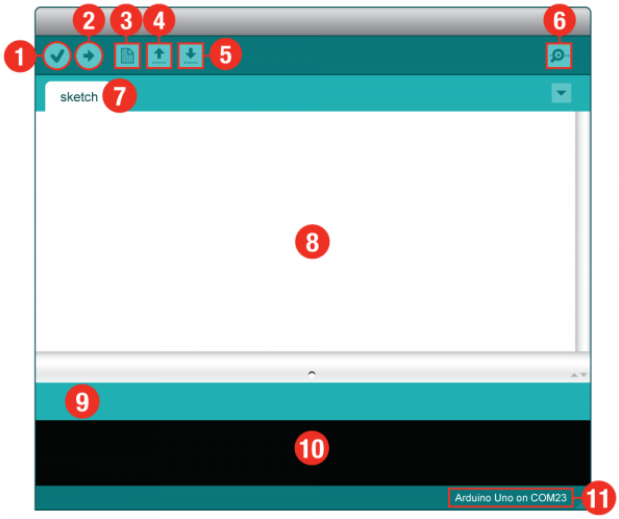
PRACTICA 1.

Materiales:

1. Computador con puerto USB , el IDE de arduino y las librerías para el ESP8266.
2. Placa de desarrollo NodeMCU con ESP8266.
3. Cable USB a mini USB para conexión de la placa.

Las placas NodeMCU incorporan un conversor USB a serie. Solamente es necesario conectar el puerto USB de la placa a uno de los puertos de nuestro ordenador con un buen cable**. (OJO algunos cables que vienen para cargadores de celulares no tienen el hilo de datos ya que solo van a llevar potencia y por tal razón no sirven para programar el dispositivo).**

El sistema operativo normalmente detecta el hardware e instala los drivers automáticamente. Programa la placa a 115200 bps. Las partes principales del IDE de arduino son:



1. **Verificar:** compila y aprueba su código. Detectará los errores de sintaxis (por ejemplo si faltan paréntesis o punto y comas).
2. **Cargar:** envía su código a la placa 101.
3. **Nuevo:** este botón abre una nueva pestaña de la ventana de código.
4. **Abrir:** este botón le permitirá abrir un boceto existente.
5. **Guardar:** guarda el boceto activo actualmente.
6. **Monitor serie:** abrirá una ventana que muestra cualquier información de serie que la placa 101 esté transmitiendo. Es muy útil para la depuración.
7. **Nombre del boceto:** muestra el nombre del boceto en el que está trabajando actualmente.

8. **Área de código:** esta es la zona donde compone el código de su boceto.
9. **Área de mensajes:** aquí el IDE le indica si existen errores en el código.
10. **Consola de texto:** la consola de texto muestra mensajes de error completos. La consola de texto es muy útil a la hora de depurar.
11. **Placa y puerto serie:** le muestra la selección de placa y puerto serie.

Para empezar de la manera más sencilla vamos a aprender cómo utilizar el botón y el led que incluye la NodeMCU. El LED está conectado, internamente, al GPIO2 y el botón al GPIO0. Debemos ingresar el siguiente código al área de código en el IDE de arduino.

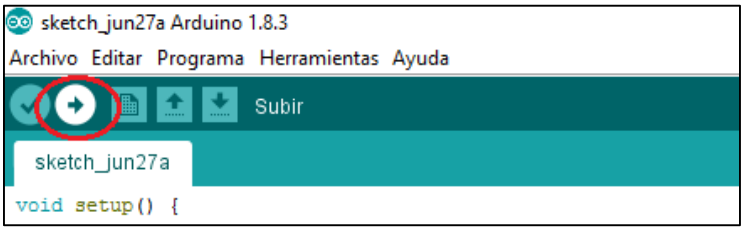
```
/*
  Este ejemplo sirve para encender y apagar el led integrado en la tarjeta de desarrollo NodeMCU
*/

void setup() {
  pinMode(02, OUTPUT);     // Se inicializa el puerto GPI02 (D4)
}

// El loop es un lazo cerrado que encenderá y apagará el led integrado

void loop() {
  digitalWrite(02, LOW);   // La configuración pull-up hará que encienda el LED
  delay(1000);                      // Espera un segundo
  digitalWrite(02, HIGH);  // Cambia a estado OFF, apagará el LED
  delay(3000);                      // Ahora espera 3segundos
}
```

Hay que tener en cuenta que el LED incorporado por la tarjeta tiene configuración Pull-UP (Se verá más adelante) al igual que la mayoría de sus puertos, por esta razón este led funciona con logica inversa, para encender el led integrado se escribe LOW y para apagar este led DOWN. **OJO solo aplica para el LED integrado en la tarjeta.**



Después de subir el código presionando la flecha como se muestra en la figura anterior, deberá encender y apagar el led integrado en la tarjeta de desarrollo NodeMCU a la velocidad especificada por el código.

Bien, ahora en el siguiente código, vamos a utilizar el botón FLASH del NodeMCU. Este botón nos permite poner en modo carga del firmware. Como ya te he comentado, esto lo hace automáticamente al igual que las placas de Arduino, en un modo normal no tenemos que pulsar nada para cargar nuestro código.

Con este código el LED integrado en la tarjeta encenderá y se apagará al presionar el botón flash de la tarjeta.

```
#define LED_BUILTIN 2
#define BUTTON_BUILTIN 0

void setup() {
  // El LED integrado está conectado al pin GPIO2. Será una salida
  // El switch se encuentra en el pinGPIO0 y será una entrada
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(BUTTON_BUILTIN, INPUT);
}

void loop() {
  // Leer el estado del botón. Cuando está pulsado el pin se pone a nivel bajo
  int estado = digitalRead(BUTTON_BUILTIN);
  // Configurar el LED de acuerdo al estado del botón
  digitalWrite(LED_BUILTIN, estado);
}
```

#define en C es un componente útil que permite al programador para dar un nombre a un valor constante antes de compilar el programa. Las constantes no ocupan ningún espacio de memoria de programa en el chip. El compilador reemplaza las referencias a estas constantes con el valor definido en tiempo de compilación.

#Int (Enteros) almacena valores numéricos de 16 bits sin decimales comprendidos en el rango 32767 a -32768. Las variables de tipo entero 'int' pueden sobrepasar su valor máximo o mínimo como consecuencia de una operación. Por ejemplo, si x = 32767 y una posterior declaración agrega 1 a x, x = x + 1 entonces el valor se x pasará a ser -32768 (algo así como que el valor da la vuelta).

El código utilizando int sería:

```
int led = 2;
int sw = 0;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(sw, INPUT);
}


void loop() {

  int valor = digitalRead(sw);
  digitalWrite(led, valor);

}
```

TRABAJO PARA LA CASA.

1. Realice un cuadro comparativo donde indique la función y el nombre de cada uno de los pines de la tarjeta de desarrollo NodeMCU versión 1.0 y versión 3.0

2. Explique las diferencias entre la versión 1.0 y la versión 3.0 de la tarjeta de desarrollo utilizada.