



**THE IMAGINATION UNIVERSITY PROGRAMME**

# **RVfpga Deney 2**

## **C Programlaması**

## 1. GİRİŞ

Bilgisayar programlarının çoğu C gibi yüksek düzeyli bir dilde yazılmıştır. Bu deney RVfpga'de çalışabilen bir C projesini PlatformIO'da nasıl oluşturacağını gösterir. İlk olarak bir C programının nasıl oluşturulup çalıştırılacağı üzerine bir öğretici sağlıyoruz. Ardından kendi C programlarını yazmayı uygulamaya dökmen için alıştırmalar tanımlıyoruz.

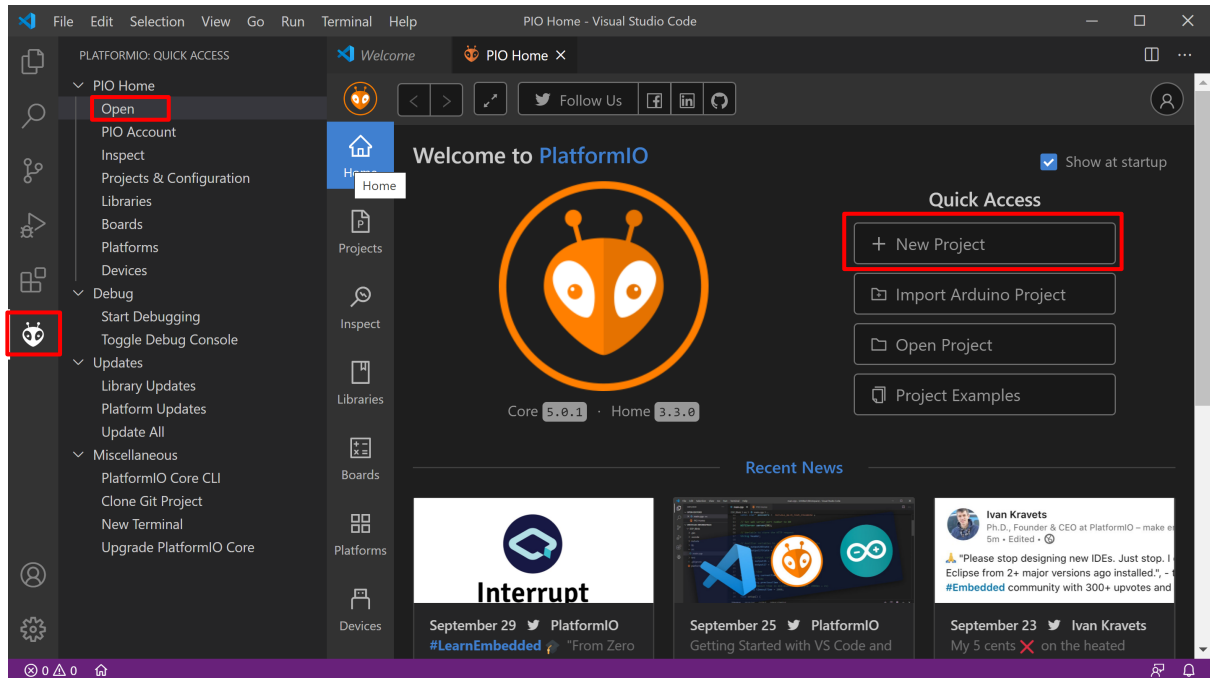
## 2. RVfpga için C Programı

PlatformIO'yu kullanarak RVfpga'de bir C programı oluşturup çalıştırmak için aşağıdaki adımları atacaksınız:

1. Bir RVfpga projesi oluştur
2. Bir C programı yaz
3. RVfpga'i Nexys A7 FPGA kartına indir
4. Bir C programını derle, indir, çalıştır

### Adım 1. Bir RVfpga projesi oluştur

VSCode'u aç (RVfpga İlk Kullanım Kılavuzunda tanımlandığı gibi). Eğer VSCode'u başlattığında PlatformIO kendiliğinden açılmıyorsa sol menü şeridindeki PlatformIO ikonuna tıklayıp ardından Home → Open'a (Figür 1'e göz at) tıkla.

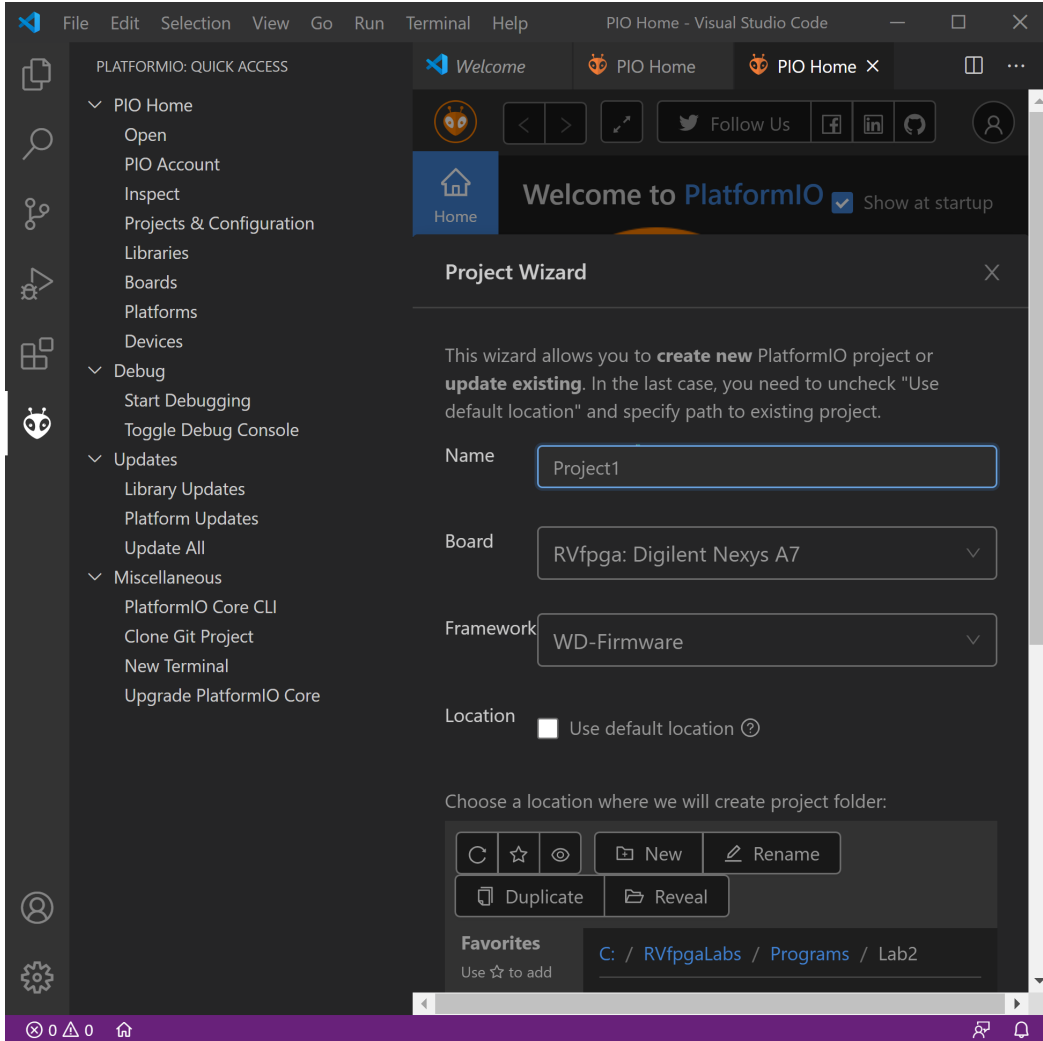


**Figür 1. PlatformIO'yu aç, yeni proje oluştur**

Şimdi PIO Home hoş geldin penceresinde, New Project'e (Yeni Proje) tıkla (Figür 1'e göz at).

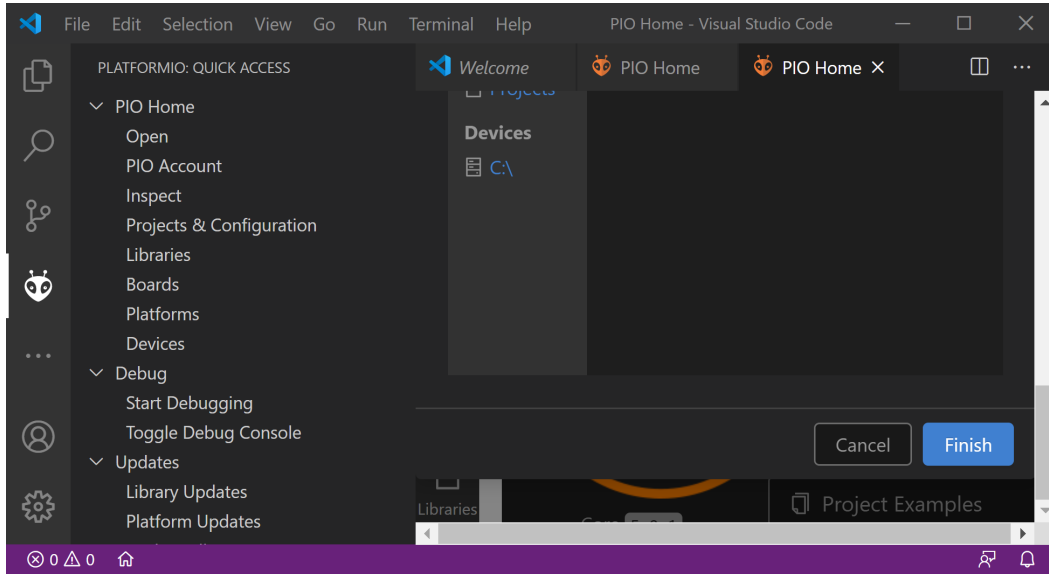
Figür 2'de gösterildiği gibi projeyi Project1 olarak adlandırıp Board'u (Kart) RVfpga: Diligent Nexys A7 olarak seç (RVfpga yazmaya başla, kart seçenek olarak belirecek). Varsayılan çerçeveyi WD-firmware olarak bırak. WD-firmware, bu deneylerde kullandığımız Freedom-E SDK (Yazılım Geliştirme Kiti) gcc, gdb ile PSP, BSP (işlemci destek paketi, kart destek paketi) de içeren Western Digital'in bellenimidir. Use default location'ın (Varsayılan yeri kullan) işaretini kaldırıp projeni şuraya yerleştir:

`[RVfpgaPath]/RVfpga/Labs/Lab2`



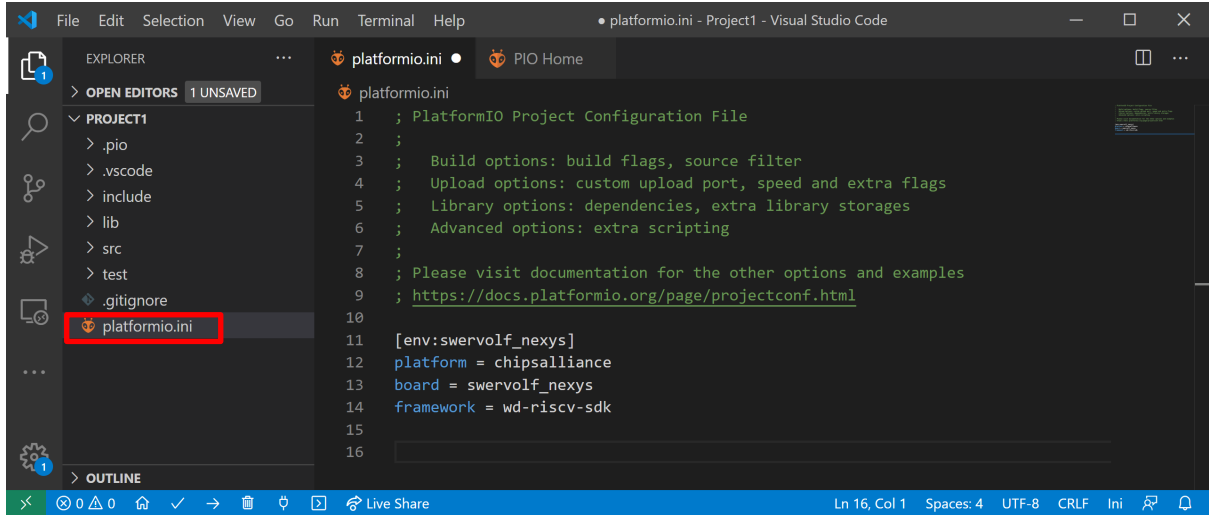
**Figür 2. Projeyi adlandır, kart ile proje klasörünü seç**

Ardından pencerenin aşağısındaki Finish'e (Bitir) tıkla (Figür 3'e göz at).



**Figür 3. Proje oluşturmaya bitir**

Soldaki Explorer (Gezgin) panelinde (genişletmen gerekebilir), platformio.ini'ye açmak için çift-tıkla (Figür 4'e göz at). Bu PlatformIO ilk değerlendirme dosyasıdır.

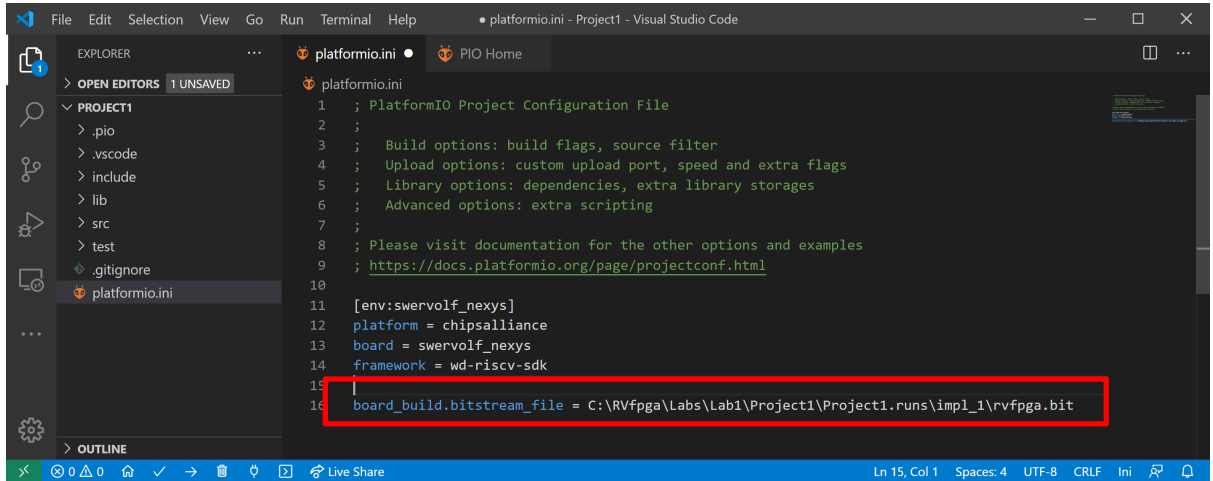


**Figür 4. PlatformIO ilk değerlendirme dosyası: platformio.ini**

platformio.ini dosyasına aşağıdaki satırı ekle, Figür 5'te gösterildiği gibi:

```
board_build.bitstream_file =
[RVfpgaPath]/RVfpga/Labs/Lab1/Project1/Project1.runs/impl_1/rvfpga.bit
```

(*[RVfpgaPath]*'i klasörün makinadaki yeriyle değiştirmeyi unutma.) Bu satır FPGA'e yüklemek için PlatformIO'nun veri akışı dosyasını nerede bulması gerektiğini gösterir. Yukarıdaki yol Deney 1'de oluşturduğunuz veri akışının yeri. (Eğer Deney 1'i bitirmediyse, İlk Kullanım Kılavuzu ile dağıtılan şuradaki RVfpga veri akışını kullanabilirsiniz: *[RVfpgaPath]/RVfpga/src/rvfpga.bit*.) platformio.ini dosyasını kaydetmek için Ctrl-s'e bas.

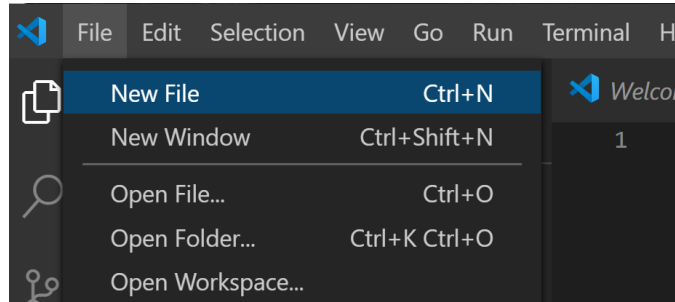


**Figür 5. RVfpga veri akışı dosyasının (rvfpga.bit) yerini ekle**

İlk Kullanım Kılavuzunda kullanılan örneklerde daha bütün bir *platformio.ini* dosyasının kullanıldığını unutma. Ek komutlar gerektiren bir işlevsellik kullanmak istersen (Verilator simütatörüne giden yol, dizisel konsolun yapılandırılması, whisper ayıklama aracı, gibi gibi.), o örneklerdeki *platformio.ini*'yi kullanabilirsin.

## Adım 2. Bir C programı yaz

Şimdi bir C programı yazacaksın. File → New File'a tıkla (Figür 6'ya göz at)



**Figür 6. Projeye dosya ekle**

Boş bir pencere açılacak. Şu C programını o pencereye yaz (ya da kopyala/yapıştır) (Figür 7'ye göz at). Bu program anahtarların değerlerini LEDlerde gösterir.

```
// bellek-eşlenmiş girdi/çıktı adresleri
#define GPIO_SWs      0x80001400
#define GPIO_LEDs     0x80001404
#define GPIO_INOUT    0x80001408

#define READ_GPIO(dir) (*(volatile unsigned *)dir)
#define WRITE_GPIO(dir, value) { (*(volatile unsigned *)dir) = (value); }

int main ( void )
{
    int En_Value=0xFFFF, switches_value;

    WRITE_GPIO(GPIO_INOUT, En_Value);

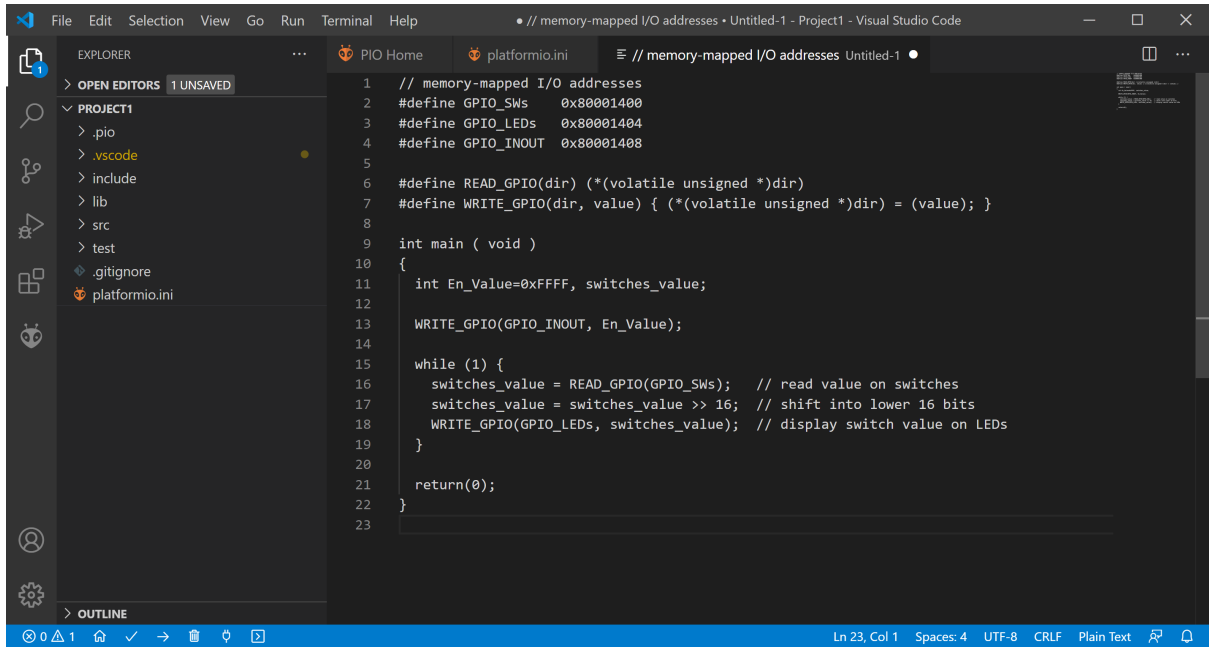
    while (1) {
        switches_value = READ_GPIO(GPIO_SWs);    // anahtarlardaki değerleri oku
        switches_value = switches_value >> 16;    // alt 16 bitlere kaydır
    }
}
```

```
WRITE_GPIO(GPIO_LEDS, switches_value); // anahtar deęerini LEDlerde
göster
}

return(0);
}
```

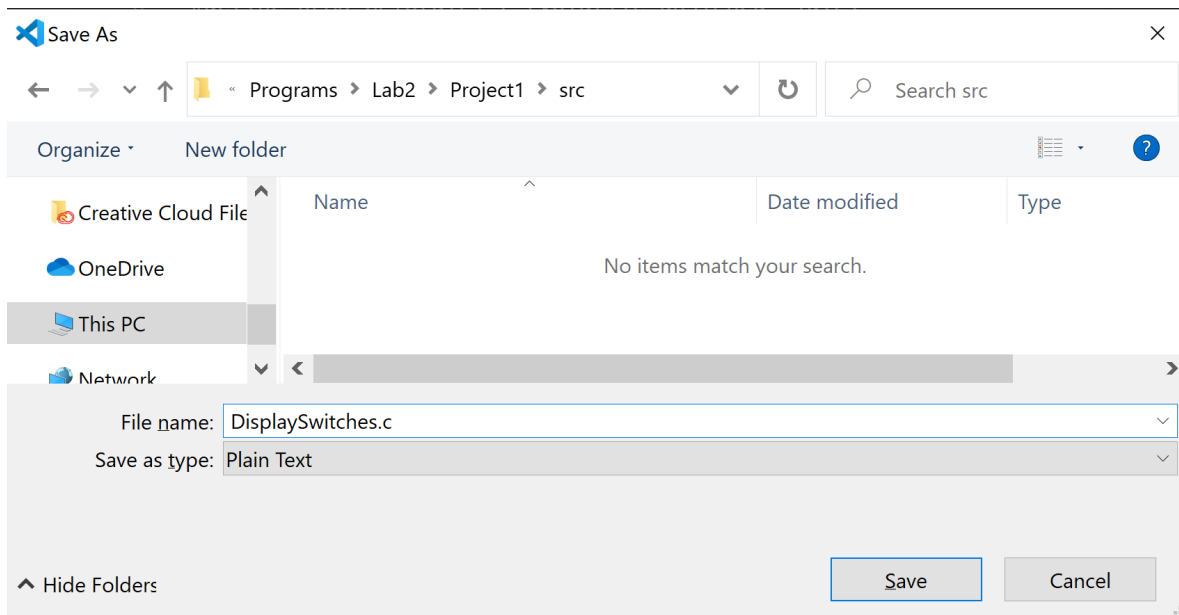
Bu program kolaylık olması için řu dosyada da var:

*[RVfpgaPath]/RVfpga/Labs/Lab2/DisplaySwitches.c*



**Figür 7. C programı gir**

Programı panele girdikten sonra dosyayı kaydetmek için Ctrl-s'e bas. DisplaySwitches.c olarak adlandırıp Project1 dizininin src klasörüne kaydet (Figür 8'e göz at).



**Figür 8. Dosyayı DisplaySwitches.c olarak kaydet**

Bu program ilk olarak Nexys A7 FPGA kartındaki LEDlerle anahtarlara bağlı bellek-eşlenmiş girdi/çıkı yazmaçlarının adreslerini şu satırları kullanarak tanımlar:

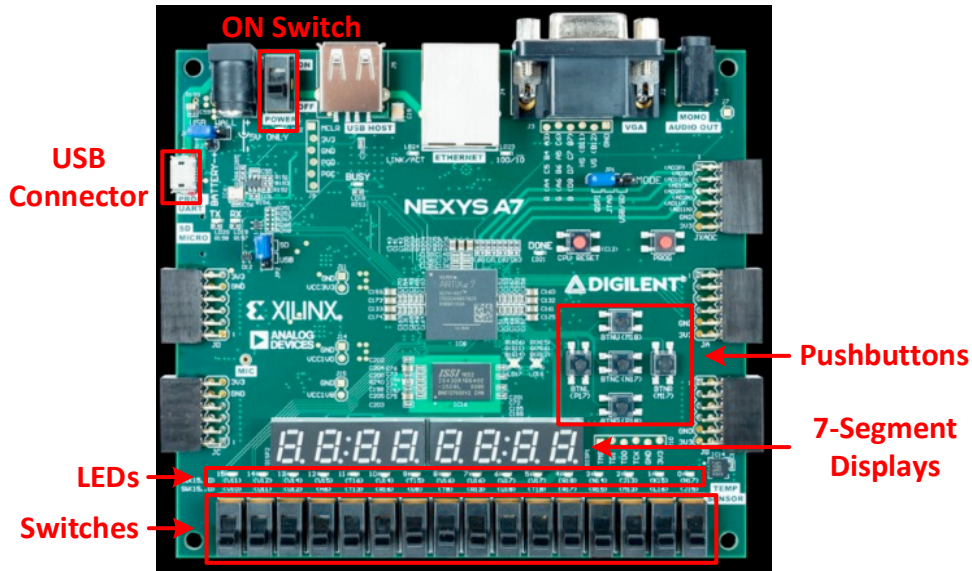
```
#define GPIO_SWs      0x80001400
#define GPIO_LEDs      0x80001404
#define GPIO_INOUT     0x80001408
```

Anahtarların değerleri 0x80001400 adresine eşlenmiş yazmacı okuyarak bulunur, değerler 0x80001404 adresine eşlenmiş yazmacı yazılarak LEDlerde gösterilir. Anahtar değerleri yazmacın üst yarımında, LEDler alt yarımındadır.

GPIO\_INOUT yazmacı genel amaçlı girdi/çıkıttın (GPIO) bir bitinin girdi ya da çıkı olduğu tanımlar. Düşük önemli 16 GPIO uçları, 15:0, Nexys A7 kartındaki 16 LEDe bağlıdır. Yüksek önemli 16 GPIO uçları, 31:16, karttaki 16 anahtar içindir. 0 girdi olduğunu gösterirken 1 çıkı olduğunu gösterir. Dolayısıyla, anahtarların RVfpga'e girdi olması, LEDlerin RVfpga'in çıkıtları olması için GPIO\_INOUT 0xFFFF ile yazılır.

Figure 9 Nexys A7 FPGA kartındaki LEDlerle (LEDs) anahtarların (Switches) yanı sıra USB bağlayıcısının (USB Connector), açma anahtarının (ON Switch), düğmelerin (Pushbuttons), 7-kesimli ekranların (7-Segment Displays) fiziksel yerlerini gösterir.

Önemli olarak Deney 6'da GPIO nitelikleriyle RVfpga GPIO donanımını detaylı olarak tanımlıyoruz. Kartın butonları, 7-kesimli ekranları gibi diğer çevre birimlerinin nasıl kullanıldığını da sonraki deneylerde tartışıyoruz (6-10 arası Deneyler).



**Figure 9. Digilent'in Nexys A7 FPGA kartının I/O (Girdi/Çıkı) arayüzleri**  
(kartın şuradan figürü <https://reference.digilentinc.com/>)

LEDler ile anahtarların bellek-eşlenmiş girdi/çıkı adreslerini tanımlamanın ardından program şunları yapar:

1. Şu kodu yürüterek GPIO\_INOUT yazmacının üst yarımını 0'a ayarlayarak yüksek önemli 16 GPIO uçlarını (anahtarlara bağlı olan) girdi olarak tanımlar, GPIO\_INOUT yazmacının alt yarımını 1'e ayarlayarak düşük önemli 16 GPIO uçlarını (LEDlere bağlı olan) çıktı olarak tanımlar:

```
int En_Value=0xFFFF;

WRITE_GPIO(GPIO_INOUT, En_Value);
```

2. Aşağıdaki kodu yürüterek durmaksızın anahtarların değerlerini okuyup, değerleri LEDlere yazar. Anahtarların değerlerinin bellek-eşlenmiş girdi/çıkı yazmacının üst yarımına okunduğunu anımsa, yani değer LEDlere fiziksel bağlı bellek-eşlenmiş girdi/çıkı yazmacına yazılmadan önce 16 bit sağa kaydırılmalıdır.

```
while (1) {
    switches_value = READ_GPIO(GPIO_SWs);    // anahtarlardaki değerleri oku
    switches_value = switches_value >> 16;    // alt 16 bitlere kaydır
    WRITE_GPIO(GPIO_LEDs, switches_value);    // anahtar değerini LEDlerde
göster
}
```

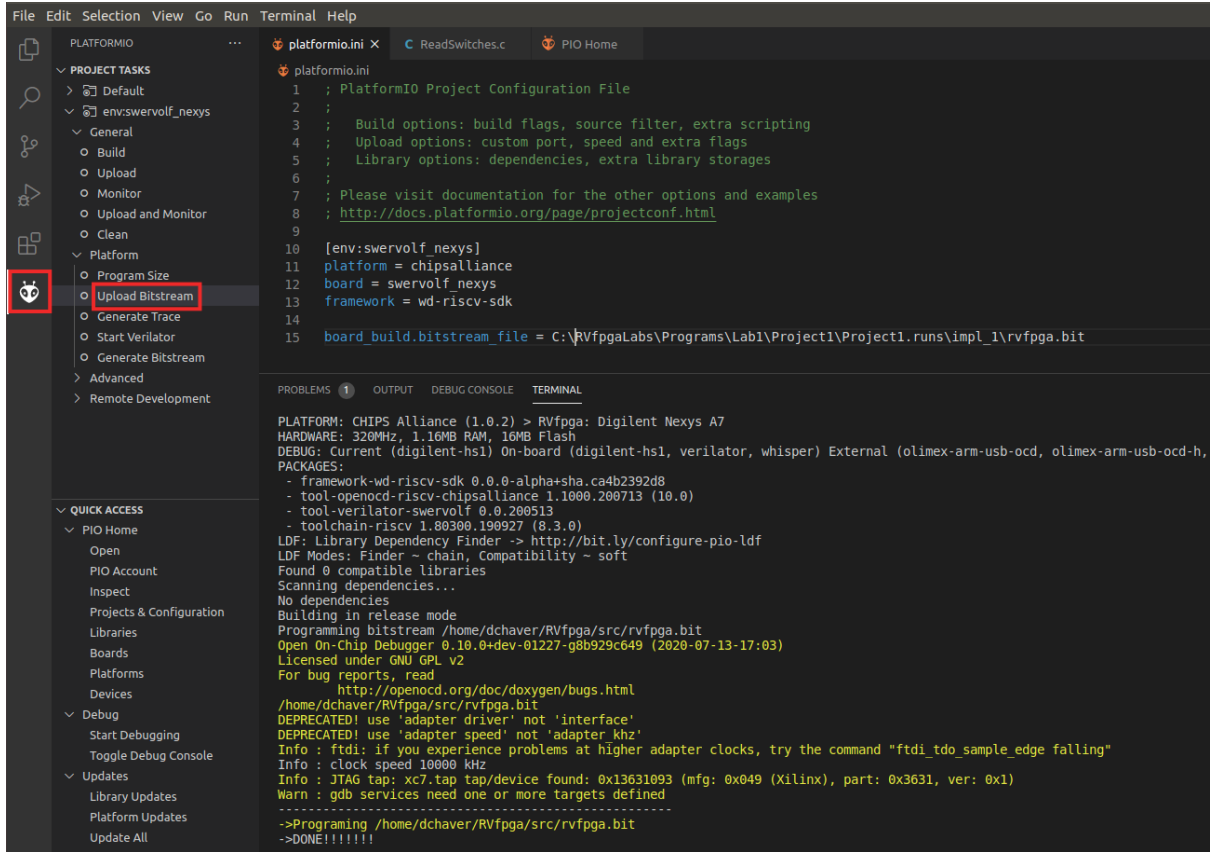
READ\_GPIO ile WRITE\_GPIO makroları belirli bellek-eşlenmiş girdi/çıkı adresindeki değeri okur ya da yazar.

### Adım 3. RVfpga'i Nexys A7 FPGA kartına indir

Şimdi RVfpga'i Nexys A7 FPGA kartına indireceksin. Sol menü şeridindeki PlatformIO ikonuna tıkla, ardından Project Tasks → env:swervolf\_nexys → Platform genişlet, Upload Bitstream'e (Veri Akışını Yükle) tıkla, Figür 10'da gösterildiği gibi.


**Önemli:** eğer bir **Windows** sistem kullanıp daha Nexys A7 FPGA kart sürücüsünü değiştirmediyse, İlk Kullanım Kılavuzunun Ek B'sindeki yönergeleri izleyerek yap.



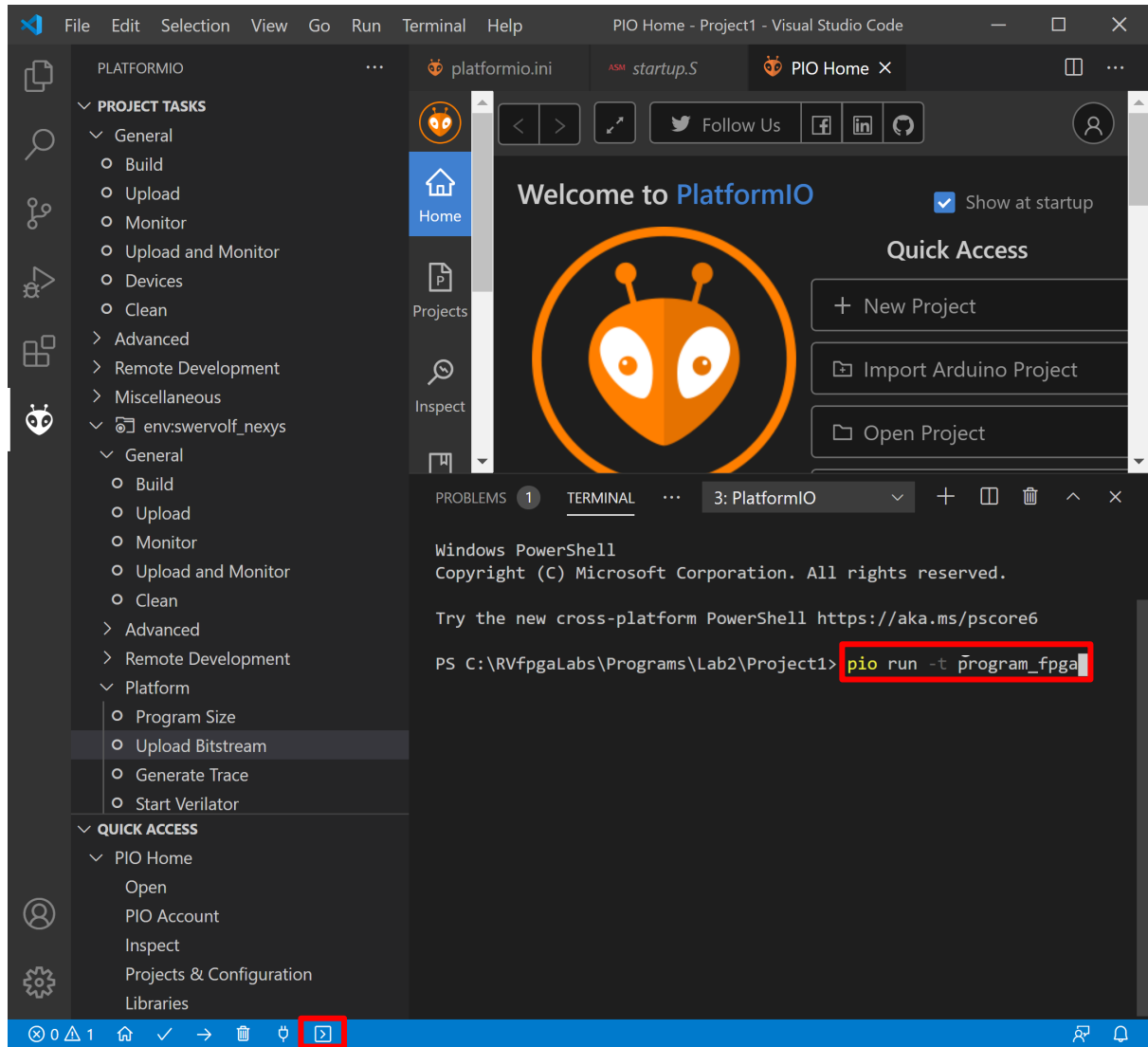


**Figür 10. PlatformIO'yu kullanarak RVfpga'i Nexys A7 FPGA Kartına yükle**

Alternatif olarak Figür 11'de gösterildiği gibi RVfpga'i bir PlatformIO terminal penceresinden de indirebilirsin. PlatformIO penceresinin aşağısındaki PlatformIO: New Terminal

(PlatformIO: Yeni Terminal) butonuna (  ) tıklayıp, ardından şunu PlatformIO terminaline yaz (ya da kopyala):

```
pio run -t program_fpga
```

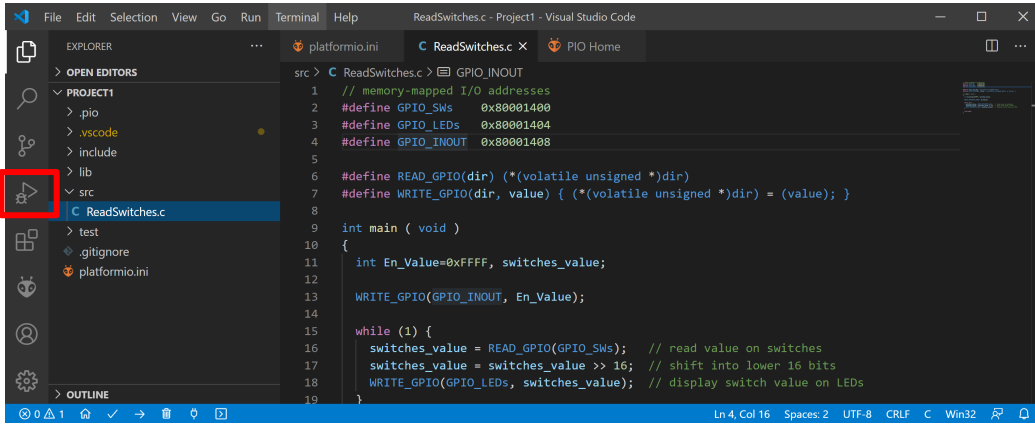


**Figür 11. PlatformIO terminalini kullanarak RVfpga’i Nexys A7 FPGA Kartına yükle**

#### **Adım 4. Bir C programını derle, indir, çalıştır**

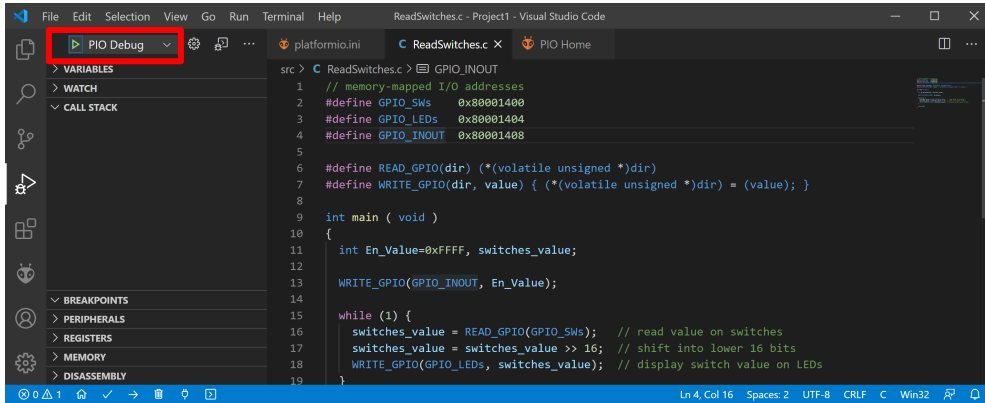
Şimdi RVfpga kartta çalıştığına göre programını derleyip, RVfpga’e indirip, çalıştırıp/ayıklayacaksın. Eğer VSCode daha açık değilse aç. Son projen, Project1, kendiliğinden açık olmalı. Değilse, PlatformIO eklentisinin açık olduğunun sağlamasını yaptıktan sonra File (Dosya) → Open Folder’a (Klasörü Aç) tıklayıp, bu deneyde önceden oluşturduğun, Project1’i seç (ancak açma).

Sol menü şeridindeki Run (Çalıştır) butonuna tıkla (Figür 12’ye göz at).



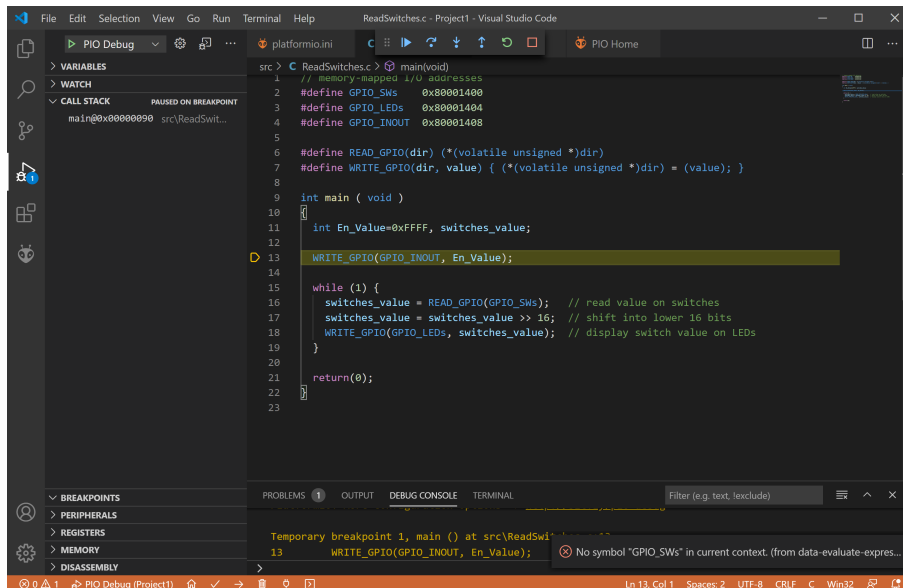
**Figür 12. RVfpga’de programı çalıştır**

Şimdi Start Debugging (Ayıklamayı Başlat) butonuna tıkla (Figür 13’e göz at).



**Figür 13. Programı çalıştırıp ayıklamayı başlat**

Program derlenip ardından Nexys A7 kartındaki FPGA’de çalışan RVfpga’e indirilecek. (Önemli olarak bir sys/cdefs.h dosyasının olmayışı ile ilgili terminalde bir deyim gözükebilir – program yine de düzgün işlevli olacaktır.) Şimdi programı çalıştırıp ayıklamaya başlayabilirsin (Figür 14’e göz at).

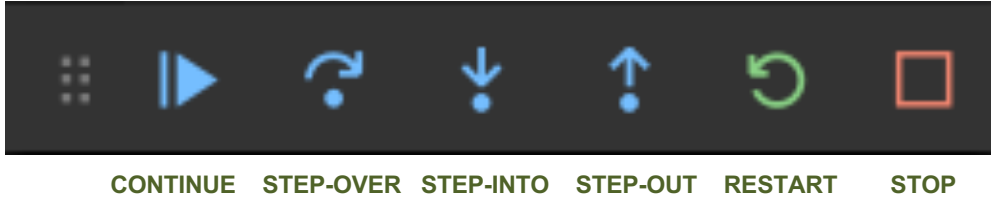


**Figür 14. Program RVfpga’de çalışıyor**

RVfpga İlk Kullanım Kılavuzunda tanımlandığı gibi, ayıklama oturumunu denetlemek için, editörün üstüne yakın görünen ayıklama araç çubuğunu kullan (Figür 15'e göz at).

Seçenekler şunlardır:


1. **Continue** (Sürdür) programı sonraki kesme noktasına dek yürütür.
2. **Breakpoints** (Kesme Noktaları) editörde satır numarasının soluna tıklayarak eklenebilir.
3. **Step Over** (Üzeri Adım) şimdiki satırı yürütüp ardından durur.
4. **Step Into** (İçeri Adım) şimdiki satırı yürütür, eğer şimdiki satırda bir işlev çağırısı varsa o işleve sıçrayıp ardından durur.
5. **Step Out** (Dışarı Adım) içinde bulunduğun işlevdeki bütün kodları yürütüp ardından işlevi dönüş yaptığında durur.
6. **Restart** (Yeniden Başlat) ayıklama oturumunu programın başından yeniden başlatır.
7. **Stop** (Durdur) ayıklama oturumunu durdurup normal düzenleme moduna döner. Önemli olarak Stop (Durdur) butonuna bastığında **program RVfpga'de çalışmayı sürdürür** ancak ayıklama oturumu sona erer.
8. **Pause** (Beklet) yürütmeyi bekletir. Program çalışırken Continue (Sürdür) butonu Pause (Beklet) butonuyla yer değiştirir.

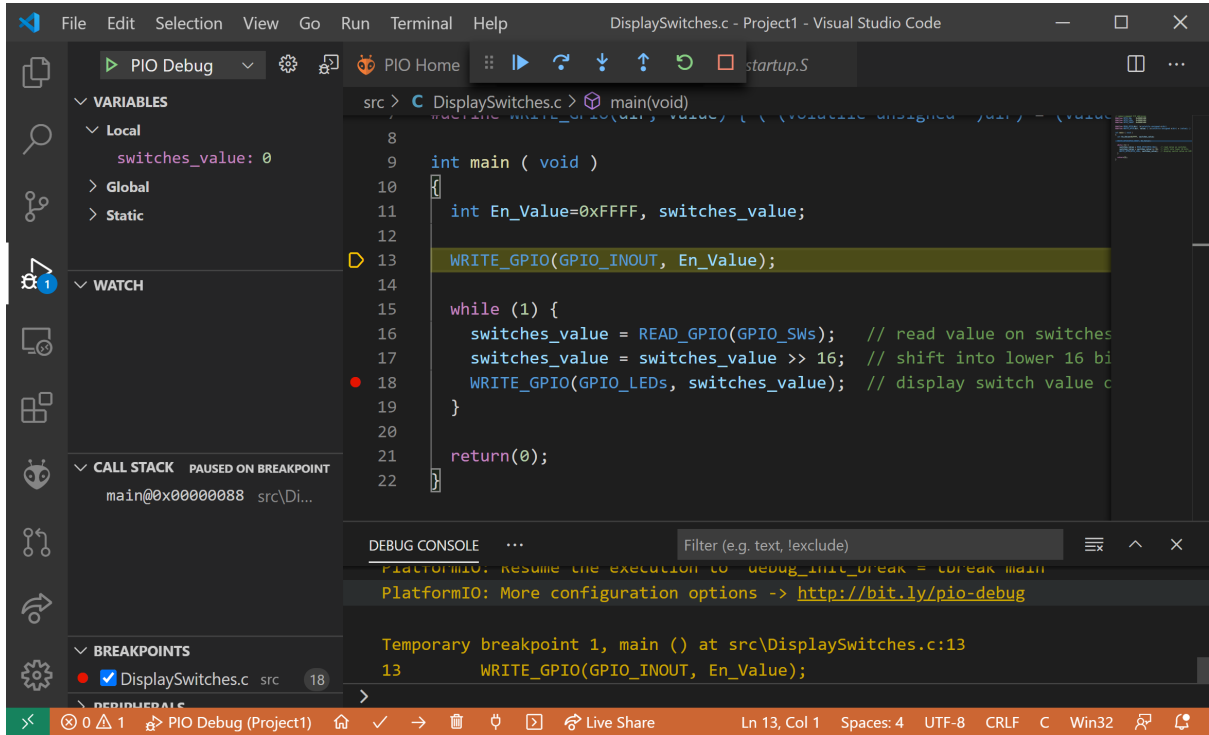


**Figür 15. Ayıklama araçları**

Sol yan çubukta, Debugger (Ayıklayıcı) seçeneklerini görebilirsin. Şu seçenekler vardır:

- **Variables** (Değişkenler): programında bulunan yerel, genel, durgun değişkenleri değerleriyle listeler.
- **Call Stack** (Çağrı Yığını): sana şimdiki çalışan işlevi, (varsa) çağırılan işlevi, şimdiki yönergenin bellekteki yerini gösterir.
- **Breakpoints** (Kesme Noktaları): ayarlanmış kesme noktalarını gösterip satır numaralarını parlatır. Kesme noktaları bu bölümde yönetilebilir. Ayrıca kesme noktaların işaret kutusu değiştirilerek geçici olarak etkinliği kaldırılabilir.
- **Peripherals** (Çevre Birimleri): aygıtın bellek-eşlenmiş çevre birimlerinin yazmaçlarının durumunu gösterir.
- **Registers** (Yazmaçlar): işlemcinin yazmaçlarının hepsinin şimdiki değerlerini listeler.
- **Memory** (Bellek): bellekteki belirli bir adresin içeriğini gösterir.
- **Disassembly** (Tersine Çeviri): belirli bir işlevin çevirici kodunu gösterir – bu C gibi yüksek-düzeyli dillerde yönergeleri bir bir ayıklayabilmek için çeviriciyi görmeni sağlar.

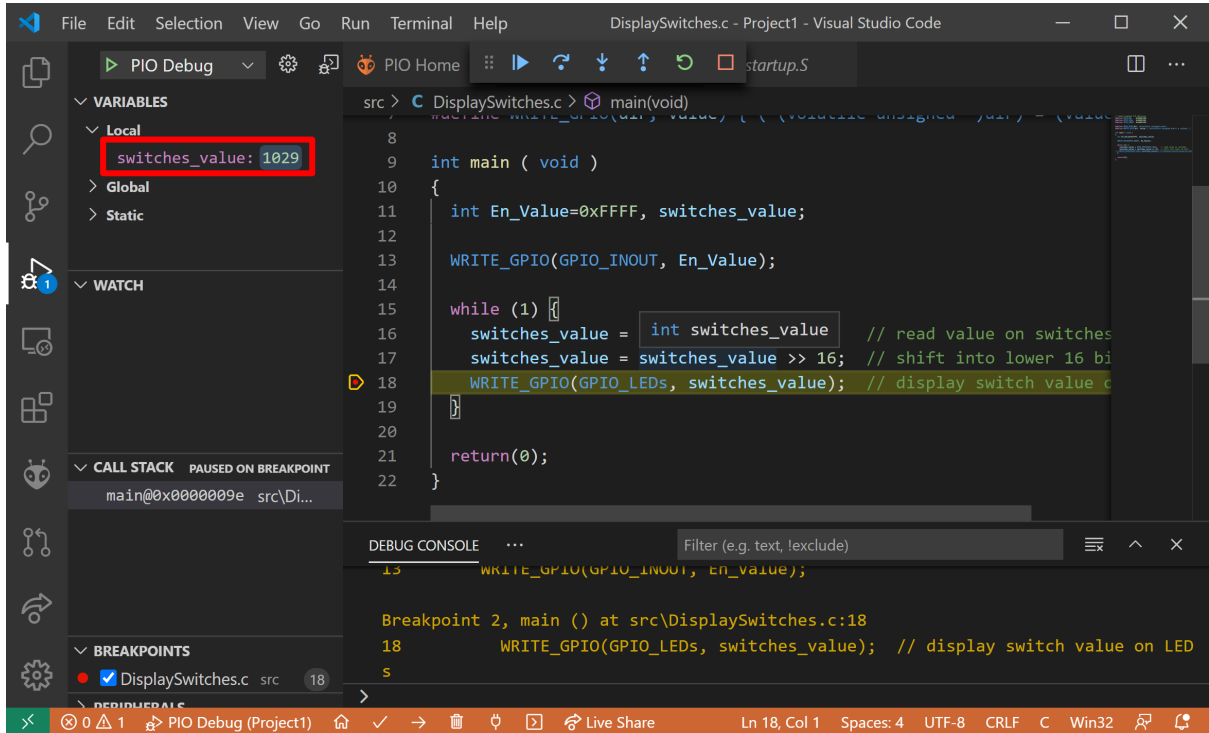
Örneğin, anahtarların değeri LEDlere yazılmadan öncesine kesme noktası ayarlamak için satır 18'in soluna tıkla, Figür 16'da gösterildiği gibi. Şimdi Continue butonuna  (ya da F5'e basarak) tıklayarak programı çalıştır. Program ayarlanmış kesme noktasına erişene kadar çalışacak. (Bir kesme noktasını kaldırmak için satır numarasının solundaki kesme noktasına tıkla.)



**Figür 16. LEDlerde değeri göstermeden önceye kesme noktası ayarlama**

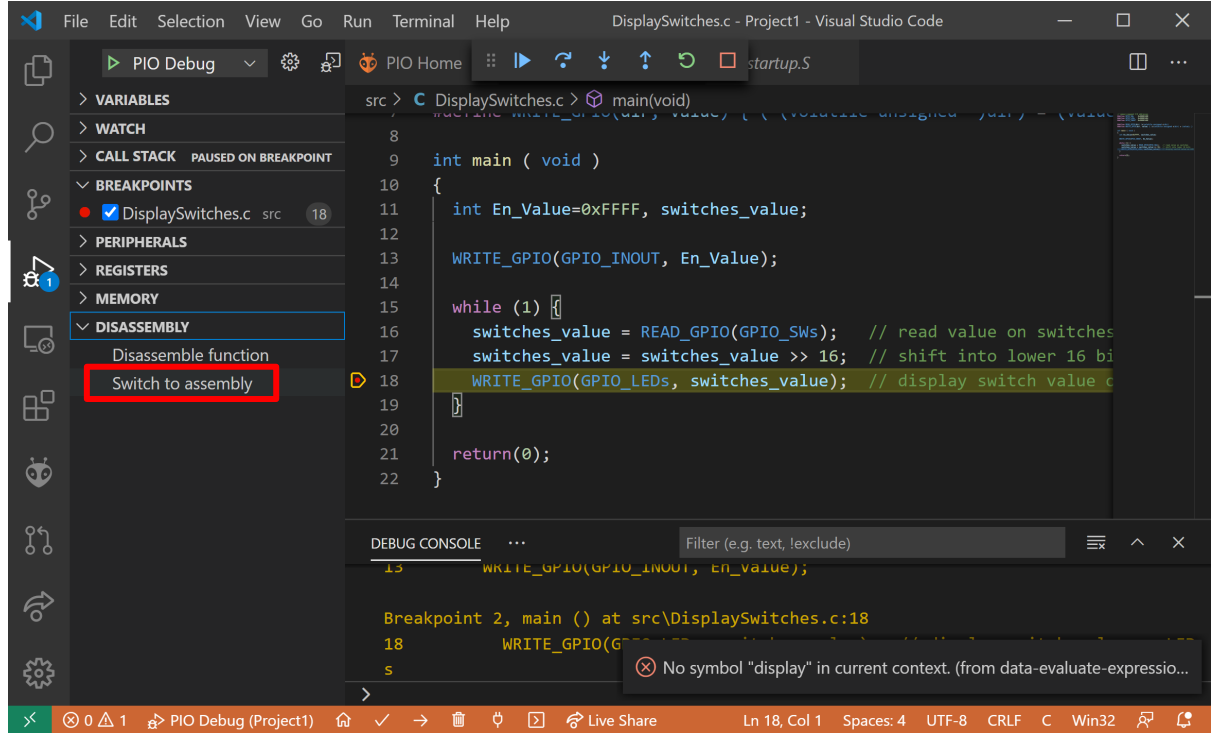
Kesme noktasına eriştikten sonra sol paneldeki Variables (Değişkenler) bölümünü genişletip switches\_value değişkeninin değerine bak, Figür 17’de gösterildiği gibi. Burada anahtarların değeri 1029 = 0x405 (ikilide 0000\_0100\_0000\_0101), şu örüntüye karşılık gelir:

Switches[15:0] = OFF-OFF-OFF-OFF-OFF-ON-OFF-OFF-OFF-OFF-OFF-OFF-OFF-ON-OFF-ON



**Figür 17. Değişkenlerin değerlerine bakma**

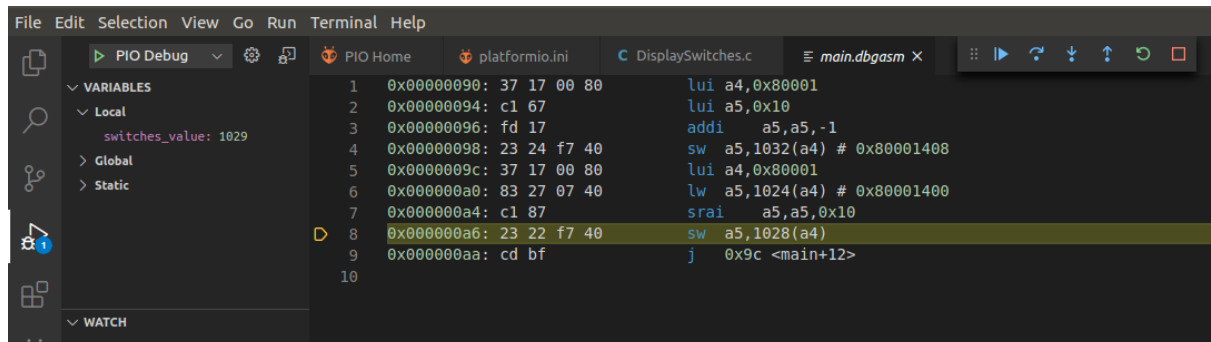
You may also view the RISC-V assembly code generated from the C program. Do so by clicking on DISASSEMBLY → Switch to assembly, as shown in Figür 18.



**Figür 18. RISC-V çevirici koduna bak**

RISC-V çeviricisi şimdi görüntüleme panelinde görünür, Figür 19'da gösterildiği gibi. Çevirici yönergenin bellek adresini, makine kodunu, çevirici kodunu gösterir. Görebileceğin üzere C kodu sıkıştırılmış (16-bit yönergeler) ile 32-bit yönergelerin bir karışımına derlenir. Aşağıdaki yorumlu çevirici kodudur.

# adres	# makine kodu	# yönerge		
0x00000090:	37 17 00 80	lui	a4,0x80001	# girdi/çıkı için taban adresi
0x00000094:	c1 67	lui	a5,0x10	# a5 = 0x10000 - 1 (=0xFFFF)
0x00000096:	fd 17	addi	a5,a5,-1	
0x00000098:	23 24 f7 40	sw	a5,1032(a4)	# girdi/çıkı yönü: [0x80001408]=0xFFFF
0x0000009c:	37 17 00 80	lui	a4,0x80001	# girdi/çıkı için taban adresi
(gereksiz)				
0x000000a0:	83 27 07 40	lw	a5,1024(a4)	# Anahtarları oku: a5 = [0x80001400]
0x000000a4:	c1 87	srai	a5,a5,0x10	# Anahtar değerini alt 16 bitlere taşı
0x000000a6:	23 22 f7 40	sw	a5,1028(a4)	# LEDlere yaz: [0x80001404] = a5
0x000000aa:	cd bf	j	0x9c <main+12>	# tekrarla




**Figür 19. RISC-V çevirici kodunu gör**

C kodunu yeniden görmek için DISASSEMBLY → Switch to code'a (Koda dön) tıkla.

Programı çalıştırmayı/ayıklamayı bitirdikten sonra ayıklama oturumunu Stop (Durdur)

butonuna basarak  (ya da Shift - F5) durdur, Explorer gezginine

en soldaki yan çubuğun üstündeki  butonuna tıklayarak dön. Önemli olarak **program RVfpga'de çalışmayı sürdürür** – yalnızca ayıklama oturumu sona erer. Projeyi üst menu çubuğundaki *File* (Dosya) → *Close Folder'a* (Klasörü Kapat) tıklayarak kapat.

### 3. printf'i kullanma, dizesel monitör

printf deyimlerini kullanmak programın ilerleyişini izlemek ya da kullanıcılara geri bildirim sunmak, örneğin işlem sonuçlarını sunmak, için kullanışlı bir yöntemdir. RVfpga İlk Kullanım Kılavuzundan (Bölüm 6.F'teki *HelloWorld\_C-Lang* örneği) anımsayabileceğin üzere normal C programlarındaki `printf` işlevi gibi olan `printfNexys` işlevini kullanabilirsin. Bunun için belirli işlemciyle kart için, burada SweRV EH1 çekirdeğiyle Nexys A7 FPGA kartı, ortak işlevleri sunan Western Digital'in PSP, BSP'sini (işlemci destek paketi, kart destek paketi) kullanmalısın.

PrintfExample adında bir PlatformIO projesini `[RVfpgaPath]/RVfpga/Labs/Lab2` klasöründe oluştur. Şu programı o projeye ekle (Figür 20'ye göz at). Program dosyasını *PrintfExample.c* olarak adlandır.

Program kolaylık olması için burada da var:

`[RVfpgaPath]/RVfpga/Labs/Lab2/PrintfExample.c`

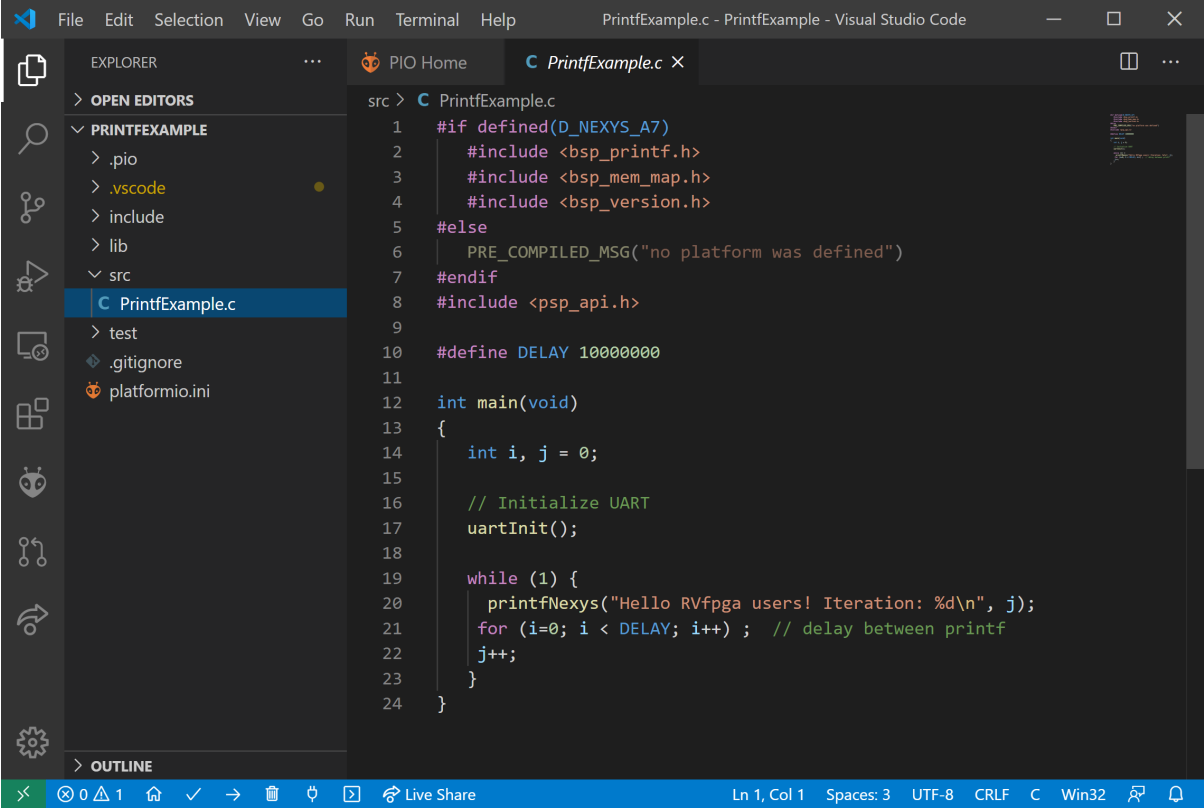
```
#if defined(D_NEXYS_A7)
#include <bsp_printf.h>
#include <bsp_mem_map.h>
#include <bsp_version.h>
#else
PRE_COMPILED_MSG("no platform was defined")
#endif
#include <psp_api.h>

#define DELAY 10000000

int main(void)
{
    int i, j = 0;

    // UART'a ilk değerini ver
    uartInit();

    while (1) {
        printfNexys("Hello RVfpga users! Iteration: %d\n", j);
        for (i=0; i < DELAY; i++) ; // printf'ler arasında geciktir
        j++;
    }
}
```



```

src > C PrintfExample.c
1  #if defined(D_NEXYS_A7)
2      #include <bsp_printf.h>
3      #include <bsp_mem_map.h>
4      #include <bsp_version.h>
5  #else
6      PRE_COMPILED_MSG("no platform was defined")
7  #endif
8  #include <psp_api.h>
9
10 #define DELAY 1000000
11
12 int main(void)
13 {
14     int i, j = 0;
15
16     // Initialize UART
17     uartInit();
18
19     while (1) {
20         printfNexys("Hello RVfpga users! Iteration: %d\n", j);
21         for (i=0; i < DELAY; i++) ; // delay between printf
22         j++;
23     }
24 }

```

**Figür 20. PrintfExample.c**

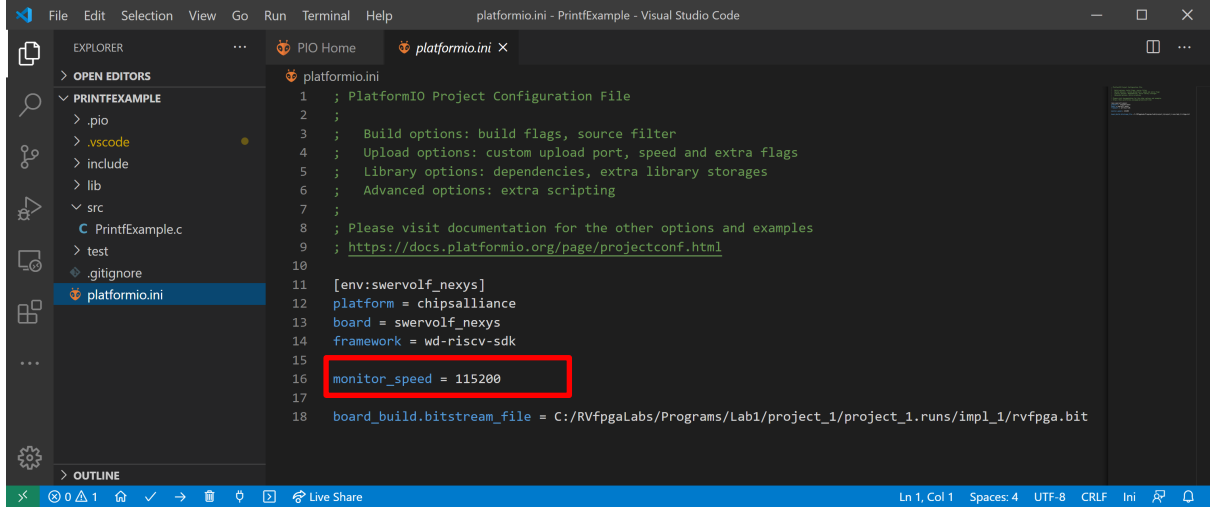
1-8 arası satırlar (Figür 20'ye göz at) `printfNexys` işlevini kullanmak için gerekli içerilecek dosyalardır. Western Digital'in BSP/PSP'sinde sunulurlar. Figür 20'deki Satır 17 `uartInit` işlevini çağırır; bu satır (Nexys A7 kartında çalışan) RVfpga ile dizesel monitör arasında iletişim için kullanılan UART bağlantısına ilk değerlerini vermek için gereklidir. Son olarak `while` döngüsü (satır 21'deki) gecikmeyle izlenen satır 20'deki `printfNexys` işlevini kullanarak dizesel monitöre sürekli yazar.

`printfNexys` işleviyle UART'ı kullanmak için `platformio.ini` dosyası UART'ın hızını içerecek biçimde değiştirilmelidir. Şu satırı `platformio.ini` dosyasına ekle, Figür 21'de gösterildiği gibi:

```
monitor_speed = 115200
```

RVfpga UART'ın 115200 baudda (saniye başına sembol) iletişim kurmasını bekler, dolayısıyla bu oran `platformio.ini`'de ayarlanmalıdır, Figür 21, satır 16'da gösterildiği gibi. Ayrıca RVfpga veri dosyanın yerini `board_build.bitstream_file = ...` kullanarak eklemeyi unutma (Figür 21, satır 18'de gösterildiği gibi).






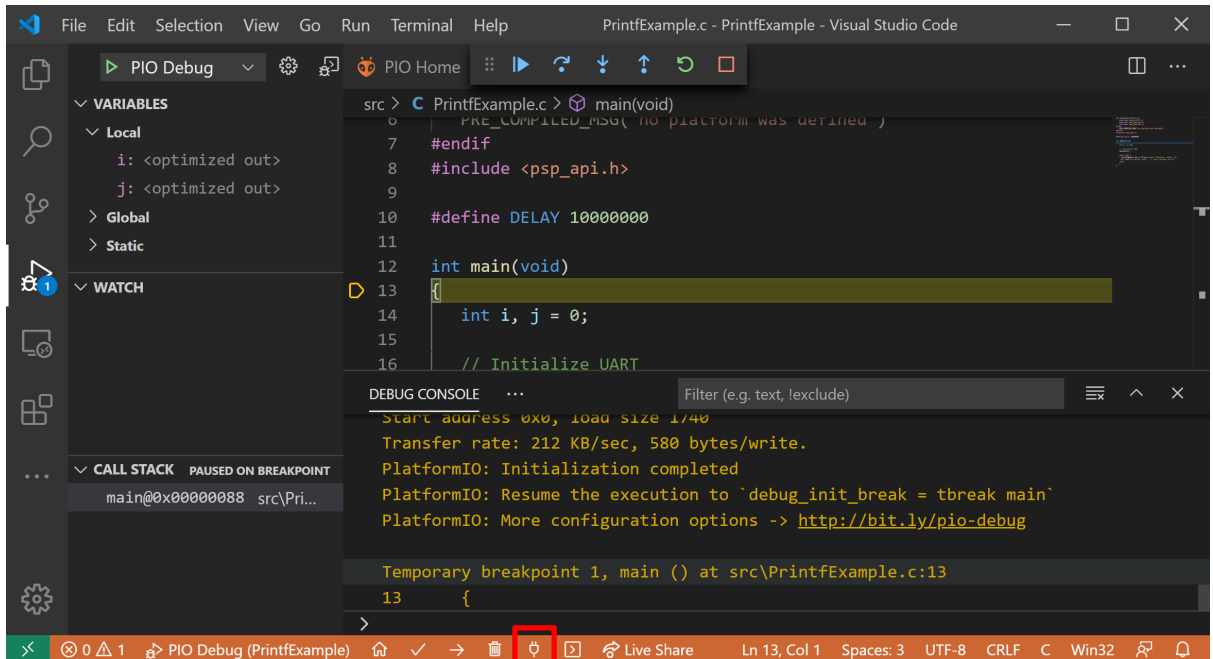
**Figür 21. UART hızını ayarlama**

**LINUX:** Unutma, eğer Linux kullanıyorsan, dizeşel monitörü kullanabilmeye başlamadan önce sistemi kullanıcıyı dialout, tty, uucp gruplarına ekleyerek hazırlamalısın, İlk Kullanım Kılavuzunun Bölüm 6.F'sinde açıklandığı gibi. GSG'deki bu işlemi yaptıysan sıkıntı yok; yapmadıysan şimdi yap.

Şimdi veri dosyasını yükle (Bölüm 2'de tanımlandığı gibi), programı çalıştır/ayıkla.

Program çalışmaya başladıktan sonra (program çalışmaya başladıktan **sonra**), PlatformIO penceresinin aşağısındaki dizeşel monitör butonuna  tıkla (Figür 22'ye göz at).

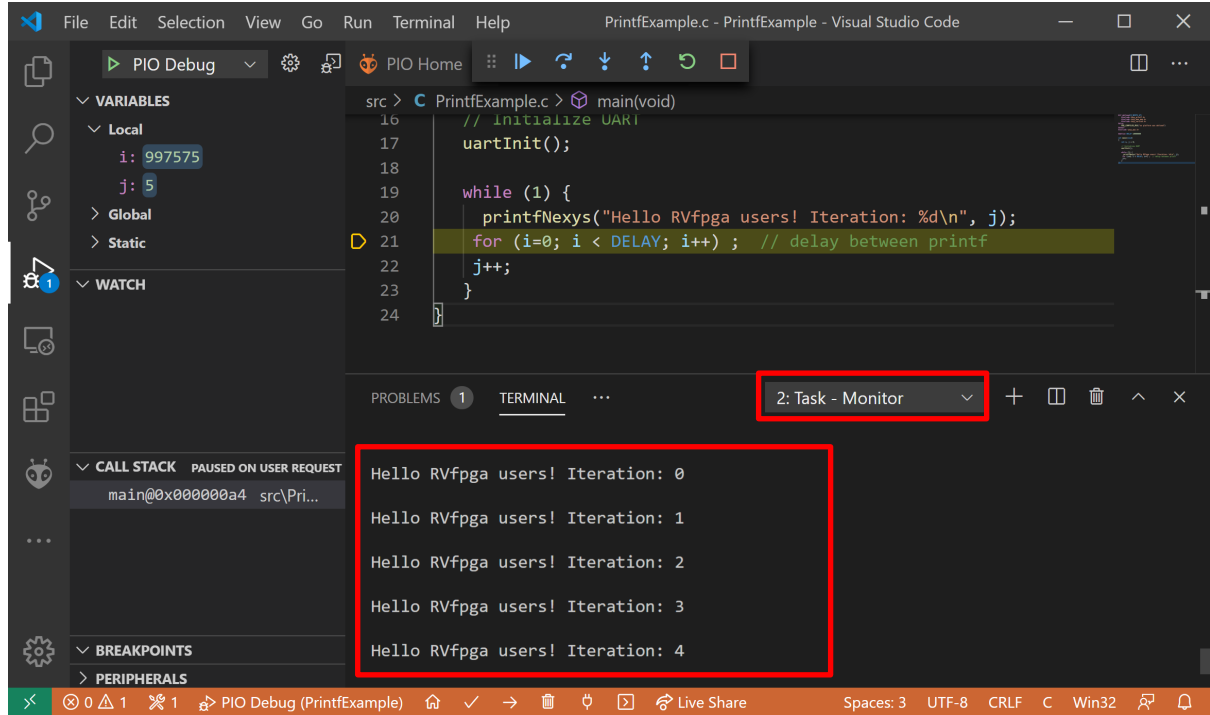
**Uyarı:** Dizeşel monitörü program çalışmaya (ardından ilk – geçici – kesme noktasına vurmaya) başlamadan açarsan UART bozulup düzgün çalışmayacaktır.



**Figür 22. Dizeşel monitörü başlatma**

Ardından programı çalıştır: .

Yazdırılan karakter dizisiyle (Hello RVfpga users!) yineleme numarasını dizisel monitörde durmaksızın göreceksin, Figür 23'te gösterildiği gibi.



**Figür 23. PlatformIO'da dizisel monitörde printfNexys işlevinin çıktısı**

## 4. Alıştırmalar

Şu alıştırmaları bitirerek kendi C programlarını oluştur. Önemli olarak Nexys A7 kartını bilgisayarına bağlı, açık bırakırsan program çalıştırmaları arasında RVfpga'yi yeniden yüklemen gerekmez. Ancak Nexys A7 kartını kapatırsan RVfpga'yi karta PlatformIO'yu kullanarak yeniden yüklemen gerekir, Bölüm 2'nin adım 3'ünde tanımlandığı gibi.

Unutma, Western Digital'in BSP işlevi `printfNexys` (Bölüm 3'e göz at) kullanarak istediğin değişkeni yazdırabilirsin.

**Alıştırma 1.** LEDlerin değerini anahtarlara yakıp söndüren bir C programı yaz. Darbe bir kişinin yanıp sönmeyi anlayabileceği yavaşlıkta olmalı. Programı **FlashSwitchesToLEDs.c** olarak adlandır.

**Alıştırma 2.** Anahtarların değerinin tersini LEDlerde gösteren bir C programı yaz. Örneğin anahtarlar şu ise (ikili sistemde): 01010101010101, LEDler şunu göstermeli: 10101010101010; anahtarlar şu ise: 1111000011110000, LEDler şunu göstermeli: 0000111100001111; gibi gibi. Programı **DisplayInverse.c** olarak adlandır.

**Alıştırma 3.** Bütün LEDler yakılana değin artan, kayan sayılar gösteren bir C programı yaz. Ardından örüntü kendini tekrarlamalı. Programı **ScrollLEDs.c** olarak adlandır.

Program şunları yapmalı:

1. İlk olarak bir yakılmış LED sağdan sola kayıp ardından soldan sağa kaymalı.
2. Ardından iki yakılmış LED sağdan sola kayıp ardından soldan sağa doğru kaymalı.
3. Ardından üç yakılmış LED sağdan sola kayıp ardından soldan sağa doğru kaymalı.
4. Gibi gibi, bütün LEDler yakılmış olana değin.
5. Ardından bu örüntü kendini tekrarlamalı.

**Alıştırma 4.** Anahtarların 4 düşük önemli bitiyle 4 yüksek önemli bitini pozitif 4-bit eklemeye gösteren bir C programı yaz. Sonucu LEDlerin 4 düşük önemli (en sağ) bitinde göster. Programı **4bitAdd.c** olarak adlandır. LEDlerin beşinci biti pozitif taşması (unsigned overflow) olunca yanmalı (yani taşınan (carry) 1 olunca).

**Alıştırma 5.** Öklit algoritmasıyla iki sayının, a ile b, en büyük ortak bölenini bulan bir C programı yaz. a ile b'nin değerleri programda statik tanımlı değişken olmalıdır. Programı **GCD.c** olarak adlandır. Şurada Öklit algoritmasıyla ilgili ek bilgi vardır:

<https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/the-euclidean-algorithm>. Google'da şunu da aratabilirsin: "Euclidean algorithm".

**Alıştırma 6.** Fibonacci dizisinin ilk 12 sayısını hesaplayıp, ardından bunu sonlu, 12 ögeli bir vektörde (bir diğer deyişle) depolayan bir C programı yaz. Fibonacci sayılarının sonsuz sekansı şöyle tanımlanır:

$$V(0)=0, V(1)=1, V(i)=V(i-1)+V(i-2) \quad (i=0,1,2,\dots)$$

Bir diğer deyişle i ögesinde denk gelen Fibonacci sayısı, dizinin önceki iki Fibonacci sayısının toplamına eşittir. Tablo 1 i = 0'dan 8'e Fibonacci sayılarını gösterir.

**Tablo 1. Fibonacci dizisi**

<i>i</i>	0	1	2	3	4	5	6	7	8
<i>V</i>	0	1	1	2	3	5	8	13	21

Vektör boyutu N programda bir sabit olarak tanımlanmalıdır. Programı **Fibonacci.c** olarak adlandır.

**Alıştırma 7.** N ögeli bir A vektöründen (dizisinden) yalnızca çift, sıfırdan büyük sayıları içeren bir başka B vektörü oluştur. C programı B'deki öge sayısını sayıp program sonunda bu değeri yazdırmalıdır. Örneğin:  $N=12$ ,  $A = [0,1,2,7,-8,4,5,12,11,-2,6,3]$ , ise:  $B = [2,4,12,6]$ . B'nin dört ögesi olduğundan program sonunda şu yazdırılır:

```
Number of elements in B = 4.
```

Yazdırmak için `printfNexys` işlevini kullan. Programı **EvenPositiveNumbers.c** olarak adlandır. Programını A'da 12 ögeyle dene.

**Alıştırma 8.** İki N ögeli vektör (dizi) A ile B verিলince şöyle bir C vektörü oluştur:

$$C(i) = |A[i] + B[N-i-1]|, \quad i = 0, \dots, N-1.$$

Yeni vektörü hesaplayan programı C'de yaz. Programında 12 ögeli diziler kullan. Programı **AddVectors.c** olarak adlandır.

**Alıştırma 9.** C'de bubble sort algoritmasını gerçekleştir. Bu algoritma şu prosedürle bir vektörün bileşenlerini yükselen biçimde sıralar:

1. Vektörü bitene değin tekrar tekrar dolaş.
2.  $V(i) > V(i+1)$  ise komşu bileşenlere yer değiştir.
3. Ardışık bileşenlerin hepsi sıralı olunca algoritma sonlanır.

Programını denemek için 12 ögeli diziler kullan. Programı **BubbleSort.c** olarak adlandır.

**Alıştırma 10.** C'de negatif olmayan bir sayı n'i yinelemeli çarpımlarla hesaplayan bir C programı yaz. Programı değişik n değerleriyle dene ancak son değer n= 7 olmalıdır. Programın sonunda factorial(n) değeri yazdırılmalıdır. n program içerisinde statik tanımlanmış bir değişken olmalı. Programı **Factorial.c** olarak adlandır.