الجامعة العربية الأمريكية
ARAB AMERICAN UNIVERSITY

EMBEDDED SYSTEMS LAB

Arduino Programming

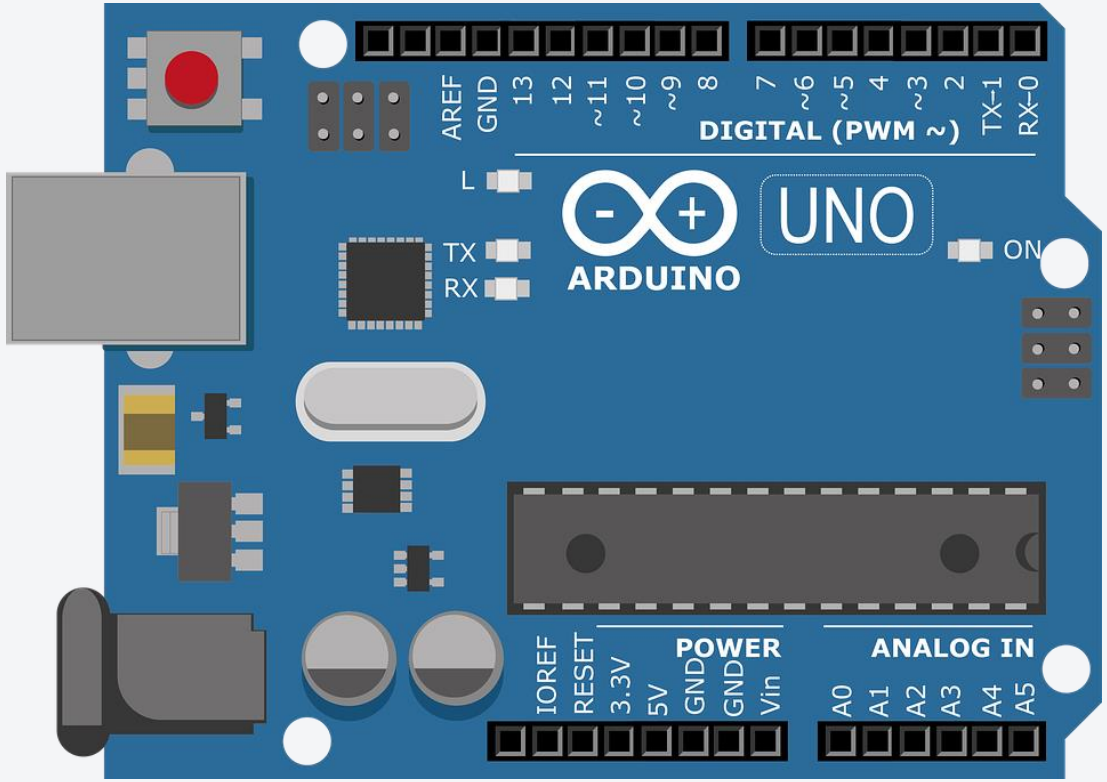
Eng. Hussein Younis

YouTube Channel

Hussein Younis
YouTube channel

SCAN ME

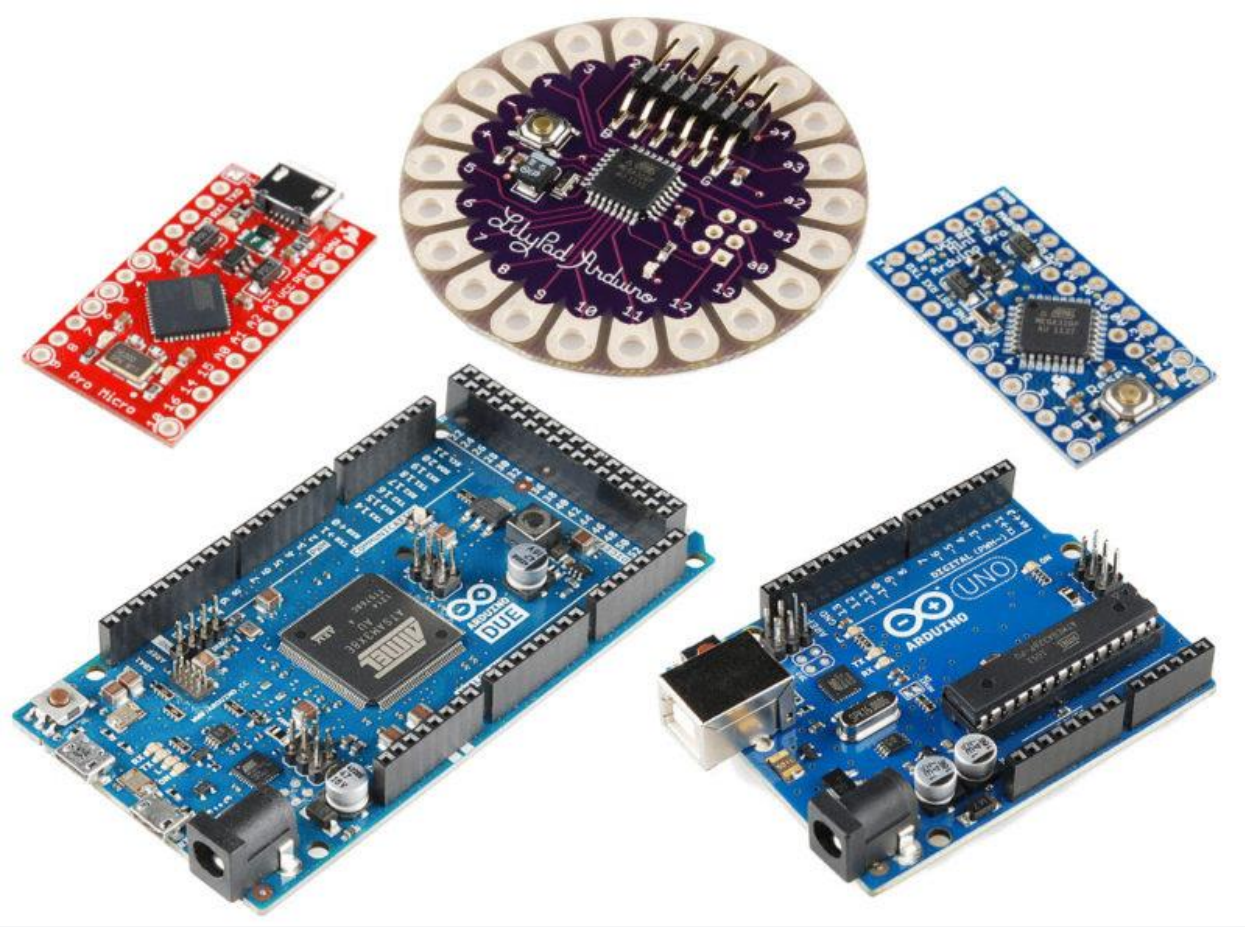
Arduino


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

What Is Arduino?

Arduino is an open source programmable circuit board that can be integrated into a wide variety of makerspace projects both simple and complex. This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world.

Types of Arduino Boards

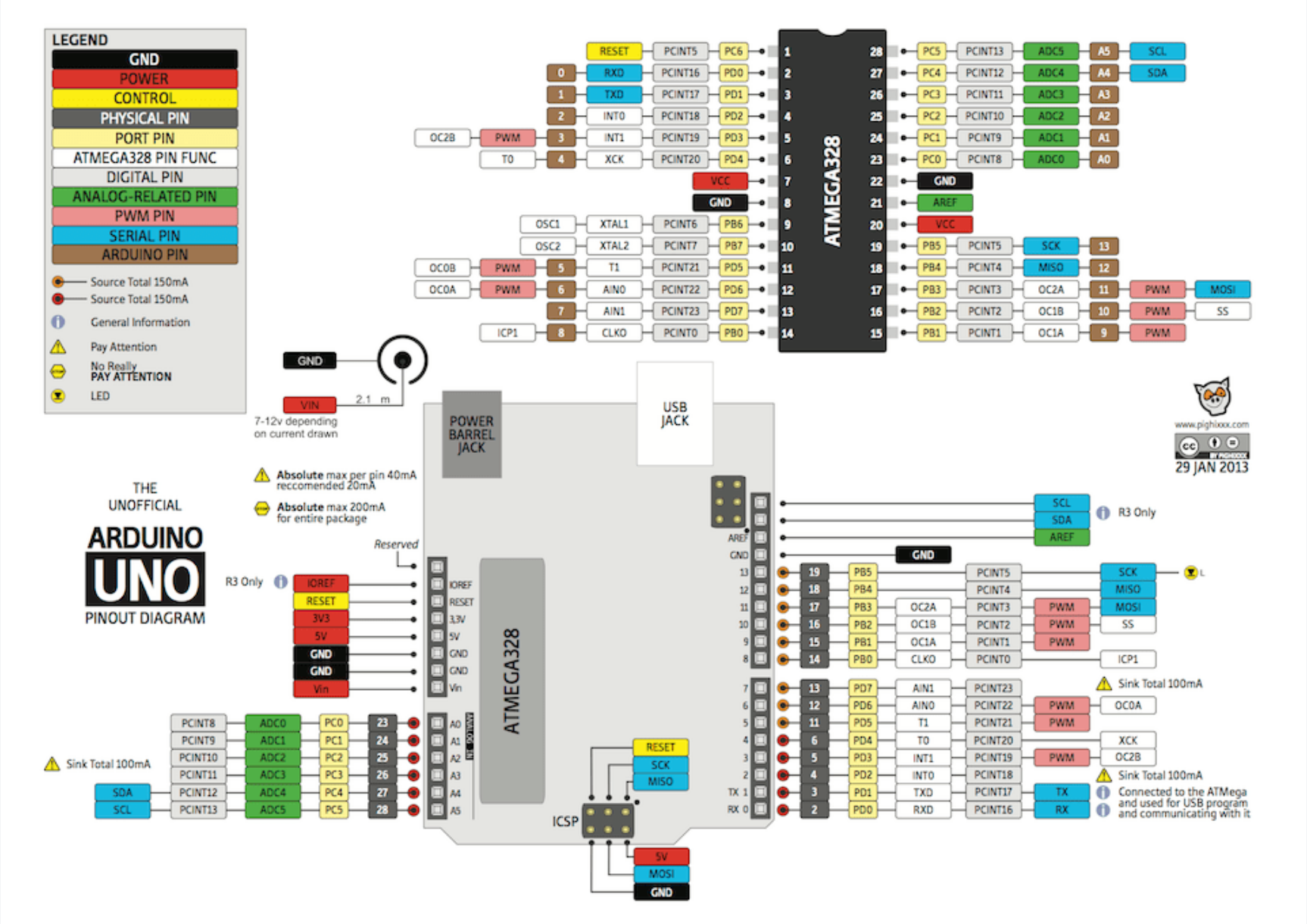


Arduino Uno

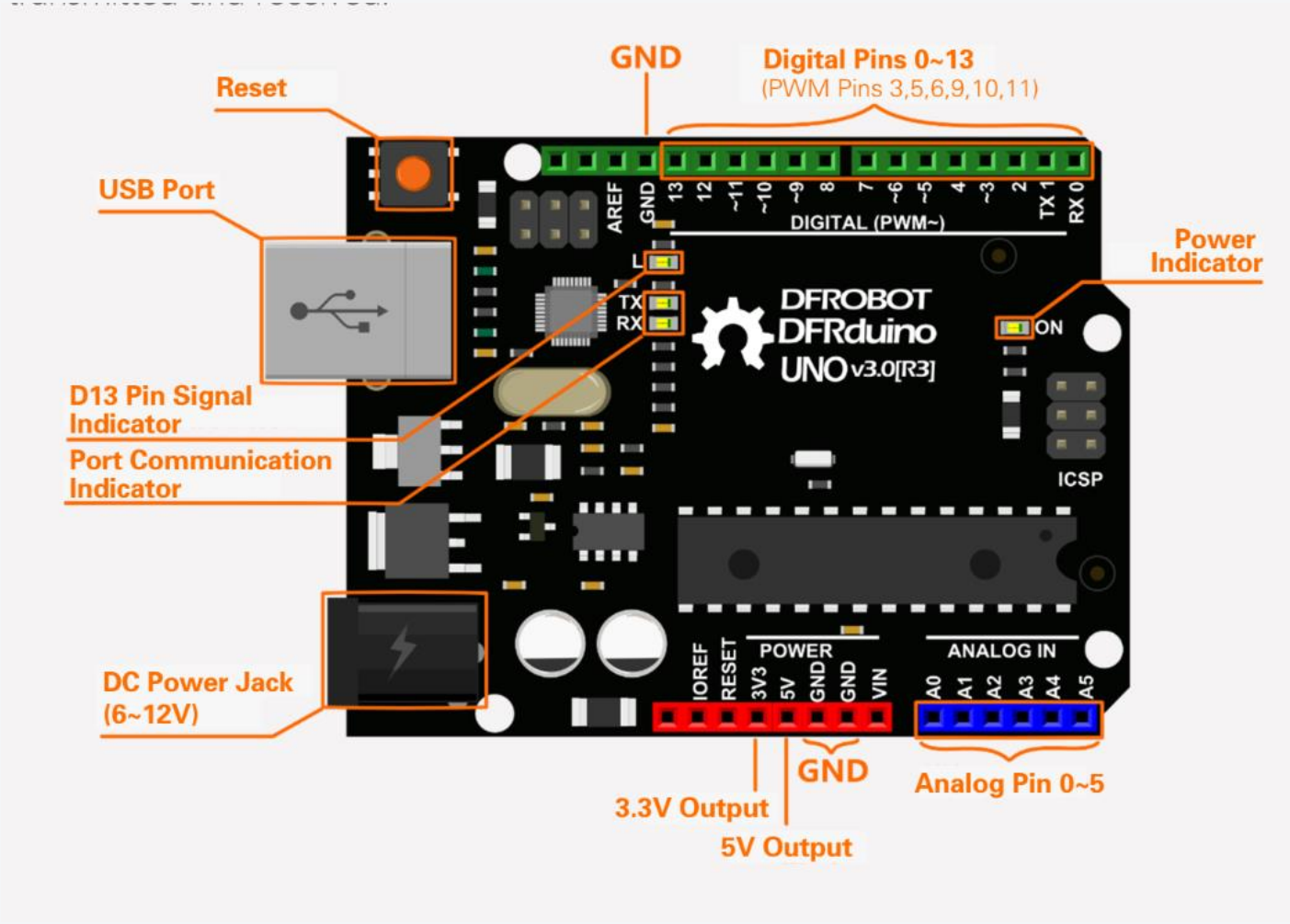
Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button.

MICROCONTROLLER	ATMEGA328P
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	14 (OF WHICH 6 PROVIDE PWM OUTPUT)
PWM DIGITAL I/O PINS	6
ANALOG INPUT PINS	6
DC CURRENT PER I/O PIN	20 MA
DC CURRENT FOR 3.3V PIN	50 MA
FLASH MEMORY	32 KB (ATMEGA328P) OF WHICH 0.5 KB USED BY BOOTLOADER
SRAM	2 KB (ATMEGA328P)
EEPROM	1 KB (ATMEGA328P)
CLOCK SPEED	16 MHZ
LED_BUILTIN	13
LENGTH	68.6 MM
WIDTH	53.4 MM
WEIGHT	25 G

Arduino Uno Pinout Diagram



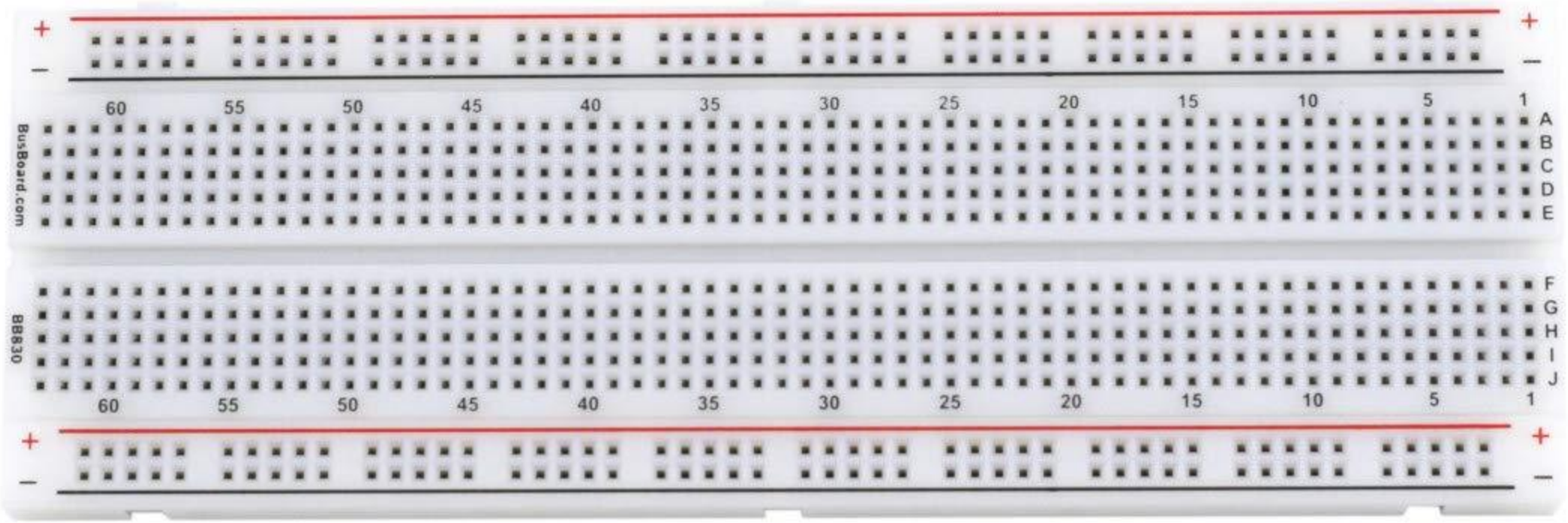
Arduino Uno Pinout



Arduino Power Supply

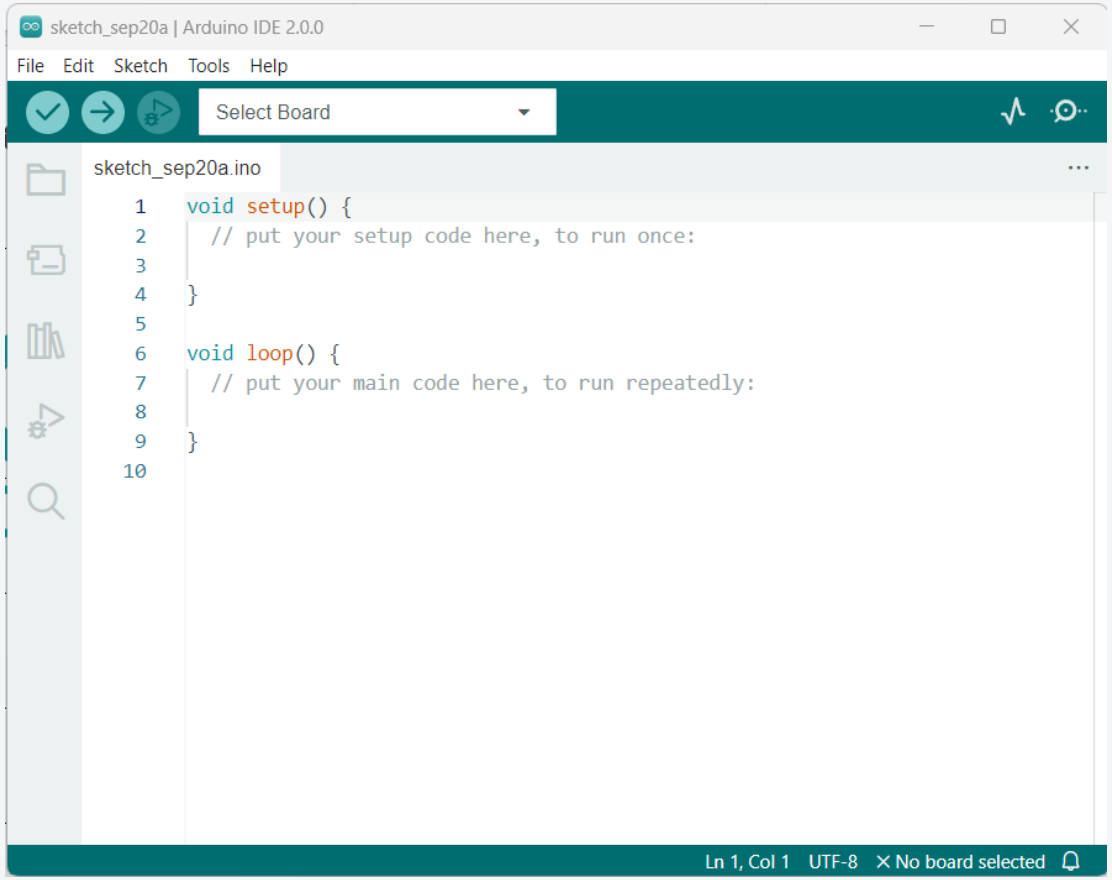
The Arduino Uno needs a power source in order for it to operate and can be powered in a variety of ways. You can do what most people do and connect the board directly to your computer via a USB cable. If you want your project to be mobile, consider using a 9V battery pack to give it juice. The last method would be to use a 9V AC power supply.

Arduino Breadboard



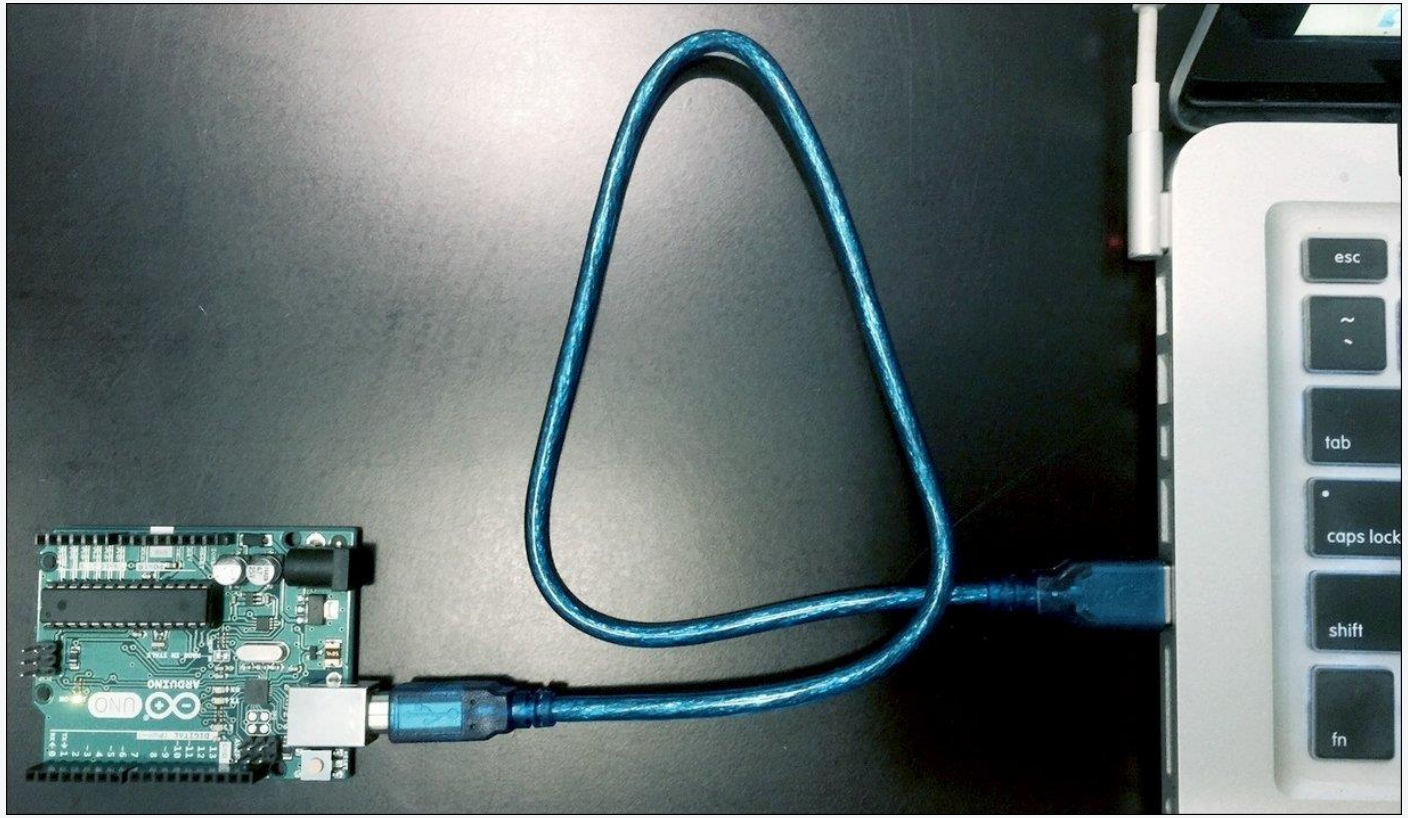
How To Program Arduino

https://www.arduino.cc/en/software



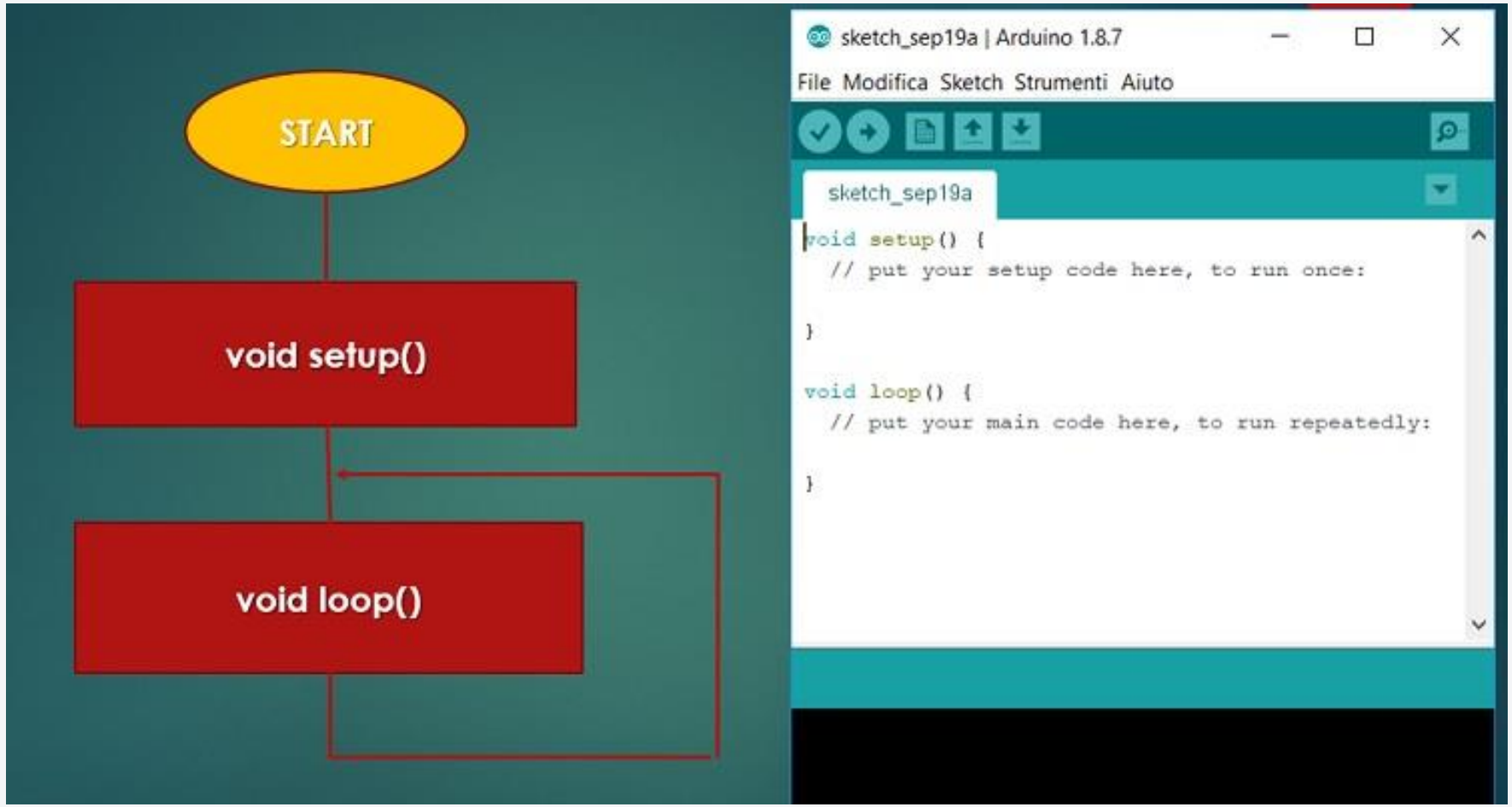
Connecting Arduino Board to the Computer

To connect the Arduino board to the computer, simply connect the appropriate cable to the Arduino board and connect the other end to the USB port of your PC. The power LED will glow indicating the board is powered. The system will automatically install the driver for the board.



Arduino IDE – Basics

Arduino program structure



Arduino constants

HIGH

The meaning of HIGH (in reference to a pin) is somewhat different depending on whether a pin is set to an INPUT or OUTPUT. When a pin is configured as an INPUT with pinMode(), and read with digitalRead(), the Arduino (ATmega) will report HIGH if:

- a voltage greater than 3.0V is present at the pin (5V boards)
- a voltage greater than 2.0V is present at the pin (3.3V boards)

LOW

The meaning of LOW also has a different meaning depending on whether a pin is set to INPUT or OUTPUT. When a pin is configured as an INPUT with pinMode(), and read with digitalRead(), the Arduino (ATmega) will report LOW if:

- a voltage less than 1.5V is present at the pin (5V boards)
- a voltage less than 1.0V (Approx) is present at the pin (3.3V boards)

false

false is defined as 0 (zero).

true

true is often said to be defined as 1, which is correct, but true has a wider definition. Any integer which is non-zero is true, in a Boolean sense. So -1, 2 and -200 are all defined as true, too, in a Boolean sense.

Arduino Conversion

CONVERSION	DESCRIPTION	SYNTAX	RETURNS
(unsigned int)	Converts a value to the unsigned int data type.	(unsigned int)x	unsigned int
(unsigned long)	Converts a value to the unsigned long data type.	(unsigned long)x	unsigned long
byte()	Converts a value to the byte data type.	byte(x)	Data type: byte.
char()	Converts a value to the char data type.	char(x)	Data type: char.
float()	Converts a value to the float data type.	float(x)	Data type: float.
int()	Converts a value to the int data type.	int(x)	Data type: int.
long()	Converts a value to the long data type.	long(x)	Data type: long.

Arduino Data Types

array

An array is a collection of variables that are accessed with an index number. Arrays in the C++ programming language Arduino sketches are written in can be complicated, but using simple arrays is relatively straightforward.

example.cpp

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[5] = {2, 4, -8, 3, 2};
char message[6] = "hello";
```

example.cpp

```
mySensVals[0] = 10;

x = mySensVals[4];

for (byte i = 0; i < 5; i = i + 1) {
  Serial.println(myPins[i]);
}

if(mySensVals[0] == 2){}
```

Arduino Data Types

byte

A byte stores an 8-bit unsigned number, from 0 to 255.

example.cpp

```
1   byte var = val;
```

Arduino Data Types

char

A data type used to store a character value. Character literals are written in single quotes, like this: 'A' (for multiple characters - strings - use double quotes: "ABC").

Characters are stored as numbers however. You can see the specific encoding in the ASCII chart. This means that it is possible to do arithmetic on characters, in which the ASCII value of the character is used (e.g. 'A' + 1 has the value 66, since the ASCII value of the capital letter A is 65). See Serial.println reference for more on how characters are translated to numbers.

The size of the char datatype is at least 8 bits. It’s recommended to only use char for storing characters. For an unsigned, one-byte (8 bit) data type, use the byte data type.

Arduino Data Types

char

A data type used to store a character value. Character literals are written in single quotes, like this: 'A' (for multiple characters - strings - use double quotes: "ABC"). The size of the char datatype is at least 8 bits. It's recommended to only use char for storing characters. For an unsigned, one-byte (8 bit) data type, use the byte data type.

example.cpp

```
char myChar = 'A';
char myChar = 65; // both are equivalent
```

Arduino Data Types

float

Datatype for floating-point numbers, a number that has a decimal point. Floating-point numbers are often used to approximate analog and continuous values because they have greater resolution than integers. Floating-point numbers can be as large as 3.4028235E+38 and as low as -3.4028235E+38. They are stored as 32 bits (4 bytes) of information.

example.cpp

```
1
2   float sensorCalbrate = 1.117;
```

Arduino Data Types

string

Text strings can be represented in two ways. you can use the String data type, which is part of the core as of version 0019, or you can make a string out of an array of type char and null-terminate it.

example.cpp

```
char Str1[15];
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[] = "arduino";
char Str5[8] = "arduino";
char Str6[15] = "arduino";
```

Arduino Data Types

unsigned char

An unsigned data type that occupies 1 byte of memory. Same as the byte datatype. The unsigned char datatype encodes numbers from 0 to 255.

example.cpp

```
unsigned char myChar = 240;
```

Arduino Functions: Digital I/O

digitalRead()

Reads the value from a specified digital pin, either HIGH or LOW.

example.cpp

```
int ledPin = 13;  // LED connected to digital pin 13
int inPin = 7;    // pushbutton connected to digital pin 7
int val = 0;      // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT);  // sets the digital pin 13 as output
  pinMode(inPin, INPUT);    // sets the digital pin 7 as input
}

void loop() {
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);  // sets the LED to the button's value
}
```

Arduino Functions: Digital I/O

digitalWrite()

Write a HIGH or a LOW value to a digital pin.

example.cpp

```
void setup() {
  pinMode(13, OUTPUT);    // sets the digital pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // sets the digital pin 13 off
  delay(1000);            // waits for a second
}
```

Arduino Functions: Digital I/O

pinMode()

Configures the specified pin to behave either as an input or an output.

example.cpp

```
void setup() {
  pinMode(13, OUTPUT);    // sets the digital pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // sets the digital pin 13 off
  delay(1000);            // waits for a second
}
```

Arduino Functions: Analog I/O

analogRead() Reads the value from the specified analog pin

example.cpp

```
int analogPin = A3; // potentiometer wiper (middle terminal) connected to analog pin 3
                    // outside leads to ground and +5V
int val = 0;  // variable to store the value read

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

void loop() {
  val = analogRead(analogPin);  // read the input pin
  Serial.println(val);          // debug value
}
```

Arduino Functions: Analog I/O

analogWrite() Writes an analog value (PWM wave) to a pin

example.cpp

```
int ledPin = 9;       // LED connected to digital pin 9
int analogPin = 3;    // potentiometer connected to analog pin 3
int val = 0;          // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT);  // sets the pin as output
}

void loop() {
  val = analogRead(analogPin);  // read the input pin
  analogWrite(ledPin, val / 4); // analogRead values go from 0 to 1023, analogWrite values from 0 to 255
}
```

Arduino Functions: Time

delay()

Pauses the program for the amount of time (in milliseconds) specified as parameter.

example.cpp

```
int ledPin = 13;              // LED connected to digital pin 13

void setup() {
  pinMode(ledPin, OUTPUT);    // sets the digital pin as output
}

void loop() {
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000);                // waits for a second
  digitalWrite(ledPin, LOW);  // sets the LED off
  delay(1000);                // waits for a second
}
```

Arduino Functions: Time

delayMicroseconds()

Pauses the program for the amount of time (in microseconds) specified by the parameter.

example.cpp

```
int outPin = 8;                // digital pin 8

void setup() {
  pinMode(outPin, OUTPUT);     // sets the digital pin as output
}

void loop() {
  digitalWrite(outPin, HIGH); // sets the pin on
  delayMicroseconds(50);       // pauses for 50 microseconds
  digitalWrite(outPin, LOW);   // sets the pin off
  delayMicroseconds(50);       // pauses for 50 microseconds
}
```

Arduino Functions: Bits and Bytes

bitRead()

Reads a bit of a number.

example.cpp

```
void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  int x = 6;
  int n = 1;
  Serial.print(bitRead(x, n)); // print the output of bitClear(x,n)
}

void loop() {
}
```

Arduino Functions: Bits and Bytes

bitSet()

Sets (writes a 1 to) a bit of a numeric variable.

example.cpp

```
void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  int x = 6;
  int n = 1;
  Serial.print(bitSet(x, n)); // print the output of bitClear(x,n)
}

void loop() {
}
```

Arduino Functions: Bits and Bytes

bitWrite()

Writes a bit of a numeric variable.

example.cpp

```
void setup() {
  Serial.begin(9600);
  while (!Serial) {}  // wait for serial port to connect. Needed for native USB port only
  byte x = 0b10000000;  // the 0b prefix indicates a binary constant
  Serial.println(x, BIN); // 10000000
  bitWrite(x, 0, 1);  // write 1 to the least significant bit of x
  Serial.println(x, BIN); // 10000001
}

void loop() {}
```