWrite(LED[x], HIGH);
    delay(100);
  }
  for (int x = 0; x < sizeof(LED); x++) { // Şimdi de sırayla söndürelim
    digitalWrite(LED[x], LOW);
    delay(100);
  }
}
```

Yukarıdaki sketch kodunu ilk gerçek Panda projemiz olarak görebiliriz. LED pinlerini bir dizi içine tanımladık, hepsini toplu halde çıkış yaptık ve ardından göze hoş gelecek bir döngü halinde sırayla yakıp yine sırayla söndürdük.

ANAHTAR OKUMAYI FONKSİYONA YAPTIRMAK

Anahtar kullanımı bölümünde Panda üzerindeki anahtarları farklı bir yöntemle okumamız gerektiğinden bahsetmiş ve bu kullanım yöntemini açıklamaları ile birlikte örnek kod ile deneyimlemiştik. Şimdi bu biraz karmaşık görünen yöntemi basitleştireceğiz.

Arduino IDE sürekli tekrarlanacak komut dizilerini fonksiyon adı verilen alt bölümlere taşıyıp istediğimiz zaman çağırabilmemize izin veriyor. Biz de anahtar okuma kodlarımızı bir fonksiyon haline getireceğiz.

```
int AnahtarX, AnahtarY;
int LED1 = 12;     // LED1 pinini tanımlıyoruz
int LED2 = 4;      // LED2 pinini tanımlıyoruz

void setup() {
  pinMode(LED1, OUTPUT);  // LED1 pinini çıkış olarak ayarlıyoruz
  pinMode(LED2, OUTPUT);  // LED2 pinini çıkış olarak ayarlıyoruz
}

void loop() {
  AnahtarOku(); // Kodumuzda kullanmadan önce anahtar konumlarını okuyalım
  if (AnahtarX = 1) digitalWrite(LED1, HIGH); // X anahtarı 1 ise LED1'i yakalım
    else digitalWrite(LED1, LOW);    // Anahtar 1 değilse 2'dir, LED1'i söndürelim.
  if (AnahtarY = 1) digitalWrite(LED2, HIGH); // Y anahtarı 1 ise LED1'i yakalım
    else digitalWrite(LED2, LOW);    // Anahtar 1 değilse 2'dir, LED1'i söndürelim.
}

void AnahtarOku() {
  AnahtarX = analogRead(A6);
  if (AnahtarX > 900) AnahtarX = 1; else if (AnahtarX < 50) AnahtarX = 2;
  AnahtarY = analogRead(A7);
  if (AnahtarY > 900) AnahtarY = 1; else if (AnahtarY < 50) AnahtarY = 2;
}
```

AnahtarOku() adını verdiğimiz fonksiyonumuzu en alta yerleştirdik ve void loop() döngüsü içinde her ihtiyacımız olduğunda tek satırda çağırıp güncel anahtar konumlarını okuyabiliyoruz. Ondan sonra tek yapmamız gereken kendi kodumuz içindeki ihtiyaca göre istediğimiz yönlendirmeleri gerçekleştirmek.

Fonksiyonları iyi öğrenip kullanmanın kolay okunur, kolay düzenlenir ve az hata çıkaran kodlar yazmak için oldukça önemli olduğunu hiç unutmayalım.

BİR TUŞA BASILANA KADAR BEKLEMEK

Panda ile hazırladığımız oyunu oynuyoruz, oyun bitti ve kazanan taraf belli oldu. Bunu bir şekilde LED'ler ile göstermeli ve yeni oyuna başlamadan önce herkesin kazananın kim olduğunu açıkça görebilmesi için beklemeliyiz.

Peki, bu bekleme ne kadar olmalı? 1 saniye? 5 saniye? 10 saniye? Bir oyun kodlarken kararları bizim almamız yerine oyunculara bırakmak her zaman daha iyi bir oyun deneyimi sunar. Bu yüzden oyunun bittiğini gösteren durumu kurduktan sonra oyuncuların oyuna yeniden başlamadan önce bir tuşa basana kadar beklemek en doğrusu olacaktır.

Bunu **while()** komutu ile yapacağız. Bu komut, parantezi içindeki kontrol şartı gerçekleşene kadar içindeki kodları çalıştırır. Bu komutu süslü parantezlerle veya parantez olmadan tek başına kullanabiliriz.

O zaman while() komutunun iki farklı kullanımını doğrudan tek bir sketch kodu üzerinde anlatalım.

```
int LED = 8;
int ButonA = A2;
int ButonS = A4;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(ButonA, INPUT);
  pinMode(ButonS, INPUT);
}

void loop() {
  while(!digitalRead(ButonA)) { // LED, A butonu basılana kadar yanıp sönecek
    digitalWrite(LED, HIGH);
    delay(100);
    digitalWrite(LED, LOW);
    delay(100);
  }
  while(!digitalRead(ButonS)); // START butonuna basılana kadar bekle...
}
```

Yukarıdaki kodda ilk `while()` döngüsü A butonunun konumuna bağlı olarak çalışıyor. Kodu kısaltmak ve pratik kullanımı için butonu okuyan `digitalRead()` komutunu doğrudan parantezler arasına yerleştirdik. Kodumuz içinde kontrol etmemiz gereken şey bir değişken ise onu `while(x = 4)` şeklinde de kullanabiliriz.

Kodumuzdaki `digitalRead()` komutunun önündeki ünlem! işareti dikkatimizi çekmiş olmalı. Bu ünlem işareti çoğu programlama dilinde önüne geldiği boolean tipi değeri terslemek için kullanılır. Yani bu durumda A butonundan okunan değer 1 ise 0, 0 ise 1 yapıyor.

Biz süslü parantezler içindeki kodumuzun butona basılana kadar çalışmasını istiyoruz. Ancak kodun çalışması için `while()` içeriğinin 1 olması gerekiyor. Bu yüzden kodun başına ünlem! ekliyor ve amacımıza ulaşıyoruz. Butona basıldığında `digitalRead()` komutu pinden 1 okuyor, ünlem hemen bu değeri 0 yapıyor ve `while()` döngüsünün şartı gerçekleşmemiş oluyor. Böylece `while()` döngüsünden çıkılıyor.

A butonuna basıp LED'i yakıp söndüren döngüden çıkıldıktan sonra kodumuz altta bulunan `while()` döngüsünde takılı kalıyor. Bu kez ButonS adıyla kullandığımız START butonuna basılana kadar bekliyor ve bu bekleme esnasında herhangi bir kod çalıştırmıyor. O zaman süslü parantezlere de gerek yok.

Panda ile oyun kodları yazarken `while()` ve tersleme ünlemini sık sık kullanmamız gerekecek. Bu yüzden bu iki komutun nasıl çalıştığını anlamak için yukarıdaki örneği iyi anlamak, yayınlanmış örnek Panda oyun kodlarındaki kullanımlarını incelemek şiddetle tavsiye edilir.

BİR DÖNGÜ İÇİNDE BUTONA BASILIP BASILMADIĞINI KONTROL ETMEK

Panda ile oyun kodları yazarken ihtiyacımız olacak en önemli şeylerden biri belirli bir kod döngüsü devam ederken bir butona basılıp basılmadığını anlayabilmek oluyor. LED'lerin yanıp söndüğü, delay() komutlarının havada uçtuğu bir döngüde butona herhangi bir an basılabilir ve bunu o an algılayıp harekete geçmek veya döngü bittiğinde buna göre bir işlem yapmak gerekiyor.

Bu durumda butonun basılabileceği tüm bekleme noktalarına, yani delay() komutlarına bir kontrol yerleştirmemiz gerekiyor. 1000 ms bekleyecek bir delay() komutunun içine buton kontrolü nasıl yerleştirilir? Aslında çok basit: neden bu 1000 ms'lik bekleme kodunu 1 ms'lik 1000 parçaya bölmüyoruz? Her bir parçanın arasına da butonun basılıp basılmadığını kontrol edecek kodumuzu ekleyebiliriz.

O zaman bunu nasıl yapabileceğimizi örnek bir sketch kodu üzerinde inceleyelim.

```
int LED1 = 8;
int LED2 = 0;
int ButonA = A2;
int Basildi = true;

void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(ButonA, INPUT);
}

void loop() {
  digitalWrite(LED1, HIGH);
  for (int x = 0; x < 1000; x++) {
    if (digitalRead(ButonA) == 1) Basildi = true;
    delay(1);
  }
  if (Basildi == 1) {
    digitalWrite(LED2, HIGH);
    Basildi = false;
  }
  else digitalWrite(LED2, LOW);
  digitalWrite(LED1, LOW);
  delay(1000);
}
```

Yukarıdaki kodun Panda'nın 8. pinine bağlı LED'i 1 saniye (1000 ms) aralıklarla yakıp söndüren basit bir blink kodu olduğunu fark etmiş olmalıyız. Özellikle LED'in söndürüldüğü ikinci bölüm tamamen aynı. Ancak ilk bölümün 1 saniyelik beklemesi bir **for** döngüsü ile değiştirilmiş.

0'dan 1000'e kadar saydığını gördüğümüz bu for döngüsünün içinde A butonuna basıldığında **Basildi** adındaki bir değişkeni true (ya da 1) yapan bir if kontrolü yer alıyor. Hemen altında da 1 milisaniye bekleme sağlayan `delay(1)` komutu bulunuyor.

Bu for döngüsü toplamda 1000 ms'lik bir bekleme oluşturuyor, ancak bunu yaparken boş durmuyor ve her 1 ms'de bir kez butonun basılı olup olmadığını kontrol ediyor. Butona basıldıysa Basildi değişkenini **true** yapıyor ve döngüden çıkıldığında döngü sürecinde butona basılıp basılmadığını anlamak mümkün oluyor. Basildi değişkenini okuduktan sonra kapatmayı, yani **false** olarak değiştirmeyi unutmamalı.

PANDA'YI RASPBERRY PI PICO İLE KODLAMAK

Şimdiye kadar tüm çalışmalarımızı Arduino Nano ile yaptık. Şimdi sıra Panda'yı **Raspberry Pi Pico** ile kullanmaya geldi. Neyse ki Arduino ekibi oldukça hızlı çalışıyor ve Pi Pico desteğini getirmeleri çok uzun sürmedi. Artık Pi Pico'yu da kolayca **Arduino IDE**'ye tanıtabiliyor ve Nano gibi kolayca kodlayabiliyoruz.

Pi Pico aslında Nano'dan çok daha yetenekli bir kart. Fiyatının klon Nano'lardan %30-50 kadar daha düşük olduğunu düşünecek olursak tam bir **fiyat/performans canavarı** olduğunu söyleyebiliriz.

Pi Pico'nun bir diğer özelliği de **CircuitPython** veya **MicroPython** dilleri ile de kodlanabiliyor olması. Yani istersek Arduino IDE ile Nano'dan alışık olduğumuz şekilde kodlayabilir, istersek popüler dillerden Python ile kodlamayı seçebiliriz.

Başlarda sadece Nano ile kullanılması planlanan Panda'ya tasarım sürecinin ortalarında Pi Pico'nun eklenme sebebi biraz da bu. Kodlamak için Pi Pico kullanırsak her iki ana programlama dili üzerinde de çalışmalar yapabilir oluyoruz.

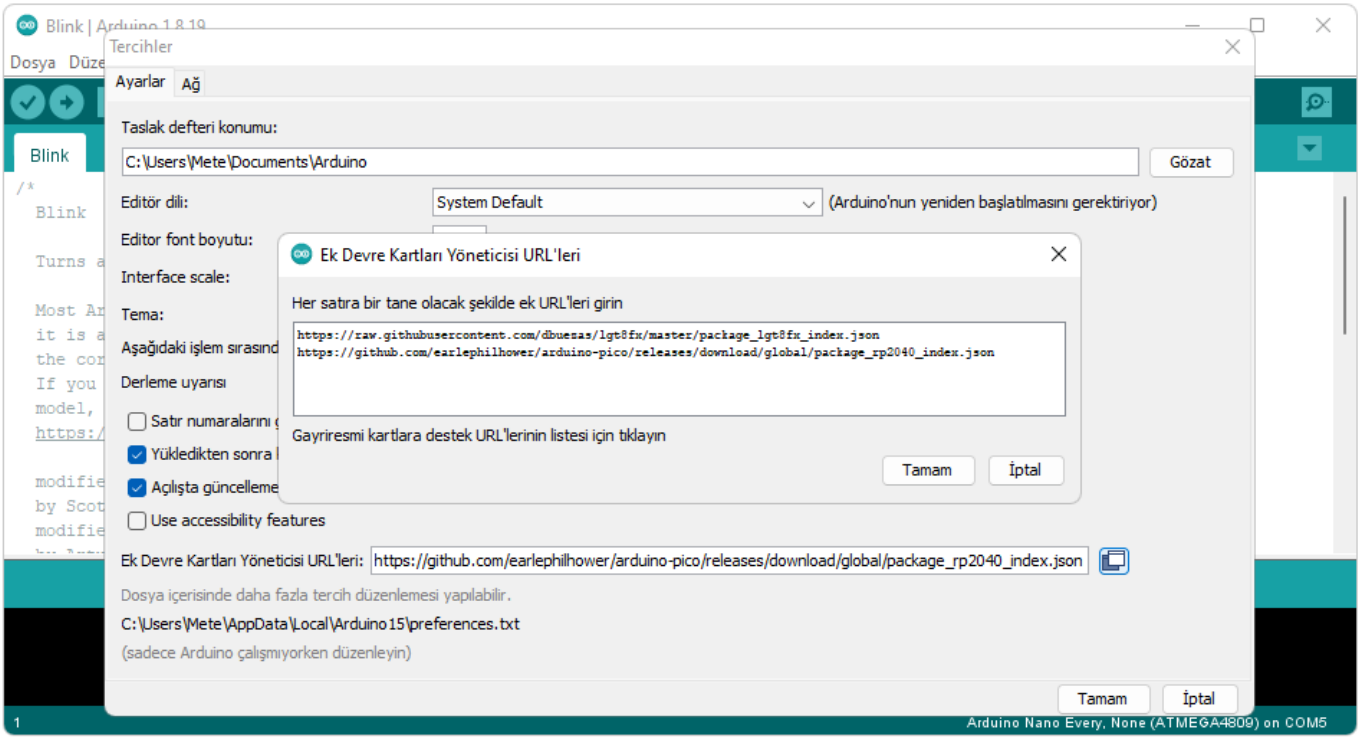
RASPBERRY PI PICO'YU ARDUINO IDE'YE TANITMAK

Pi Pico'yu Arduino IDE'ye tanıtmak alışık olduğumuz kart seçiminden biraz daha karmaşık. Bu işlem için üçüncü parti bir eklenti kurmamız gerekiyor.

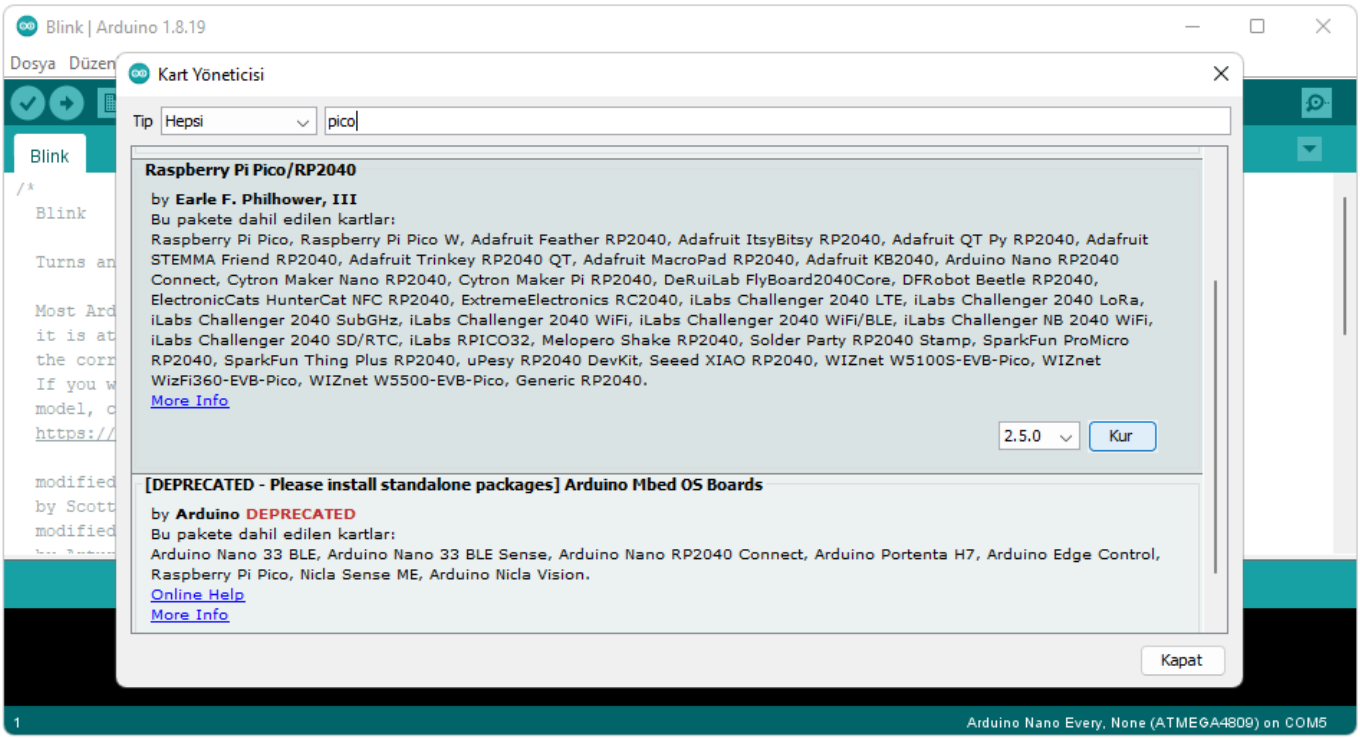
Bu eklentiyi kurmak için önce harici eklenti linkini Arduino IDE'ye girmemiz gerekiyor. Bunu da **Dosya** > **Tercihler** tıklayarak açılan pencerenin alt kısmındaki Ek Devre Kartları Yöneticisi URL'leri bölümünün sağındaki küçük butona tıklayarak yapıyoruz.

Açılan pencereye aşağıdaki linki yapıştırıyoruz. Listede başka linkler varsa her bir linki yeni satıra yapıştıralım.

https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json

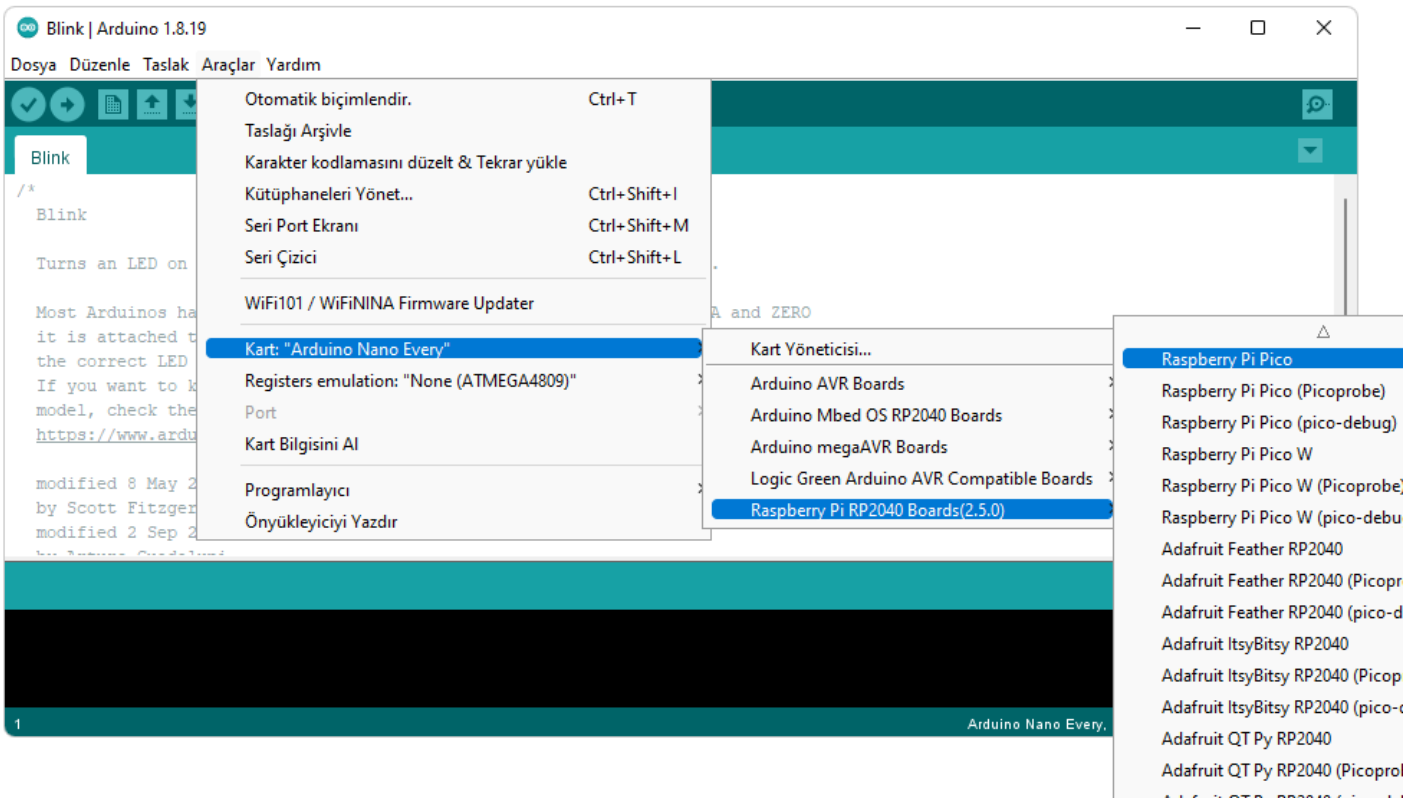


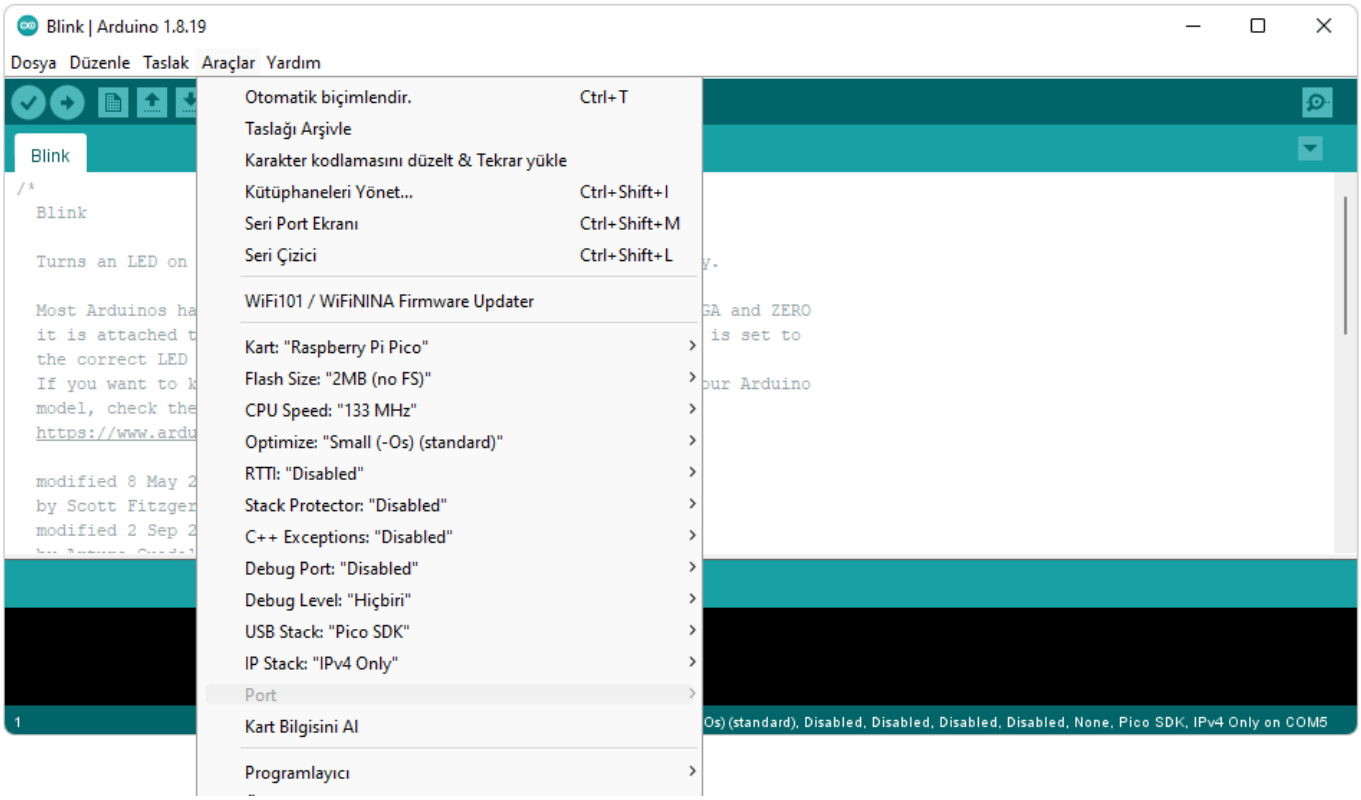
Arduino IDE'nin ana menüsünde **Araçlar** > **Kart** > **Kart Yöneticisi** seçerek ekleyebileceğimiz kartların listesini açalım. Arama kısmına **pico** yazmamız ihtiyacımız olan paketi bulmamız için yeterli. Önümüze gelen listede **Earle F. Philhower III** tarafından hazırlanmış **Raspberry Pi Pico/RP2040** paketini seçip **Kur** butonuna tıkladığımızda gereken eklentiyi indirecek kurulum başlayacaktır.



Kurulum tamamlandığında paket versiyonunun yanında yeşil renkte **INSTALLED** yazacak. Artık bu pencereyi kapatabiliriz.

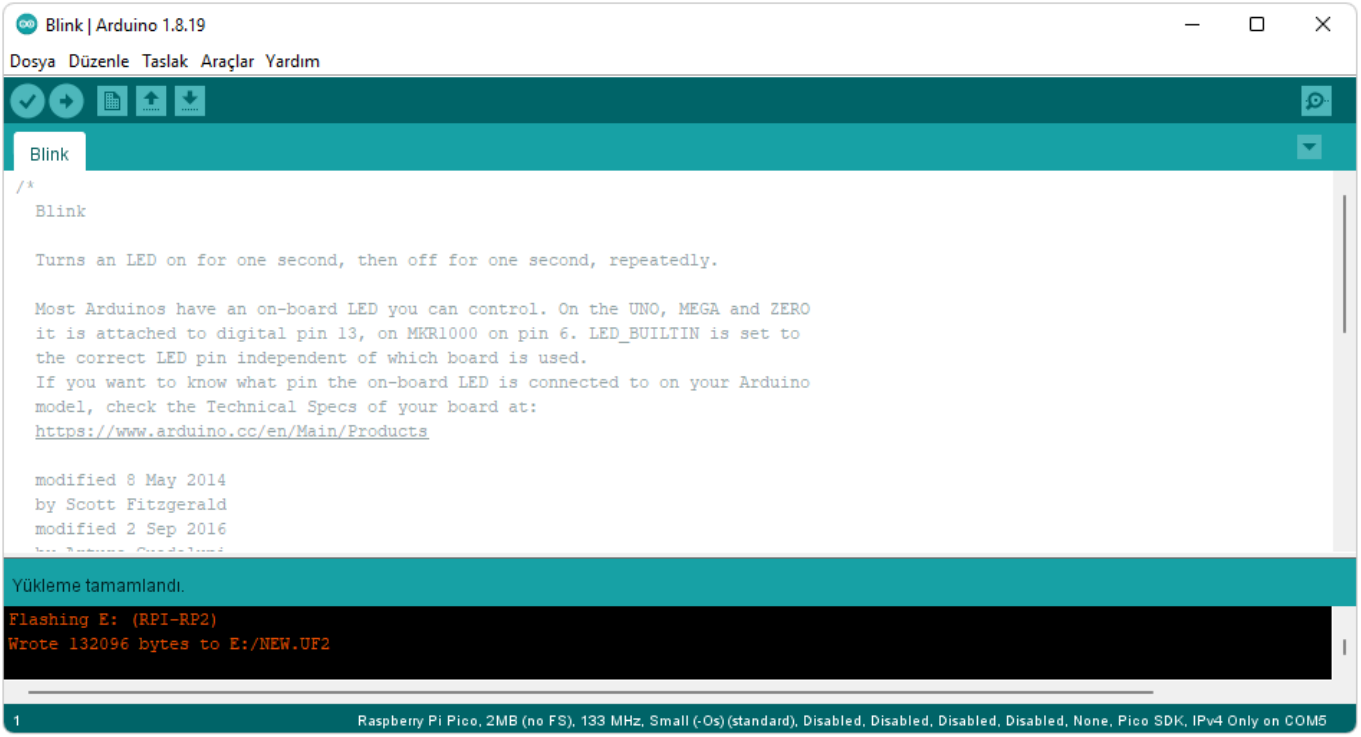
Bundan sonra yapmamız gereken şey **Araçlar** > **Kart** menüsünün altına gelmiş olan **Raspberry Pi RP2040 Boards** menüsünün altından **Raspberry Pi Pico**'yu seçmek.





Pi Pico'yu Arduino'da ilk kez kullanıyorsak Araçlar menüsünde seçebileceğimiz bir portunun olmadığını göreceğiz. Çünkü Pi Pico'nun içinde henüz Arduino'nun temel yazılımı yok ve bu yüzden bir seri port göremiyoruz. Pi Pico'ya ilk yükleyeceğimiz sketch öncesinde Arduino IDE gerekli temel yazılımı Pi Pico'ya kuracak ve bir seri porta sahip olmasını sağlayacak. İlk sketch kodumuzu yüklemek için fazla beklememize gerek yok. Eski dostumuz Blink sketch'ini açıp yükle butonuna bastığımızda kod yüklemesi öncesinde gerekli işlemler yapılacak ve bir seri portumuz olacak.

Aslında Arduino ekibi de Pi Pico'yu doğrudan destekleyen bir yazılım eklentisine sahip, ancak bu eklentinin henüz tone() gibi temel fonksiyonlarda ciddi sorunları var ve bu sorunlar çözülene kadar **Earle Philhower**'ın yazılımını kullanmamız sorun yaşamamamız için önemli.

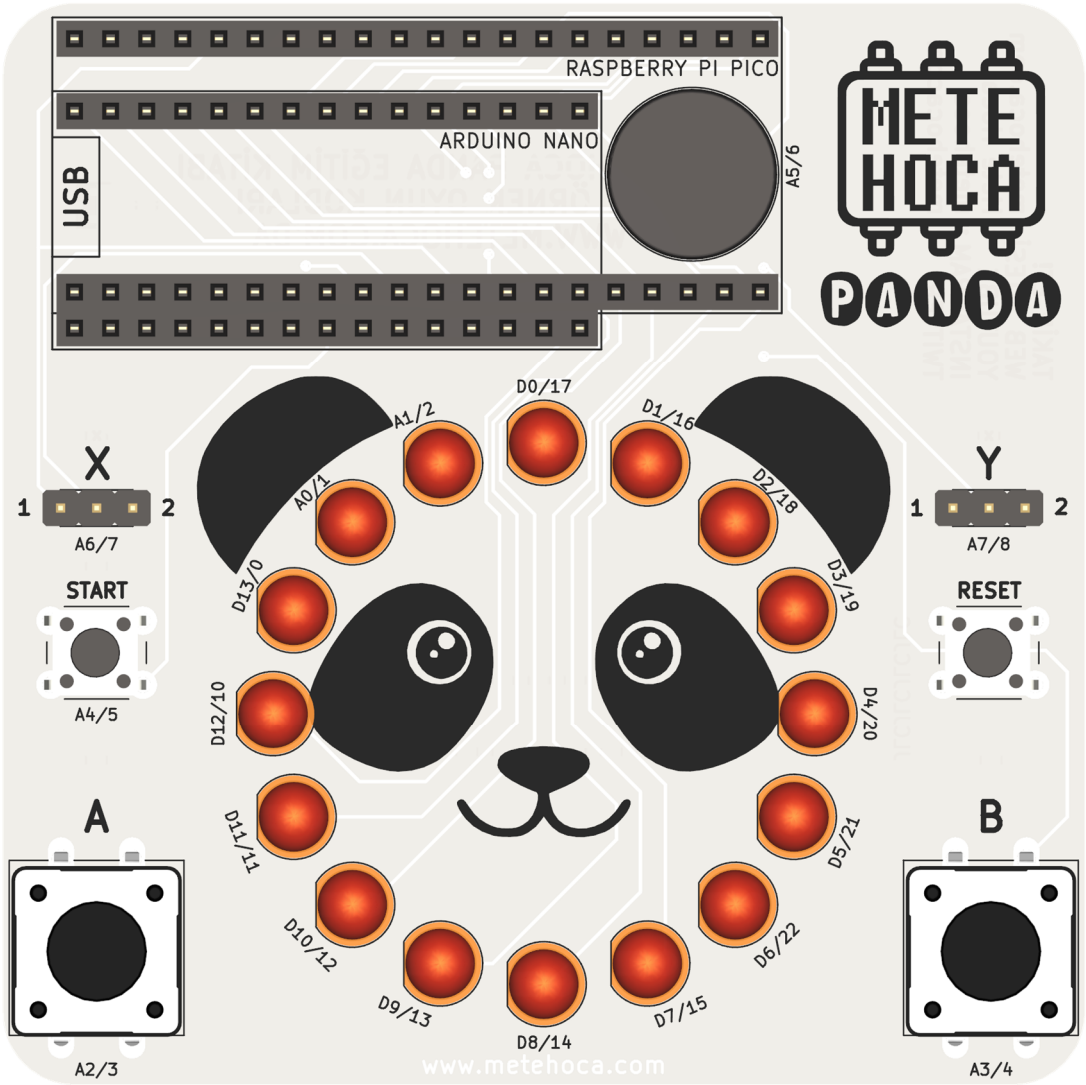


RASPBERRY PI PICO'YU ARDUINO IDE'DE KULLANMAK

Pi Pico'yu bir kez Arduino IDE'ye tanıttıktan sonra kodlama şekli birkaç küçük detay dışında Arduino Nano ile tamamen aynı. Pinleri yine aynı şekilde giriş veya çıkış olarak tanımlıyoruz, pine çıkış vermek veya pini okumak için yine aynı Arduino komutlarını kullanıyoruz.

Pi Pico'nun alt kısmında pin isimlerinin GP1, GP2… şeklinde yazıldığını görüyoruz. Arduino IDE'de pinleri tanımlarken pin isimlerini bu şekilde yazmıyoruz. Örneğin GP17 için sadece 17 yazmalıyız.

Panda üzerindeki bileşenlerin Pi Pico'nun hangi pin çıkışlarına bağlı olduğunun Nano pinleri ile birlikte yazılı olduğundan bahsetmiştik.



Örneğin en tepede duran ilk LED'imizin yanında D0/17 yazılı. Bu yazı, LED'in Nano'nun 0 numaralı pinine ve Pi Pico'nun 17 numaralı pinine bağlı olduğunu işaret ediyor.

Pi Pico'da LED'leri yakmak ve butonları okumak Arduino Nano ile birebir aynı komutlarla yapılıyor. Yani LED yakmak için yine **digitalWrite(pin numarası, HIGH)** kullanırken, butonları okumak için yine **digitalRead(pin numarası)** komutunu kullanıyoruz.

Pi Pico'nun pin sayısı Arduino Nano'dan epey fazla olduğu için Panda üzerindeki anahtarların konumlarını okumak için analog pinlerden analog değerler okumamıza gerek yok. Pi Pico üzerinde 7 ve 8 numaralı pinlere bağlı olan X ve Y anahtarlarının konumlarını doğrudan HIGH veya LOW olarak okuyabiliriz. Pinden okunan değer HIGH ise anahtar 1 konumunda, LOW ise 2 konumunda demektir.

METE HOCA

Projelerinizden **#projezamanı** ile bahsedin!

Twitter: **metehocacom**

Instagram: **metehoca**

Daha fazla bilgi, videolu anlatımlar ve sorularınıza cevaplar için;

www.metehoca.com