Microcontrollers and Robotic

Week 2: MCU Architecture

Fenerbahçe University

Professor & TAs

Prof: Dr. Vecdi Emre Levent

Office: 311

Email: emre.levent@fbu.edu.tr

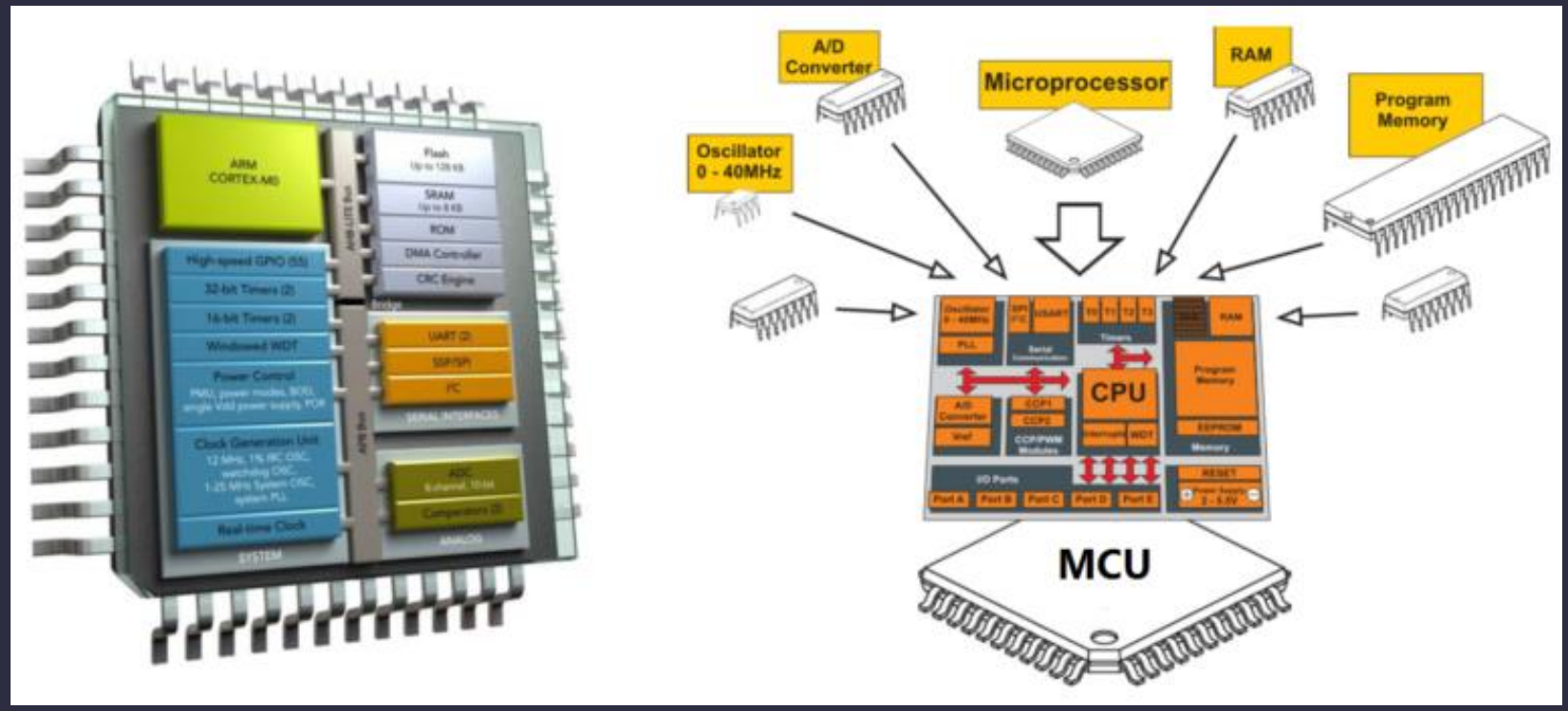
TA: Arş. Gör. Ezgi Çakmak

Office: 311

Email: ezgi.cakmak@fbu.edu.tr

MCU Architecture

- A microcontroller (MCU for microcontroller unit) is a small computer on a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip.



MCU Architecture

- A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals.

- Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

MCU Architecture

There are lost of MCU Vendors

- Cypress Semiconductor
- Infineon Technologies
- Microchip Technology
- NXP Technologies
- Renesas Electronics
- STMicoelectronics

MCU Architecture

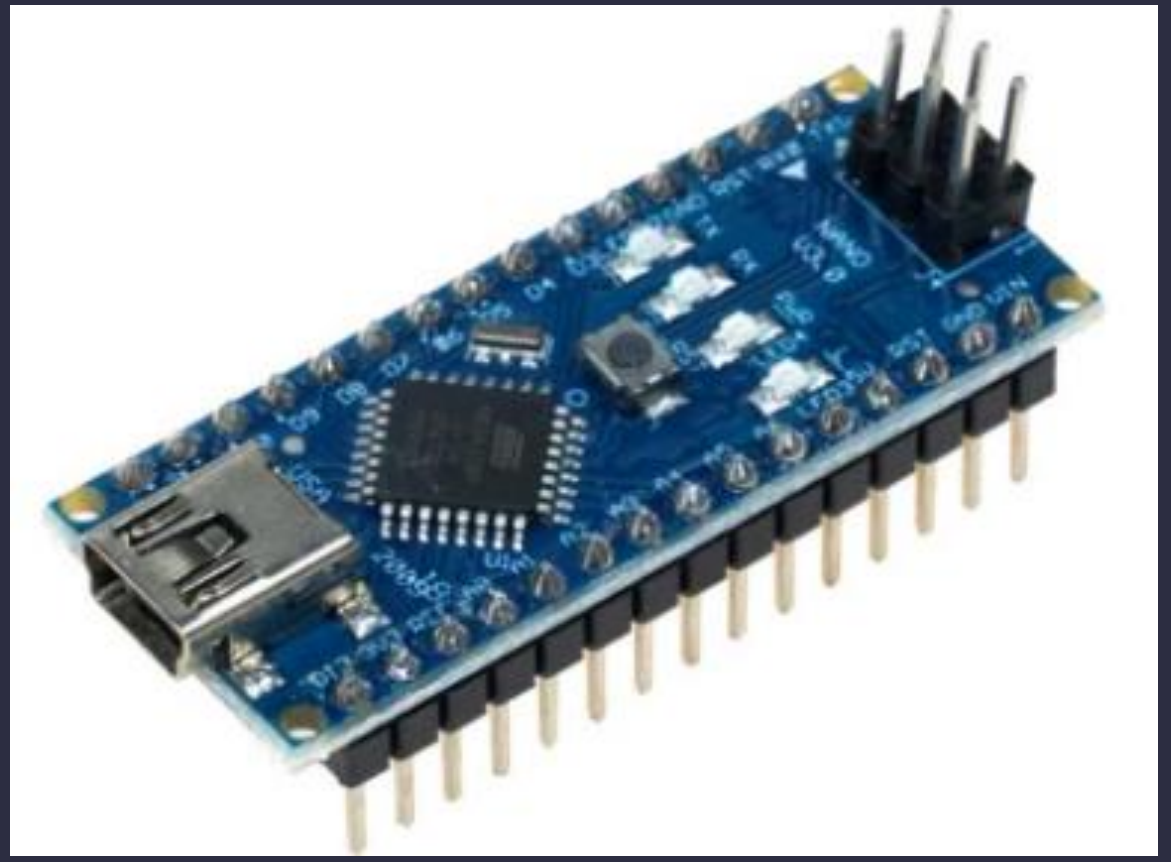
There are lost of MCU Vendors

- Cypress Semiconductor
- Infineon Technologies
- Microchip Technology
- NXP Technologies
- Renesas Electronics
- STMicoelectronics

We will use MicroChip Technology based MCUs

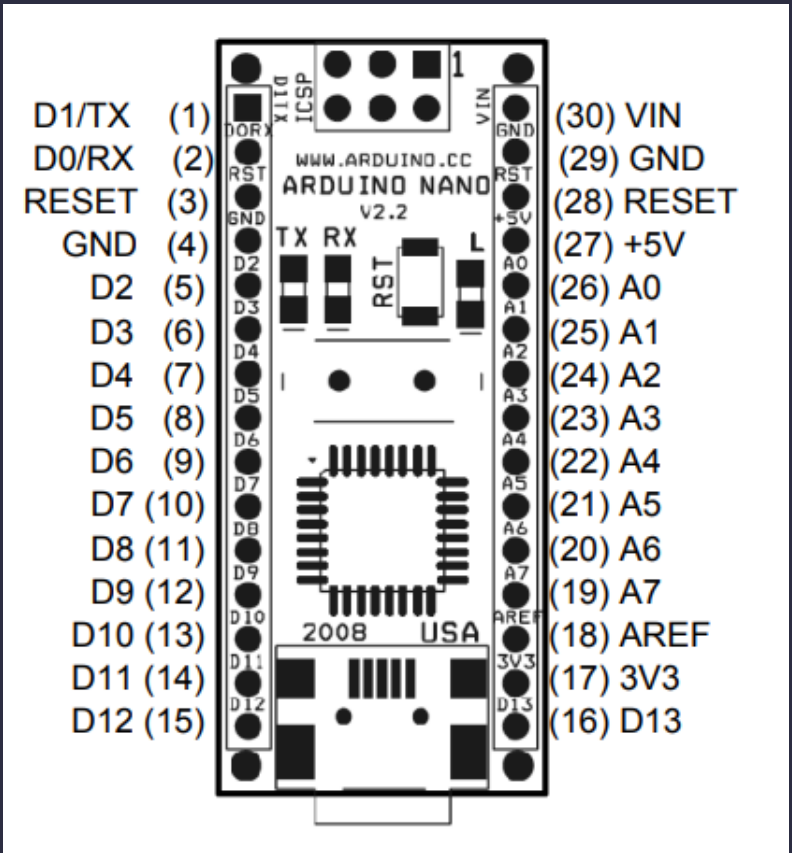
MCU Architecture

- Ardunio Nano



MCU Architecture

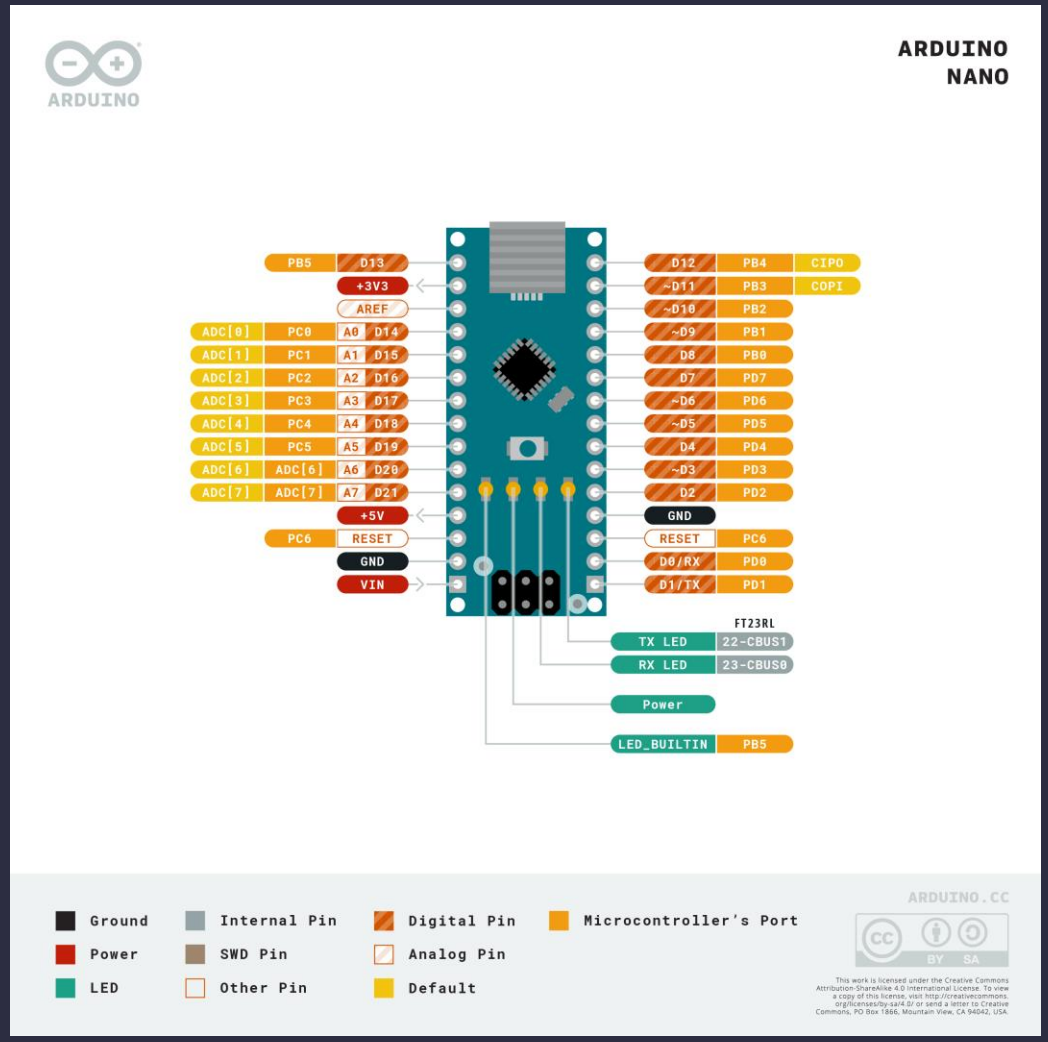
- Ardunio Nano



Pin No.	Name	Type	Description
1-2, 5-16	D0-D13	I/O	Digital input/output port 0 to 13
3, 28	RESET	Input	Reset (active low)
4, 29	GND	PWR	Supply ground
17	3V3	Output	+3.3V output (from FTDI)
18	AREF	Input	ADC reference
19-26	A7-A0	Input	Analog input channel 0 to 7
27	+5V	Output or Input	+5V output (from on-board regulator) or +5V (input from external power supply)
30	VIN	PWR	Supply voltage

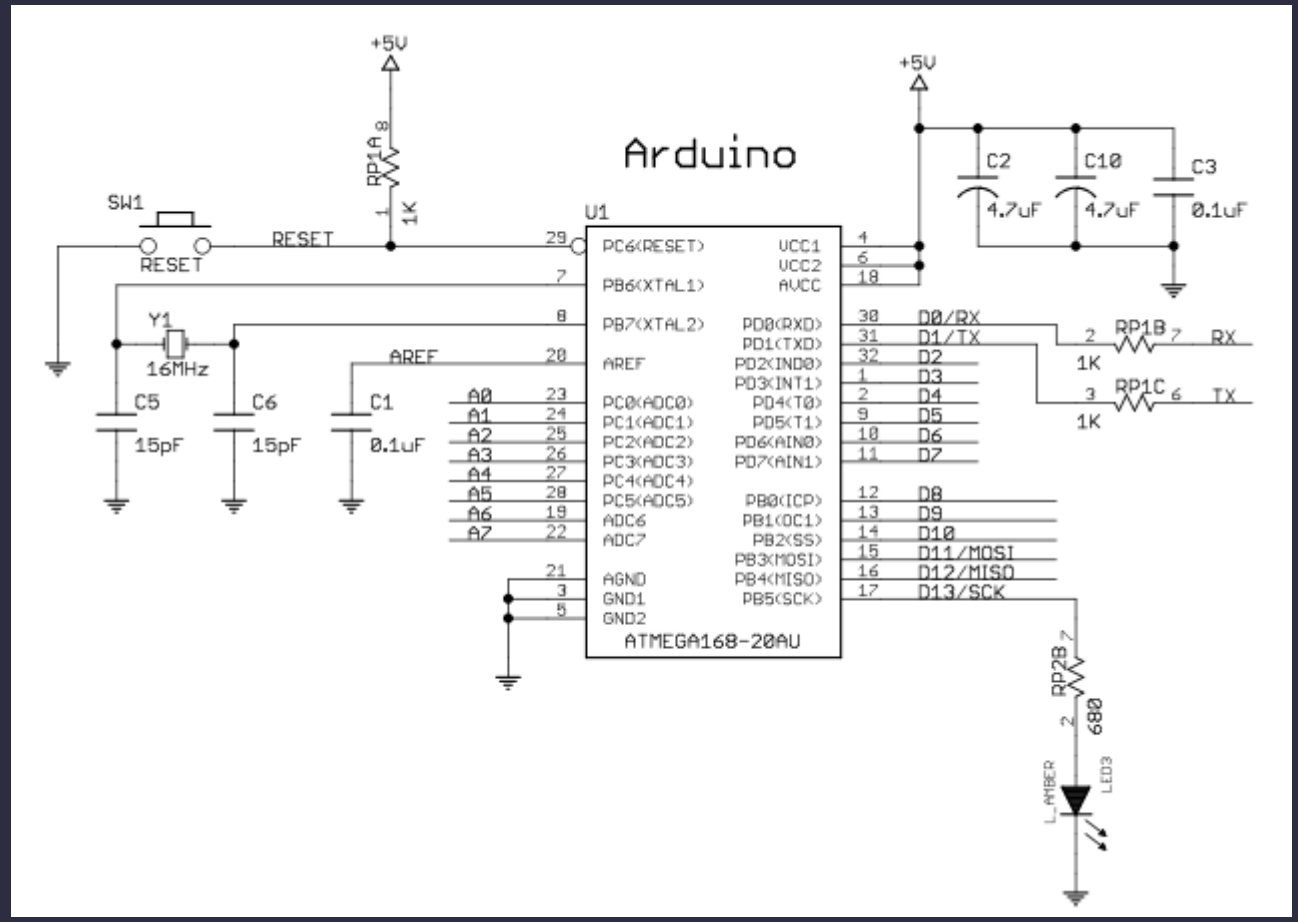
MCU Architecture

- Ardunio Nano



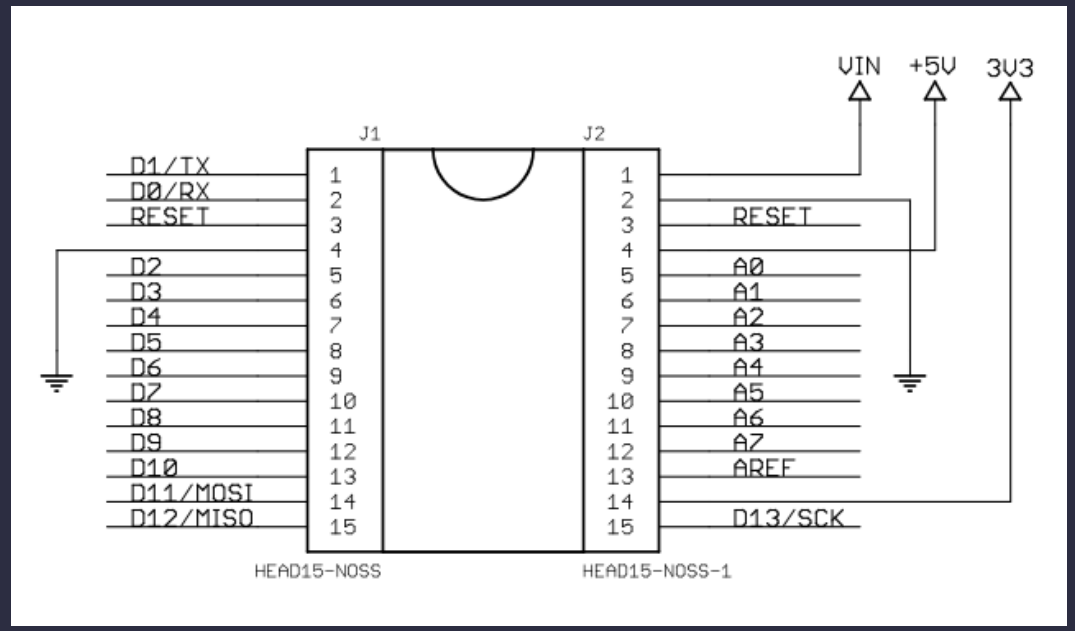
MCU Architecture

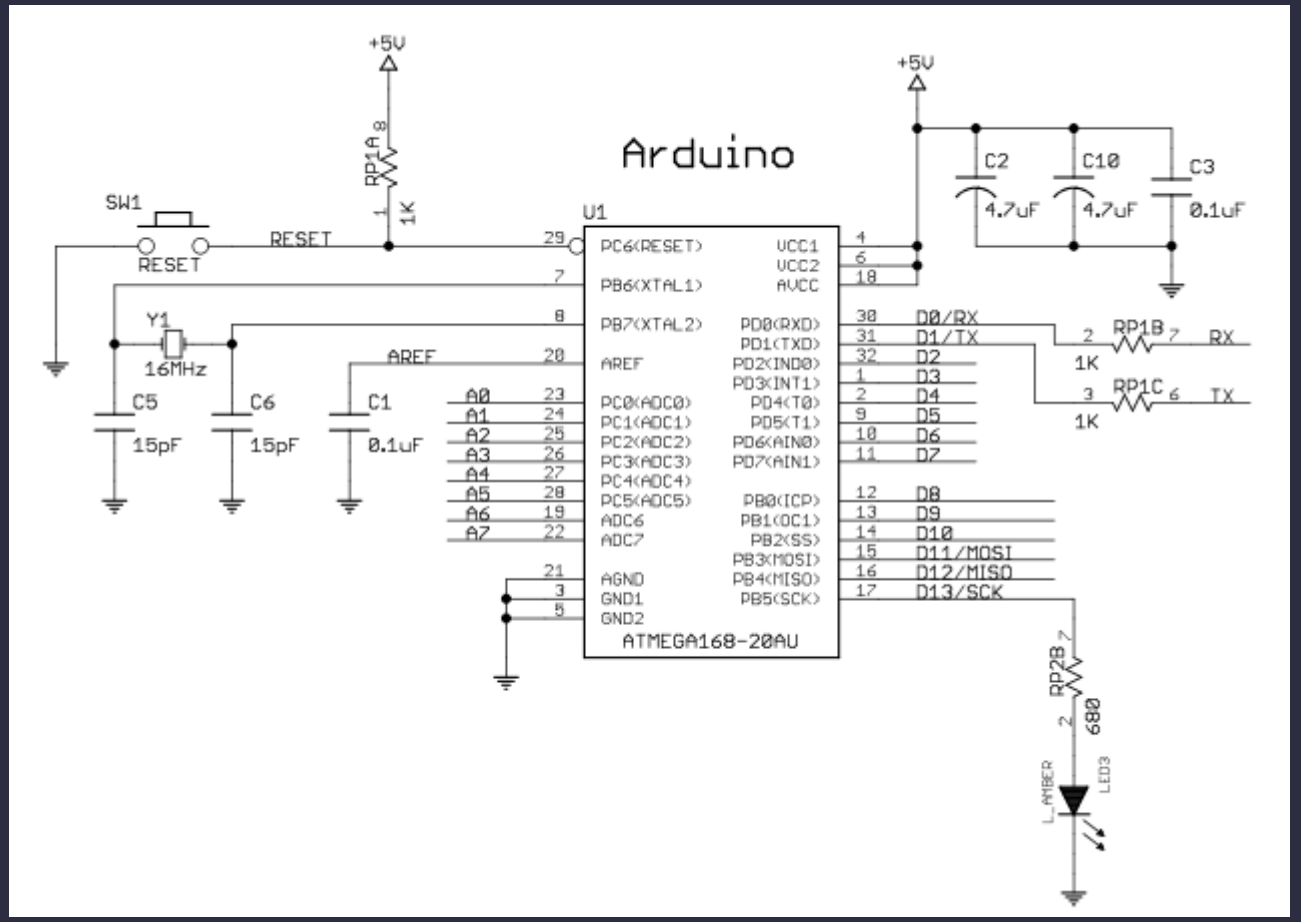
- Ardunio Nano Schematics



MCU Architecture

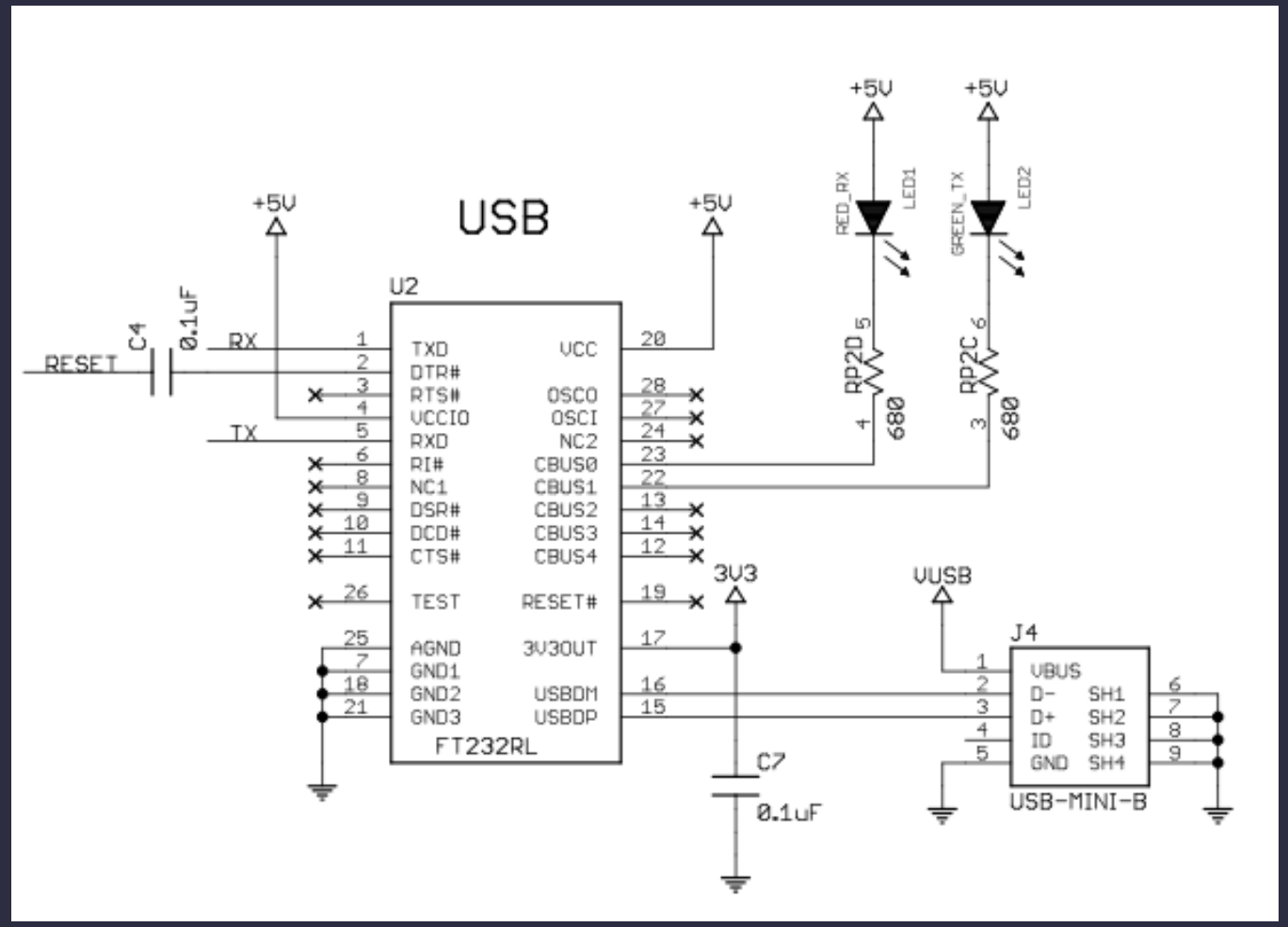
- Ardunio Nano Schematics





MCU Architecture

- Ardunio Nano Schematics



MCU Architecture

- Ardunio Nano

ATmega328 microcontroller

The ATMega328 CPU runs with 16 MHz and features 32 KB of Flash Memory (of which 2 KB used by bootloader).

Communication	UART	Yes
	I2C	Yes
	SPI	Yes

MCU Architecture

- Ardunio Nano

ATmega328 microcontroller

- 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash
- Advanced RISC architecture
  - 131 powerful instructions – most single clock cycle execution
  - 32 x 8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier

MCU Architecture

- Ardunio Nano

ATmega328 microcontroller

Peripheral features

- Two 8-bit Timer/Counters with separate prescaler and compare mode
- One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
- Real time counter with separate oscillator
- Six PWM channels
- 8-channel 10-bit ADC in TQFP and QFN/MLF package
- Temperature measurement

Low power consumption

- Active mode: 1.5mA at 3V - 4MHz
- Power-down mode: 1µA at 3V

- Programmable serial USART
- Master/slave SPI serial interface
- Byte-oriented 2-wire serial interface (Phillips I2 C compatible)
- Programmable watchdog timer with separate on-chip oscillator
- On-chip analog comparator
- Interrupt and wake-up on pin change

MCU Architecture

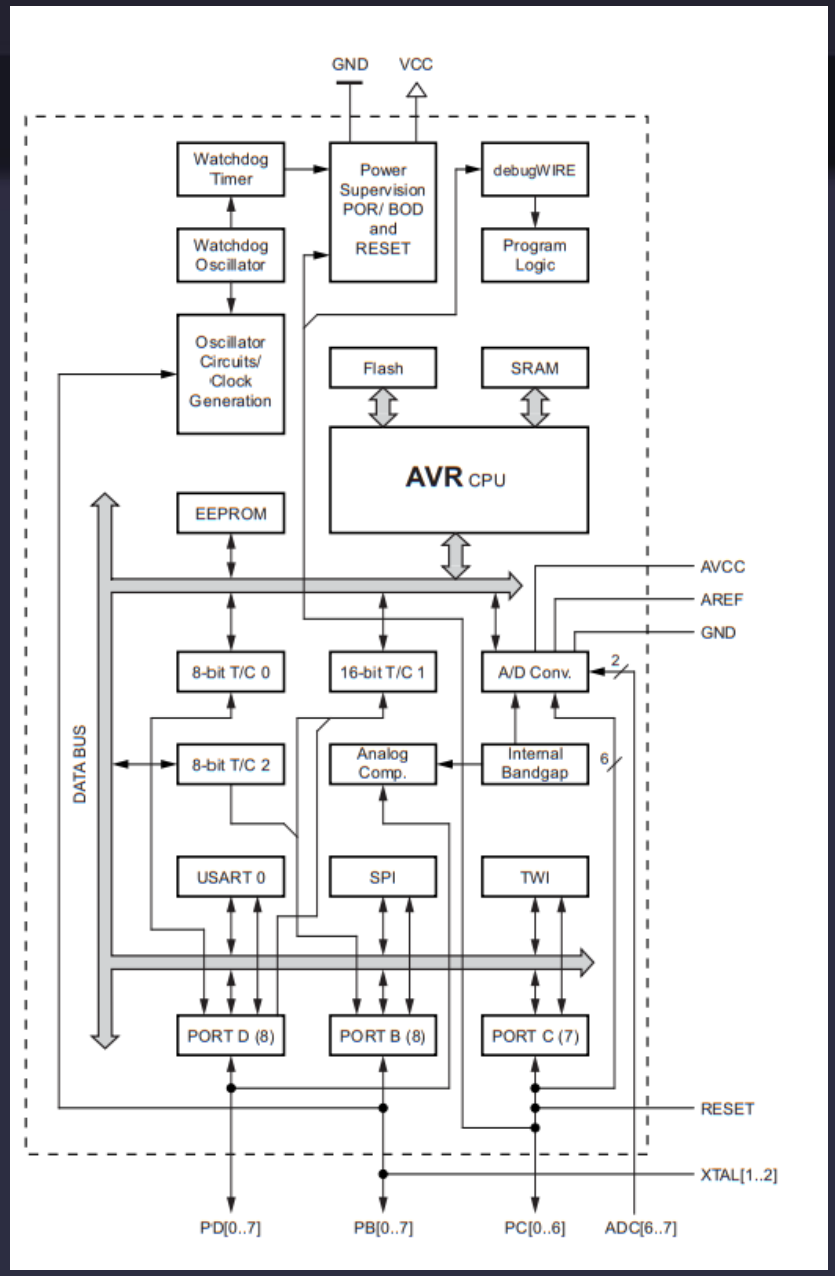
- Ardunio Nano

ATmega328 microcontroller

The Atmel® ATmega328P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By

executing powerful instructions in a single clock cycle, the ATmega328P achieves throughputs approaching 1MIPS per MHz

allowing the system designer to optimize power consumption versus processing speed.



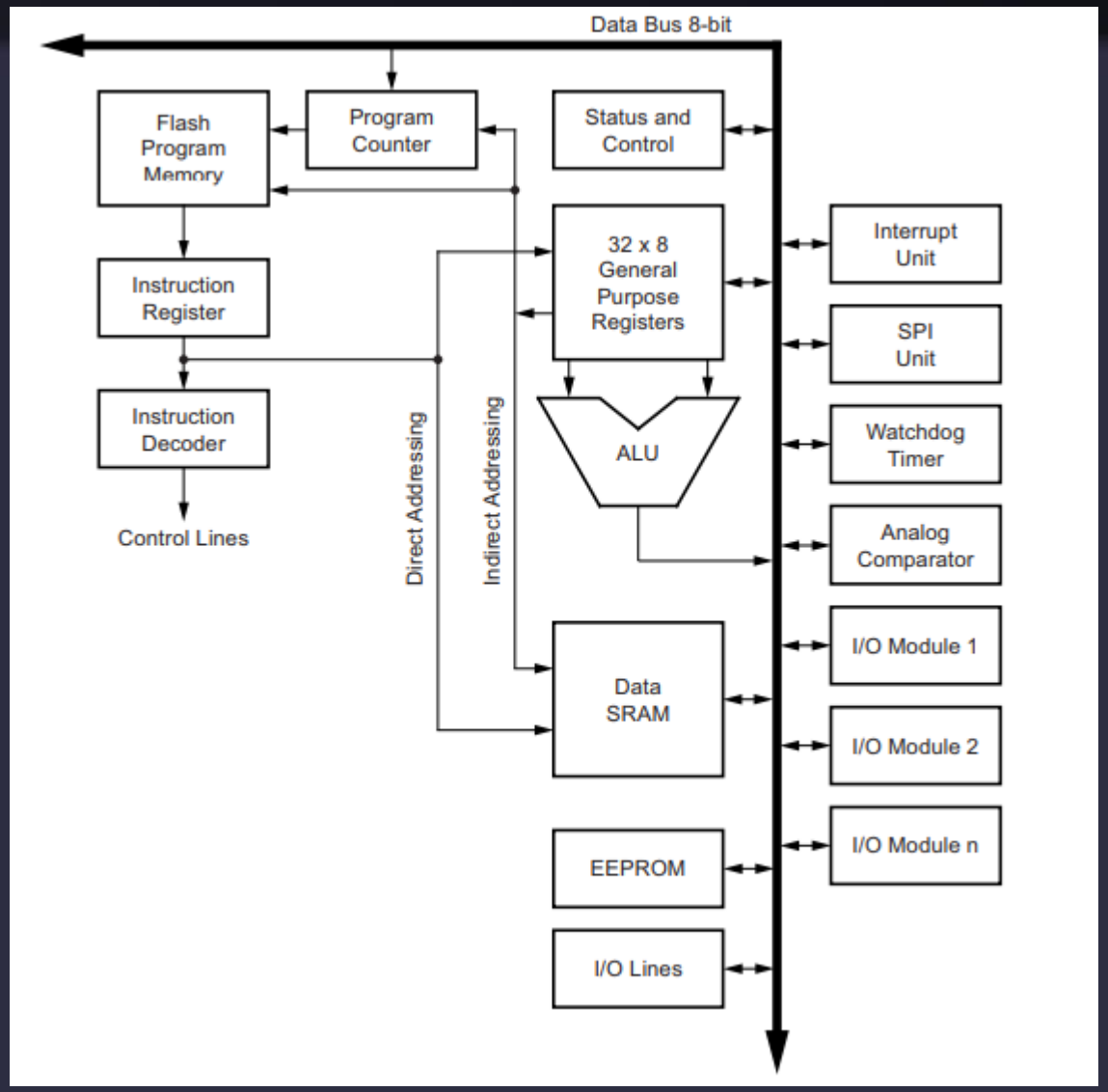
MCU Architecture

- Ardunio Nano

ATmega328 microcontroller CPU Core

The main function of the CPU core is to ensure correct program execution.

The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

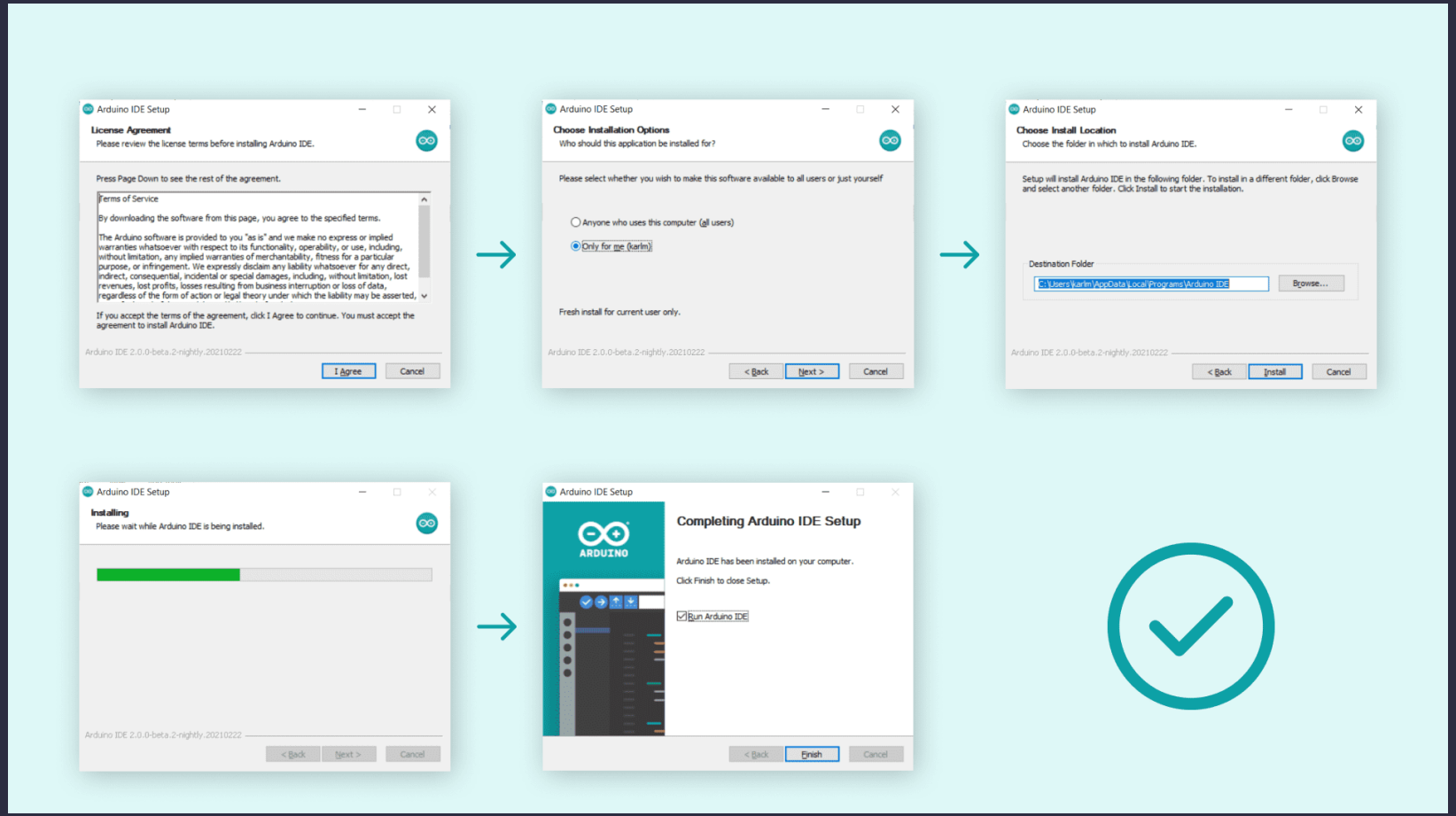


MCU Architecture

- Ardunio IDE

Download and Install

https://www.arduino.cc/en/software#experimental-software

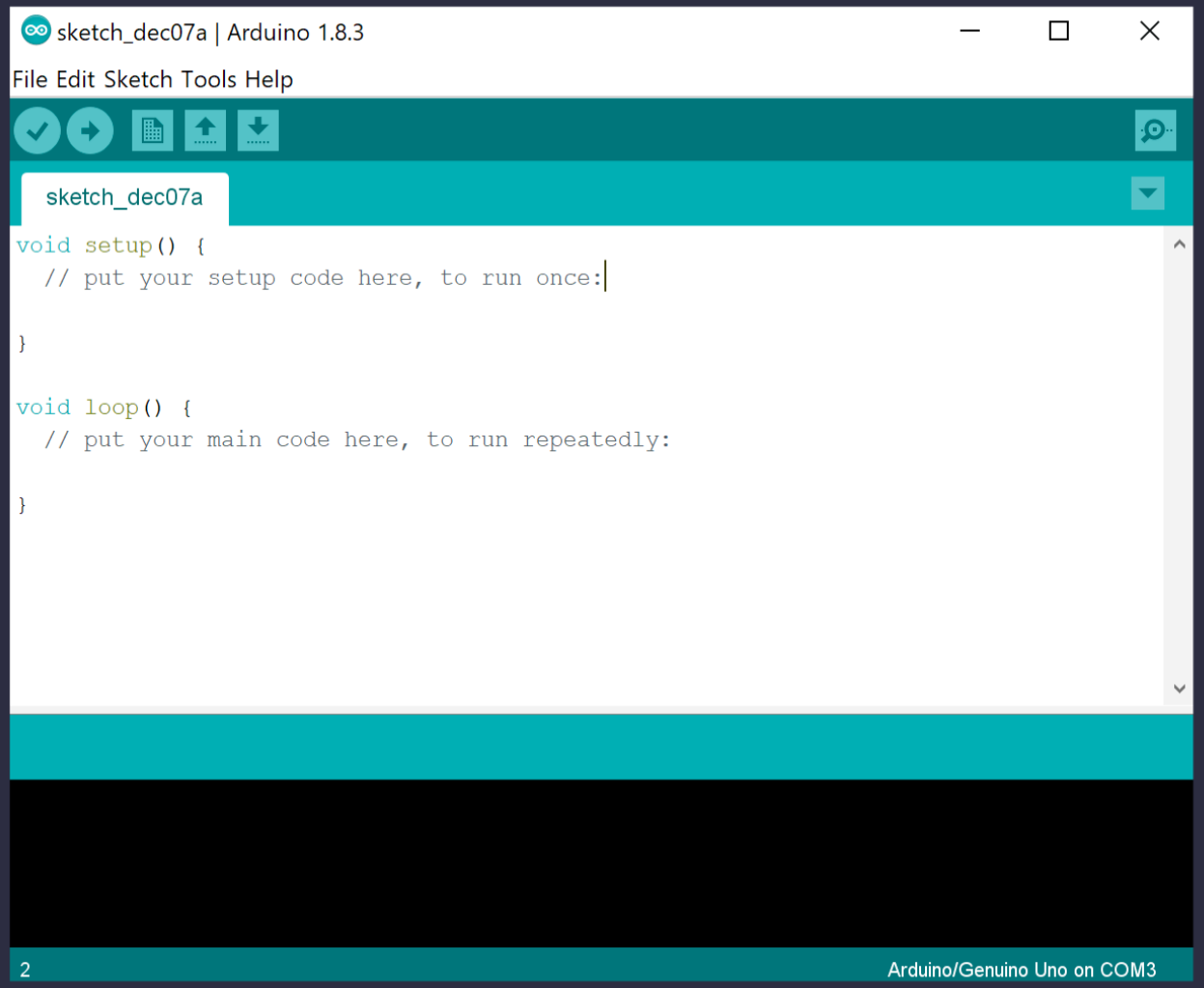


MCU Architecture

- Ardunio IDE

There are two main tools when uploading a sketch to a board: verify and upload.

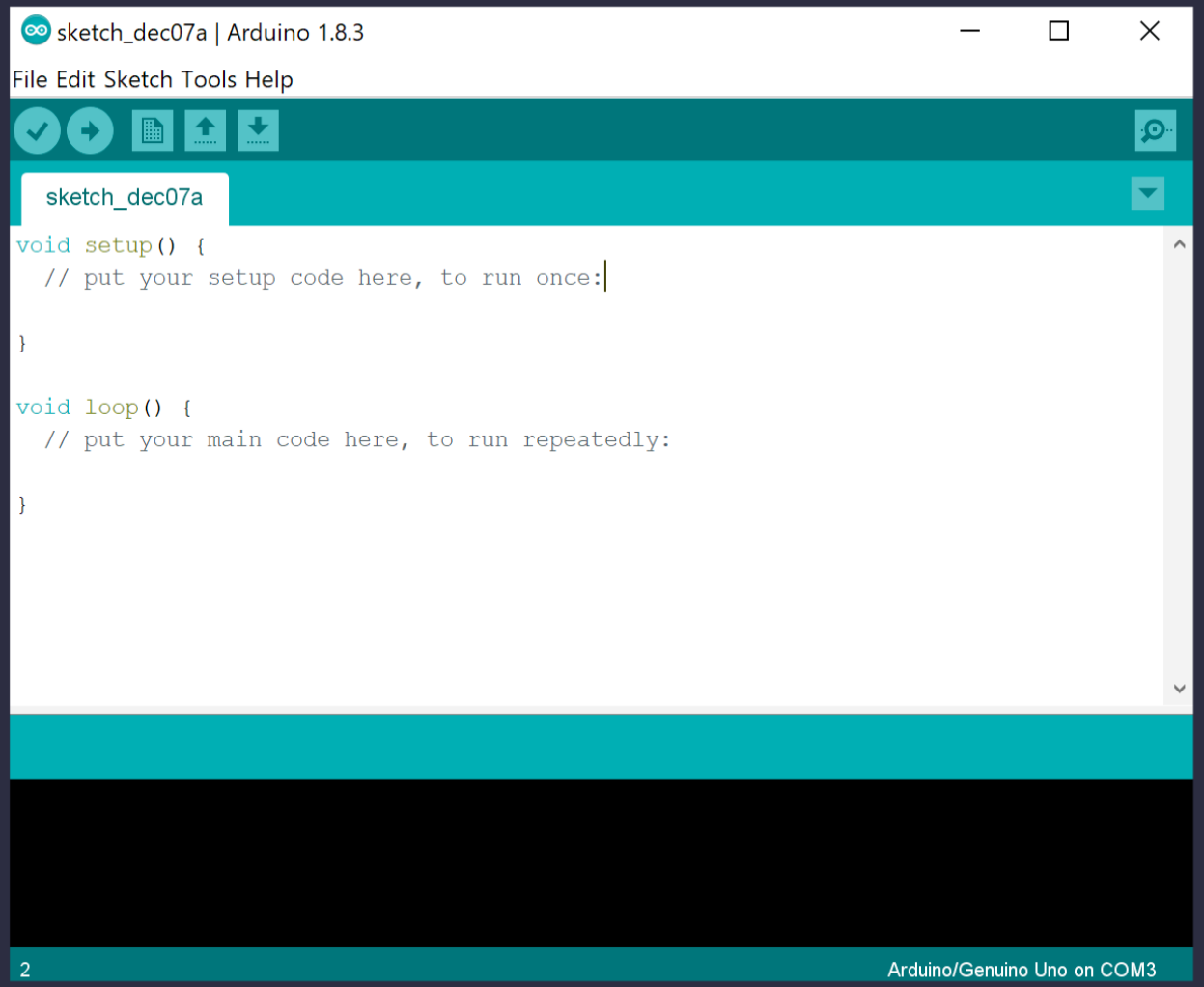
- The verify tool simply goes through your sketch, checks for errors and compiles it.
- The upload tool does the same, but when it finishes compiling the code, it also uploads it to the board



MCU Architecture

- Ardunio IDE

A good practice is to use the verifying tool before attempting to upload anything. This is a quick way of spotting any errors in your code, so you can fix them before actually uploading the code.



MCU Architecture

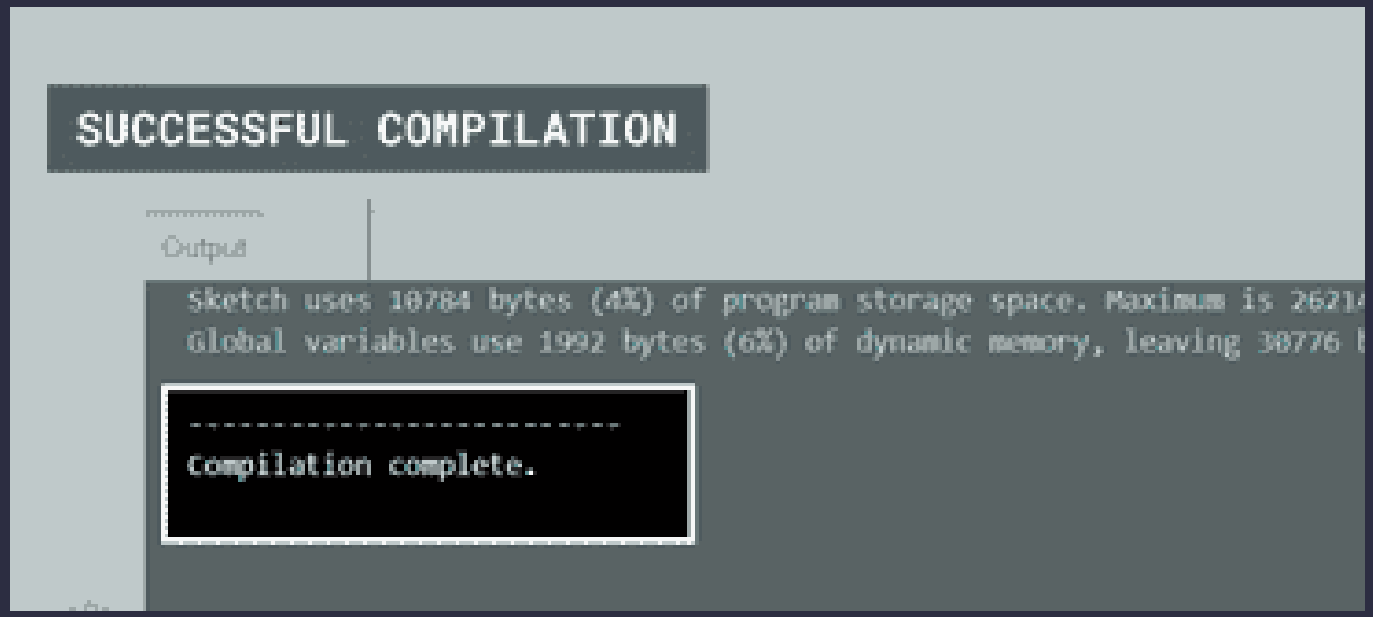
- Ardunio IDE

At the very left, there is a checkmark and an arrow pointing right. The checkmark is used to verify, and the arrow is used to upload.

MCU Architecture

- Ardunio IDE

Click on the verify tool (checkmark).

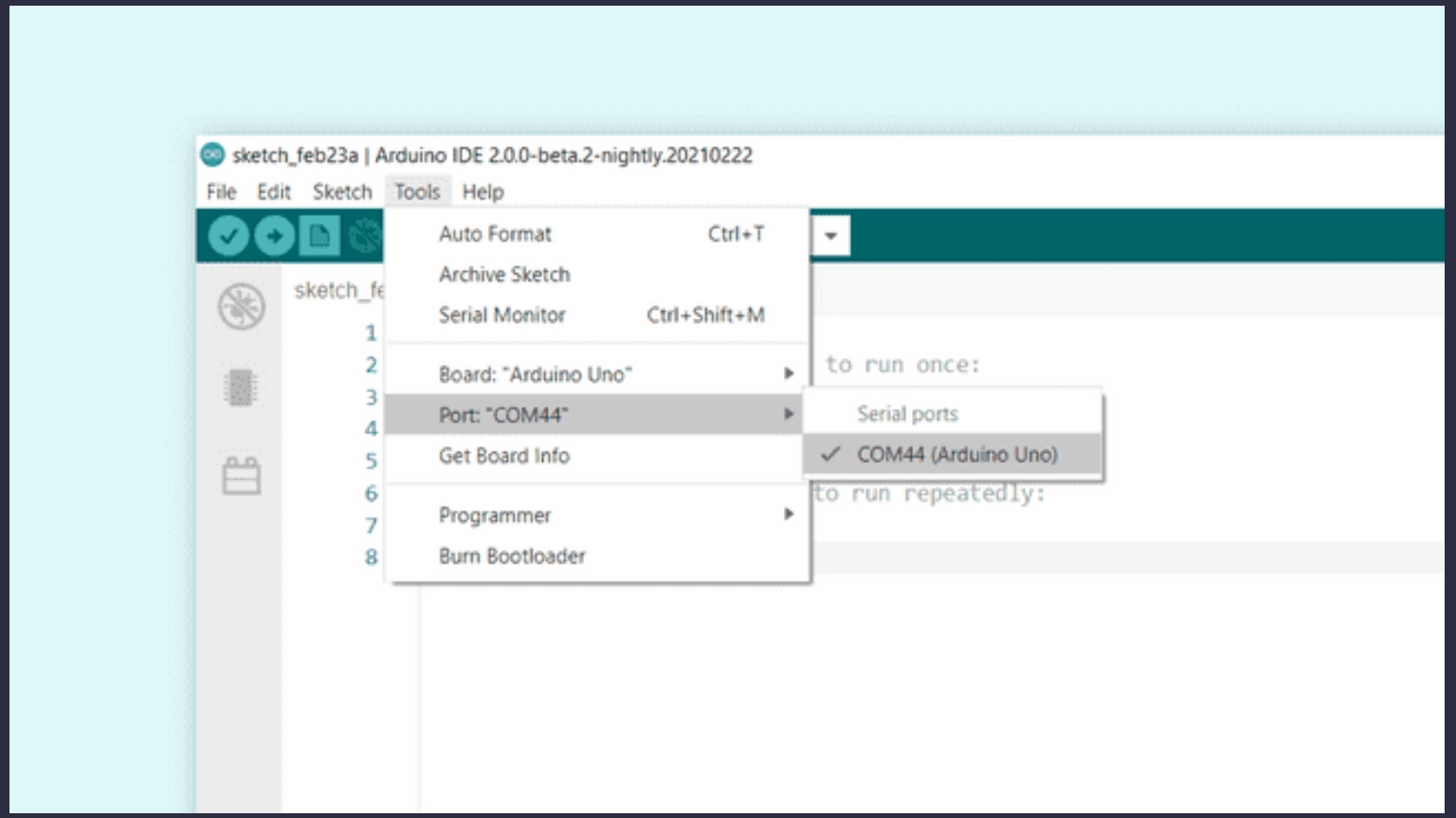


MCU Architecture

- Ardunio IDE

Select the board that we are using

Tools > Port > {Board}

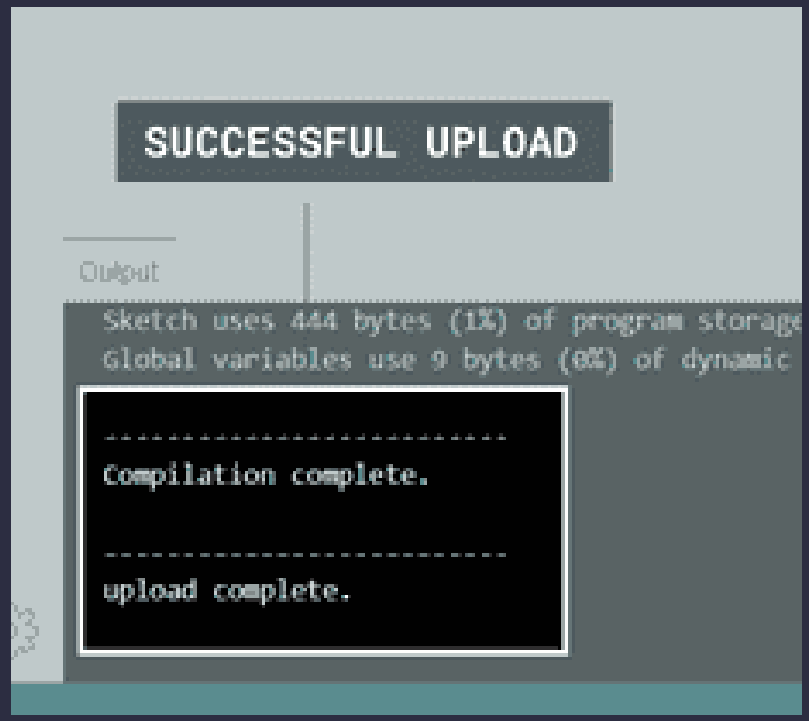


MCU Architecture

- Ardunio IDE

Click on the upload button, and it will start uploading the sketch to the board

When it is finished, it will notify you in the console log.

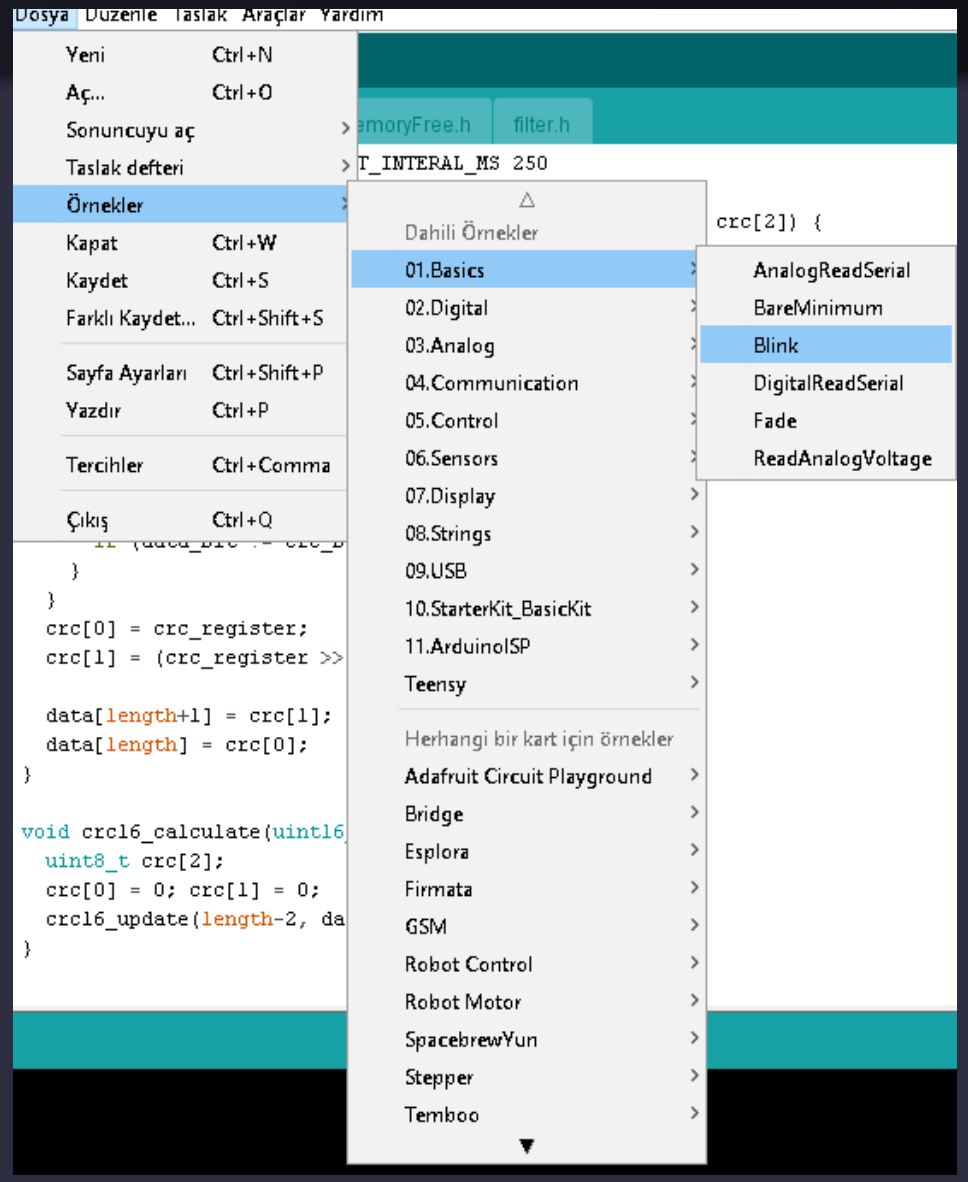


MCU Architecture

- Ardunio IDE

Examples under

File -> Examples ->

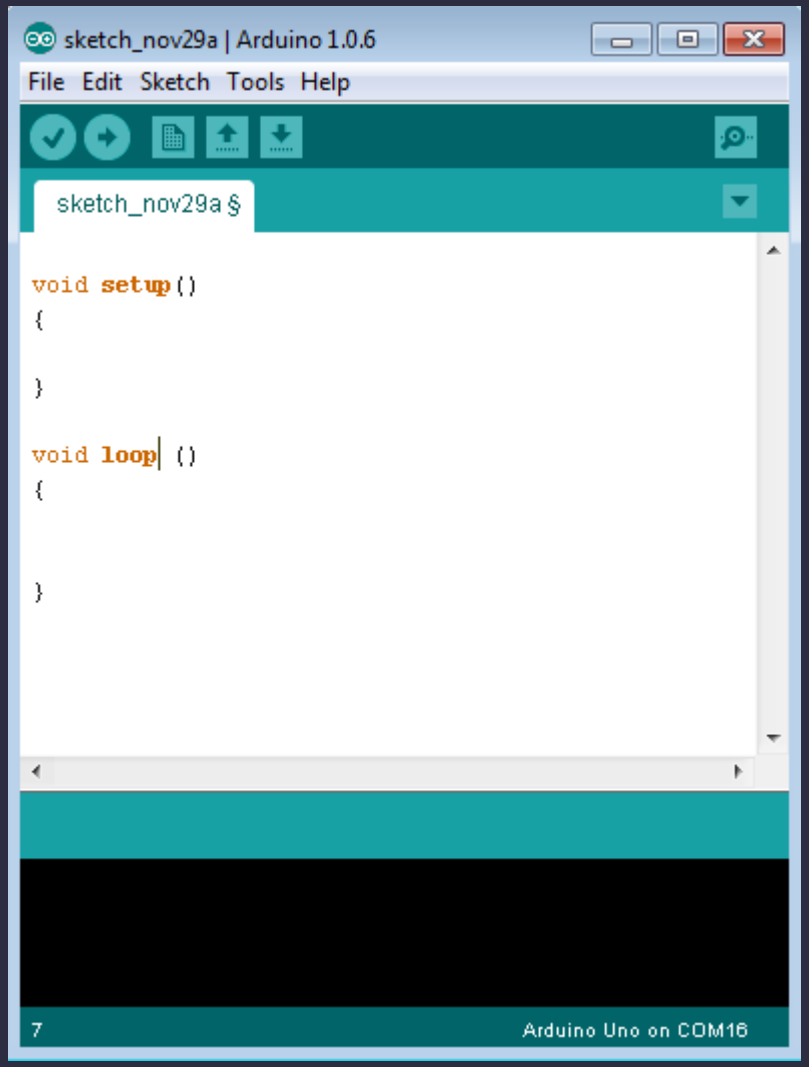


MCU Architecture

- Ardunio IDE

Software structure consist of two main functions

- Setup( ) function
- Loop( ) function



MCU Architecture

- Ardunio IDE

```
void setup ( ) {


}
```

PURPOSE – The setup() function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

MCU Architecture

- Ardunio IDE

```
void loop ( ) {

}
```

PURPOSE – After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

MCU Architecture

- Ardunio IDE

Data types

void	**Boolean**	**char**	**Unsigned char**	**byte**	**int**	**Unsigned int**	**word**
long	**Unsigned long**	**short**	**float**	**double**	**array**	**String-char array**	**String-object**

MCU Architecture

- Ardunio IDE

Arduino provides four different time manipulation functions

	Function & Description
1	delay () function The way the **delay()** function works is pretty simple. It accepts a single integer (or number) argument. This number represents the time (measured in milliseconds).
2	delayMicroseconds () function The **delayMicroseconds()** function accepts a single integer (or number) argument. There are a thousand microseconds in a millisecond, and a million microseconds in a second.
3	millis () function This function is used to return the number of milliseconds at the time, the Arduino board begins running the current program.
4	micros () function The micros() function returns the number of microseconds from the time, the Arduino board begins running the current program. This number overflows i.e. goes back to zero after approximately 70 minutes.

MCU Architecture

- Ardunio IDE

The pins on the Arduino board can be configured as either inputs or outputs.

pinMode() function can set a pin to input or output

pinMode(3,INPUT) ; // set pin to input without using built in pull up resistor

MCU Architecture

- Ardunio IDE

The pins on the Arduino board can be configured as either inputs or outputs.

```
int led = 5 ; // led connected to pin 5

void setup () {
  pinMode(led , OUTPUT); // set the digital pin as output
}
void setup () {
     digitalWrite(led,HIGH); // turn on led
    delay(500); // delay for 500 ms
    digitalWrite(led,LOW); // turn off led
    delay(500); // delay for 500 ms
}
```

MCU Architecture

- Ardunio IDE

digitalWrite() Function

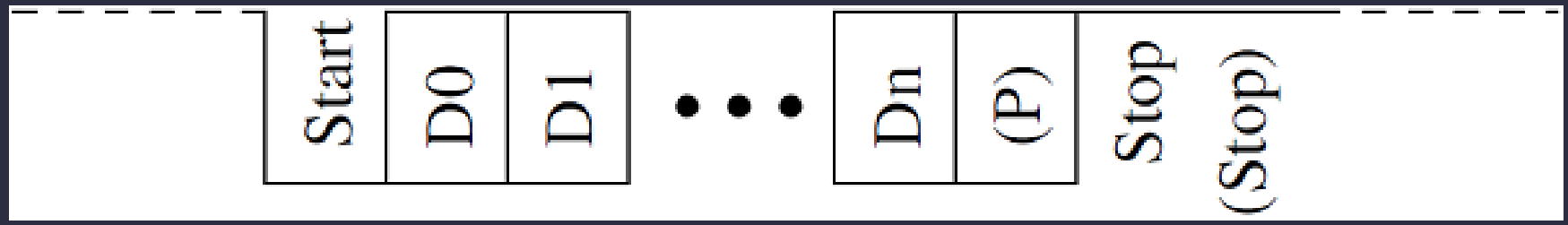
The digitalWrite() function is used to write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode()

its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

MCU Architecture

- Ardunio IDE

UART (Universal Asynchronous Receiver Transmitter)



MCU Architecture

- Ardunio IDE

UART (Universal Asynchronous Receiver Transmitter)

```
void setup() {
  Serial.begin(9600); //set up serial library baud rate to 9600
  Serial.println("hello world"); //print hello world
}

void loop() {

}
```

MCU Architecture

- Ardunio IDE

UART (Universal Asynchronous Receiver Transmitter)

```
void setup() {
  Serial.begin(9600); //set up serial library baud rate to 9600
}

void loop() {
  if(Serial.available()){
    Serial.print("I received:"); //print I received
    Serial.write(Serial.read()); //send what you read
  }
}
```