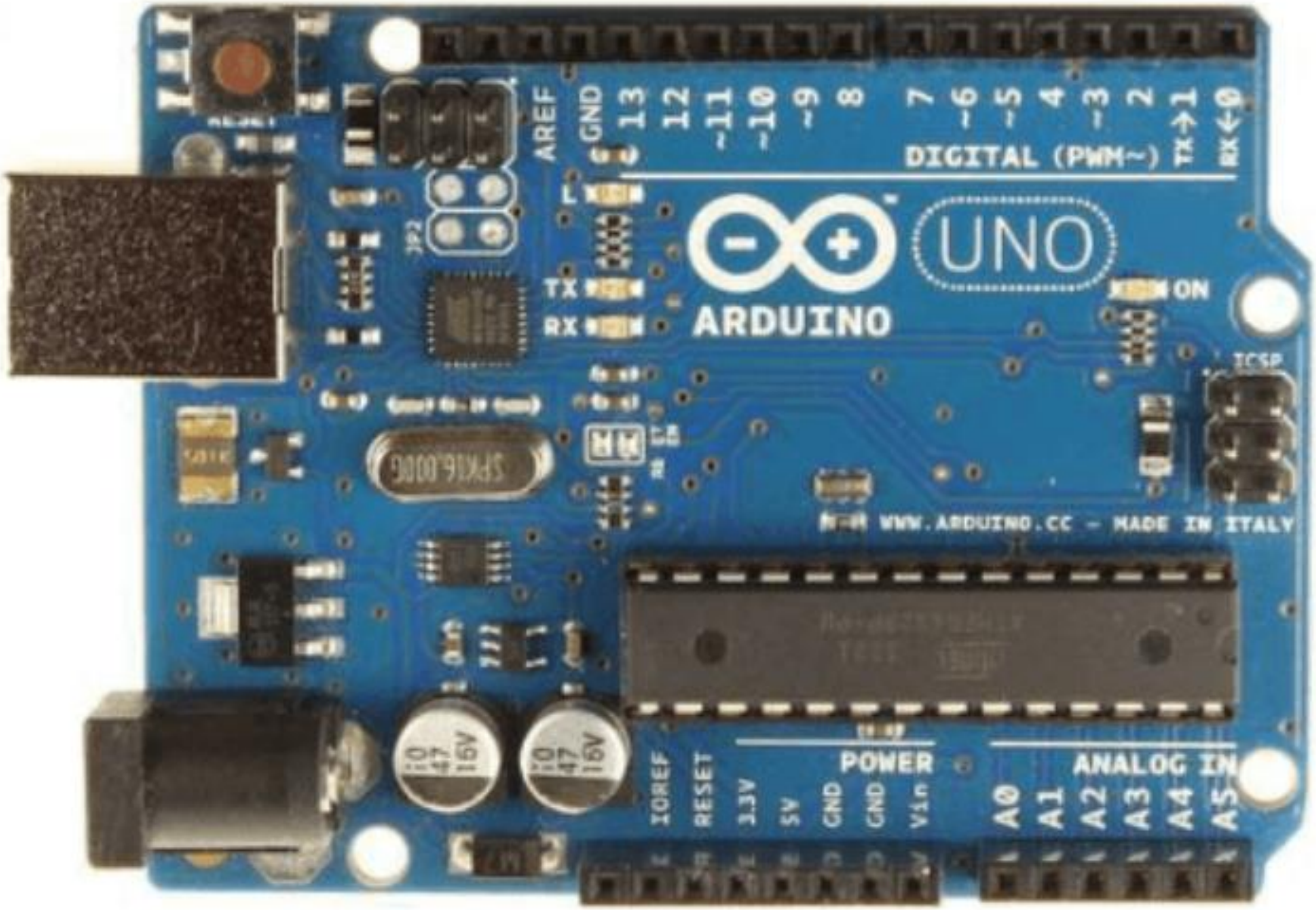
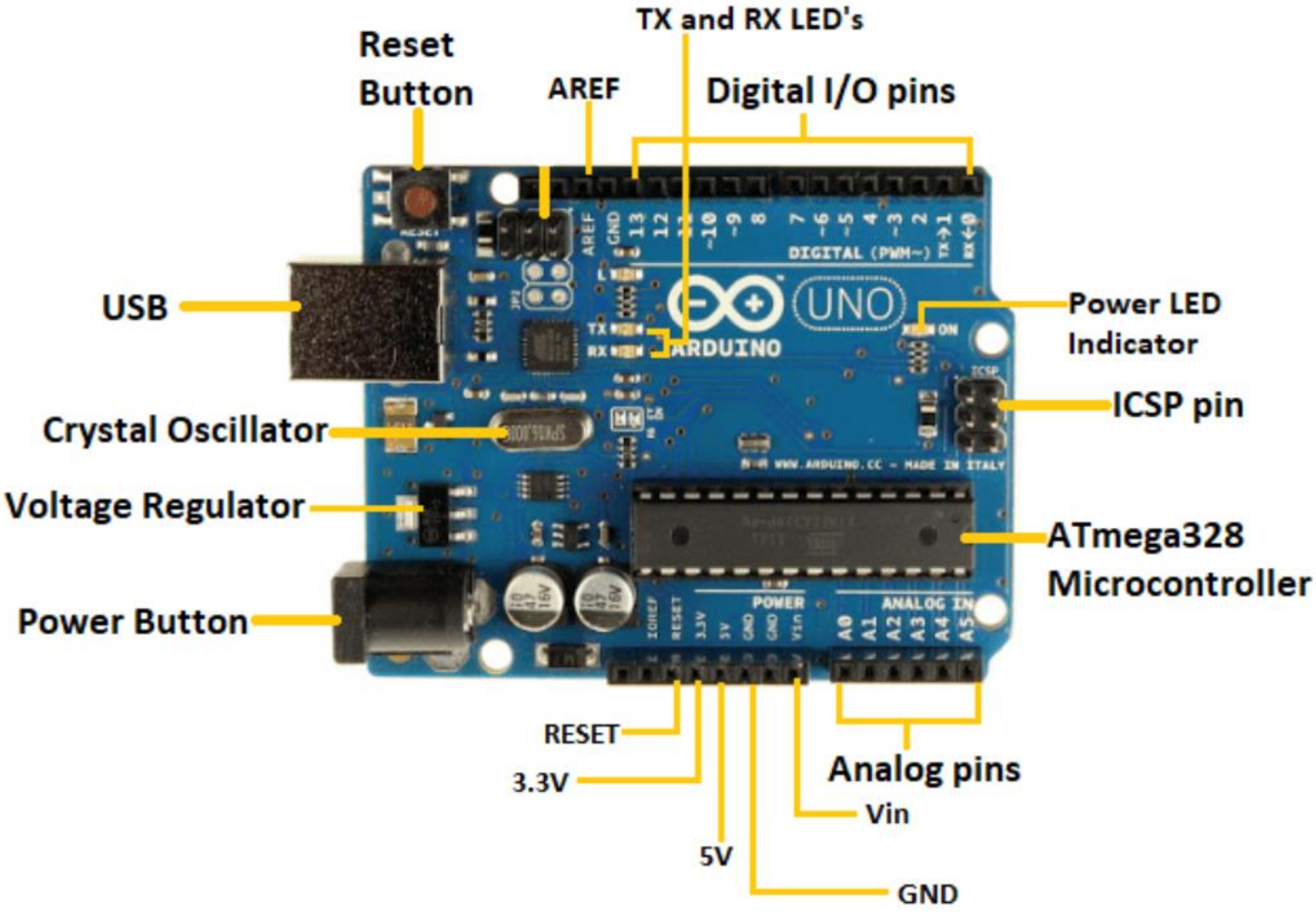
Arduino UNO

- Arduino UNO is a microcontroller board based on the **ATmega328P**.
- It has 14 digital input/output pins (of which 6 can be used as PWM outputs),
- 6 analog inputs,
- a 16 MHz ceramic resonator,
- a USB connection,
- a power jack,
- an ICSP (In-Circuit Serial Programming) header and
- a reset button.
- It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

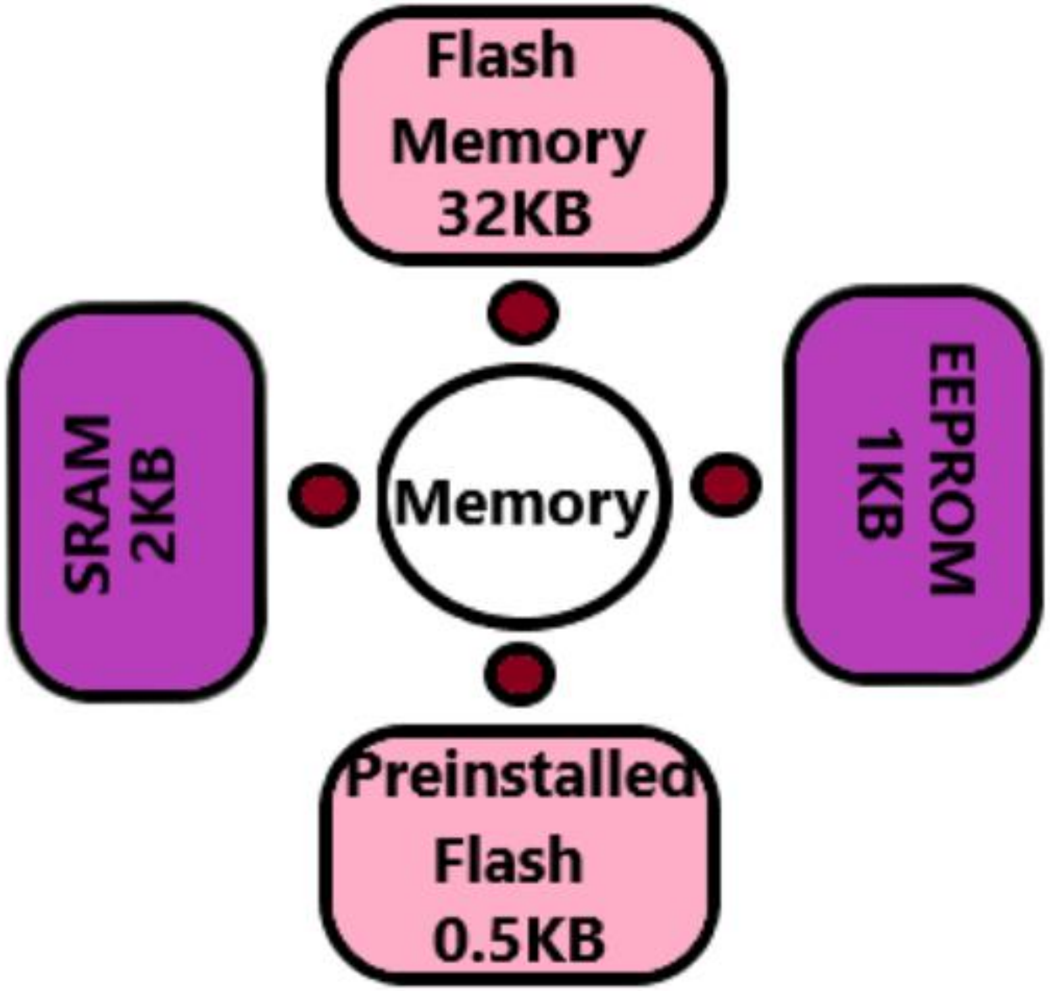

AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
UNO
ARDUINO
ON
ICSP
WWW.ARDUINO.CC - MADE IN ITALY
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5


TX and RX LED's
Reset Button
AREF
Digital I/O pins
USB
Power LED Indicator
ICSP pin
Crystal Oscillator
Voltage Regulator
ATmega328 Microcontroller
Power Button
RESET
3.3V
5V
GND
Vin
Analog pins

- **ATmega328 Microcontroller**- It is a single chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines **Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.**
- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator**- The ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- **Digital I/O pins**- The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.
- **TX and RX LED's**- The successful flow of data is represented by the lighting of these LED's.

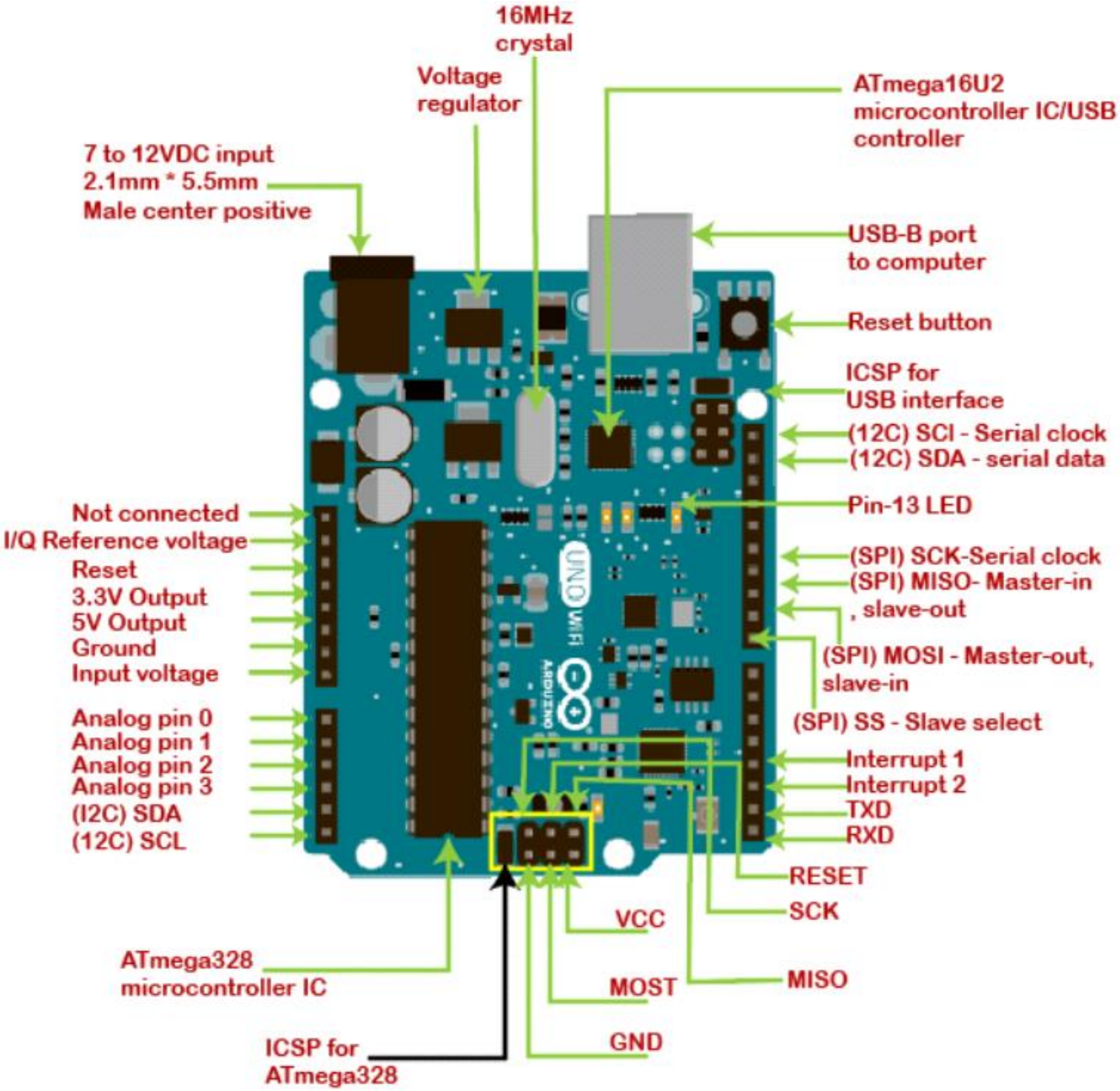
- **AREF-** The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button**- It is used to add a Reset button to the connection.
- **USB**- It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V.
- **GND**- Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin**- It is the input voltage.
- **Analog Pins**- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

Memory


Flash Memory 32KB
SRAM 2KB
Memory
EEPROM 1KB
Preinstalled Flash 0.5KB

Technical Specifications of Arduino UNO

- There are 20 Input/Output pins present on the Arduino UNO board. These 20 pis include 6 PWM pins, 6 analog pins, and 8 digital I/O pins.
- The PWM pins are Pulse Width Modulation capable pins.
- The crystal oscillator present in Arduino UNO comes with a frequency of 16MHz.
- It also has a Arduino integrated WiFi module. Such Arduino UNO board is based on the Integrated WiFi ESP8266 Module and ATmega328P microcontroller.
- The input voltage of the UNO board varies from 7V to 20V.
- Arduino UNO automatically draws power from the external power supply. It can also draw power from the USB.


16MHz crystal
Voltage regulator
ATmega16U2 microcontroller IC/USB controller
7 to 12VDC input 2.1mm * 5.5mm Male center positive
USB-B port to computer
Reset button
ICSP for USB interface
(12C) SCI - Serial clock
(12C) SDA - serial data
Pin-13 LED
Not connected
I/Q Reference voltage
Reset
3.3V Output
5V Output
Ground
Input voltage
(SPI) SCK-Serial clock
(SPI) MISO- Master-in , slave-out
(SPI) MOSI - Master-out, slave-in
(SPI) SS - Slave select
Analog pin 0
Analog pin 1
Analog pin 2
Analog pin 3
(I2C) SDA
(12C) SCL
Interrupt 1
Interrupt 2
TXD
RXD
RESET
SCK
VCC
MISO
ATmega328 microcontroller IC
MOST
GND
ICSP for ATmega328

Arduino Functions

- The functions allow a programmer to divide a specific code into various sections, and each section performs a particular task. The functions are created to perform a task multiple times in a program.
- The function is a type of procedure that returns the area of code from which it is called.
- For example, to repeat a task multiple times in code, we can use the same set of statements every time the task is performed.

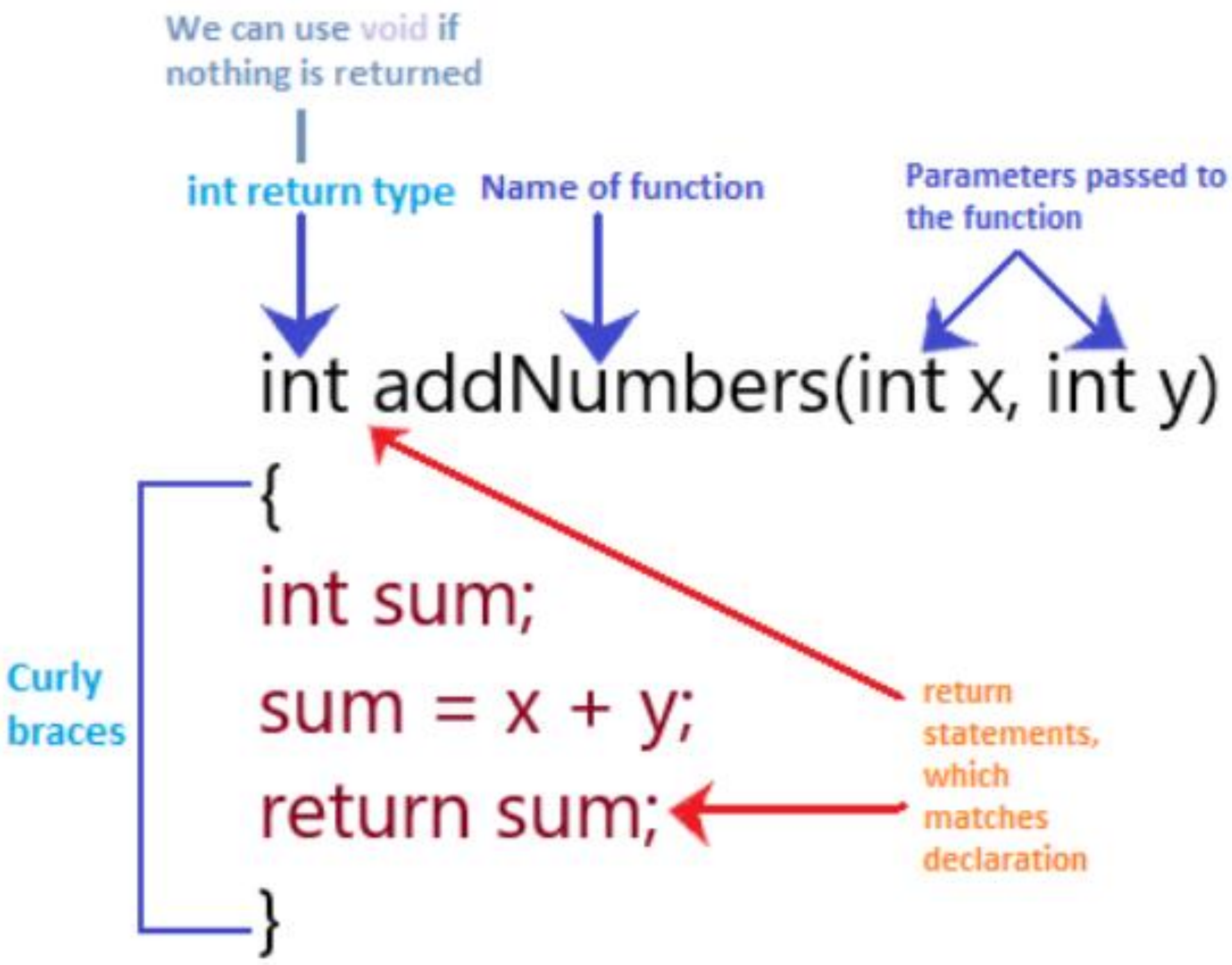
Advantages of using Functions

- It increases the readability of the code.
- It conceives and organizes the program.
- It reduces the chances of errors.
- It makes the program compact and small.
- It avoids the repetition of the set of statements or codes.
- It allows us to divide a complex code or program into a simpler one.
- The modification becomes easier with the help of functions in a program.

- The Arduino has two common functions **setup()** and **loop(),** which are called automatically in the background. The code to be executed is written inside the curly braces within these functions.
- **void setup()** - It includes the initial part of the code, which is executed only once. It is called as the **preparation block**.
- **void loop()** - It includes the statements, which are executed repeatedly. It is called the **execution block**.
- But sometimes, we need to write our own functions.

- Function Declaration
- The method to declare a function is listed below:
- **Function return type**
- We need a return type for a function. For example, we can store the return value of a function in a variable.
- We can use any data type as a return type, such as **float, char**, etc.

- **Function name**
- It consists of a name specified to the function. It represents the real body of the function.
- **Function parameter**
- It includes the parameters passed to the function. The parameters are defined as the special variables, which are used to pass data to a function.
- The function must be followed by **parentheses ( )** and the **semicolon ;**
- The actual data passed to the function is termed as an argument.



```
int addNumbers(int x, int y)
{
int sum;
sum = x + y;
return sum;
}
```

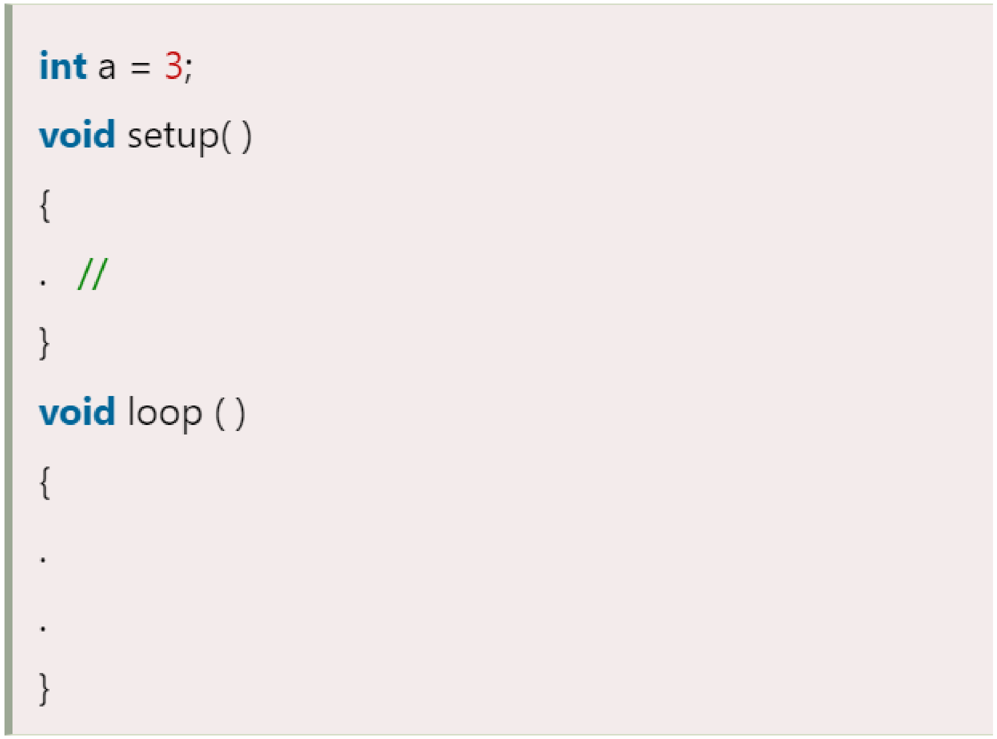
We can use void if nothing is returned
int return type
Name of function
Parameters passed to the function
Curly braces
return statements, which matches declaration

Arduino Data Types

- The data types are used to identify the types of data and the associated functions for handling the data. It is used for declaring functions and variables, which determines the bit pattern and the storage space.
- The data types that we will use in the Arduino are listed below:
- void Data Type
- int Data Type
- Char Data Type
- Float Data Type
- Double Data Type
- Unsigned int Data Type
- short Data Type
- long Data Type
- Unsigned long Data Type
- byte data type
- word data type

void Data Type

- The void data type specifies the empty set of values and only used to declare the functions. It is used as the return type for the functions that do not return any value.



```
int a = 3;
void setup( )
{
.  //
}
void loop ( )
{
.
.
}
```

Int Data Type

- The integer data types are the whole numbers like 5, -6, 10, -123, etc. They do not have any fractional part. The integer data types are represented by **int**. It is considered as the primary data type to store the numbers.
- The size of int is 2 bytes (16 bits).
- Minimal range: -32768 to 32767 or - (2^ 15) to ((2 ^ 15) - 1)

```
int var = val;
```

where,

var= variable

value = the value assigned to the variable

For example,

```
int a;

int b = 3;
```

Char Data Type

- The char data type can store any number of character set. An identifier declared as the char becomes a character variable. The literals are written inside a single quote.
- The char type is often said to be an integer type. It is because, symbols, letters, etc., are represented in memory by associated number codes and that are only integers.
- The size of character data type is **minimum of 8 bits**. We can use the byte data type for an unsigned char data type of 8 bits or 1 byte.

The syntax is:

```
char var = val;
```

where,

var= variable

val = The value assigned to the variable.

```
char myvariable = ' B ';
char myvariable = 66 ;  // both the value are equivalent
```

Float Data Type

- A number having the fractional part and a decimal part is considered as a floating-point number.
- For example, 4.567 is a floating-point number.
- The number 13 is an integer, while 13.0 is a floating-point number.
- Due to their greater resolution, fractional numbers are used to approximate the contiguous and analog values.
- Floating point numbers can also be written in the exponent form.
- The numbers can be as large as 3.4028235E+38 and as small as -3.4028235E+38.
- The size of float data types is 4 bytes or 32 bits.

The syntax is:

```
float var = val;
```

where,

var= variable

val = The value assigned to the variable

```
int a ;
int b ;
float c ;
void setup ( )
{
Serial.begin (9600);
}
void loop ( )
{
a = 3 ;
b = a/2 ;  // b is an integer. It cannot hold fractions. The output will be 1.
c = (float) a / 2.0 ;  // c now contains 1.5.
// Here, we have to use 2.0 instead of 2.
}
```

Double Data Type

- The double data type is also used for handling the decimal or floating-point numbers.
- It occupies twice as much memory as float.
- It stores floating point numbers with larger precision and range. It stands for double precision floating point numbers.
- It occupies 4 bytes in ATmega and UNO boards, while 8 bytes on Arduino Due.

The syntax is:

```
double var = val;
```

where,

var= variable

val = The value assigned to the variable

Unsigned int Data Type

- The unsigned int stores the value up to 2 bytes or 16 bits.
- It stores only positive values.
- The range of unsigned int data type is from 0 to 65,535 or 0 to ((2 ^ 16) - 1).
- Arduino Due stores the unsigned data value of 4 bytes or 32-bits.
- The difference between Unsigned and signed data type is the sign bit.
- The int type in Arduino is the signed int.
- In a 16-bit number, 15 bits are interpreted with the 2's complement, while the high bit is interpreted as the positive or negative number.
- If the high bit is '1', it is considered as a negative number.

The syntax is:

```
unsigned int var = val;
```

where,

var= variable

val = The value assigned to the variable

short Data Type

- The short is an integer data type that stores two bytes or 16-bit of data.

- The range of short data types is from -32768 to 32767 or - (2^ 15) to ((2 ^ 15) - 1).
- The ARM and ATmega based Arduino's usually stores the data value of 2 bytes.

The syntax is:

```
short var = val;
```

where,

var= variable

val = the value assigned to the variable

<u>long Data Type</u>

- The long data types are considered as the extended size variables, which store 4 bytes (32 -bits).
- The size ranges from -2,147,483,648 to 2,147,483,647.
- While using integer numbers, at least one of the numbers should be followed by L, which forces the number to be a long data type.

The syntax is:

```
long var = val;
```

where,

var= variable

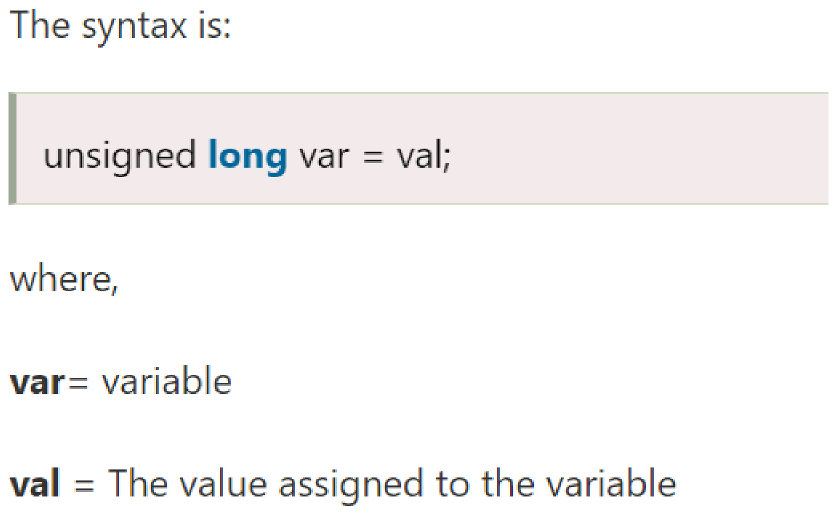
val = The value assigned to the variable

For example,

```
long speed = 186000L;
```

Unsigned long Data Type

- The unsigned long data types are also considered as the extended size variables, which store 4 bytes (32 -bits).
- It does not store negative numbers like other unsigned data types, which makes their size ranges from 0 to 4,294,967,295 or (2^32 - 1).



The syntax is:

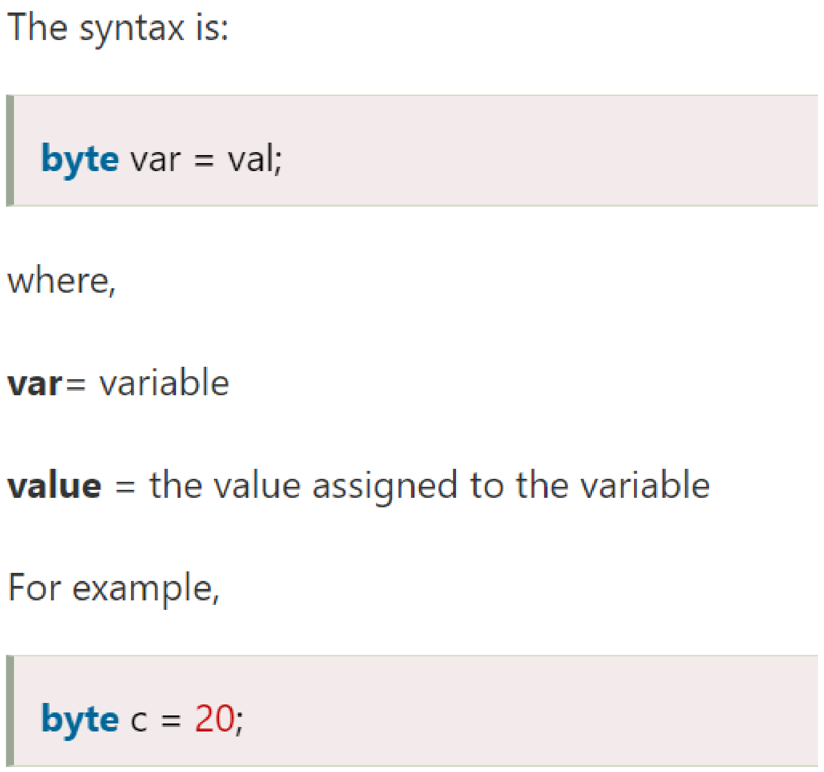
```
unsigned long var = val;
```

where,

var= variable

val = The value assigned to the variable

byte

- 1 byte = 8 bits.
- It is considered as an unsigned number, which stores values from 0 to 255.
- 

The syntax is:

```
byte var = val;
```

where,

var= variable

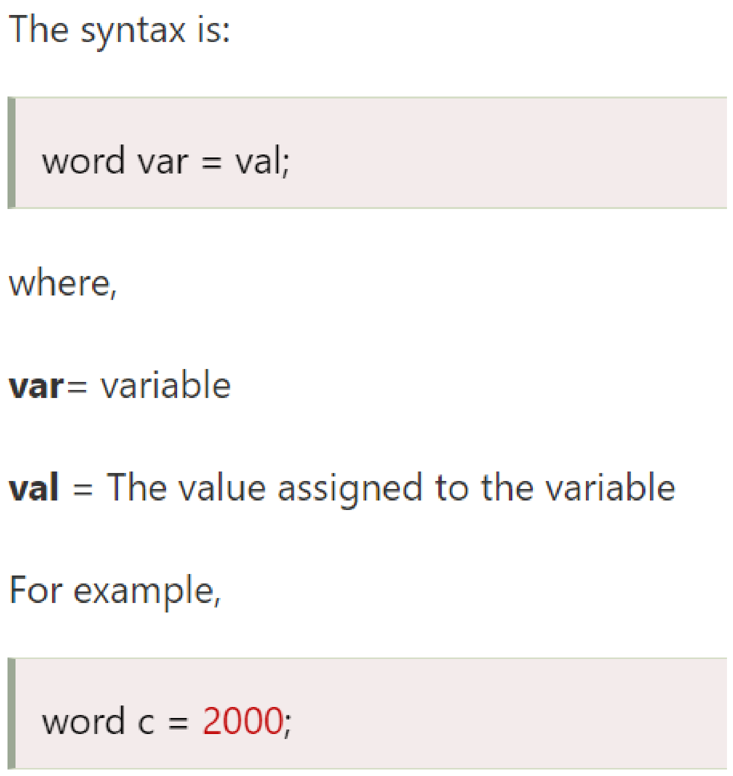
value = the value assigned to the variable

For example,

```
byte c = 20;
```

word

- It is considered as an unsigned number of 16 bits or 2 bytes,
- which stores values from 0 to 65535.



The syntax is:

```
word var = val;
```

where,

var= variable

val = The value assigned to the variable

For example,

```
word c = 2000;
```