**INTERNET DE LAS COSAS GRUPO B01**

**ENTREGA 3**

**Articulo**

**Profesor**:

Logo, company name

Description automatically generatedJohn Olarte

INTEGRANTES:

**Andrés Felipe Villegas Corrales. Código: 100248416**

**Nelson David Caso Diaz**

**Julieth Tatiana Betancur Moreno**

.

**INSTITUCIÓN UNIVERSITARIA POLITÉCNICO GRANCOLOMBIANO**

**COLOMBIA**

**2022**

**TABLA DE CONTENIDO**

[**OBJETIVOS** 3](#_Toc121167586)

[**INTRODUCCIÓN** 3](#_Toc121167587)

[**ACTIVIDAD 3** 5](#_Toc121167588)

[**3.1 INVESTIGAR SOBRE EL FUNCIONAMIENTO Y DISEÑO DE CIRCUITOS CON CONEXIÓN INALÁMBRICA** 5](#_Toc121167589)

[**3.2 DISEÑAR, IMPLEMENTAR Y CONFIGURAR UN CIRCUITO CON UN ARDUINO Y UN MÓDULO ESP8266, MINIPLACA WI- FI.** 7](#_Toc121167590)

[**3.3 SE PUEDEN ENVIAR DATOS POR VÍA IP.** 13](#_Toc121167591)

[**CONCLUSIONES** 19](#_Toc121167592)

[**BIBLIOGRAFIA** 19](#_Toc121167593)

**URL de nuestro proyecto en Tinkercad y Git hub para Proteus y Arduino IDE**: <https://github.com/villegas984/Internetofthings.git>

<https://www.tinkercad.com/things/kk6HqEVjrgd-frantic-snaget/editel?sharecode=Ba2kTDcNGeTD2C01DyY4v2gpByFmXnpb_4mzXNwPIy0>

**Resumen**— Con el presente trabajo se realizará la TERCERA entrega entrega del curso Internet de las cosas (IoT), realizando un artículo el cual describe el proceso para garantizar la conectividad de los equipos en simulacion.

# **OBJETIVOS**

* Con el presente trabajo se busca realizar la tercera entrega del curso Internet de las cosas (IoT).
* De igual manera se desea Diseñar, implementar y configurar un circuito con un Arduino y un módulo ESP8266, miniplaca Wi- Fi..
* Investigar sobre el funcionamiento y diseño de circuitos con conexión inalámbrica.

# **INTRODUCCIÓN**

Hoy en día, debido a los avances en tecnología e Internet de las Cosas, tareas paso a paso como correo, acceso a registros bancarios, seguimiento del grado de otras tareas, salud, hogar, transporte inteligente, redes inalámbricas de sensores, red inteligente, objetos de control remoto, sistemas de monitoreo, son posibles. Sin embargo, para cualquier aplicación de Internet de las cosas (IoT), es importante tener en cuenta algunas cuestiones relacionadas a esta tecnología como seguridad y eficiencia energética especialmente en Redes de Conexión inalámbrica, que es un ad hoc o adyacente.

Las Redes de Sensores Inalámbricos son canales donde los datos del mundo físico pueden ser accedido y utilizado por cualquier dispositivo de computación en el mundo. En líneas similares, Internet de las cosas es la conexión de varios dispositivos de este tipo con la capacidad de compartir información a través de múltiples plataformas a través de un marco unificado como la nube informática.

Estos dos flujos son similares, de modo que ambos recopilan datos para el análisis y Procesando. Sorprendentemente, estas tecnologías aún no están totalmente en acción debido a su problemas, tales como los costos de un proyecto, la eficiencia energética y la simplificación en interfaz. Lo que hace que esto sea interesante es que Internet de las Cosas (IoT) y Las Redes de Sensores Inalámbricos (WSN) son el futuro debido al rápido crecimiento de la recopilación y el análisis de datos (Mehta, M. 2015).

La red inalámbrica consta de varios nodos que pueden recibir y emitir datos de forma autónoma. Estos nodos están organizados en la red según diferentes topologías. En este artículo, presentamos una realización WSN que permite enviar datos por vía IP.

**Terminología:**

ESP8266, Espressif Systems, Sistemas de comunicación de Bajo poder, sistema en chip (SoC), Wi-Fi, inalámbrico, Tecnologías y Red de Sensores Inalámbricos (WSN).

# **ACTIVIDAD 3**

# **3.1 INVESTIGAR SOBRE EL FUNCIONAMIENTO Y DISEÑO DE CIRCUITOS CON CONEXIÓN INALÁMBRICA**

NodeMCU también se llama chip Wi-Fi ESP8266. Esta tarjeta puede actuar en modo estación o punto de acceso. En el modo de estación, desempeña el papel de sensor inalámbrico, se conecta al Wi-Fi disponible redes En el modo Punto de acceso, puede actuar como un punto de acceso para que otros dispositivos puedan conectarse a este módulo. La comunicación Wi-Fi funciona bajo el protocolo estándar IEEE 802.11 b/g/n.

Utiliza el comando AT estándar establecido puede ser conectándolo a cualquier microcontrolador. El módulo acepta comandos a través de una interfaz serial simple, entonces responde con el resultado de la operación. Además, una vez conectado y configurado puede aceptar conexiones, enviará mensajes no solicitados, mensajes cada vez que una nueva conexión o solicitud sea emitida.

El ESP8266 es un microprocesador de bajo coste con Wifi integrado fabricado por Espressif. Podemos usar el ESP8266 para conectar proyectos de electrónica y robótica con Arduino.

Los sensores inalámbricos como es el caso del módulo NodeMCU (ESP8266), juegan un papel principal para el intercambio de datos. La creación de una red WiFi es a través de la interconexión entre el cliente y el servidor a través de un punto de acceso. El prototipo realizado consta de cuatro NodeMcu (ESP8266) que son módulos de bajo costo y bajo consumo.

En realidad, el ESP8266 es mucho más que un módulo Wifi para Arduino. Es un procesador completo, con mucha más potencia que la mayoría de las modelos de Arduino. De hecho es uno de los principales “competidores” a los que se enfrenta Arduino, ya que cumple con la misma función que los chips que llevan las placas Arduino. Nos permiten leer sus entradas y activar sus salidas siguiendo las instrucciones que le hayamos programado.

Las especificaciones del chip ESP8266 son:

* Utiliza una CPU Tensilica L106 32-bit
* Voltaje de operación entre 3V y 3,6V
* Corriente de operación 80 mA
* Temperatura de operación -40ºC y 125ºC
* Soporta IPv4 y los protocolos TCP/UDP/HTTP/FTP
* No soporta HTTPS en un principio. Si que lo hace mediante software tanto en cliente como servidor TLS1.2
* Tiene 17 puertos GPIO pero solo se pueden usar 9 o 10. El GPIO16 es especial ya que está conectado al RTC (Real Time Clock)
* Pueden ser configurados con resistencia Pull-up o Pull-down.
* Soporta los principales buses de comunicación (SPI, I2C, UART)
* Voltaje de trabajo entre 3V y 3,6V.

El consumo dependerá de diferentes factores como el modo en el que esté trabajando el ESP8266, de los protocolos que estemos utilizando, de la calidad de la señal WiFi y sobre todo de si enviamos o recibimos información a través de la WiFi. Oscilan entre los 0,5 μA (microamperios) cuando el dispositivo está apagado y los 170 mA cuando transmitimos a tope de señal.

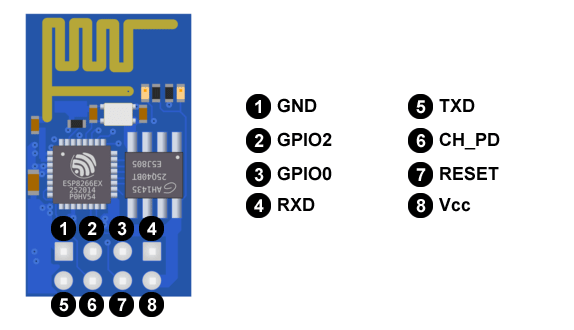


Ilustración 1 PUERTO GPIO del chip ESP8266

Existen varios modos de operación para esta clase de Chip:

* **Active mode o modo activo:** a pleno rendimiento.
* **Sleep mode o modo dormido**: solo el RTC (Real Time Clock) está activo para mantener la sincronización. Se queda en modo alerta de los posibles eventos que le hagan despertar. Mantiene en memoria los datos de conexión y así no hace falta volver a establecer la conexión con la WiFi. Consume entre 0,6 mA y 1 mA.
* **Deep sleep o modo en sueño profundo:** el RTC está encendido pero no operativo. Debe pasar por el modo dormido antes de despertar. Hay que llevar especial cuidado con los datos ya que en este estado es como si estuviera apagado y todos los datos que no estén almacenados se pierden. Consume alrededor de 20 μA.

Al igual que con Arduino, donde trabajamos con la placa o circuito integrado, con el ESP8266 ocurre exactamente lo mismo. El fabricante AI-Thinker proporciona la serie ESP con diferentes modelos para diferentes usos. A parte han ido surgiendo diferentes placas que incorporan algún módulo ESP como el NodeMCU.

Algunos módulos ESP-XX son:

* ESP-01
* ESP-05
* ESP-12
* ESP-201
* NodeMCU

# **3.2 DISEÑAR, IMPLEMENTAR Y CONFIGURAR UN CIRCUITO CON UN ARDUINO Y UN MÓDULO ESP8266, MINIPLACA WI- FI.**

Para programar el módulo Wifi 8266 debemos conectarlo al puerto USB, y para hacerlo debemos conseguir un convertidor de USB a TTL, sin embargo nuestro microcontrolador Arduino lo tiene incluido, esto nos permite encender el módulo wifi con comandos AT y utilizamos el Arduino para la comunicación USB a serial para habilitar la comunicación por comandos AT.

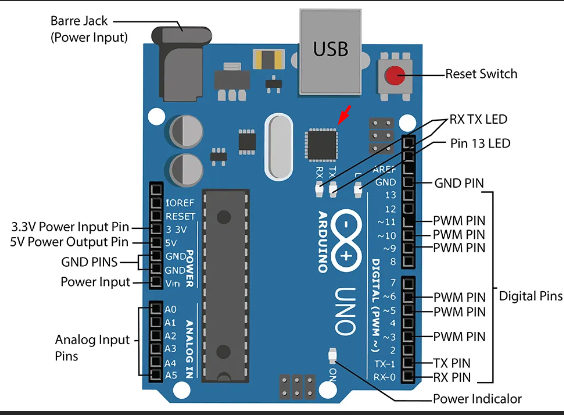


Ilustración 2 USB A serial para comunicación

Para las conexiones de los cables conectamos la recepción pin 3 (PWM PIN) del Arduino con la recepción (RXD) del módulo Wifi, porque nos estamos conectamos directamente como lo mostramos arriba al conector TTL del Arduino integrado. Lo mismo hacemos para la transmisión conectamos el pin (2D) al TXD del módulo Wifi, como se describe en la parte inferior.

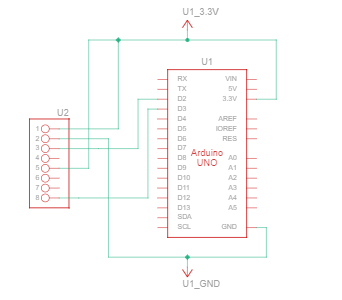
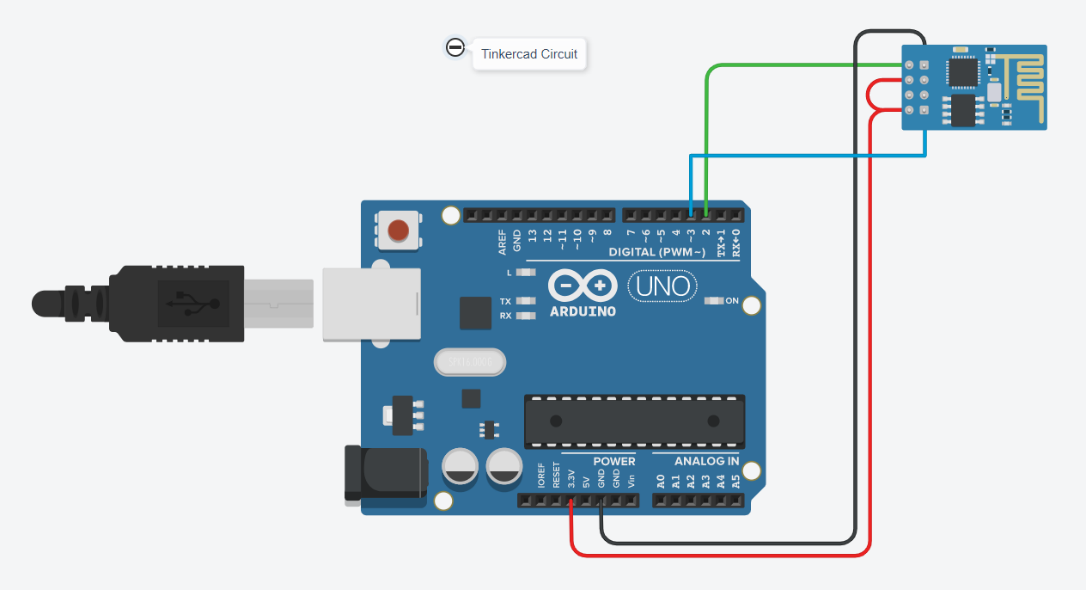


Ilustración 3 Tinkercad Circuit <https://www.tinkercad.com/things/kk6HqEVjrgd-frantic-snaget/editel?sharecode=Ba2kTDcNGeTD2C01DyY4v2gpByFmXnpb_4mzXNwPIy0>

COMPIM se utiliza para modelar interfaces COM físicas en Proteus. Funciona capturando y almacenando en búfer señales en serie que luego presenta al circuito eléctrico. Los puertos seriales de la computadora se utilizarán para conducir todos los datos seriales que se originan en la CPU o el modelo UART.

Para el desarrollo de esta entrega nos limitaremos a emplear Arduino junto con un módulo NodeMCU, uno de los varios modelos con el chip ESP8266.

Hemos tenido dificultades para simular NodeMCU /esp8266 en proteus. La razón es que no hay bibliotecas de terceros NodeMCU en proteus. Entonces, hemos decidido imitar el comportamiento de NodeMCU usando algunos de los componentes de Proteus como lo es COMPIM.

Para la simulación se conecta el NodeMCU a la PC usando un cable USB y se captura el puerto COM en Proteus usando el módulo COMPIM de Proteus, el Arduino va a leer esos datos y ejecutar las instrucciones necesarias. De esta manera podemos simular las funcionalidades de wifi y también completar el proyecto.

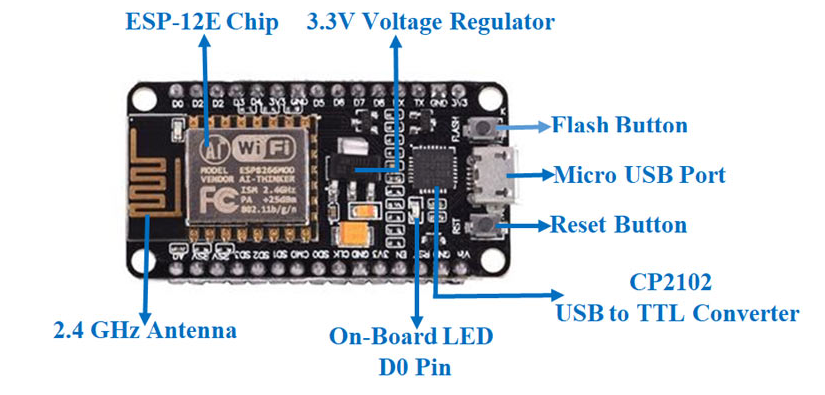


Ilustración 4 NodeMCU ESP8266

La placa de desarrollo NodeMCU ESP8266 viene con el módulo ESP-12E que contiene el chip ESP8266 con microprocesador Tensilica Xtensa LX106 RISC de 32 bits. Este microprocesador es compatible con RTOS y funciona con una frecuencia de reloj ajustable de 80 MHz a 160 MHz. NodeMCU tiene 128 KB de RAM y 4 MB de memoria Flash para almacenar datos y programas. Su alta potencia de procesamiento con funciones integradas de Wi-Fi/Bluetooth y Deep Sleep Operating lo hacen ideal para proyectos de Internet de las cosas.

NodeMCU se puede alimentar con un conector Micro USB y un pin VIN (pin de suministro externo). Es compatible con la interfaz UART, SPI e I2C.

La placa de desarrollo NodeMCU se puede programar fácilmente con Arduino IDE ya que es fácil de usar. Una vez que Arduino IDE esté instalado en la computadora, conecte la placa con la computadora usando el cable USB. Ahora abra el IDE de Arduino y elija la placa correcta seleccionando Tools>Boards>NodeMCU1.0 (Módulo ESP-12E), y elija el puerto correcto seleccionando Tools>Port.

Para nuestro desarrollo decidimos utilizar Proteus junto con Arduino IDE, descartamos Tinkercad porque actualmente el módulo Wifi no conecta con Tinkercad.

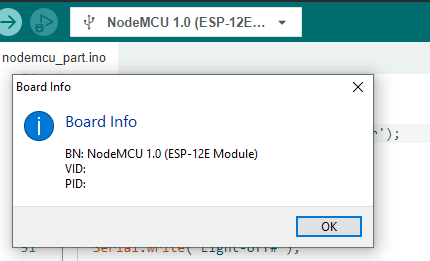


Ilustración 5 Arduino IDE configuración

También agregamos un LED porque queríamos encenderlo a través de instrucciones enviadas por IP, a través de una página WEB, pero desafortunadamente no pudimos cargar el código como queríamos y a pesar de que nos conectamos a módulo COMPIM, este no estaba aceptando los comando AT para identificar las redes inalámbricas.

Este es nuestro código Arduino, el cual necesitamos compilar a través de Arduino IDE, para luego buscar la ubicación del archivo hexadecimal y colocarlo en el archivo del programa Proteus Arduino.

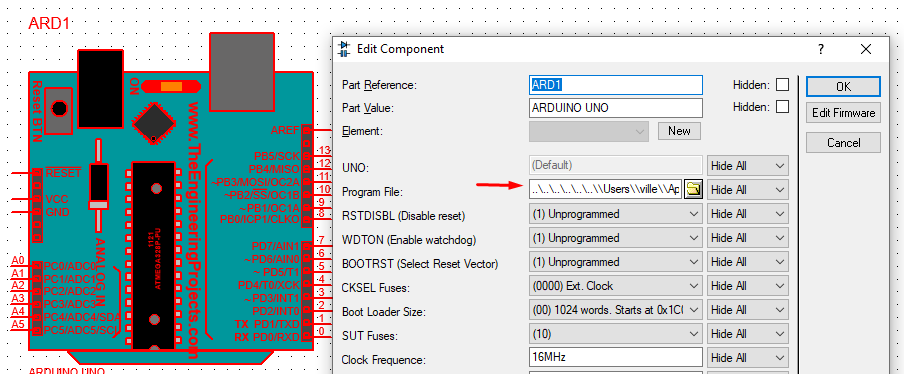
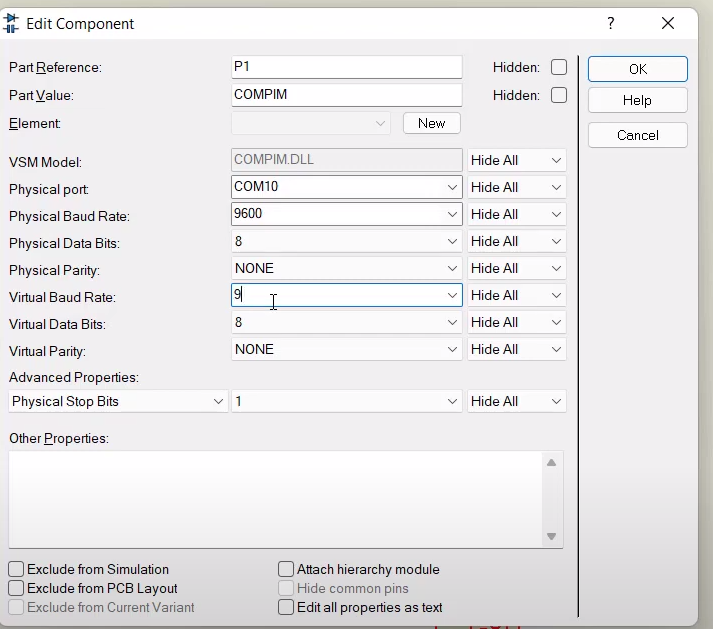


Ilustración 6Archivo Hex para el programa el componente Arduino de Proteus

Configuramos nuestro COMPIM con el mismo puerto que en nuestro Arduino IDE

Ilustración 7 Configuración COMPIM y Arduino IDE



Nuestro Desarrollo fue simple queríamos enviarle una señal a través de IP a nuestro componente COMPIM para que apagara o encendiera un LED en Proteus ubicado en el PIN 7 de nuestro Arduino Uno, también incluimos una terminal virtual para ver los comandos como se muestra abajo.

#include <SoftwareSerial.h>

int light = 7; // usamos un pin de salida al LED o light

void setup() {

  pinMode(light, OUTPUT);

  digitalWrite(light, LOW); //Normalmente Apagado

  Serial.begin(9600);

}

void loop() {

  if (Serial.available() > 0) {

    String str = Serial.readStringUntil('#');

    if (str == "Light-on") {

      digitalWrite(light, HIGH);

      Serial.println("Light is on");

    }

    else if (str == "Light-off") {

      digitalWrite(light, LOW);

      Serial.println("Light is off");

    }

  }

}



Ilustración 8 Módulo NodeMCU1.0 Adquirido

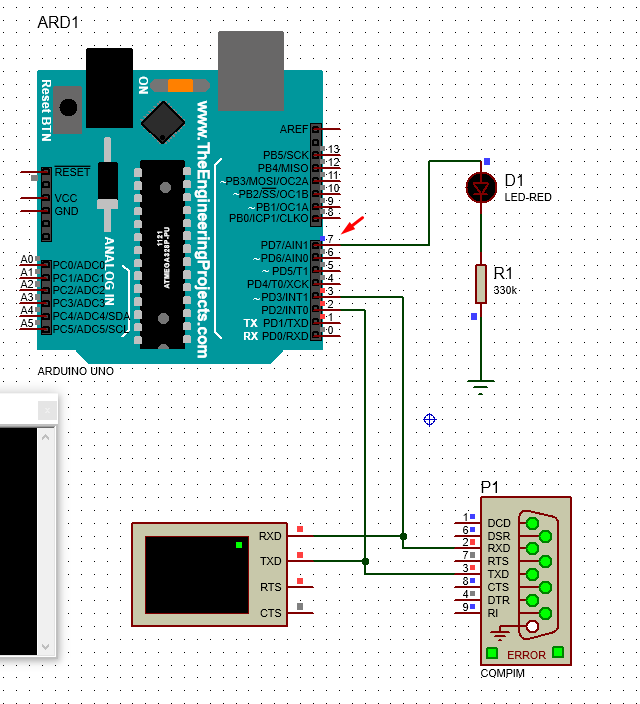


Ilustración 9 COMPIM Encendido y pin 7 de nuestro Arduino conectado al LED

Como lo comentamos anteriormente por alguna razón el componente recibe los comandos pero no nos responde, como se aprecia en la parte inferior.

Graphical user interface

Description automatically generated

Ilustración 10 Comandos AT no responden

Como se aprecia abajo esta era la idea, el LED siempre apagado y cuando se seleccionara “LED ON”, el circuito tendría que entender que el string “lights on” esta siendo ejecutado y la luz debería encenderse, y en caso contrario LED OFF es el string lights off para apagar la luz. Todo esto iba a ser por comunicación IP.

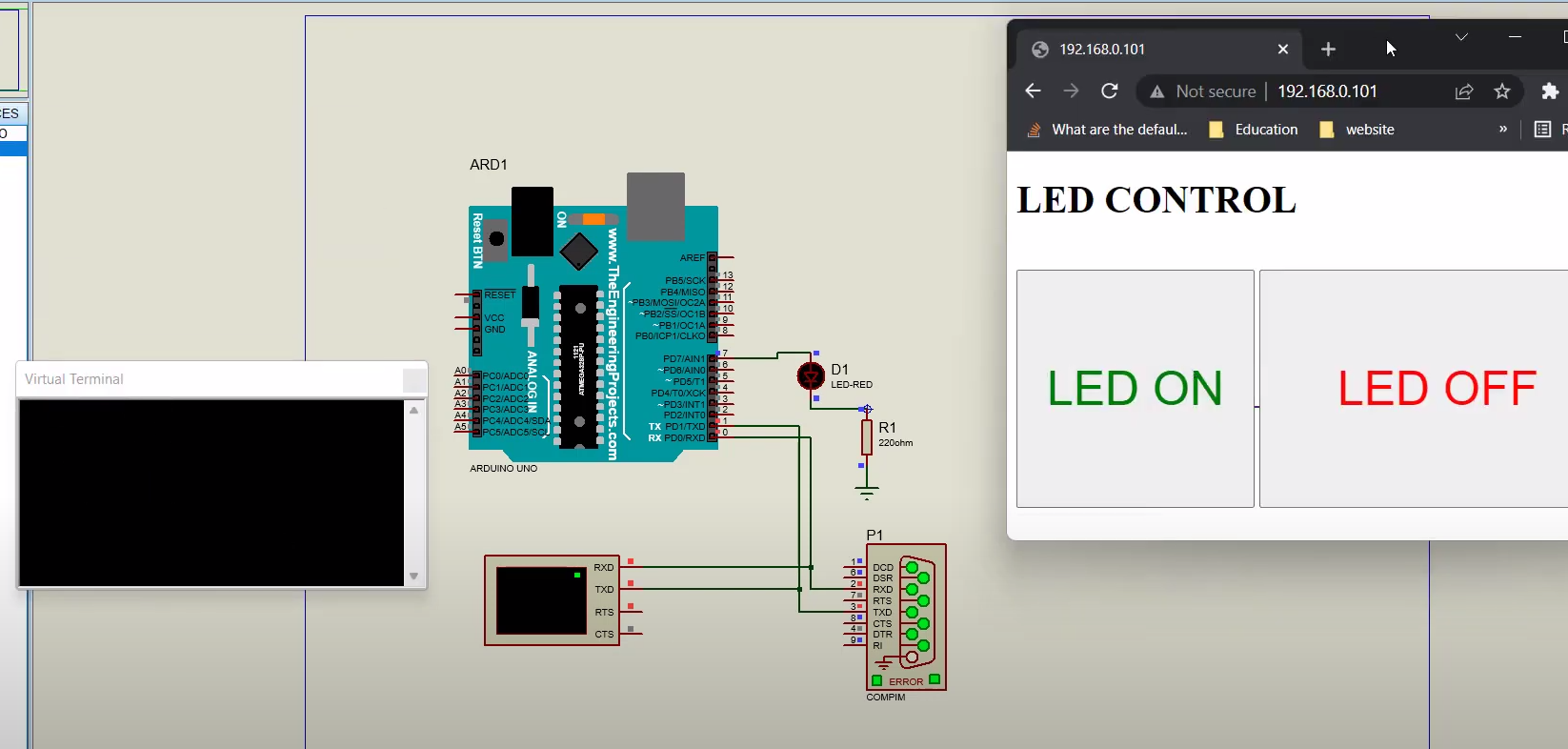


Figure 10 Programa Ejecutándose

# **3.3 SE PUEDEN ENVIAR DATOS POR VÍA IP.**

De acuerdo con los diferentes tutoriales que hemos visto si es posible hacerlo desafortunadamente, a; principio estábamos teniendo muchos errores.

<https://www.youtube.com/watch?v=7gXcTBHLCRc>

<https://www.youtube.com/watch?v=ex468UWer3E&t=5009s>

Continuamos reibiendo el siguiente error cuando intentamos “Upload” nuestro desarrollo al componente COMPIM, y esto es porque no tenemos los componente físicos para conectarlo a nuestro PC ni ninguno componente de Proteus funciona para este propósito, de acuerdo con lo que hemos leído, también hemos intentado con diferentes boards y modificando el puerto para que resista diferentes velocidades de subida, pero el mismo resultado siempre aparece.

File "C:/Users/ville/AppData/Local/Arduino15/packages/esp8266/hardware/esp8266/3.0.2/tools/esptool\esptool.py", line 529, in connect raise FatalError('Failed to connect to %s: %s' % (self.CHIP\_NAME, last\_error)) esptool.FatalError: Failed to connect to ESP8266: Timed out waiting for packet header Failed uploading: uploading error: exit status 1

raise SerialException("could not open port {!r}: {!r}".format(self.portstr, ctypes.WinError()))

Hemos decidido adquirir el módulo NodeMCU1.0 (Módulo ESP-12E) y cuando lo conectamos inmediatamente habilito el COM4 y nos permitió hacer el Upload del desarrollo en la placa Wifi.

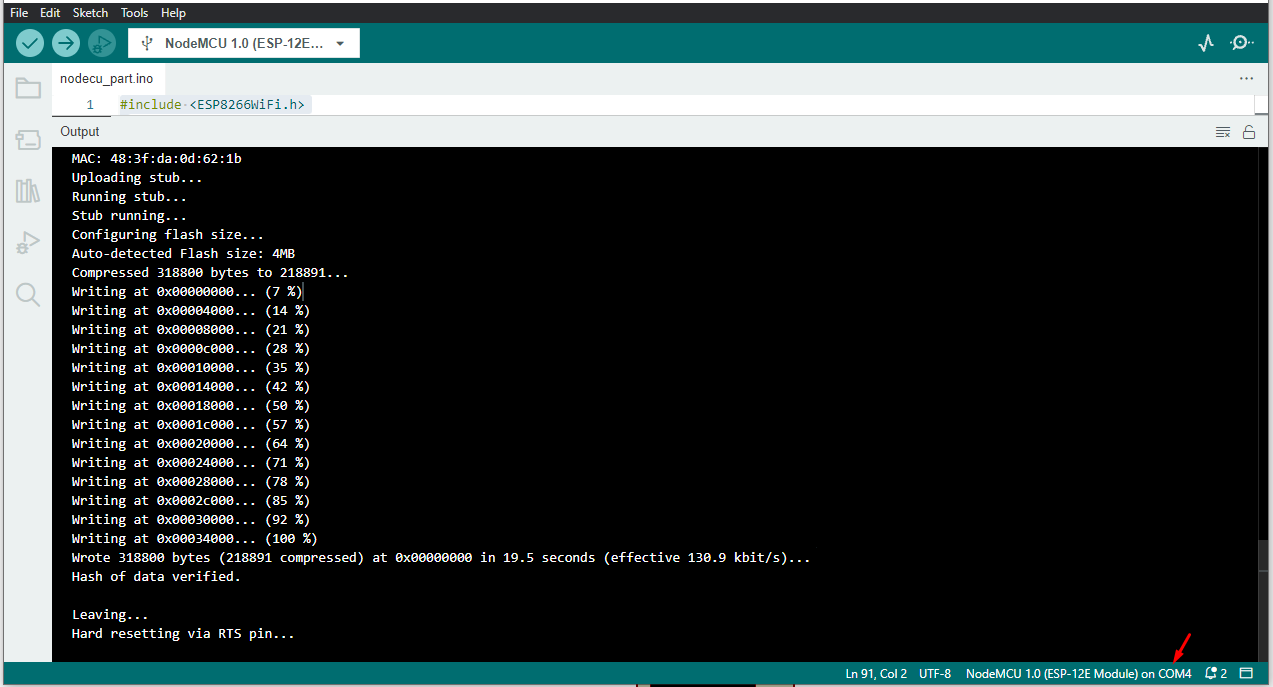


Ilustración 111 Upload en Módulo NodeMCU1.0

Al resetear el módulo NodeMCU inmediatamente obtenemos respuesta:

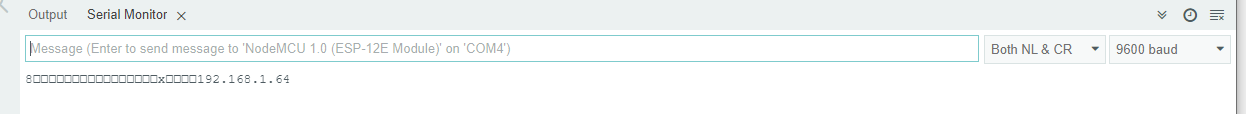
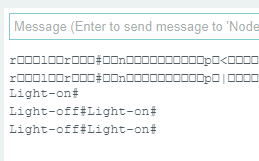


Ilustración 122 Serial Port terminal

Este es nuestro Desarrollo para la placa NodeMCU.

#include <ESP8266WiFi.h>

const char\* ssid = "SSID";   //enter your wi-fi name

const char\* password = "Password";    //enter the wifi password

unsigned char status\_led=0;

WiFiServer server(80);

void setup() {

  Serial.begin(9600);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

  }

  server.begin();

  Serial.println(WiFi.localIP());

}

void loop() {

  WiFiClient client = server.available();

  if (!client) {

    return;

  }

  while (! client.available())

  {

    delay (1);

  }

  String req = client.readStringUntil('\r');

  client.flush();

  if (req.indexOf("/ledoff") != -1)  {

    status\_led=0;

    Serial.write("Light-off#");

  }

  else if(req.indexOf("/ledon") != -1)

  {

    status\_led=1;

    Serial.println("Light-on#");

  }

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println("Connection: close");

client.println("");

client.println("<!DOCTYPE HTML>");

client.println("<HTML>");

client.println("<H1> LED CONTROL </H1>");

client.println("<br />");

 client.println("<a href=\"/ledon\"\"> <button style='FONT-SIZE: 40px; color: green; HEIGHT: 200px; align: center; WIDTH: 200px; 126px; Z-INDEX: 0; TOP: 200px;'> LED ON </button> </a>");

 client.println("<a href=\"/ledoff\"\"> <button style='FONT-SIZE: 40px; color: red; HEIGHT: 200px; align: center; HEIGHT: 200px; WIDTH: 300px; 126px; Z-INDEX: 0; TOP: 200px;'> LED OFF </button> </a>");

 client.println("</html>");

  delay(1);

}

Después de cargar nuestro código en el módulo NodeMCu, logramos que funcionara nuestro proyecto y hemos enviado comandos a través de IP a nuestro circuito para el encendido y apagado de nuestro LED, pero esto lo hicimos haciendo una conexión normal del puerto TXD a TXD y RXD a RXD del módulo Wifi y la placa Arduino, como se aprecia a continuación.

Graphical user interface

Description automatically generated

Ilustración 13 LED apagado y COMPIM corriendo

Graphical user interface

Description automatically generated with low confidence

Ilustración 14 Comando de encendido

Schematic

Description automatically generated

Ilustración 15 Comando de apagado

**Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedAhora para propósitos de este ejercicio conectamos la recepción pin 3 (PWM PIN) del Arduino con la recepción (RXD) del módulo Wifi, porque nos estamos conectamos directamente al conector TTL del Arduino integrado. Lo mismo hacemos para la transmisión conectamos el pin (2D) al TXD del módulo Wifi, como se describe en la parte inferior y se logra él envió de datos por protocolo IP como se aprecia abajo en la terminal virtual de Proteus**.

Ilustración 16 PIN 2 y 3 Recibiendo datos

Ilustración 17 PIN 2 y 3 de Arduino recibiendo datos

# **CONCLUSIONES**

Hemos llegado a la conclusión de que nuestro circuito funciona con éxito después de haber adquirido el módulo Wifi NodeMcu. El envío y recepción de datos por medio del módulo Wifi decimos que si es posible con el protocolos IP porque lo hemos visto y ya lo hemos logrado.

Con este prototipo se pretendía realizar una interconexión con la placa NodeMcu a través del puerto 80 del protocolo HTTP. El cliente HTTP envía una solicitud HTTP al servidor que activa el módulo COMPIM, permitiendo el encendido y apagado de un LED a través del protocolo IP.

# **BIBLIOGRAFIA**

* Mehta, M. (2015). ESP8266: A Breakthrough in wireless sensor networks and internet of things. International Journal of Electronics and Communication Engineering & Technology, 6(8), 7-11.
* Online reference URL: <https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/>