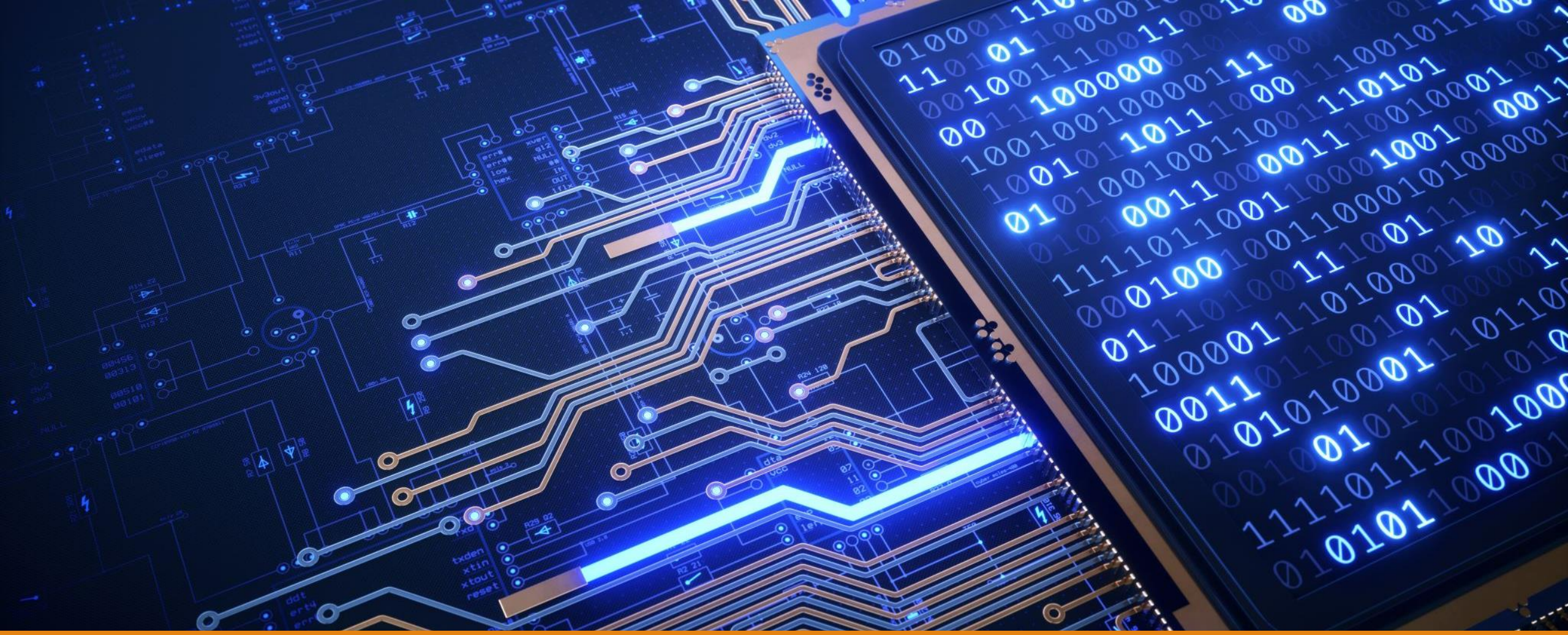


Embedded Systems

Dr. Abdelhay Ali

Lecture 1

Introduction

Outlines

1. Course Information
2. Course outline
3. Introduction to Embedded System
4. AVR Family
5. Arduino

Outlines

1. Course Information
2. Course outline
3. Introduction to Embedded System
4. AVR Family
5. Arduino

Course Information

- Instructor Dr Abdelhay Ali
- Abdelhay@eng.aun.edu.eg
- 2nd floor EE department
- Office: Second Floor EE dept, Faculty of Engineering, Assiut university, Egypt.

- Lectures: Thursday 9AM :12 PM
- Lab:-

- Midterm (Quizzes, Exam and PROJECTs)
- Final Exam (Open Book Exam)

Outlines

1. Course Information
2. Course outline
3. Introduction to Embedded System
4. AVR Family
5. Arduino

Course outline

- Introduction to embedded systems
- PWM and ADC
- UART, I2C, SPI
- Project 1: CNC and 3D Printer
- Internal and Ex. Interrupt
- Timers and Counters
- ESP8266 Wi-Fi microchip
- Project 2: Smart Home
- LabVIEW
- FPGA and ASIC Design for Embedded Systems

Outlines

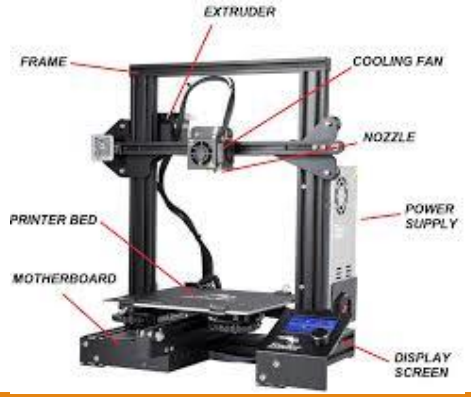
1. Course Information
2. Course outline
3. Introduction to Embedded System
4. AVR Family
5. Arduino

Introduction to Embedded System

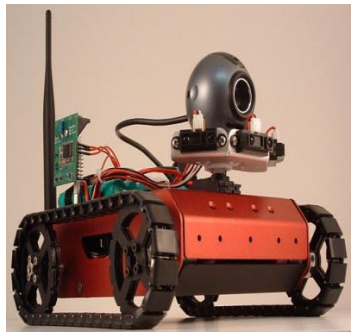
- **What is Embedded System?**

An Electronic/Electro mechanical system which is designed to perform a specific function and is a combination of both hardware and firmware (Software)

E.g. Electronic Toys, Mobile Handsets, Washing Machines, Air Conditioners, Automotive Control Units, Set Top Box, DVD Player etc…




CNC Machine Prototype used as the testing

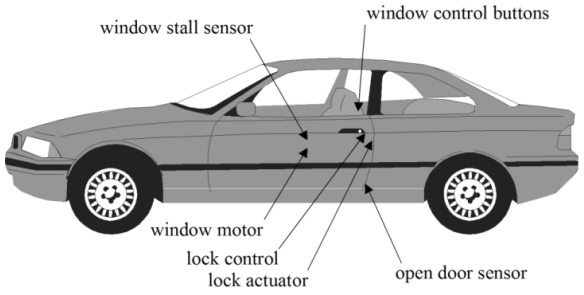


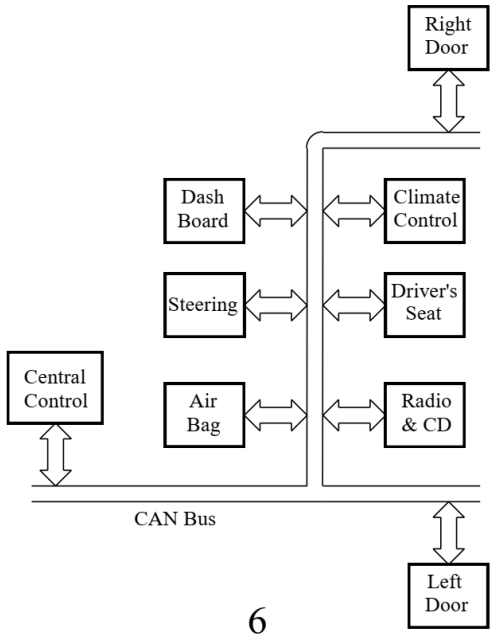
Introduction to Embedded System

Embedded System is a **special-purpose computer system** designed to perform one or a few dedicated functions.

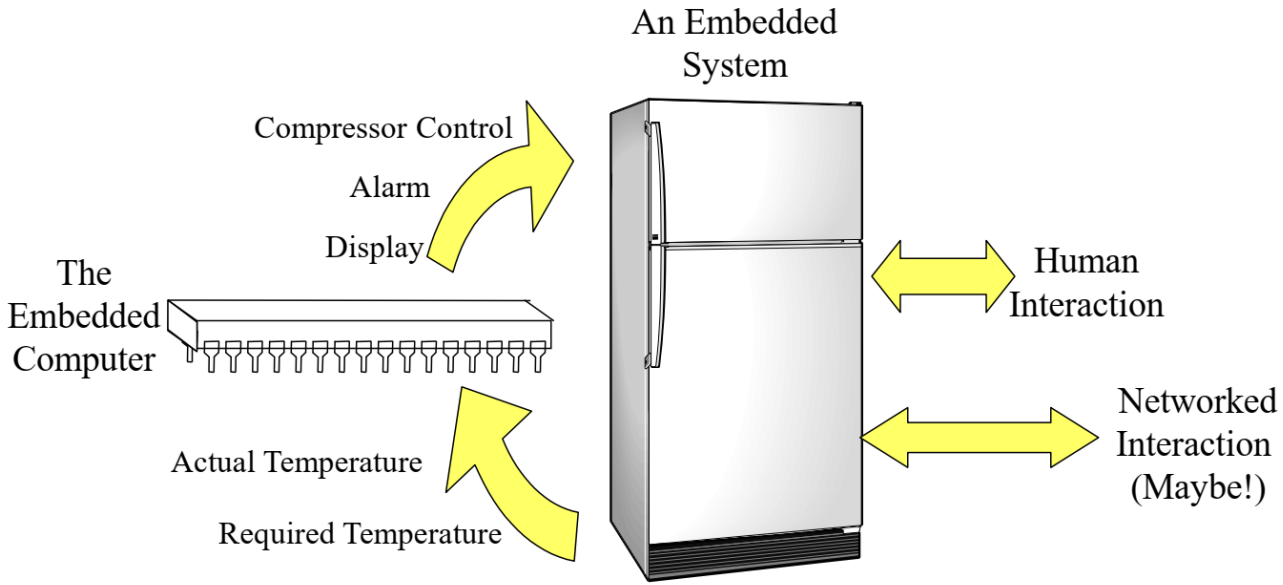
In general, it does not provide programmability to users, as opposed to general purpose computer systems like PC

Introduction to Embedded System



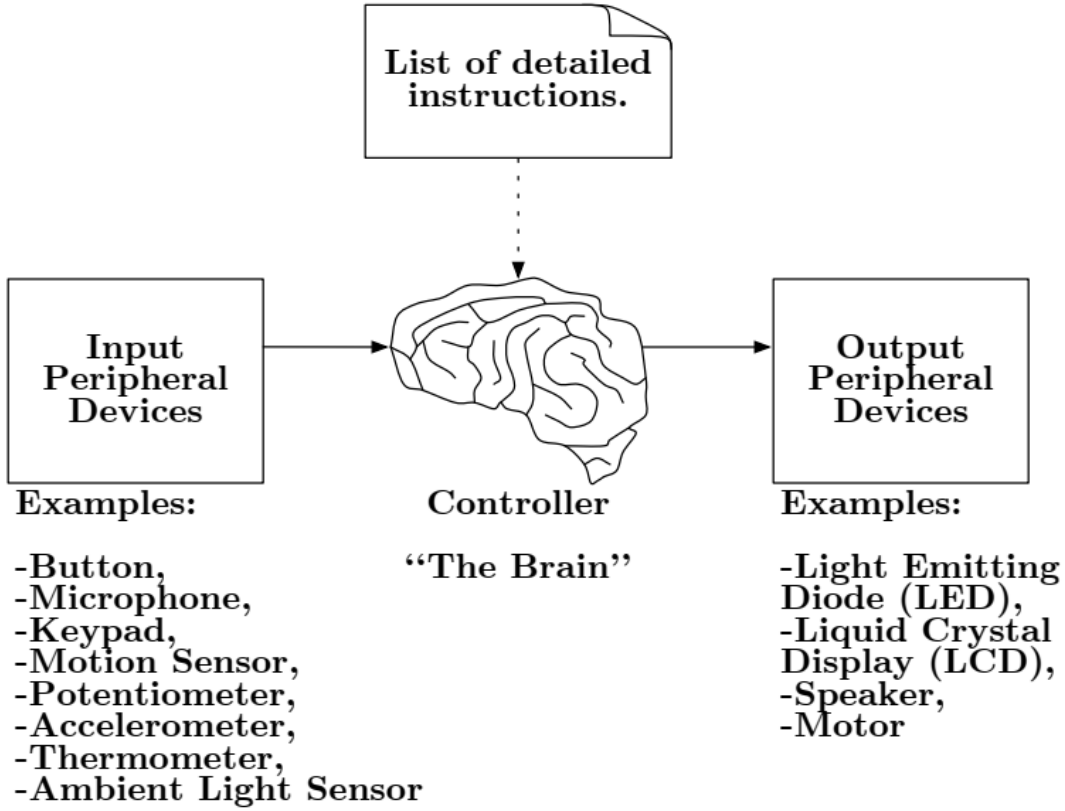


6



Introduction to Embedded System

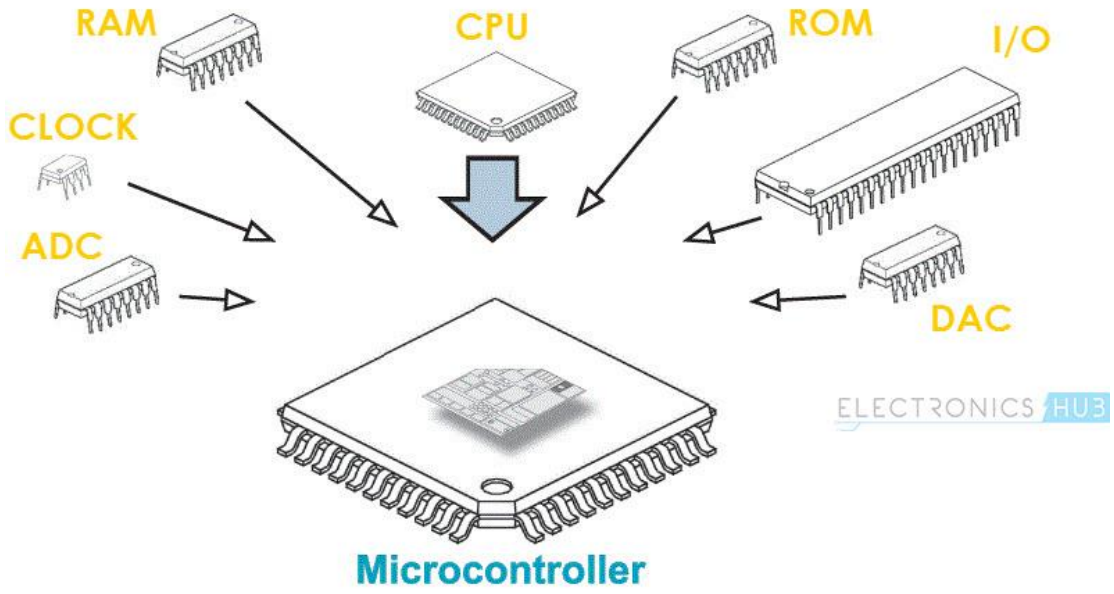
- **Conceptual embedded system block diagram.**



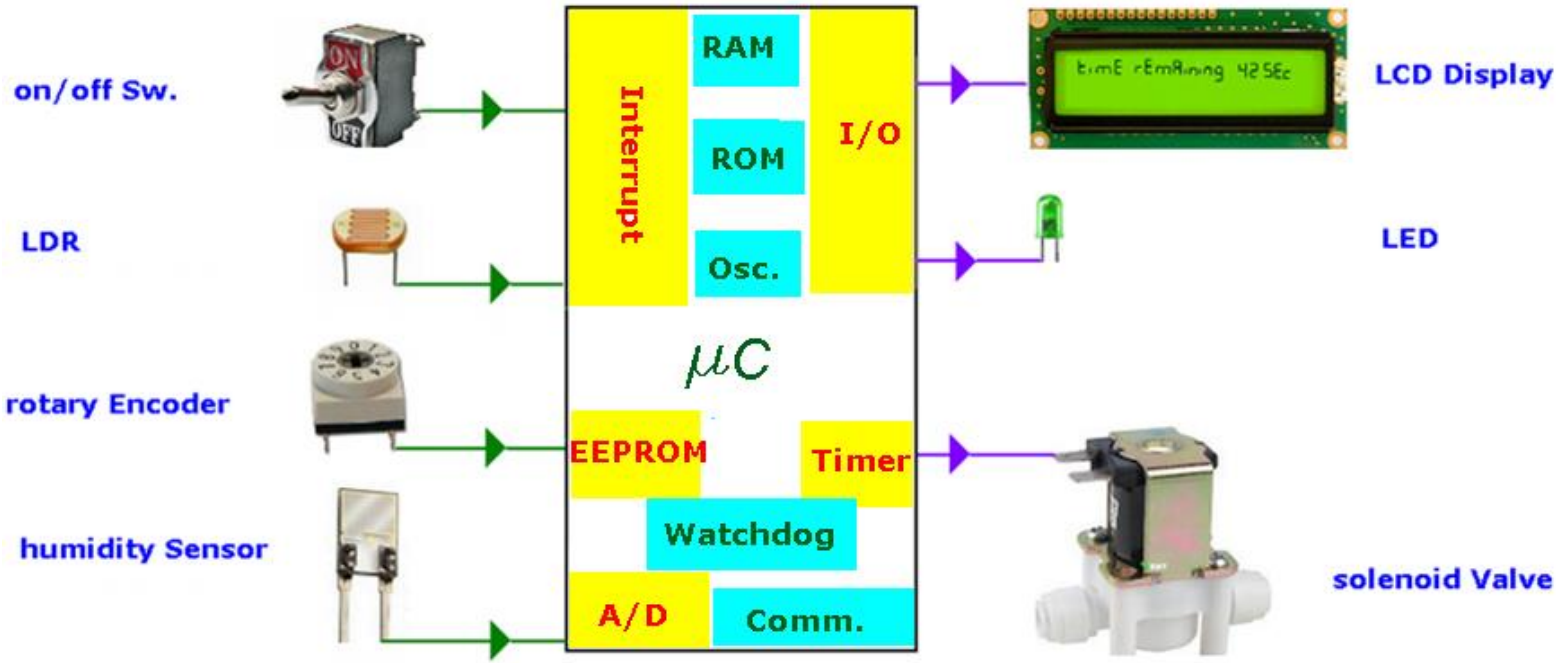
Introduction to Embedded System

❑Microcontroller

A microcontroller (MCU for microcontroller unit) is a small computer on a single integrated circuit (IC) chip. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals



Introduction to Embedded System



Outlines

1. Course Information
2. Course outline
3. Introduction to Embedded System
4. AVR Family
5. Arduino

AVR Family

AVR Microcontroller

AVR stand for?

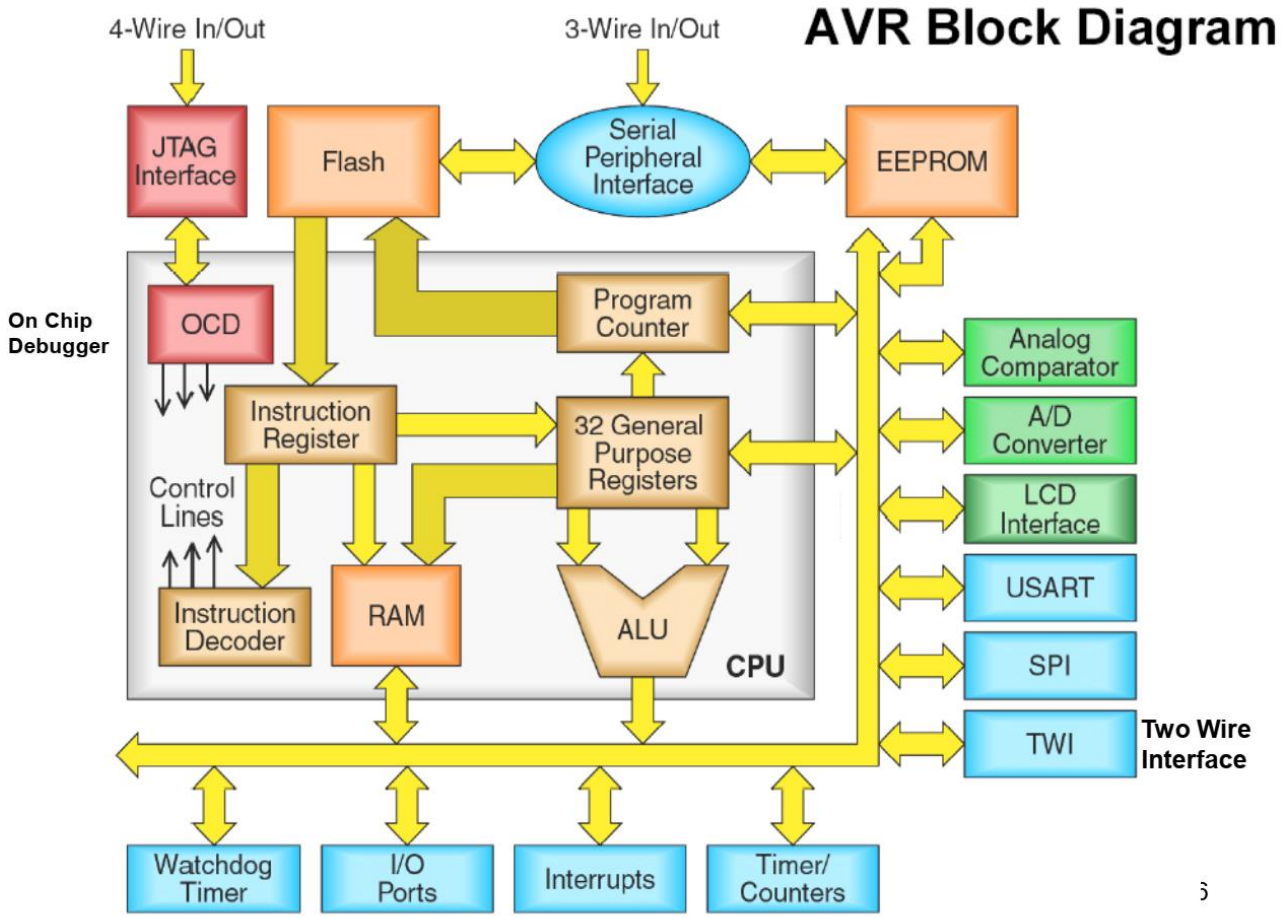
- Advanced Virtual RISC,

AVR Micro controllers is Family of RISC Microcontrollers from Atmel.

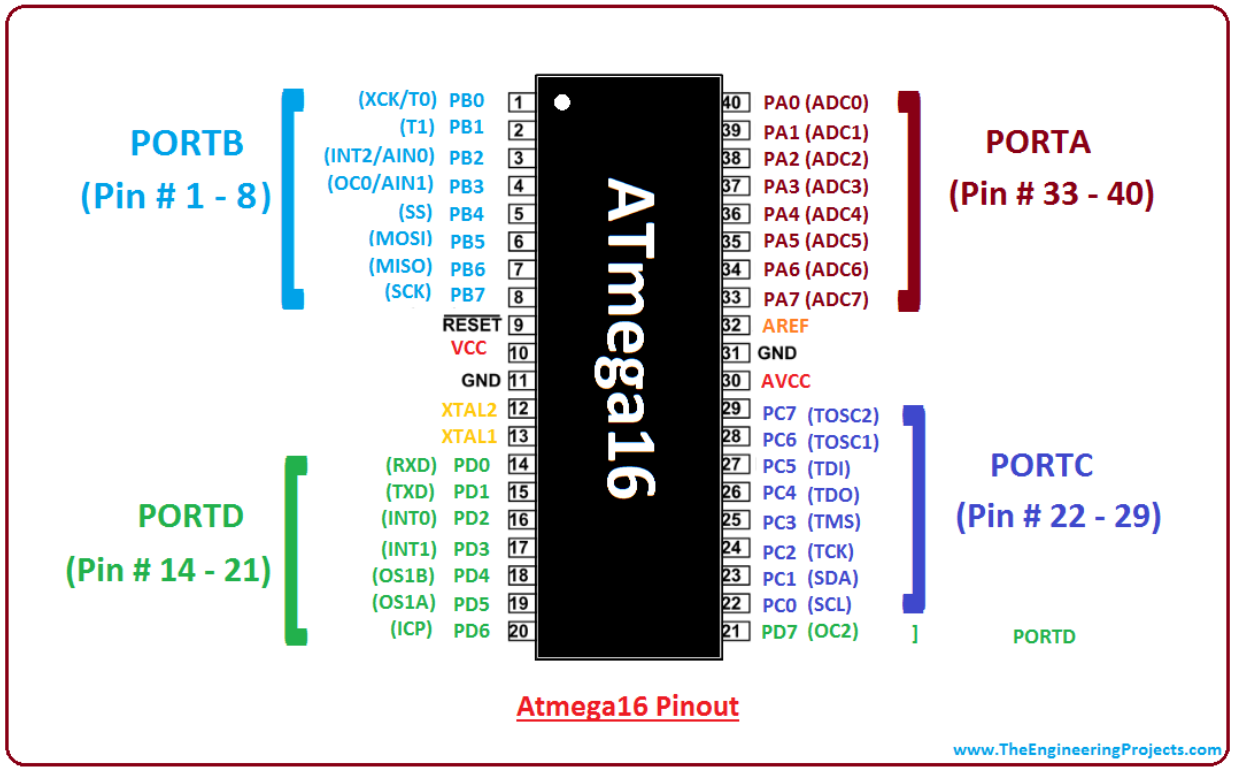
- There are multiple architectures

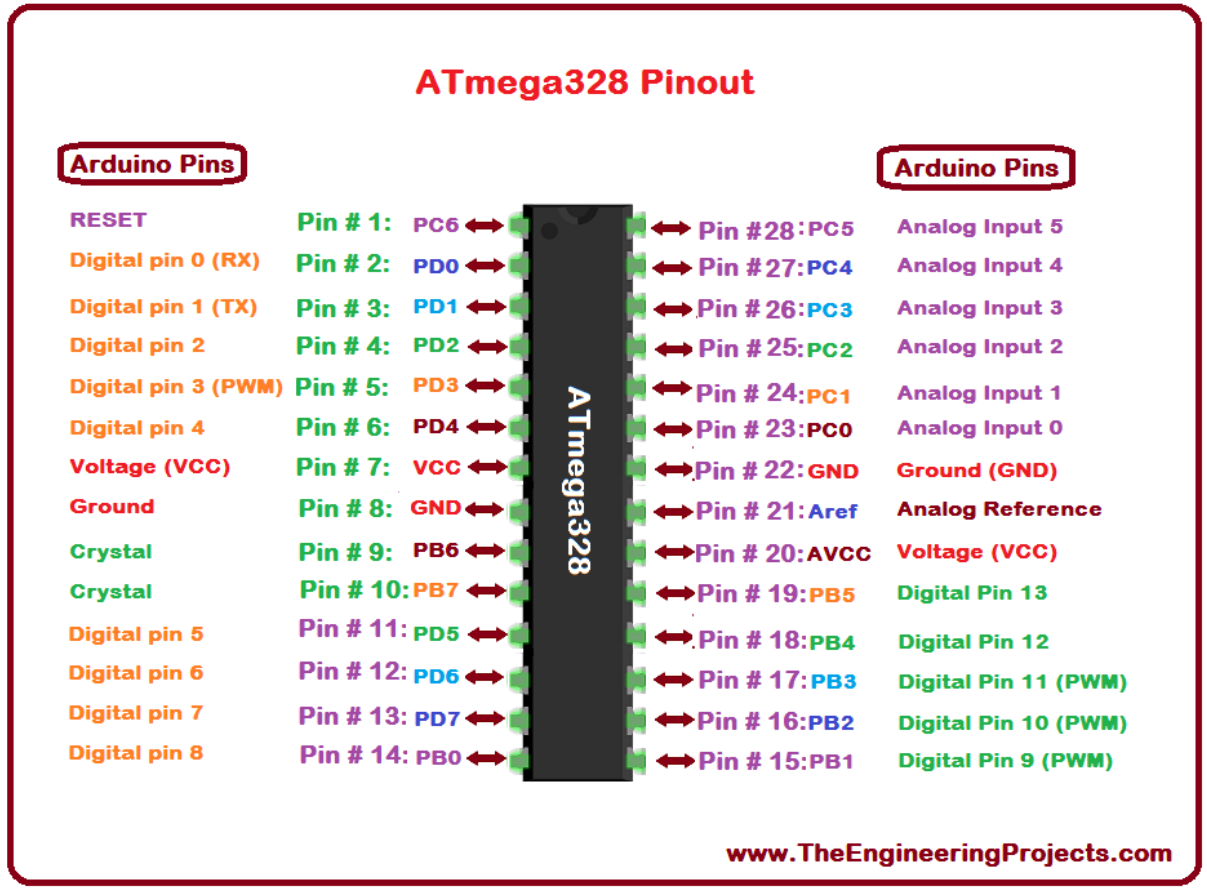
RISC (Reduced Instruction Set Computer)
CISC (Complex Instruction Set Computer)

AVR Family


AVR Block Diagram
4-Wire In/Out
3-Wire In/Out
JTAG Interface
Flash
Serial Peripheral Interface
EEPROM
On Chip Debugger
OCD
Program Counter
Analog Comparator
Instruction Register
32 General Purpose Registers
A/D Converter
Control Lines
LCD Interface
USART
Instruction Decoder
RAM
ALU
CPU
SPI
TWI
Two Wire Interface
Watchdog Timer
I/O Ports
Interrupts
Timer/ Counters

AVR Family


PORTB
(Pin # 1 - 8)
PORTD
(Pin # 14 - 21)
PORTA
(Pin # 33 - 40)
PORTC
(Pin # 22 - 29)
ATmega16
(XCK/T0) PB0 1
(T1) PB1 2
(INT2/AIN0) PB2 3
(OC0/AIN1) PB3 4
(SS) PB4 5
(MOSI) PB5 6
(MISO) PB6 7
(SCK) PB7 8
RESET 9
VCC 10
GND 11
XTAL2 12
XTAL1 13
(RXD) PD0 14
(TXD) PD1 15
(INT0) PD2 16
(INT1) PD3 17
(OS1B) PD4 18
(OS1A) PD5 19
(ICP) PD6 20
40 PA0 (ADC0)
39 PA1 (ADC1)
38 PA2 (ADC2)
37 PA3 (ADC3)
36 PA4 (ADC4)
35 PA5 (ADC5)
34 PA6 (ADC6)
33 PA7 (ADC7)
32 AREF
31 GND
30 AVCC
29 PC7 (TOSC2)
28 PC6 (TOSC1)
27 PC5 (TDI)
26 PC4 (TDO)
25 PC3 (TMS)
24 PC2 (TCK)
23 PC1 (SDA)
22 PC0 (SCL)
21 PD7 (OC2)
Atmega16 Pinout
www.TheEngineeringProjects.com


ATmega328 Pinout
Arduino Pins
Arduino Pins
RESET Pin # 1: PC6 Pin # 28: PC5 Analog Input 5
Digital pin 0 (RX) Pin # 2: PD0 Pin # 27: PC4 Analog Input 4
Digital pin 1 (TX) Pin # 3: PD1 Pin # 26: PC3 Analog Input 3
Digital pin 2 Pin # 4: PD2 Pin # 25: PC2 Analog Input 2
Digital pin 3 (PWM) Pin # 5: PD3 Pin # 24: PC1 Analog Input 1
Digital pin 4 Pin # 6: PD4 Pin # 23: PC0 Analog Input 0
Voltage (VCC) Pin # 7: VCC Pin # 22: GND Ground (GND)
Ground Pin # 8: GND Pin # 21: Aref Analog Reference
Crystal Pin # 9: PB6 Pin # 20: AVCC Voltage (VCC)
Crystal Pin # 10: PB7 Pin # 19: PB5 Digital Pin 13
Digital pin 5 Pin # 11: PD5 Pin # 18: PB4 Digital Pin 12
Digital pin 6 Pin # 12: PD6 Pin # 17: PB3 Digital Pin 11 (PWM)
Digital pin 7 Pin # 13: PD7 Pin # 16: PB2 Digital Pin 10 (PWM)
Digital pin 8 Pin # 14: PB0 Pin # 15: PB1 Digital Pin 9 (PWM)
ATmega328
www.TheEngineeringProjects.com

AVR Family

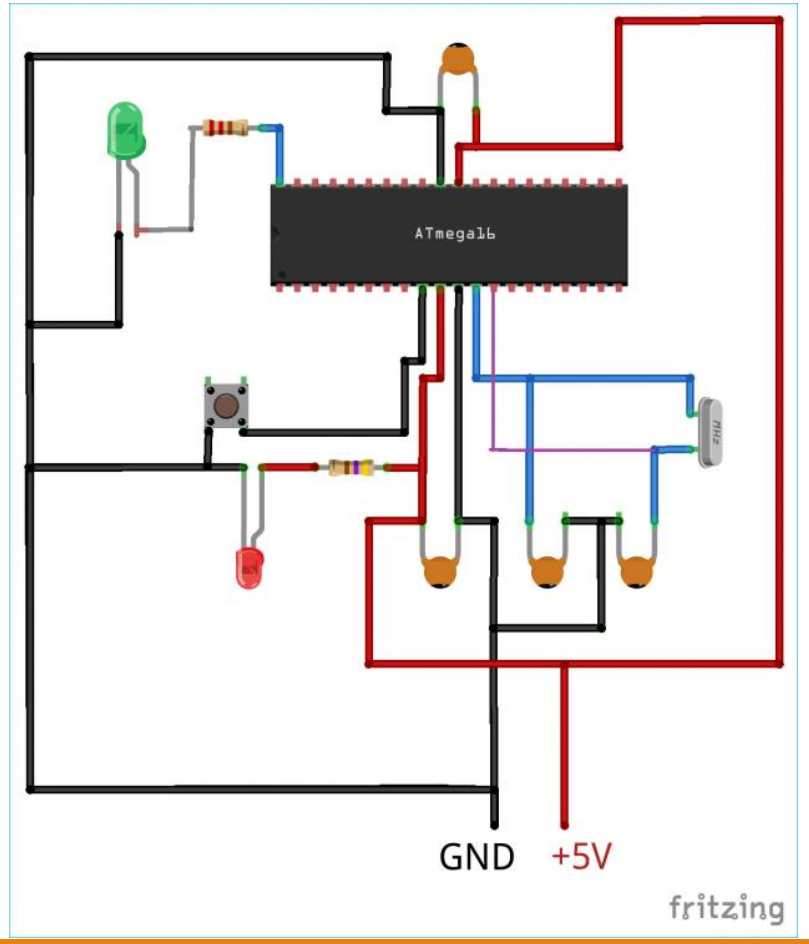
- 8 bit Microcontroller
- High performance - 16MIPS @ 16MHz
- EEPROM – non volatile memory
- Two 8 bit, One 16 bit timer with total 4 PWM channels

	ATmega16
Flash	16k bytes
RAM	1k bytes
EEPROM	512 bytes

Microcontroller	ATmega328
Operating Voltage	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

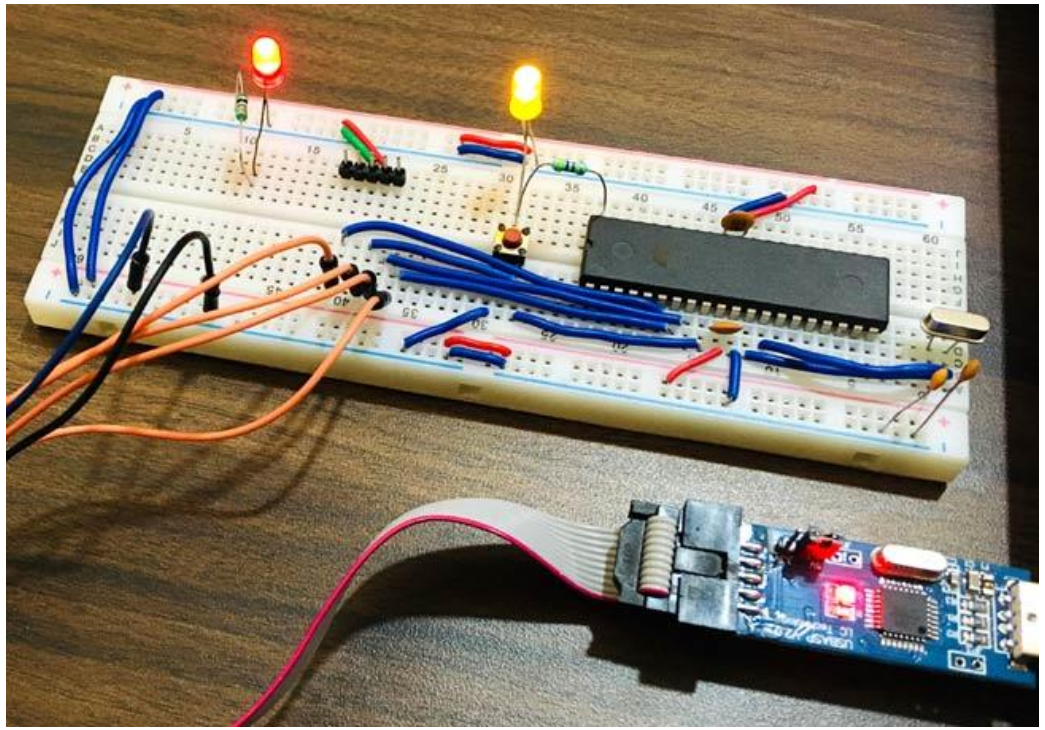
Introduction to Embedded System

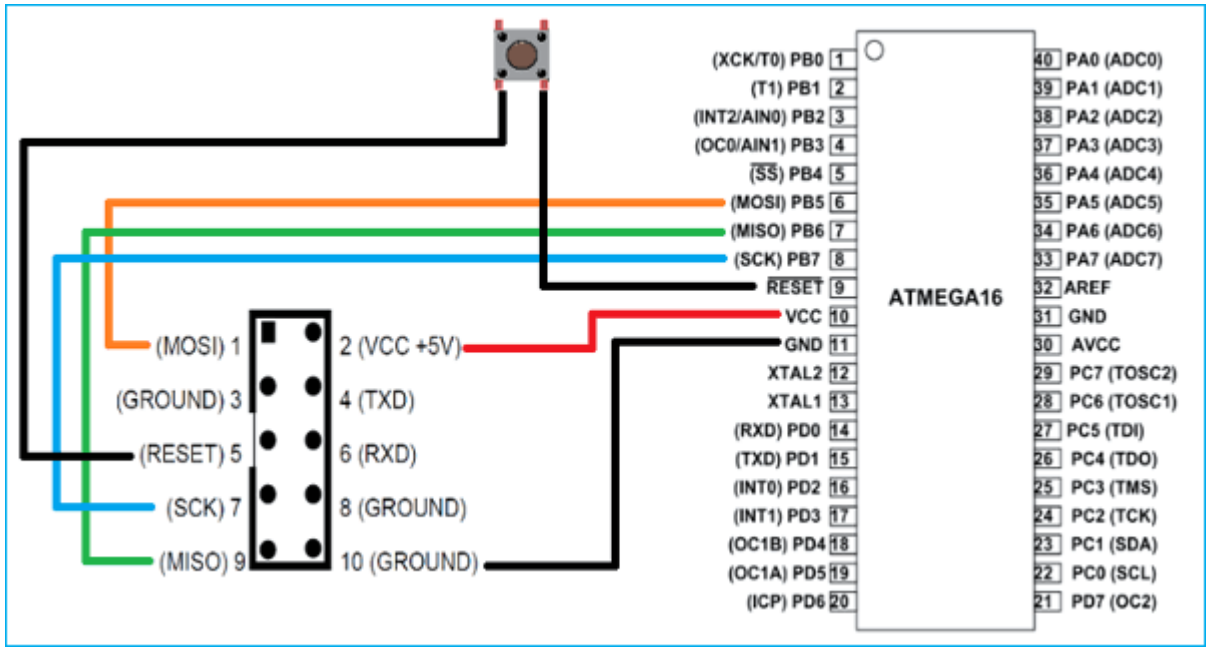
Setting up Atmega16 with oscillator



AVR Family

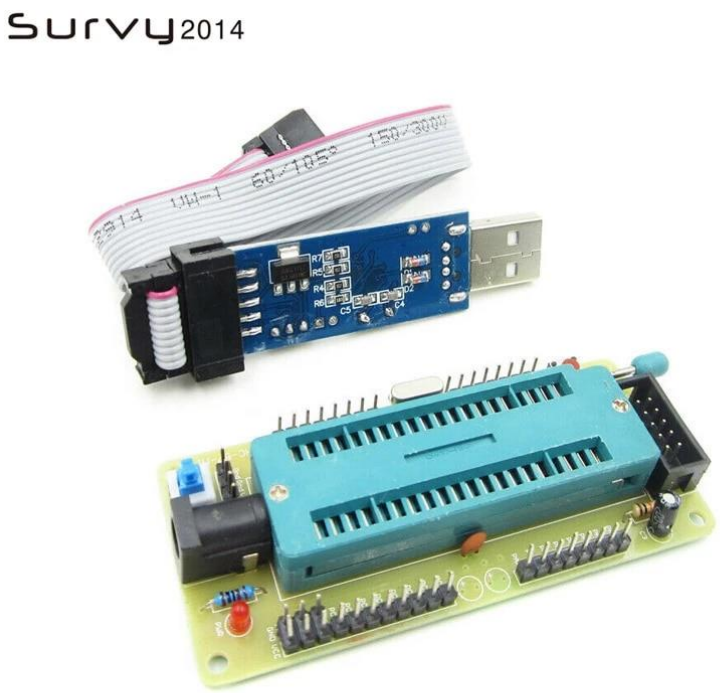
Programming Atmega16

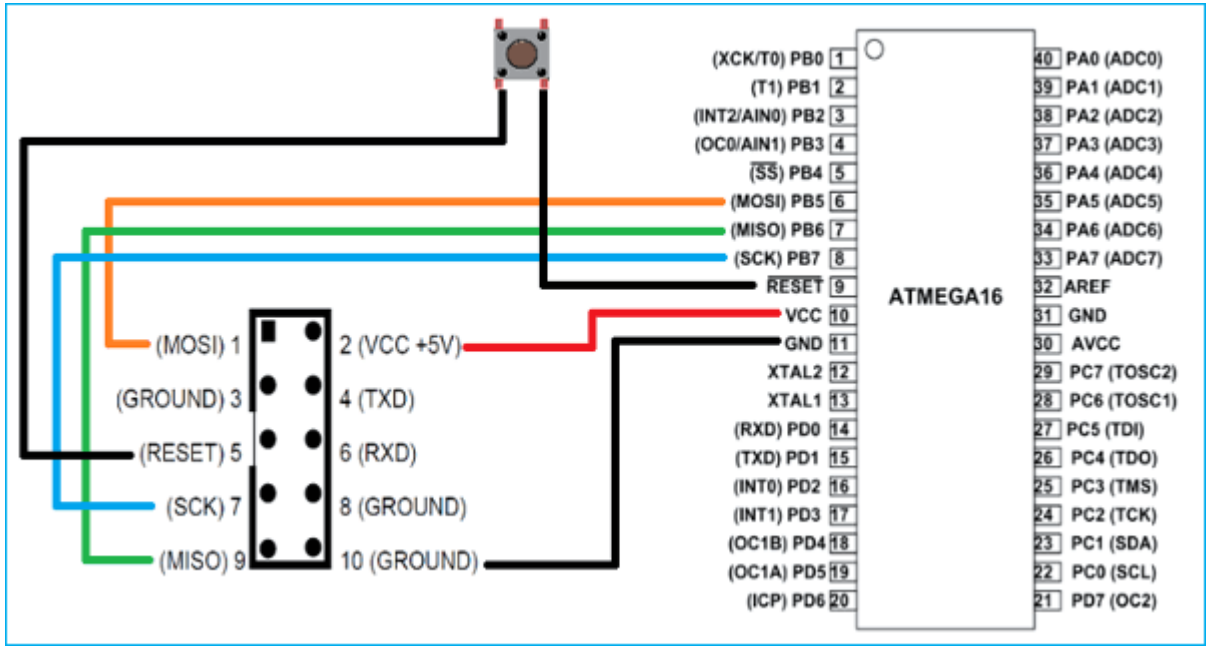




AVR Family

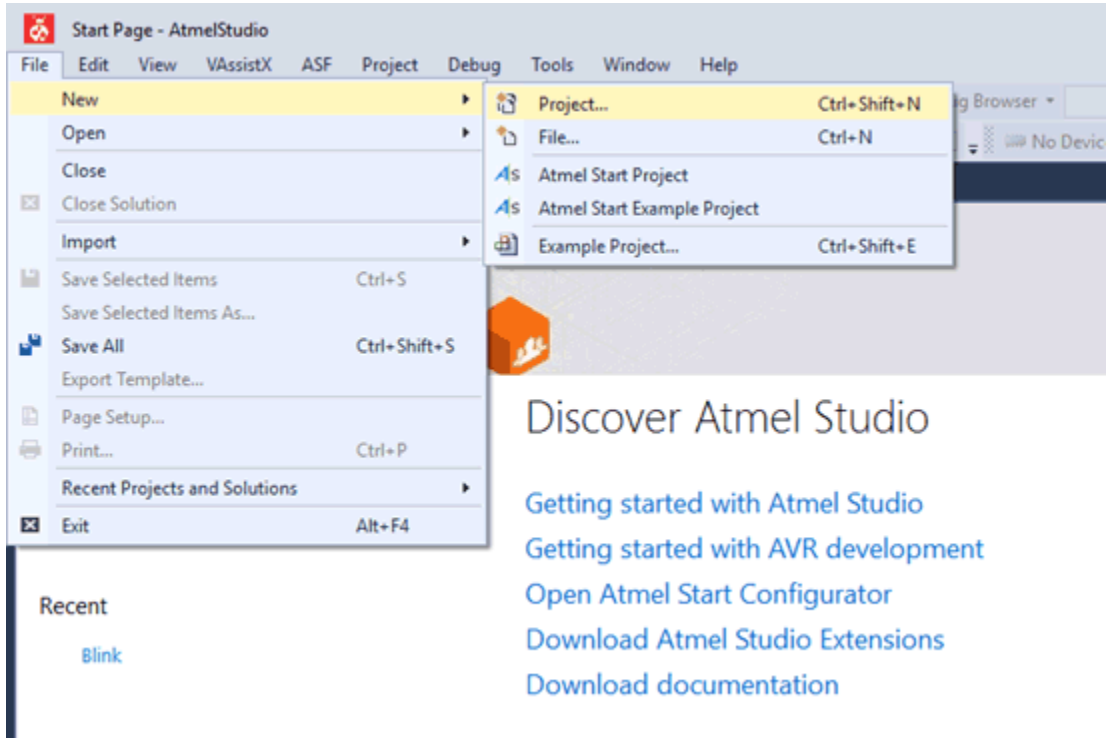
Programming Atmega16

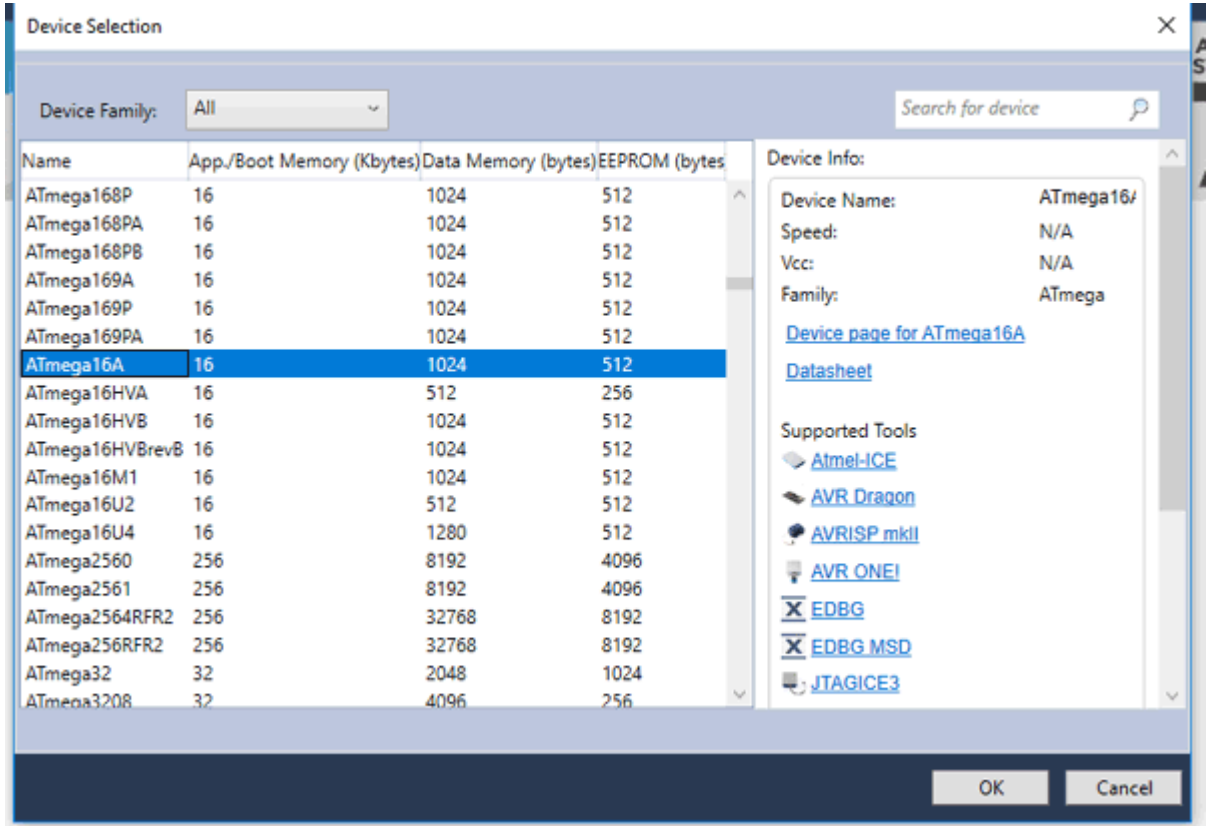




AVR Family

Atmel Studio

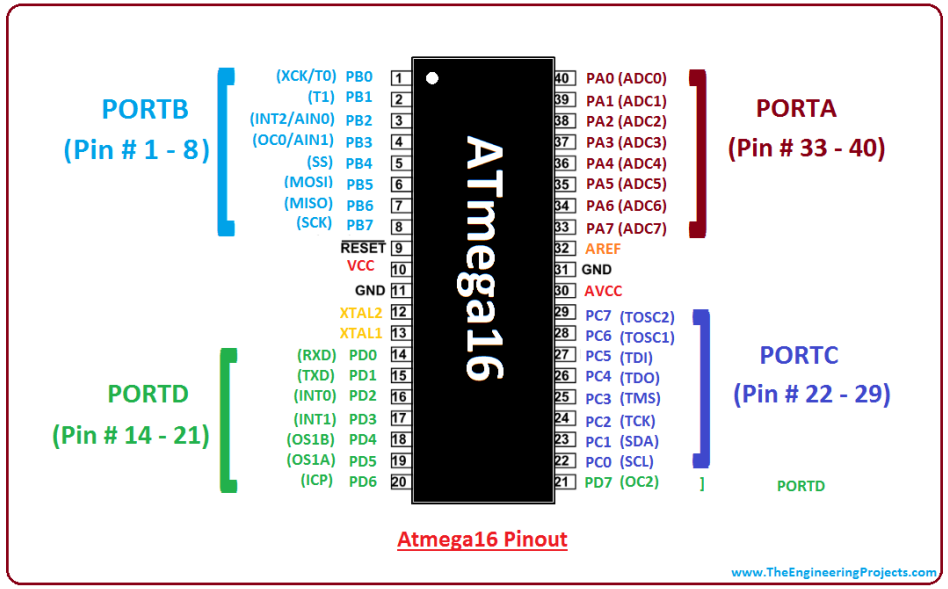




Atmel Studio
I/O ports

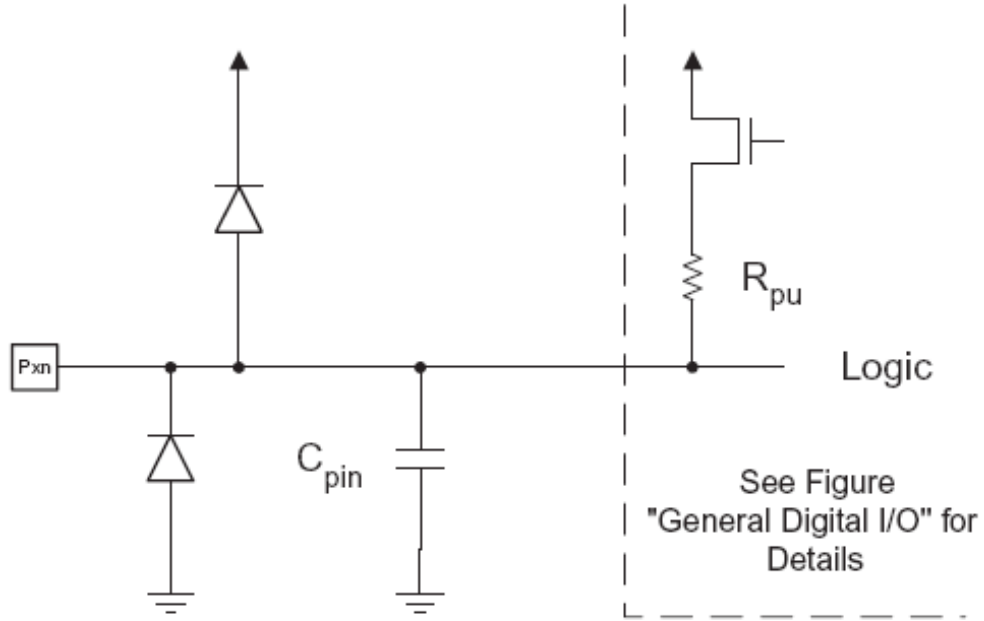
PORT: Group of 8 pins, or set of pins used for exchanging data with external world

- Width of almost all registers : 8 bits (some 16 bits)
- In port related registers, every bit corresponds to one pin of the port. Bit 0 corresponds to Pin 0 Bit 1 corresponds to Pin 1 .. Etc
- Remember direct one to one correspondence between HEX and BINARY numbers. 0xFF = 0b11111111 0xAA = 0b10101010 0x11 = 0b00010001



Atmel Studio
I/O ports

- General Purpose IO : Data Direction Input or Output
- Internal Pullup can be used for Input Pins
- Output driver can source 20mA current
- protection diodes to GND and VCC



Atmel Studio
I/O ports

Three registers :

- DDRx (R/W): Configure the Data Direction of the port pins.
- PORTx (R/W): Write the values to the port pins in output mode.
- PINx (R): Read data from port pins in input mode

Where x : A,B,C,D depending on the available ports in your AVR.

Data Reg.

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Direction Reg.

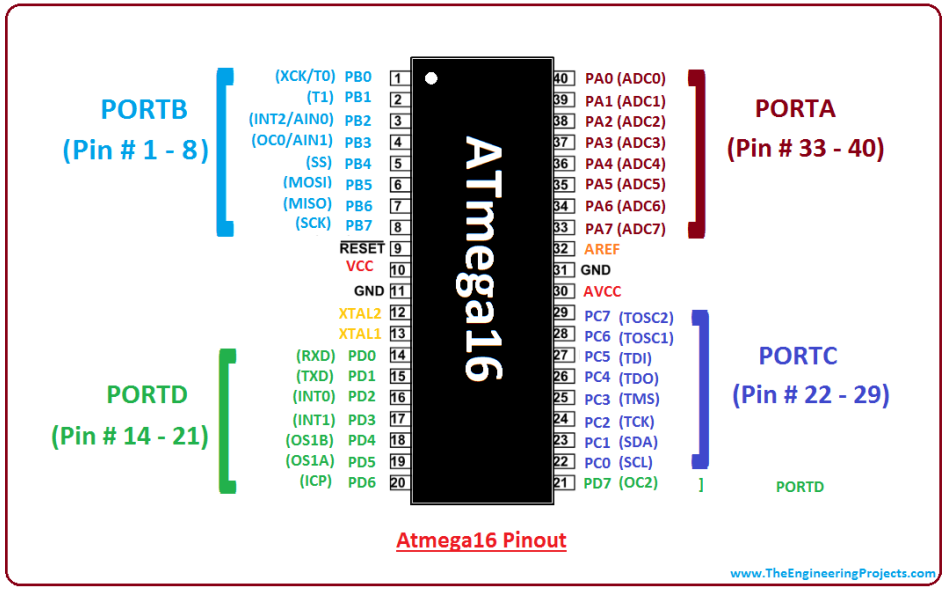
0→ Input

1→ Output

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

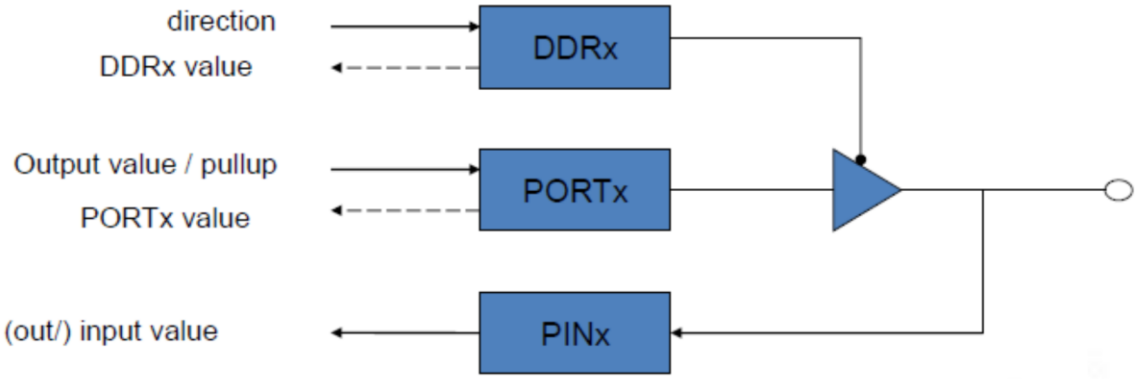
Pin Reg.

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

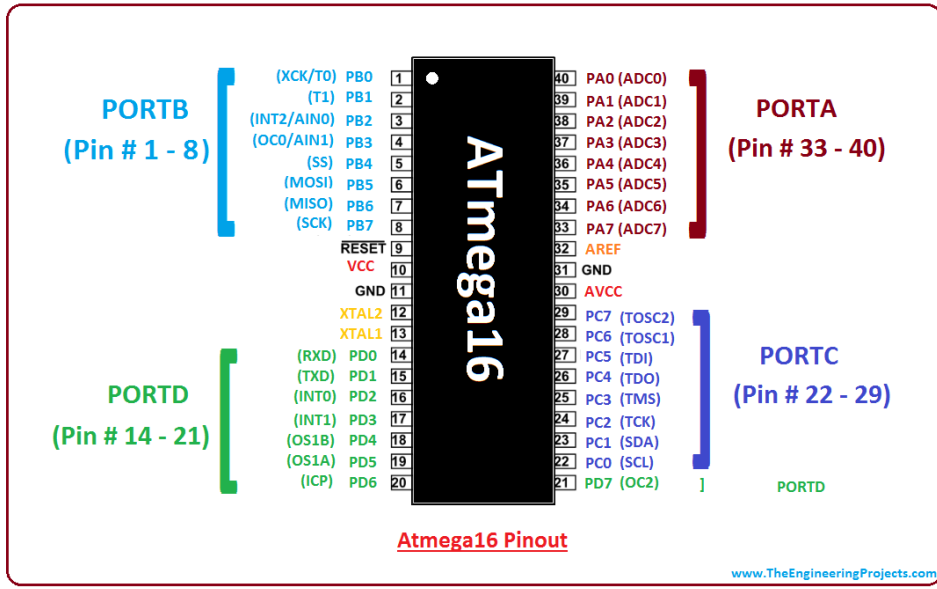


Atmel Studio
I/O ports

Three registers :

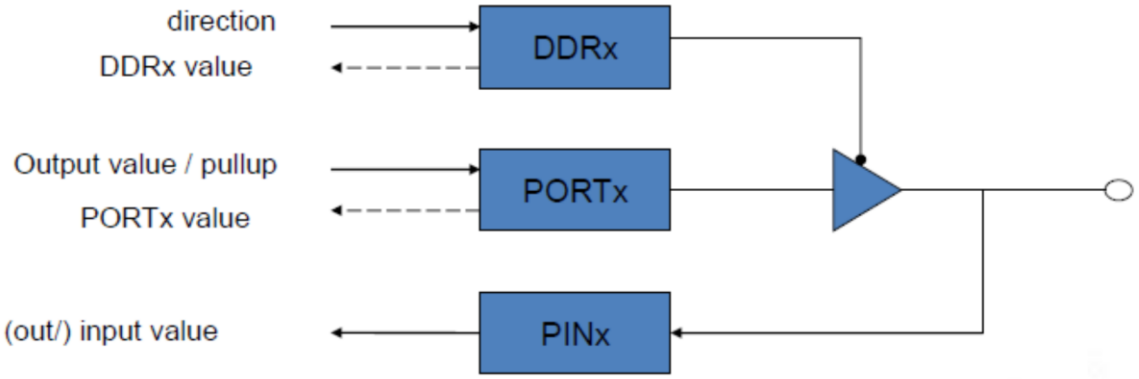


DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

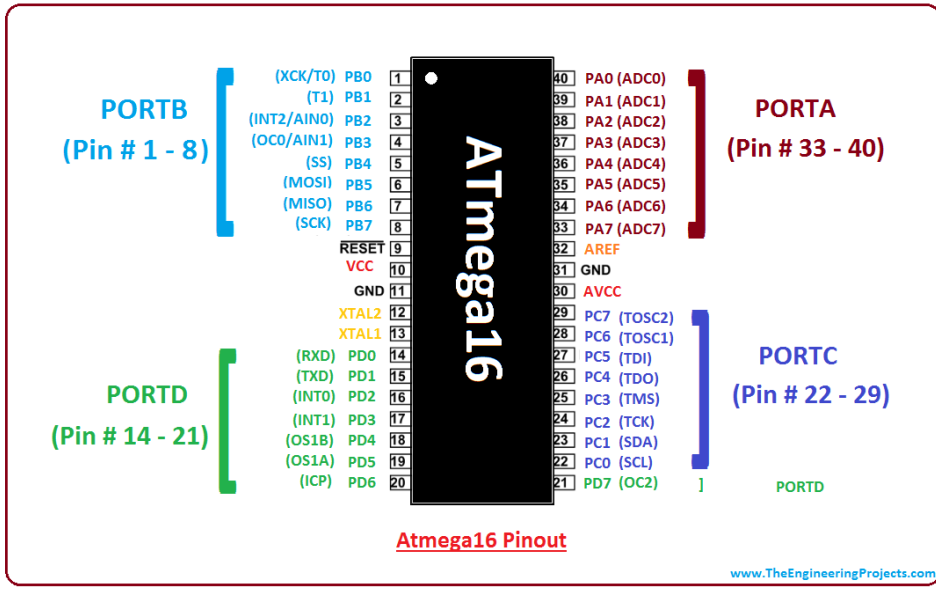


Atmel Studio
I/O ports

Three registers :



DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)



Atmel Studio
I/O ports

Port direction register DDRx

Configures data direction of the port - Input / Output

- DDRx.n = 0 > makes corresponding port pin as input

DDRx.n = 1 > makes corresponding port pin as output

- Examples :

–Set all pins of port A as input pins :

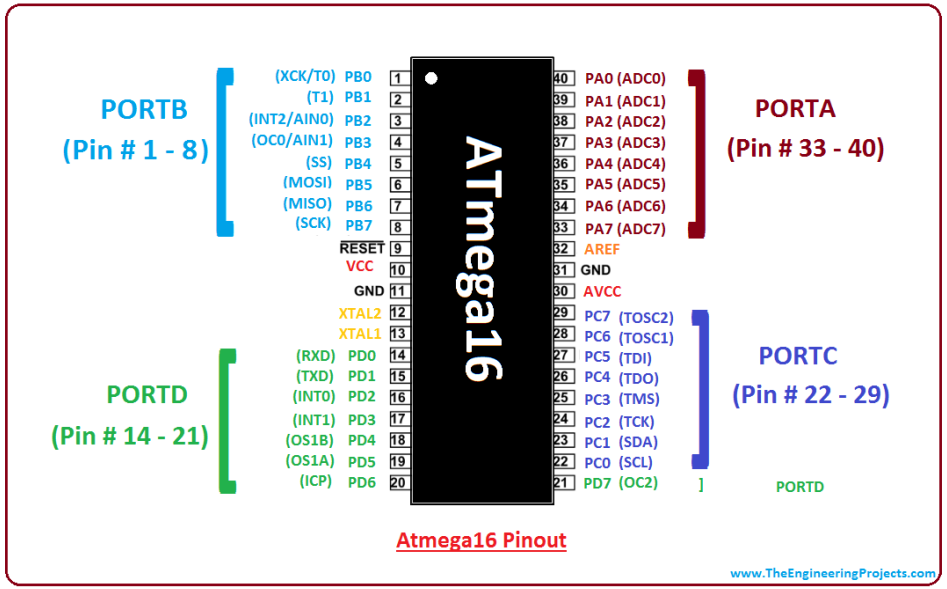
- `DDRA = 0b00000000; OR 0x00`

–Set all pins of port A as output pins

- `DDRA = 0b11111111; OR 0x11`

–Set lower nibble of port B as output and higher nibble as input

- DDRB = `0b00001111; OR 0x01`



Atmel Studio
I/O ports

PIN register

Used to read data from port pins, when port is configured as input.

First set DDRx to zero, then use PINx to read the value.

If PINx is read, when port is configured as output, it will give you data that has been outputted on port.

There two input modes :

- Tristated input
- Pullup input

•Example :

```
DDRA = 0x00; //Set PA as input
x = PINA; //Read contents of PA
```

Atmel Studio
I/O ports

Port register

Used for two purposes

1) Output data (when port is configured as output):

Writing to PORTx.n will immediately (in same clock cycle) change state of the port pins according to given value.

Do not forget to load DDRx with appropriate value for configuring port pins as output.

–Examples :

to output 0xFF data on PB

```
DDRB = 0b11111111;    //set all pins of port b as outputs
PORTB = 0xFF;         //write data on port
```

- to output data in variable x on PA

```
DDRA = 0xFF;          //make port a as output
PORTA = x;             //output 8 bit variable on port
```

Atmel Studio
I/O ports

2) Configuring pin as tristate/pullup (when port is configured as input) :

When port is configures as input (i.e DDRx.n=0), then PORTx.n controls the internal pull-up resistor.

PORTx.n = 1 : Enables pullup for nth bit PORTx.n = 0 : Disables pullup for nth bit, thus making it tristate

–Examples :

- Set PA as input with pull-ups enabled and read data from

```
DDRA = 0x00;        //make port a as input
PORTA = 0xFF;       //enable all pull-ups
y = PINA;           //read data from port a pins
```

- Set PB as tri stated input

```
DDRB = 0x00;        //make port b as input
PORTB = 0x00;       //disable pull-ups and make it tri state
```

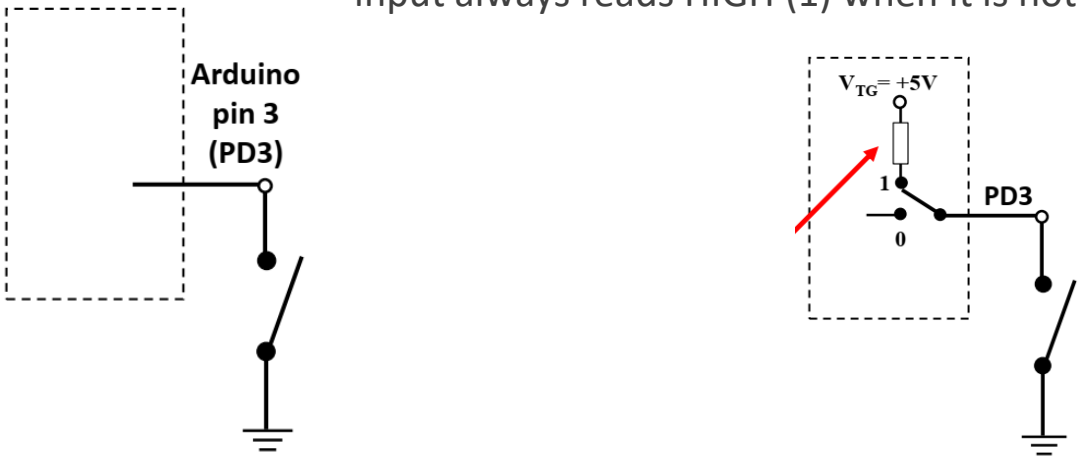
Atmel Studio
I/O ports

What is pull up ?

Tri-state input pin offers very high impedance and thus can read as logic 1/ logic 0 (according to the input signal connected to it).

Pin state changes rapidly and this change is unpredictable, this may cause your program to go out of control if it depends on input from such tri-state pin.

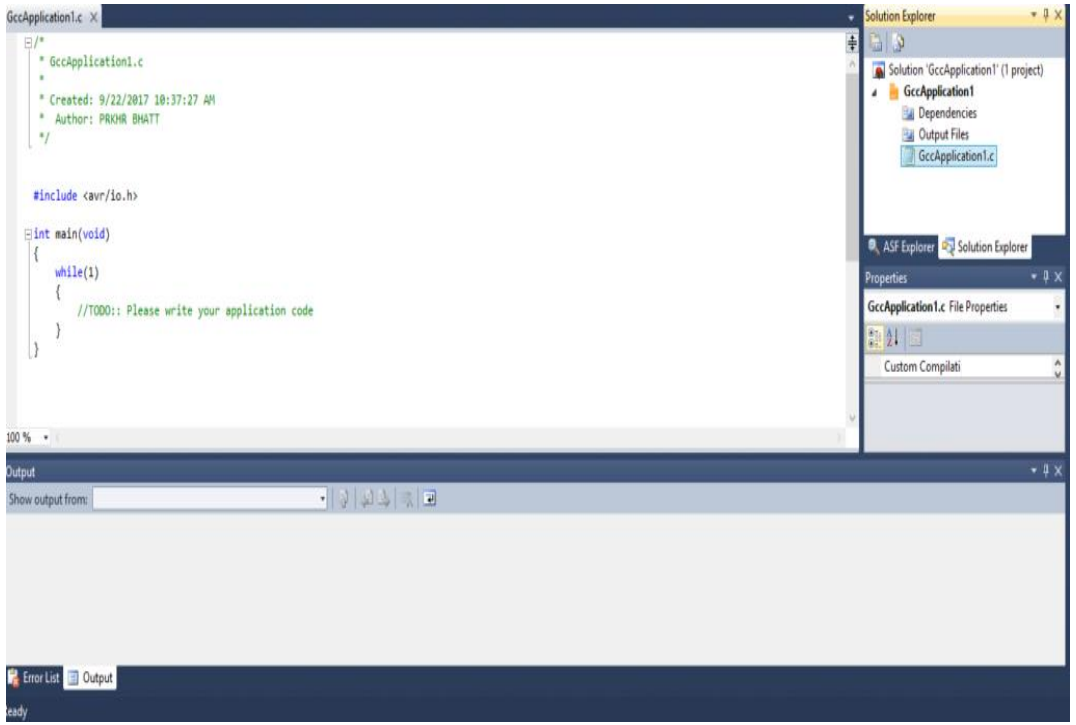
Pull-up resistor is used to ensure that tri-stated input always reads HIGH (1) when it is not driven by any external entity.

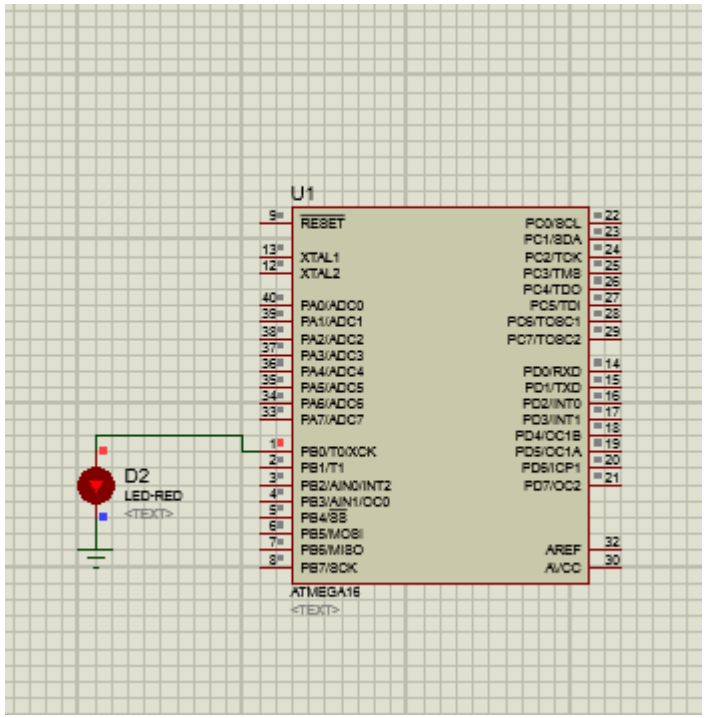


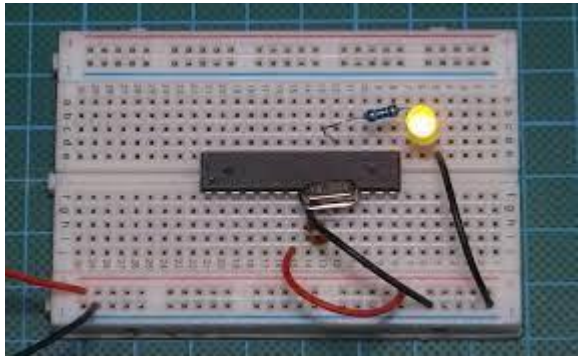
Atmel Studio
I/O ports

Simple project 1

LED Blinking Program in Atmega16-AVR







Atmel Studio
I/O ports

Simple project

LED Blinking Program in Atmega16-AVR

```
#include <avr/io.h>
#include <util/delay.h>//it has delay function
int main(void)
{
DDRB=0b00000001;
while(1)
        {
     PORTB=0b00000001;
     _delay_ms(1000); //so that LED would remain in ON state for 1 sec and then  turns off
     PORTB=0b00000000;
     _delay_ms(1000);
        }
}
```

Outlines

1. Course Information
2. Course outline
3. Introduction to Embedded System
4. AVR Family
5. Arduino

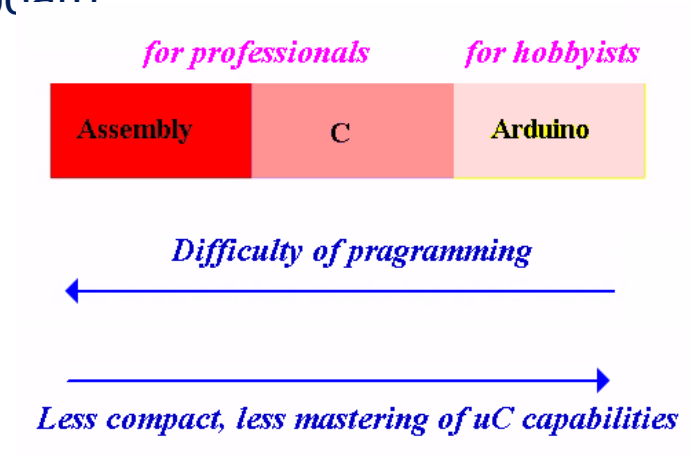
Introduction to Embedded System

- **What is a Development Board**

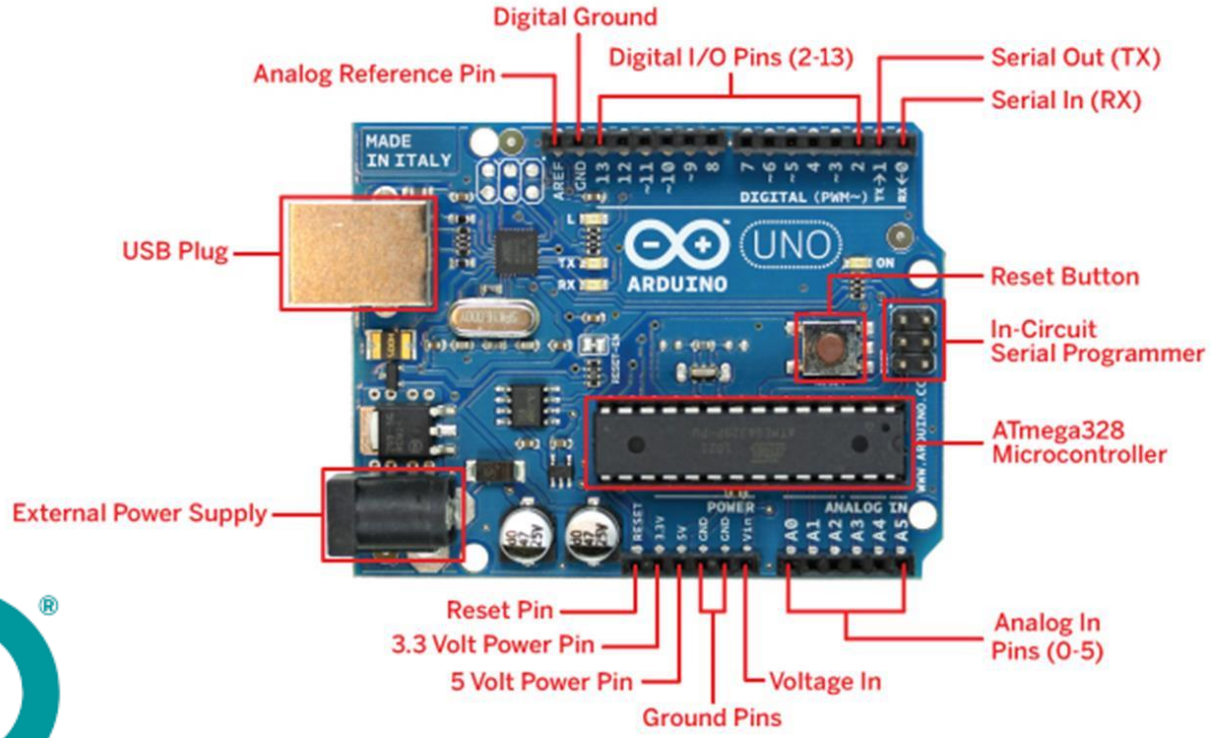
A printed circuit board designed to facilitate work with a particular microcontroller.

Introduction to Embedded System

- **The Arduino Development Board**
- **Arduino (The name is an Italian , meaning "stron friend"**) is an opensource platform used for buildir electronics projects. Arduino consists of both a physic programmable circuit board (often referred to as microcontroller) and a piece of software, or IDE (Integrate Development Environment) that runs on your compute used to write and upload computer code to the physic board



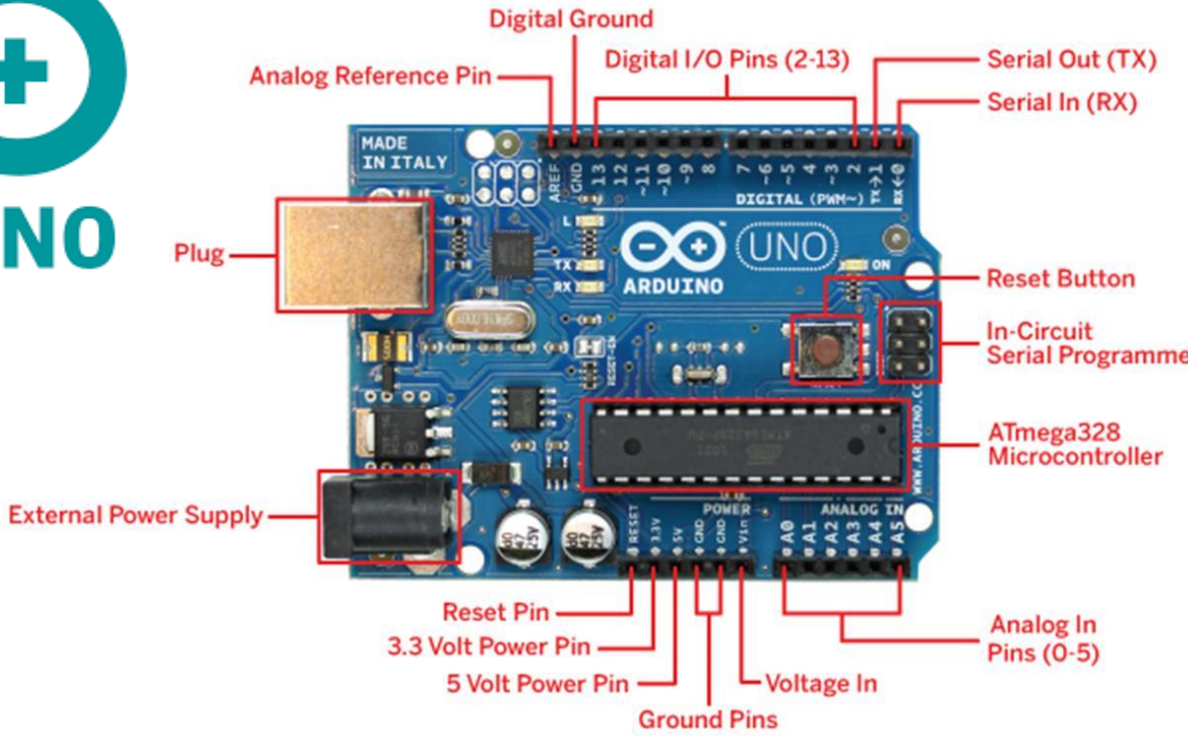
ARDUINO®



Introduction to Embedded System

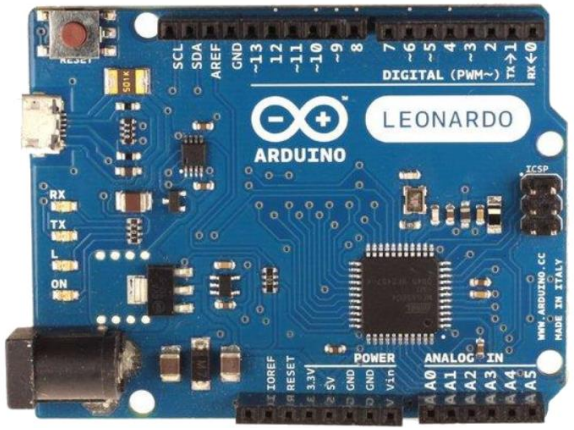
- **The Arduino UNO Development Board**

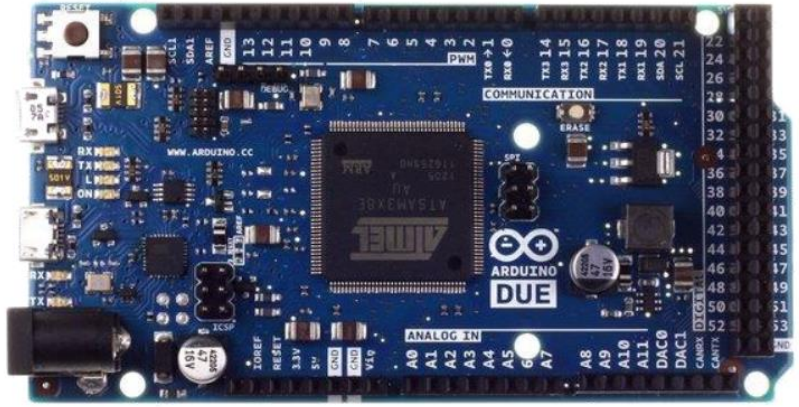
Microcontroller	ATmega328
Operating Voltage	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

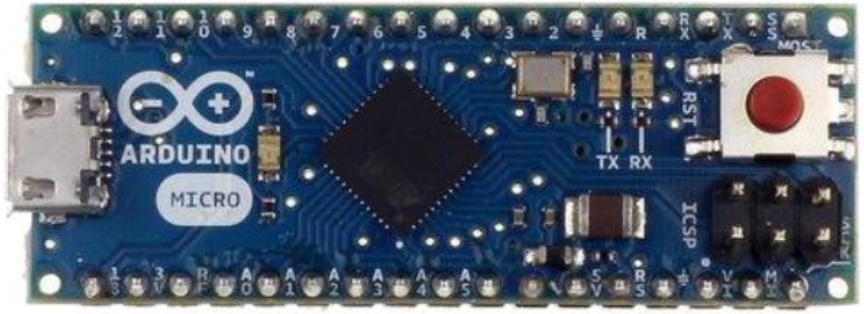


Introduction to Embedded System

- **The Arduino Development Boards**

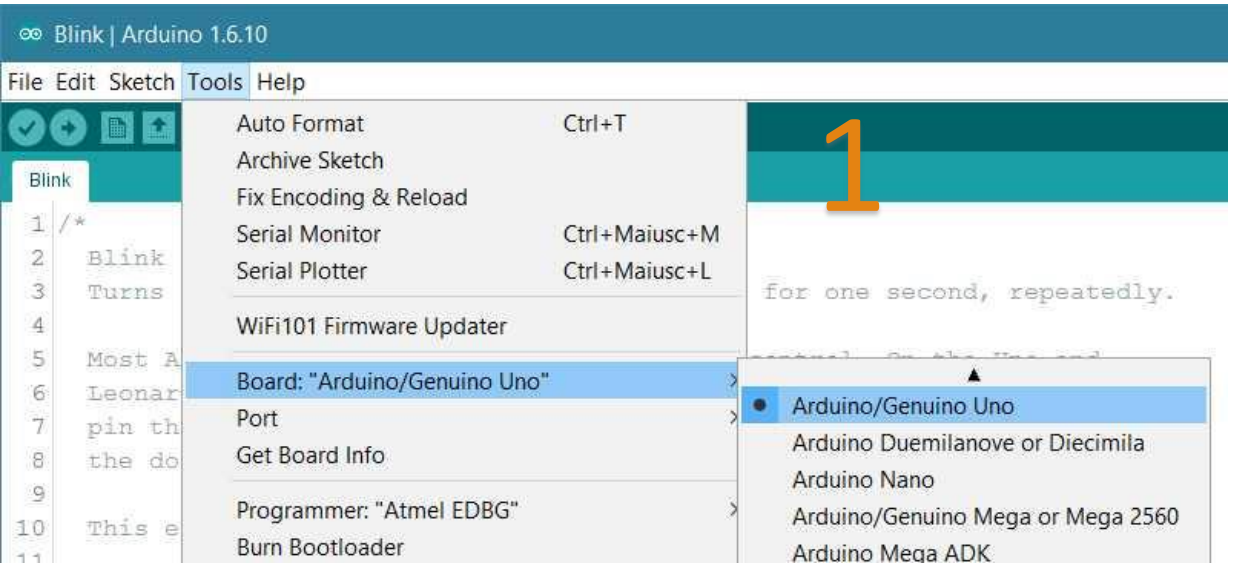


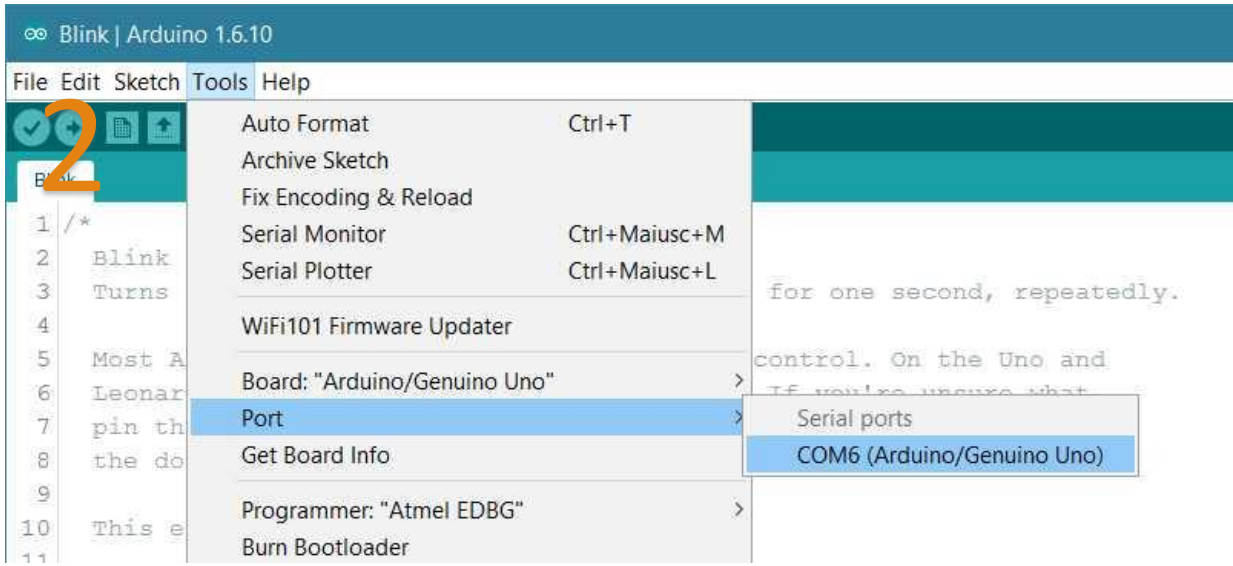


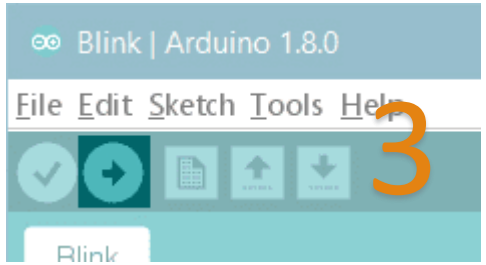


Introduction to Embedded System

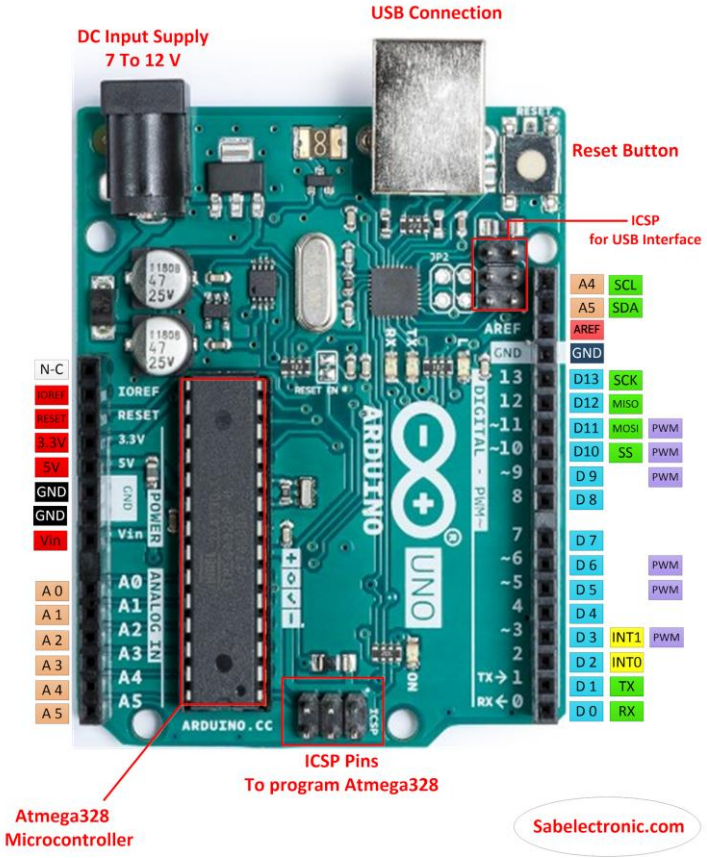
- **Arduino IDE**

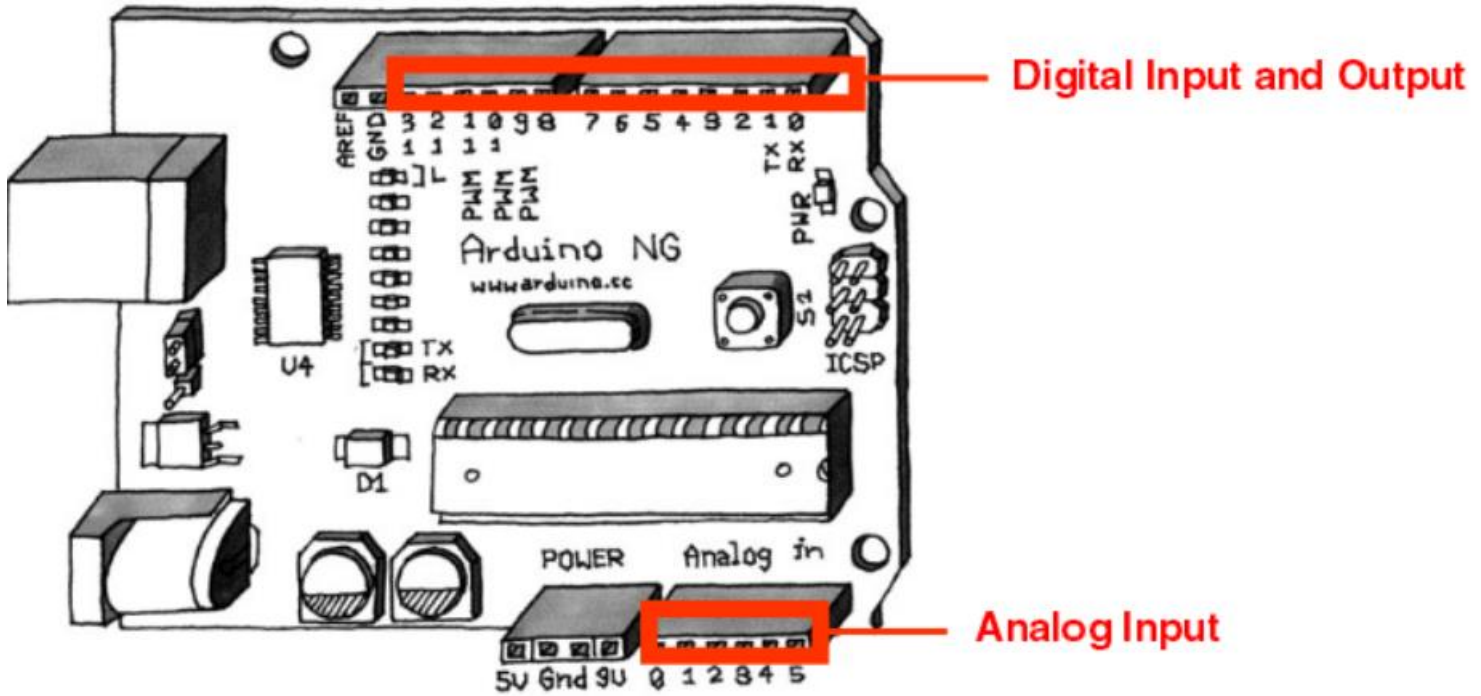






Arduino IDE
IO Pins





Arduino IDE
IO Pins

Arduino Digital I/0

pinMode(pin_no., dir)

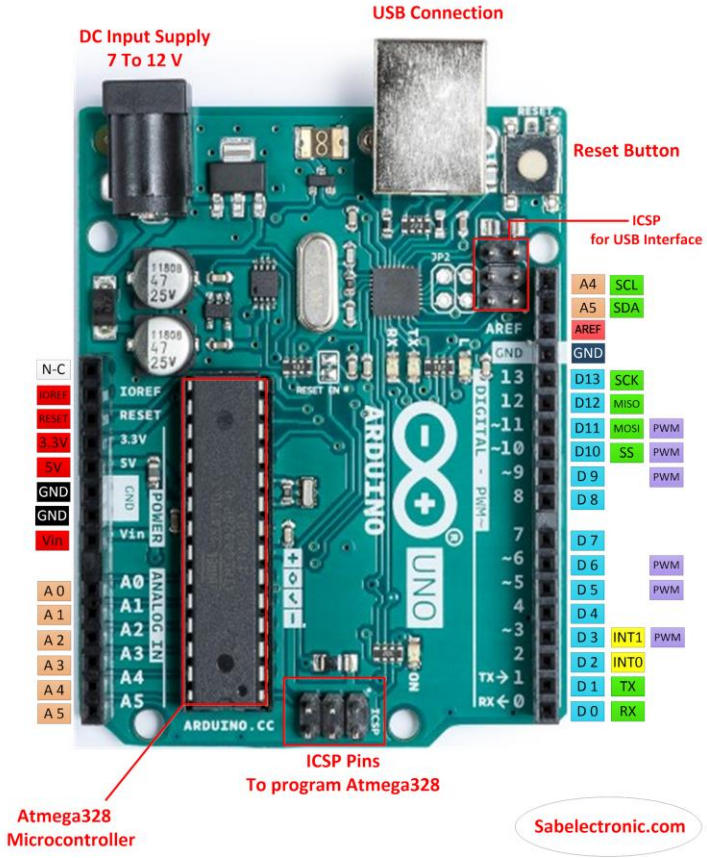
Sets pin to either INPUT or OUTPUT

digitalRead(*pin*)

Reads HIGH or LOW from a pin

digitalWrite(*pin*, *value*)

Writes HIGH or LOW to a pin

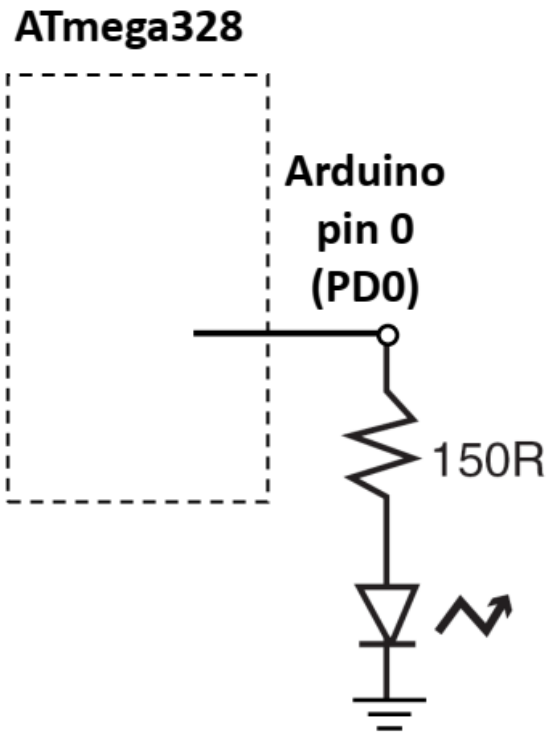


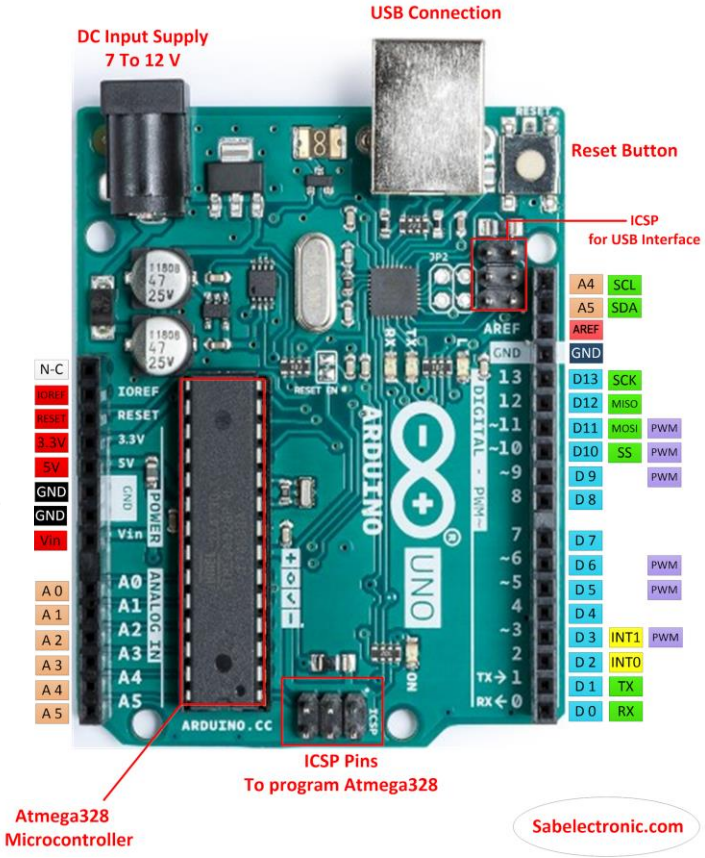
Arduino IDE
IO Pins

Output

pinMode(0,OUTPUT);

- Turn on the LED
  - `digitalWrite(0,HIGH) ;`
  - Turn off the LED
  - `digitalWrite(0,LOW);`



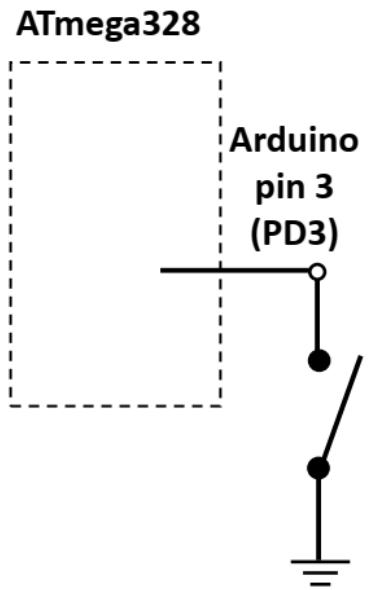


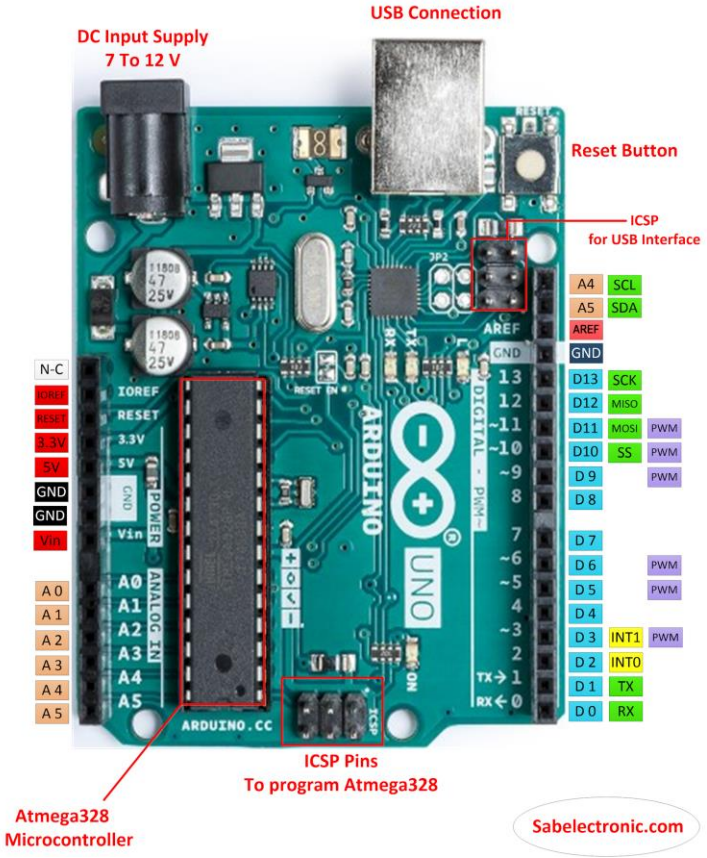
Arduino IDE
IO Pins

Input

pinMode(3, INPUT);

X=digitalRead(3);



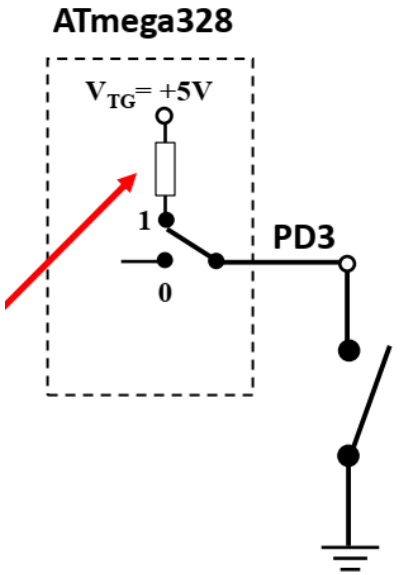


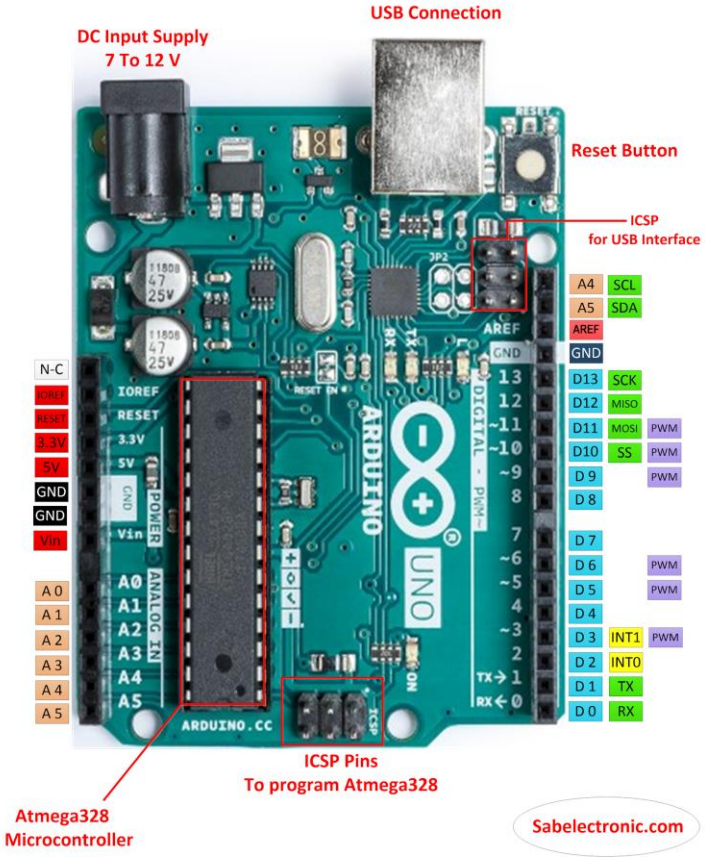
Arduino IDE
IO Pins

Input

```
pinMode(3,INPUT_PULLUP);
```

X=digitalRead(3);

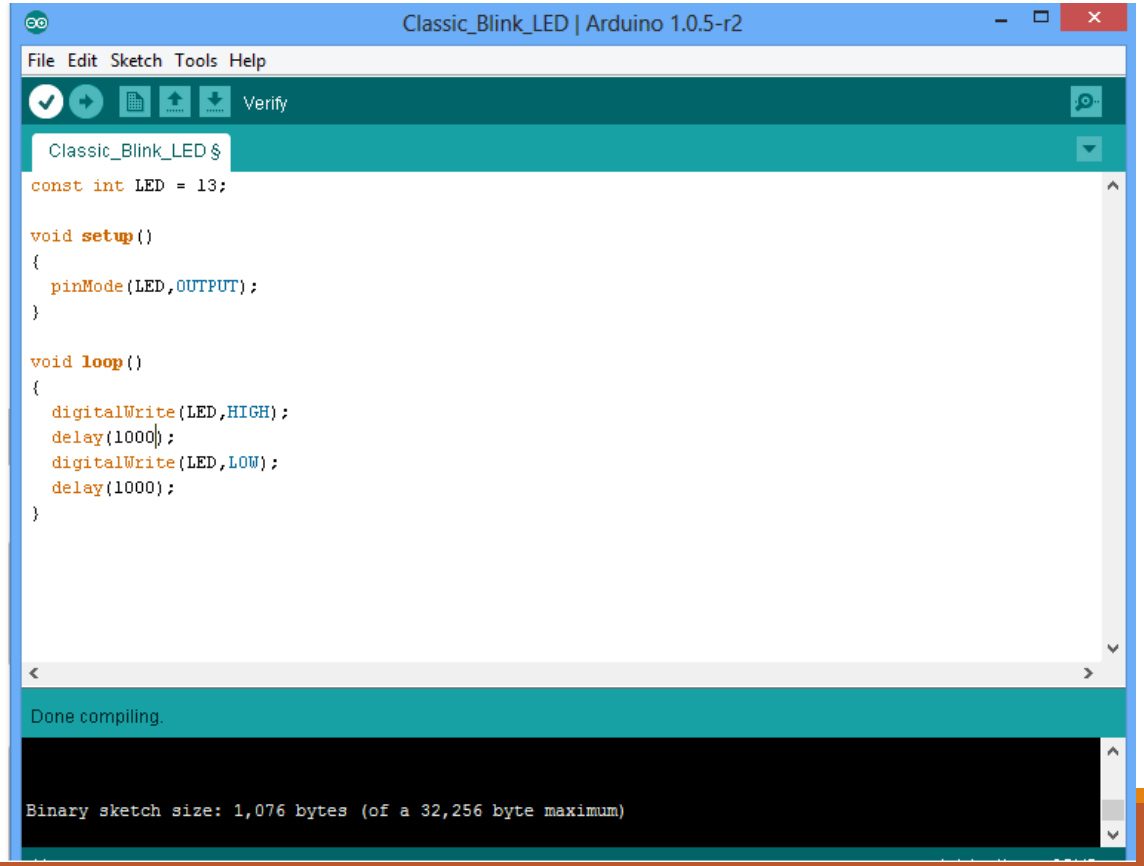


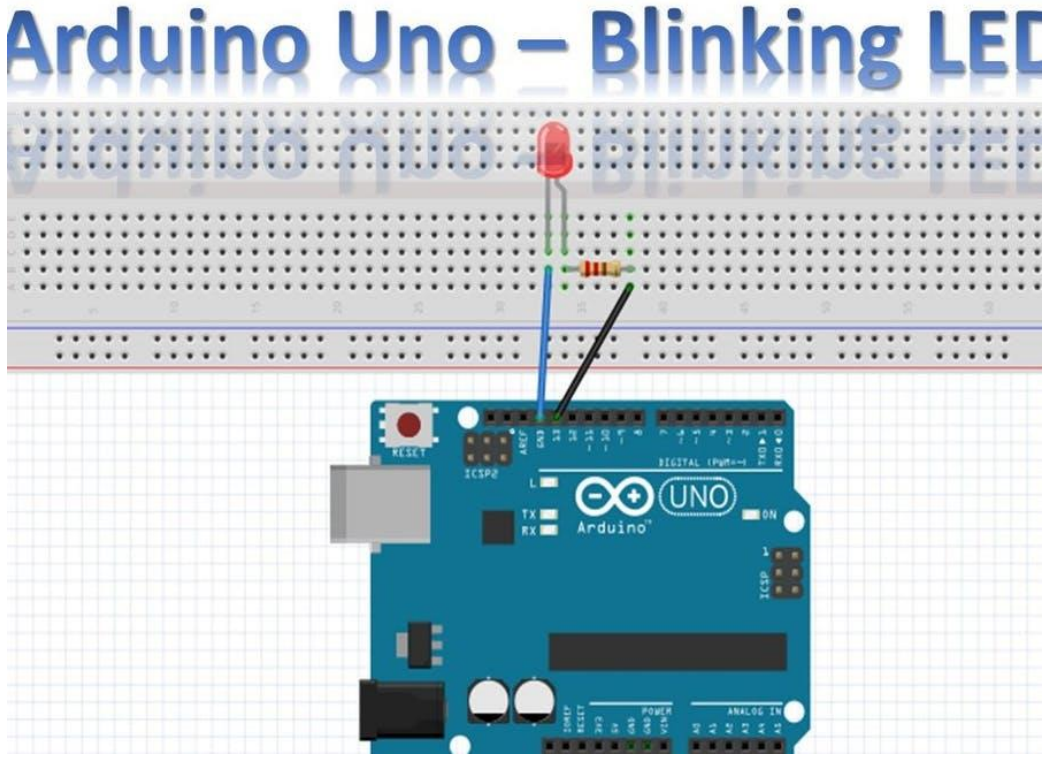


Arduino IDE

Simple project

LED Blinking Program using Arduino UNO

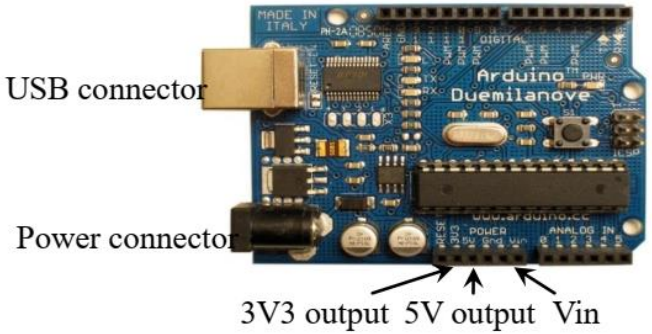




Arduino IDE

The power pins are as follows:

- **Vin.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from Vin via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board FTDI chip. Maximum current draw is 50 mA.
- **GND.** Ground pins.



Thanks