a) **New Sketch:** Abrir el Arduino IDE. Crear un nuevo sketch. Cambiarle el nombre al archivo según su gusto. (blink)

b) **Boards Manager:** Ingresar al menú de Tools-> Board:-> Boards Manager. Buscar Node y verificar que la librería para poder utilizar el esp8266 ya esta instalada.

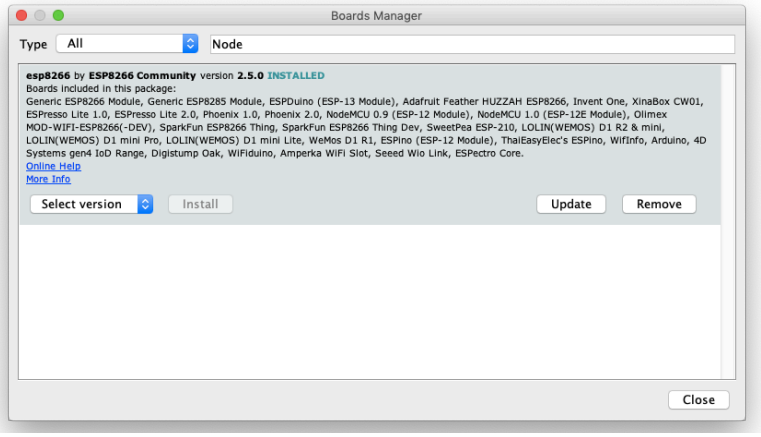

Ilustración 3. Librerías ESP8266

A continuación, conectar el microcontrolador al puerto USB. Después, ir al menú Tools-> Board: y seleccionar NodeMCU v1. Además, escoger el puerto USB activo donde se tiene conectado el microcontrolador.

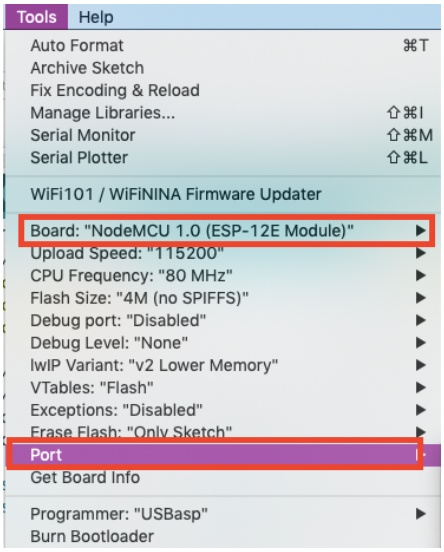

Ilustración 4. Menú Herramientas

c) **Conectar un LED:** el cátodo del LED (positivo) se conecta a uno de los pines digitales de la NodeMcu y el ánodo al pin de tierra (GND) como se ve en la imagen. La alimentación del NodeMCU es directa desde el computador a través de cable USB.

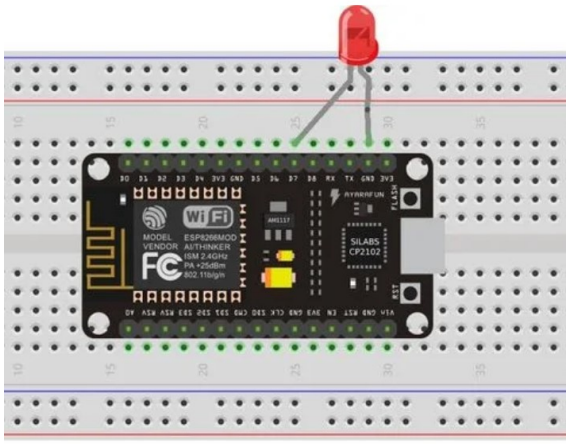

Ilustración 5. Diagrama de conexiones

Una señal digital es una onda que permite representar datos, usualmente de manera binaria, a través de valores fijos de voltaje. Para este primer tutorial se hará parpadear un LED utilizando una señal digital de lógica binaria, esto significa que se enviarán valores de 0V y 5V para encender y apagar el LED, lo que en lógica binaria significa 1 (HIGH) y 0 (LOW).

Antes de comenzar a programar se dará una introducción a los comandos básicos necesarios:

- const
  - Sintaxis: const <tipo><nombre>= <dato>;
  - Uso: Declara constantes globales en Arduino, que toman valores que no se pueden modificar una vez inicializados.
- pinMode
  - Sintaxis: pinMode(<pin>, <modo>);
  - Uso: Indica si un pin del Arduino será utilizado como entrada (INPUT), salida (OUTPUT), o entrada con resistencia de pull-up (INPUT_PULLUP).
- digitalWrite
  - Sintaxis: digitalWrite(<pin>, <valor>);
  - Uso: Escribe un valor de HIGH (5V) o LOW (0V) en un pin digital.
- delay
  - Sintaxis: delay(<tiempo en ms>);
  - Uso: Espera un número de milisegundos antes de continuar la ejecución del código.

Asimismo, se puede observar en la estructura del sketch dos métodos importantes, setup() y loop(). De igual manera, como se puede leer en los comentarios de cada método, el primero se ejecuta únicamente una vez al iniciar el programa, mientras que el segundo se ejecuta infinitamente (de manera similar a un loop while).

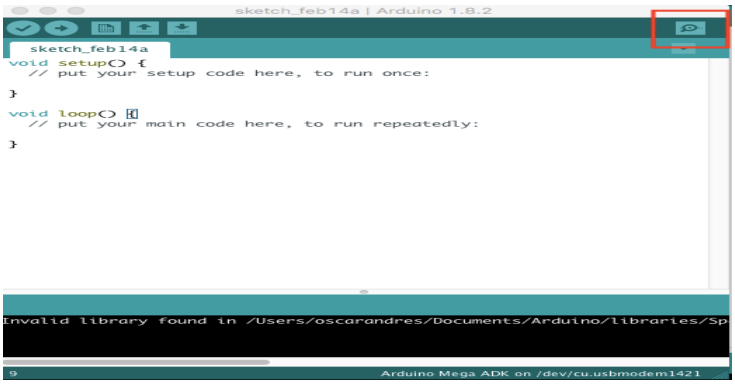

Ilustración 6. Arduino IDE

d) **Programar:** Realizar un sketch que se encargue de prender y apagar un LED cada cierto tiempo que será determinado por el programador.

Para ello, se sugiere declarar constantes que permitan mantener orden y legibilidad en el código. Los valores más importantes para la tarea a desarrollar son el pin al cual se conectará el LED y el periodo con el cual parpadea el mismo. Estas constantes se declaran al inicio del sketch y están disponibles en toda la ejecución.

```
//Configuracion
const int PIN_LED = 11; //Pin al que se conecta el LED
const int PERIODO = 100; //Periodo con el que parpadea el LED

void setup() {
  // Modo de operacion del pin PIN_LED
  pinMode(PIN_LED, OUTPUT);

}

void loop() {

  digitalWrite(PIN_LED, HIGH);// Encender el LED
  delay(PERIODO / 2); // Esperar medio periodo

  digitalWrite(PIN_LED, LOW);// Apagar el LED
  delay(PERIODO / 2); // Esperar medio periodo
}
```

Ilustración 7. Ejemplo sketch Blink

e) Cargar programa: Una vez sea finalizada la programación, se carga el programa al NodeMCU. Con el segundo botón de la parte superior izquierda.

f) New Sketch: Abrir el Arduino IDE. Crear un nuevo sketch. Cambiarle el nombre al archivo según su gusto. (Serial)

Un último concepto importante para tener en cuenta cuando se habla de comunicación serial es la tasa de baudios o baud rate. Dado que este protocolo transmite un bit detrás de otro, es necesario definir cuanto tiempo se demorará la tarjeta en comenzar a transmitir el siguiente dato. La tasa de baudios indica el número de datos que se transmite en un segundo de comunicación, y como se verá más adelante es esencial que ambas partes la conozcan para entender la información que se recibe.

Comandos básicos para llegar a implementar comunicación serial:

- Serial.begin
  - Sintaxis: Serial.begin(<baudrate>);
  - Uso: Inicia las comunicaciones seriales en el Arduino a la tasa de baudios seleccionada.
- Serial.available
  - Sintaxis: Serial.available();
  - Uso: Indica el número de datos que han llegado por comunicación serial y no han sido leídos aún.
- Serial.println
  - Sintaxis: Serial.println(<texto>);
  - Uso: Envía el texto bit a bit por el puerto serial, y le agrega un salto de línea al final.

g) Programar: Utilizando una variable entera “int” imprima los números del 1 al 100 en el Serial, cuando sea mayor a 100 reinicie la variable a 0. Utilice un delay para que sea posible visualizar los números de manera adecuada en el monitor Serial.