Microcontrollers & Applications

Lecture 1.3: Programming Languages of C, Assembly, and Python

Programming Languages

C / C++: Arduino UNO

- Arduino IDE

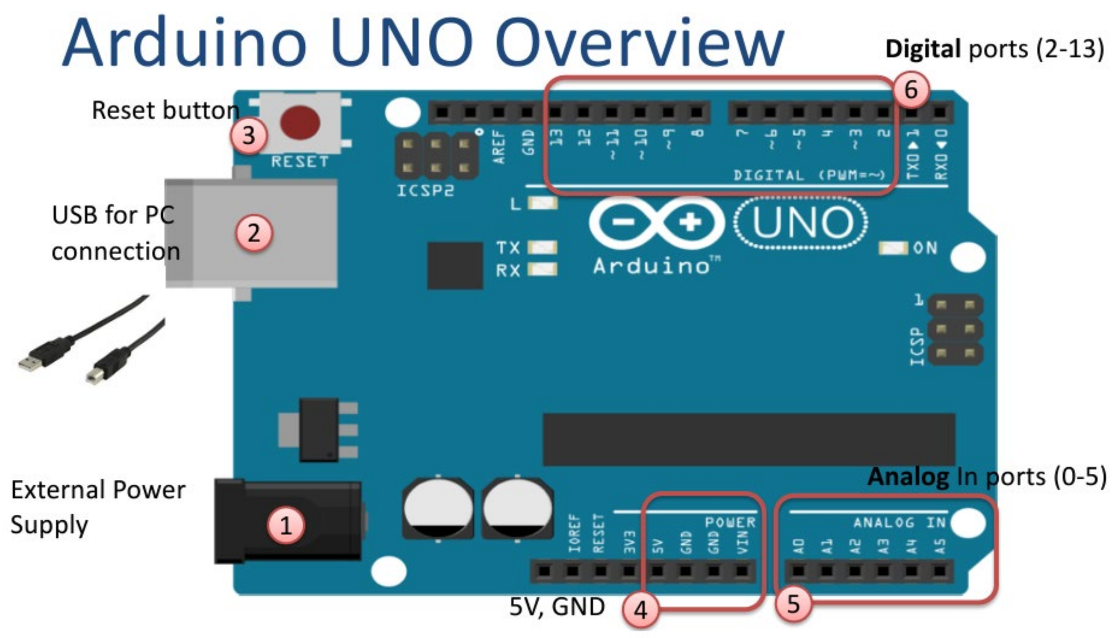
Assembly: Arduino UNO

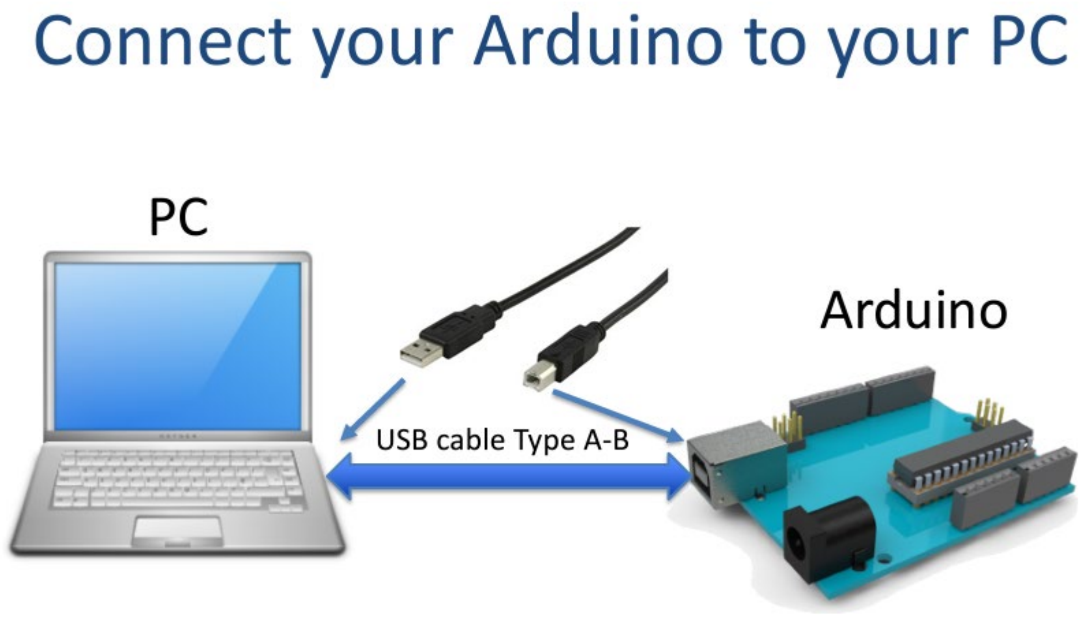
- Arduino IDE

Python: Raspberry Pi Pico

- MicroPython
- CircuitPython

Arduino UNO and Connection





Arduino: Choose Correct Board and Port

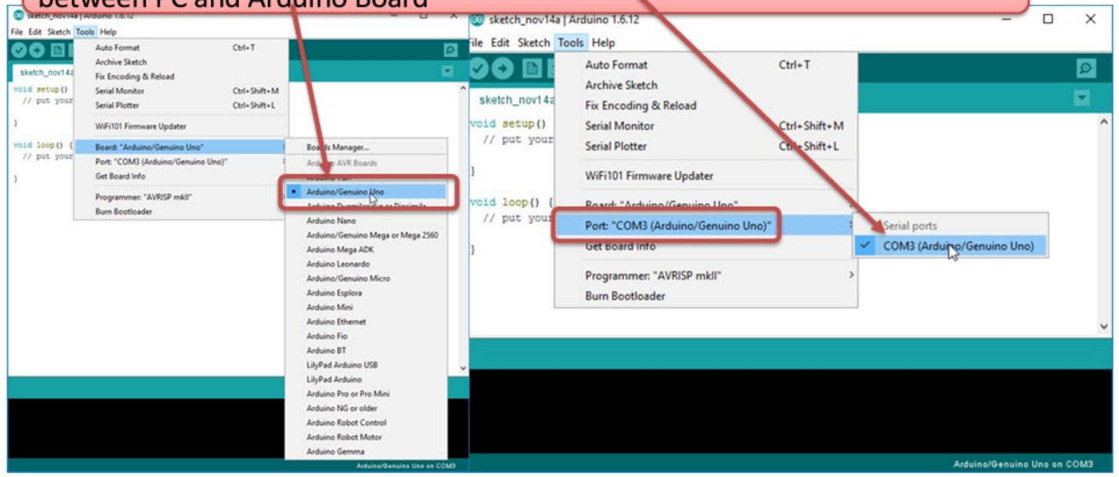
Arduino Programs

All Arduino programs must follow the following main structure:

```
// Initialization, define variables, etc.

void setup()
{
   // Initialization
   ...
}

void loop()
{
   //Main Program
   ...
}
```

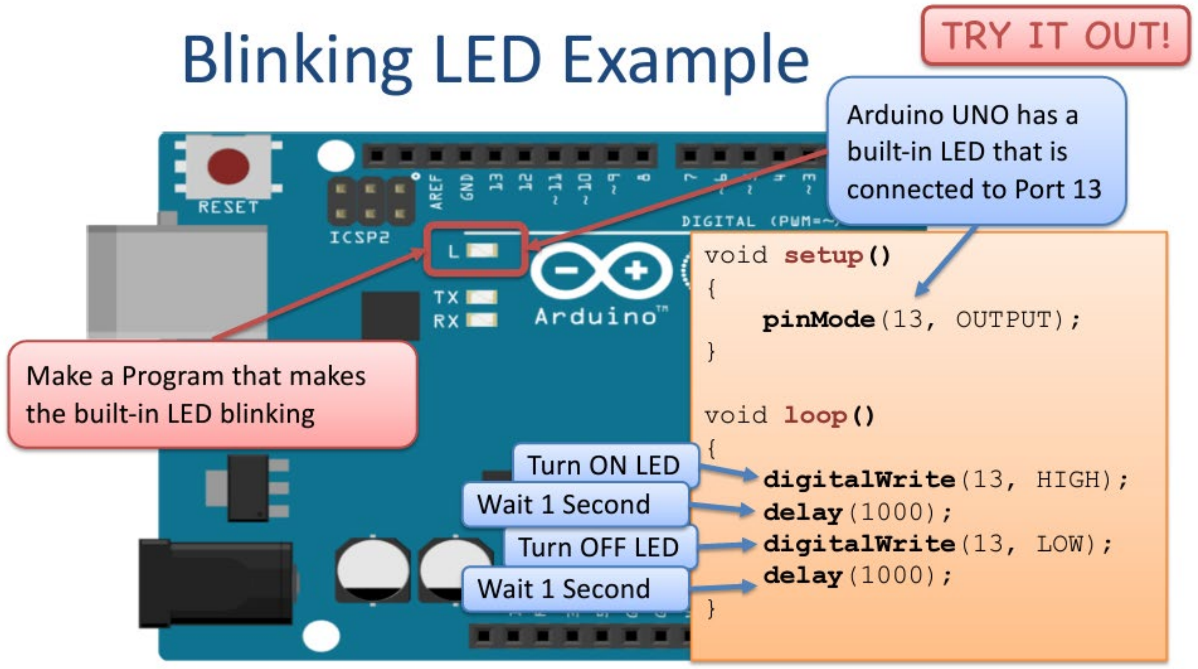
Do you get an Error Message?

Choose correct Board (Arduino UNO) and the correct Port for Communication between PC and Arduino Board



```
void setup() {
  // put your
}

void loop() {
  // put your
}
```

Arduino: The Blink Example



```
void setup()
{
    pinMode(13, OUTPUT);
}

void loop()
{
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
}
```

Blinking LED Example

```
void setup()
{
   pinMode(13, OUTPUT);
}

void loop()
{
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13, LOW);
   delay(1000);
}
```

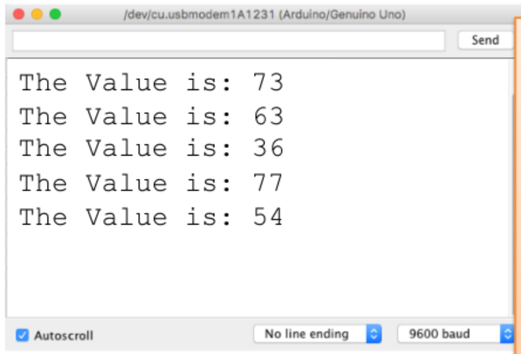
This Program makes the built-in LED blinking

Try to change from 1000 to 100
– What happens then?

Arduino IDE: Serial Communication Example

Serial Monitor

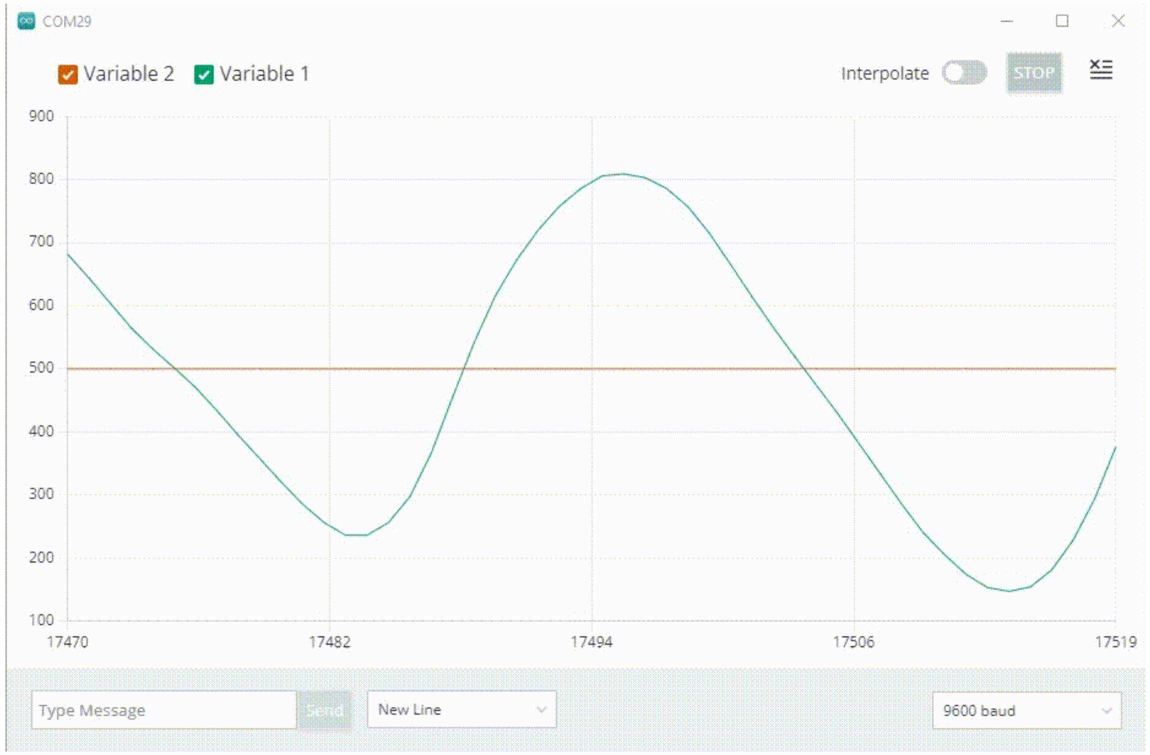
TRY IT OUT!



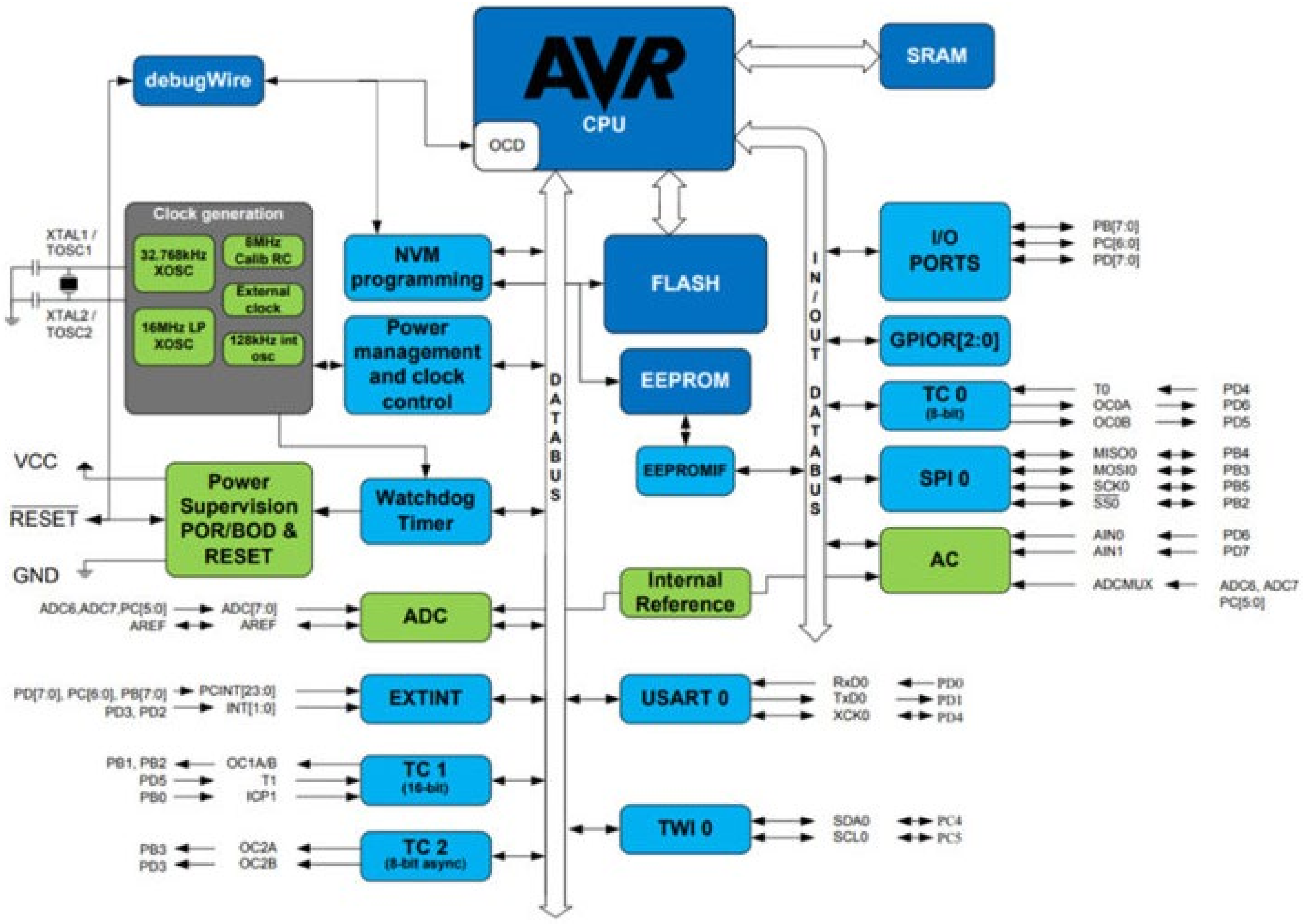
```
The Value is: 73
The Value is: 63
The Value is: 36
The Value is: 77
The Value is: 54
```

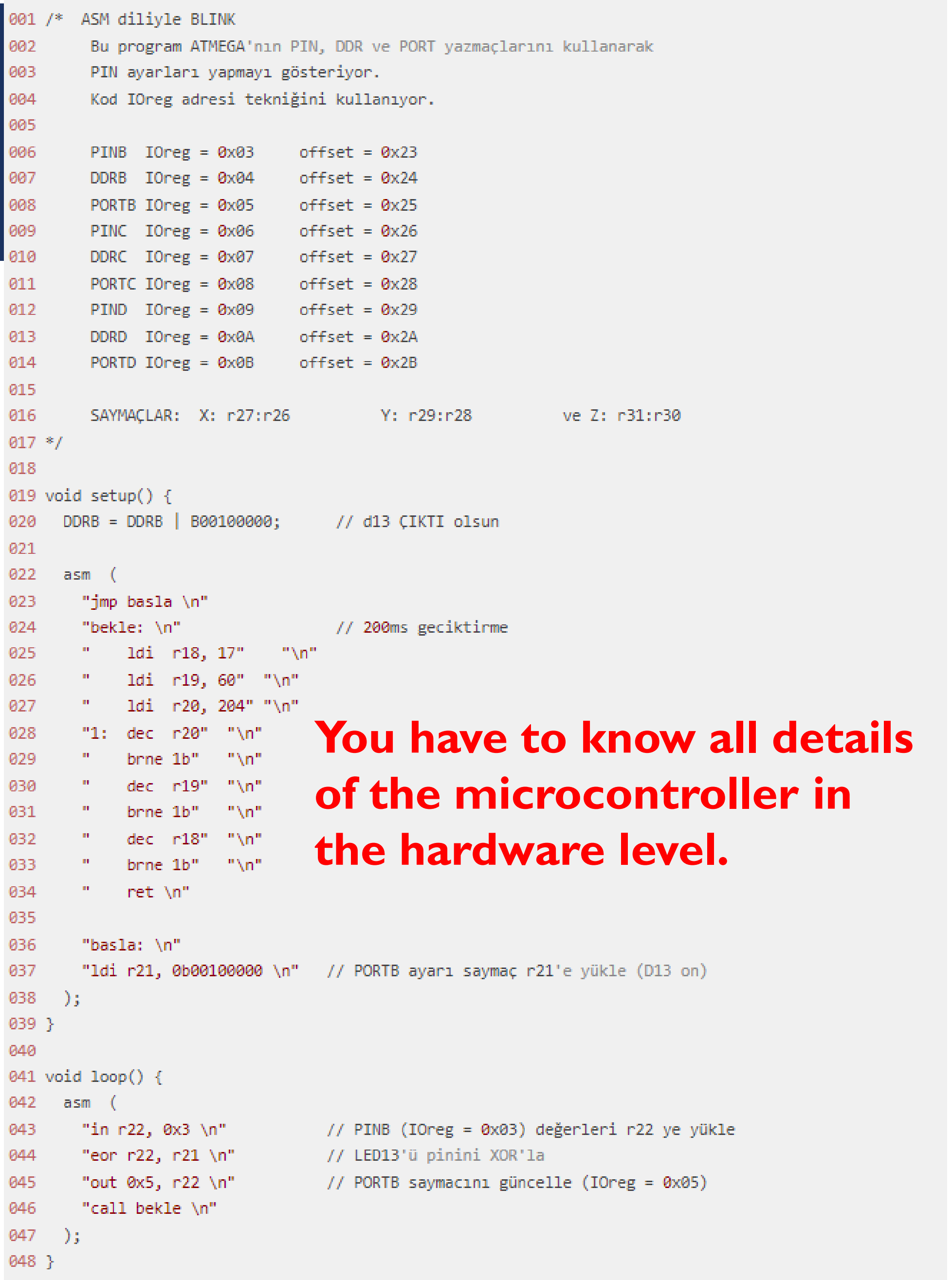
```
int myValue = 0;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    myValue = random(100);
    Serial.print("The Value is: ");
    Serial.println(myValue);
    delay(1000);
}
```

Here you see how we can write a value to the Serial Monitor. This can be a value from a sensor, e.g., a temperature sensor.



Assembly: Arduino IDE



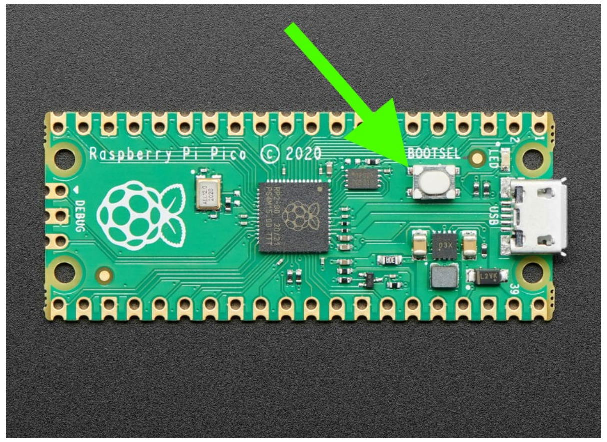


```
/*  ASM diliyle BLINK
     Bu program ATMEGA'nın PIN, DDR ve PORT yazmaçlarını kullanarak
     PIN ayarları yapmayı gösteriyor.
     Kod IOreg adresi tekniğini kullanıyor.

     PINB  IOreg = 0x03     offset = 0x23
     DDRB  IOreg = 0x04     offset = 0x24
     PORTB IOreg = 0x05     offset = 0x25
     PINC  IOreg = 0x06     offset = 0x26
     DDRC  IOreg = 0x07     offset = 0x27
     PORTC IOreg = 0x08     offset = 0x28
     PIND  IOreg = 0x09     offset = 0x29
     DDRD  IOreg = 0x0A     offset = 0x2A
     PORTD IOreg = 0x0B     offset = 0x2B

     SAYMAÇLAR:  X: r27:r26          Y: r29:r28          ve Z: r31:r30
*/

void setup() {
  DDRB = DDRB | B00100000;      // d13 ÇIKTI olsun

  asm  (
    "jmp basla \n"
    "bekle: \n"                 // 200ms geciktirme
    "    ldi  r18, 17"    "\n"
    "    ldi  r19, 60"   "\n"
    "    ldi  r20, 204" "\n"
    "1:  dec  r20"   "\n"
    "    brne 1b"    "\n"
    "    dec  r19"   "\n"
    "    brne 1b"    "\n"
    "    dec  r18"   "\n"
    "    brne 1b"    "\n"
    "    ret \n"

    "basla: \n"
    "ldi r21, 0b00100000 \n"    // PORTB ayarı saymaç r21'e yükle (D13 on)
  );
}

void loop() {
  asm  (
    "in r22, 0x3 \n"            // PINB (IOreg = 0x03) değerleri r22 ye yükle
    "eor r22, r21 \n"           // LED13'ü pinini XOR'la
    "out 0x5, r22 \n"           // PORTB saymacını güncelle (IOreg = 0x05)
    "call bekle \n"
  );
}
```

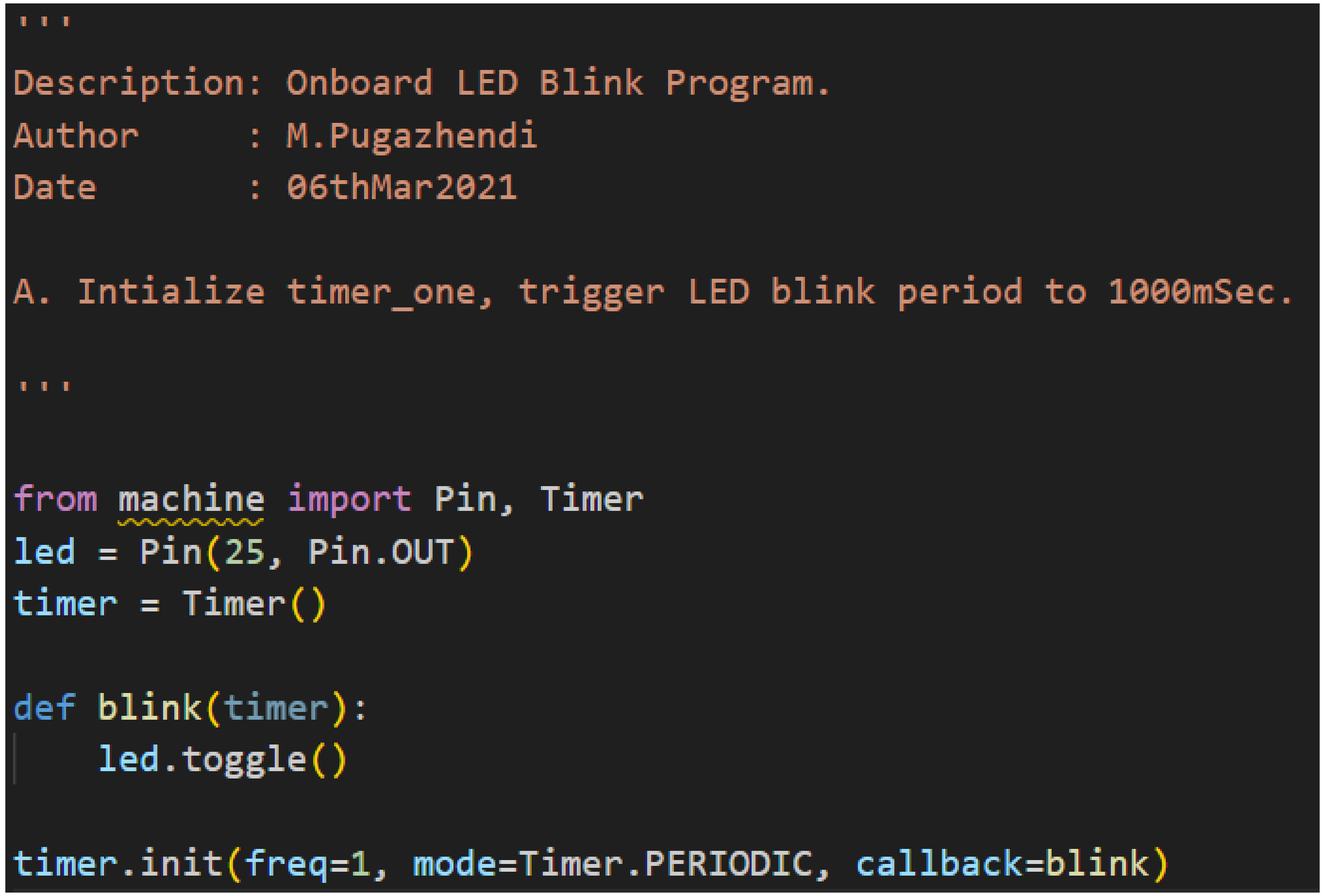
You have to know all details of the microcontroller in the hardware level.

Python: Raspberry Pi Pico

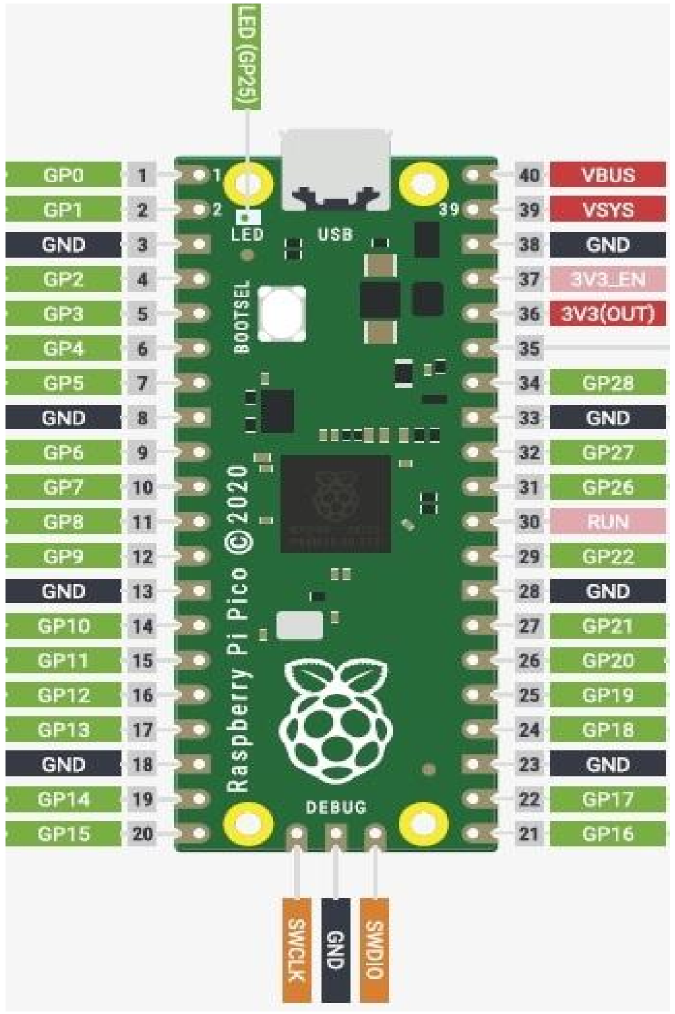
- Download and install one of the editors:
  - Thonny
  - Visual Studio Code
  - Mu
  - …
- Preperation:
  - Download the latest UF2 image file from either MicroPython or CircuitPython website for your board ((Raspberry Pi Pico, for example).
  - While continue to hold BOOTSEL button, connect the board to the computer via USB.
  - You will see a new disk drive appear called RPI-RP2.
  - Drag the adafruit_circuitpython_etc.uf2 file to RPI-RP2.
  - The RPI-RP2 drive will disappear, and a new disk drive called CIRCUITPY will appear.
- Coding:
  - Write your program into the code.py file, and copy it under CIRCUITPY drive.
  - If you need, you can copy required library files under the lib folder.



Circuitpython: Blink Example



```
'''
Description: Onboard LED Blink Program.
Author      : M.Pugazhendi
Date        : 06thMar2021

A. Intialize timer_one, trigger LED blink period to 1000mSec.

'''

from machine import Pin, Timer
led = Pin(25, Pin.OUT)
timer = Timer()

def blink(timer):
    led.toggle()

timer.init(freq=1, mode=Timer.PERIODIC, callback=blink)
```



CircuitPython

- Why would I use CircuitPython?
  - CircuitPython is designed to run on microcontroller boards.
  - All you need is that little board, a USB cable, and a computer with a USB connection.
  - You want to get up and running quickly.
  - You're new to programming.
  - Easily update your code.
  - The serial console and REPL.
  - File storage.
  - Strong hardware support.
  - **It's Python!**

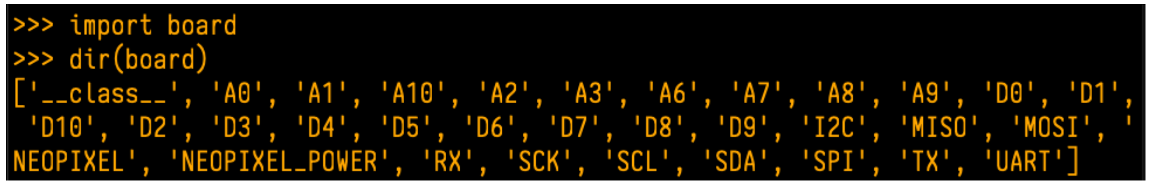
- CircuitPython is also available for the Pico and generally RP2040 boards. You load it just like loading MicroPython.
- While CircuitPython is based on MicroPython, there are some key differences.

CircuitPython Versus MicroPython

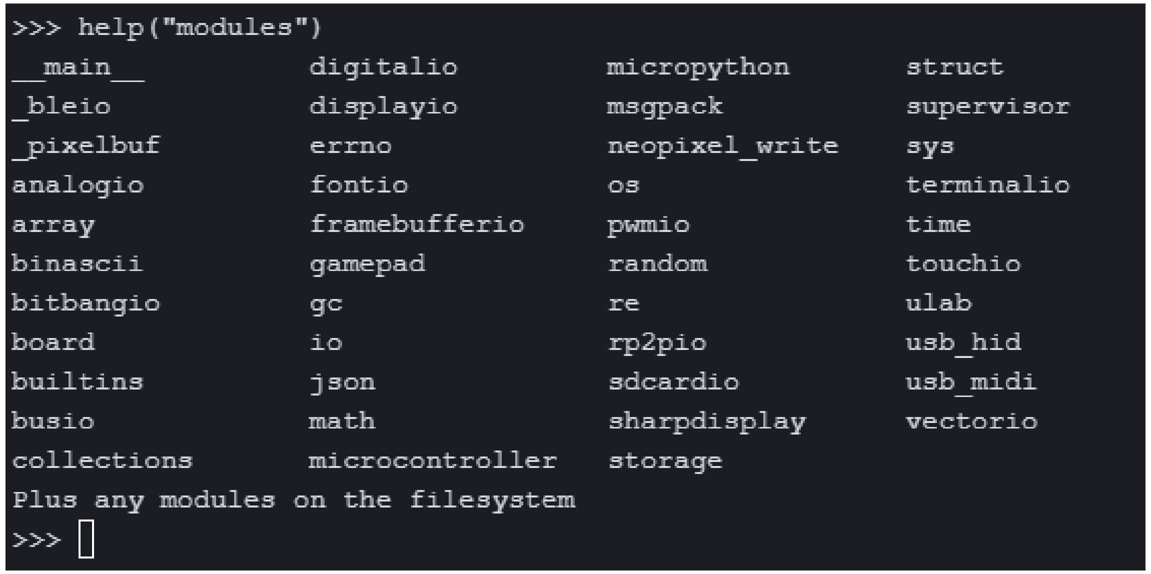
- To see all supported libraries: https://circuitpython.readthedocs.io/projects/bundle/en/latest/drivers.html
- To see all differences between CircuitPython and MicroPython: https://github.com/adafruit/circuitpython#differences-from-micropython
- To see quick start document for using Raspberry Pi Pico with CircuitPython: https://circuitpython.org/board/raspberry_pi_pico/

- You may want to use MicroPython for
  - Advanced APIs such as interrupts and threading.
  - Complete PIO API (CircuitPython's support is incomplete)
  - Using existing MicroPython code
- You may want to use CircuitPython since
  - CircuitPython was designed to have a USB disk drive that appears when you plug in the board.
  - CircuitPython will restart your code when you save files to the disk drive.
  - CircuitPython has a consistent API across all boards.
  - CircuitPython has a lot of examples and support!
- Of course, it’s great to know both! ☺

Circuitpython: Board Module & Modules



```
>>> import board
>>> dir(board)
['__class__', 'A0', 'A1', 'A10', 'A2', 'A3', 'A6', 'A7', 'A8', 'A9', 'D0', 'D1',
 'D10', 'D2', 'D3', 'D4', 'D5', 'D6', 'D7', 'D8', 'D9', 'I2C', 'MISO', 'MOSI', '
NEOPIXEL', 'NEOPIXEL_POWER', 'RX', 'SCK', 'SCL', 'SDA', 'SPI', 'TX', 'UART']
```



```
>>> help("modules")
__main__          digitalio         micropython       struct
_bleio            displayio         msgpack           supervisor
_pixelbuf         errno             neopixel_write    sys
analogio          fontio            os                terminalio
array             framebufferio     pwmio             time
binascii          gamepad           random            touchio
bitbangio         gc                re                ulab
board             io                rp2pio            usb_hid
builtins          json              sdcardio          usb_midi
busio             math              sharpdisplay      vectorio
collections       microcontroller   storage
Plus any modules on the filesystem
>>> 
```

Thanks for listening ☺

YALÇIN İŞLER

Assoc. Prof.