Conexión WiFi

1. Objetivo de la práctica

Establecer la comunicación mediante WiFi que permita controlar un robot móvil inalámbricamente desde la computadora. Programar una tarjeta de desarrollo con el chip ESP8266 para recibir datos mediante la red y controlar los actuadores del robot móvil.

2. Metas

Para la realización de la práctica se deben cumplir las siguientes metas:

- Generar una red local para la comunicación.
- Conectar la tarjeta de desarrollo con ESP8266 a la red.
- Programar en Simulink el envío de datos.
- Programar el ESP8266 para recibir los datos y controlar los actuadores.

3. Antecedentes

En el área de la robótica móvil resulta de gran utilidad el uso de tecnologías de comunicación inalámbricas, ya que los robots se encuentran en movimiento dentro de un espacio y el uso de cables para la conexión con algún sistema o sensor exterior podría entorpecer el funcionamiento del robot móvil. Afortunadamente, existen diversas tecnologías que permiten el intercambio de información de manera inalámbrica, entre las más utilizadas se encuentran las siguientes:

- Bluetooth
- ZigBee
- RFID
- Ultra Wide Band (UWB)
- Wi-Fi
- WiMAX

La comunicación Wi-Fi es un estándar internacional para comunicaciones inalámbricas denominado IEEE 802.11. Este estándar se ha ido mejorando y modificando con el tiempo, ajustándose a los requerimientos y recursos tecnológicos, por lo que actualmente existen muchas versiones indicadas por una o más letras después del número del estándar, por ejemplo, IEEE 802.11a.

Hoy en día, la mayoría de los dispositivos móviles como celulares y laptops cuentan con este tipo de tecnología para interconectarse en redes y también al internet. Además, existen diversos módulos con comunicación Wi-Fi que son económicos y relativamente fáciles de utilizar, por lo que resulta una buena alternativa para la implementación en aplicaciones de robótica móvil.

A continuación, se presenta un enfoque práctico para la implementación de esta tecnología con la finalidad de lograr la comunicación entre una computadora y un módulo con Wi-Fi (ESP8266).

4. Conocimientos previos

Los conocimientos necesarios para la realización de la práctica:

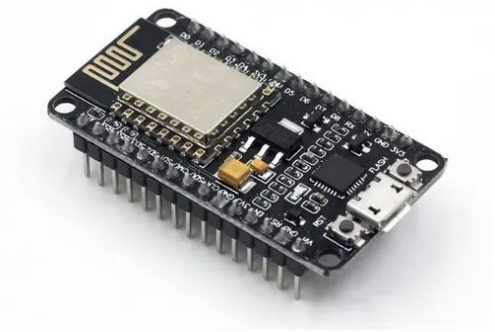
- Conocimientos básicos de computación y programación.
- Conocimientos básicos de Matlab.
- Conocimientos básicos de Simulink.

5. Materiales y Equipo

Para la realización de la práctica es necesario contar con lo siguiente:

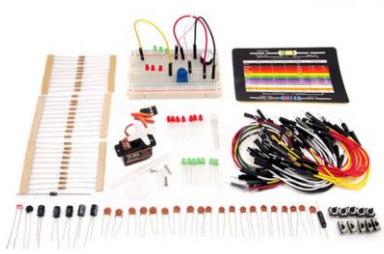
- Una computadora con Arduino IDE, Matlab y Simulink. (1)
- Tarjeta de desarrollo NodeMCU con Módulo WiFi ESP 8266. (2)
- Cable de conexión entre tarjeta y computadora. (3)
- Componentes electrónicos básicos para prueba de funcionamiento: protoboard, cables, leds, resistencias, fuente de alimentación, etc. (4)

(1)



(2)

(3)



(4)

6. Preparativos previos y recomendaciones

- Es recomendable establecer una carpeta específica en la computadora para el desarrollo de la práctica.

7. Desarrollo de la práctica

a) Descarga e instalación del Arduino IDE

Durante esta práctica se programará el módulo ESP8266 haciendo uso de la interfaz de programación de Arduino. Si bien existen diferentes maneras de realizar la programación de este dispositivo, la interfaz Arduino IDE resulta muy práctica y fácil de utilizar. Además, existe un buen soporte y documentación para su uso con el módulo ESP8266, por lo que resulta una buena opción para la implementación.

Antes que nada, es necesario descargar e instalar el Arduino IDE, el cual puede ser descargado directamente desde la página oficial de Arduino en la sección de software: https://www.arduino.cc/en/software.

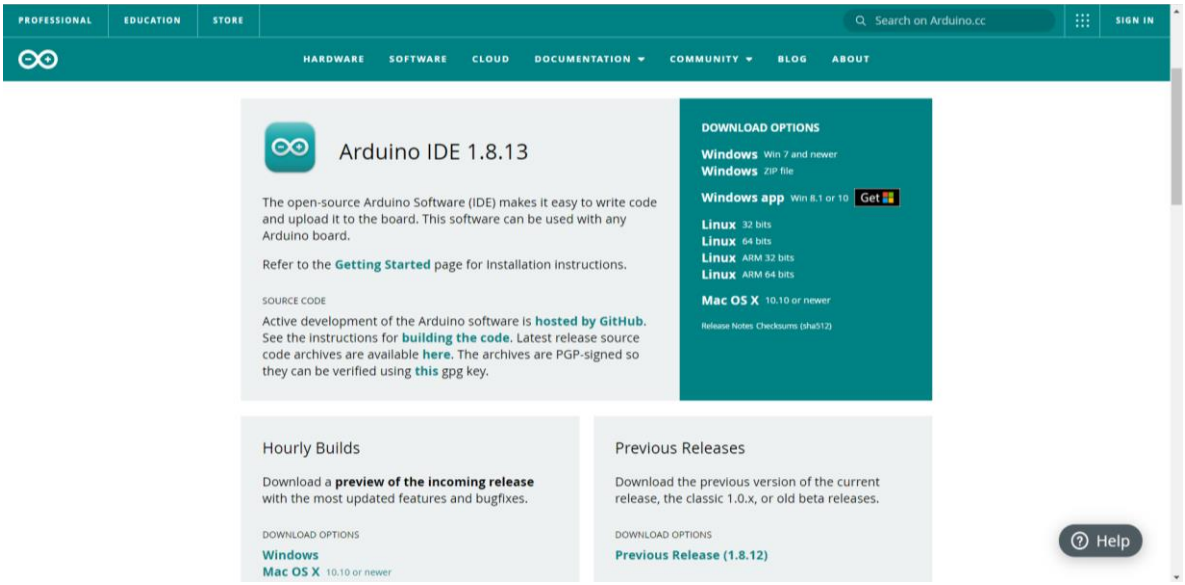


Figura 1. Página de descargas de Arduino.

Tras ingresar a la página se deberá seleccionar y descargar la versión correspondiente al sistema operativo que se está utilizando y se deberá seguir el proceso de instalación en el equipo. Una vez concluida la instalación del Arduino IDE se deberá abrir la aplicación y realizar la configuración que se presenta a continuación para poder utilizar esta interfaz de programación con la placa ESP8266.

b) Configuración de ESP8266 en Arduino IDE

Para poder programar la tarjeta de desarrollo con ESP8266 utilizando el Arduino IDE es necesario configurar la aplicación para que reconozca esta tarjeta. Para ello se debe abrir la ventana de preferencias desde la barra de menú → Archivo → Preferencias. (Figura 2)

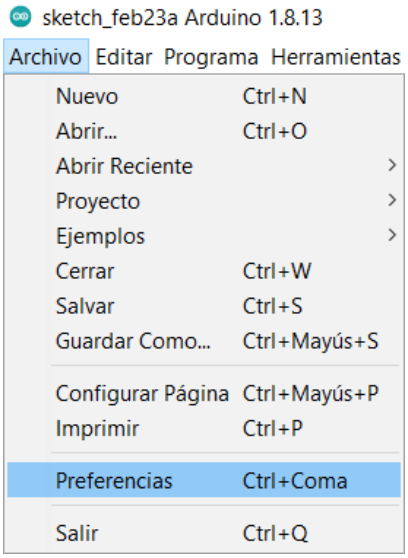


Figura 2. Acceso a Preferencias dentro de Arduino IDE.

Al dar clic en esta opción se abrirá una nueva ventana en dónde pueden modificarse distintas propiedades de la aplicación. En la sección inferior de la ventana (Figura 3) aparece un recuadro con la leyenda “Gestor de URLs Adicionales de Tarjetas”, seguido de un recuadro en el que se puede introducir texto. Esta área se puede utilizar para hacer referencia a URLs de soporte para tarjetas de terceros que no son propias de Arduino, pero que ya tienen desarrollado el soporte para poder utilizarlas. En este caso, deberemos introducir la dirección correspondiente a la tarjeta ESP8266, esta es:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

Tras haber realizado esta modificación se deberá dar clic en el botón “ok” para que queden guardados los cambios.

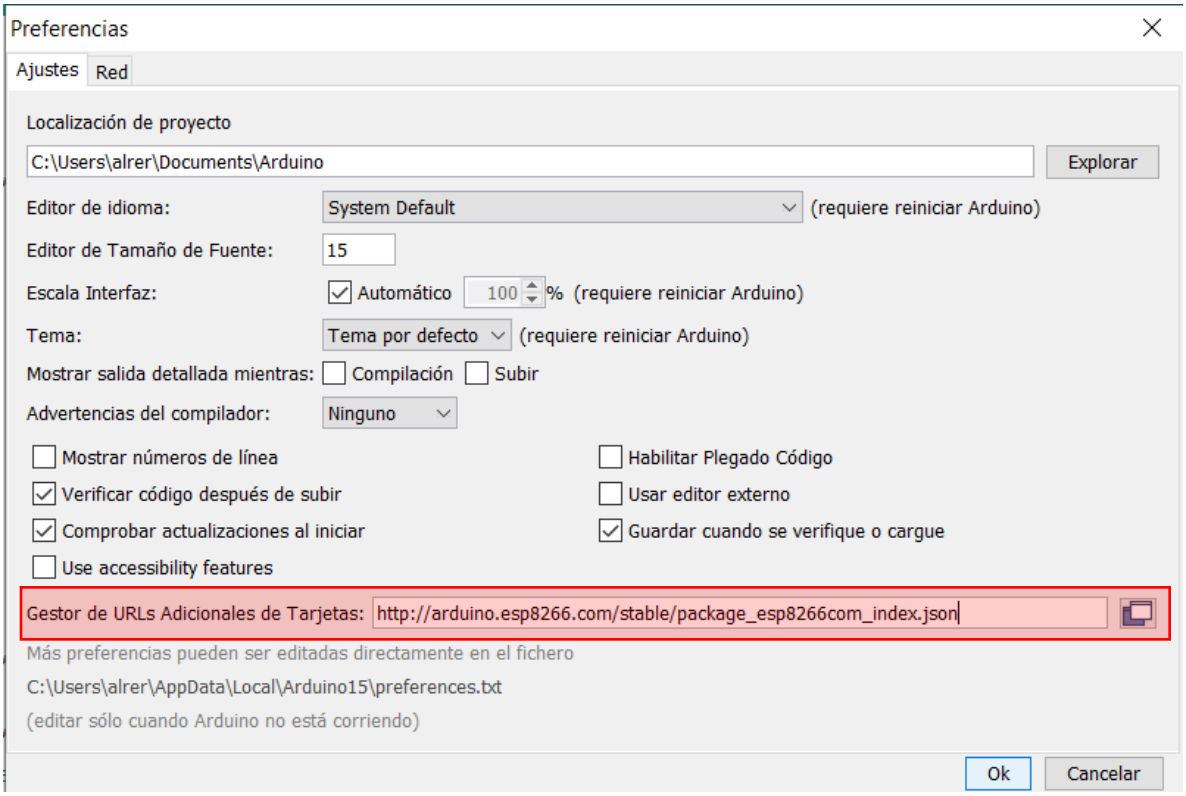


Figura 3. Agregar URL adicional de Tarjeta ESP8266.

Después de introducir esta dirección la aplicación será capaz de reconocer dónde encontrar los recursos para el soporte de dicha tarjeta, por lo que se puede realizar su instalación. Para ello, es necesario abrir el gestor de tarjetas, al cual se puede acceder de la barra de menú → Herramientas → Placa → Gestor de Tarjetas. (Figura 4)

Con ello deberá abrirse una nueva ventana en la que se puede instalar la placa correspondiente, por lo que bastará con buscar el paquete correspondiente a la placa ESP8266 y dar clic en el botón para instalar. (Figura 5)

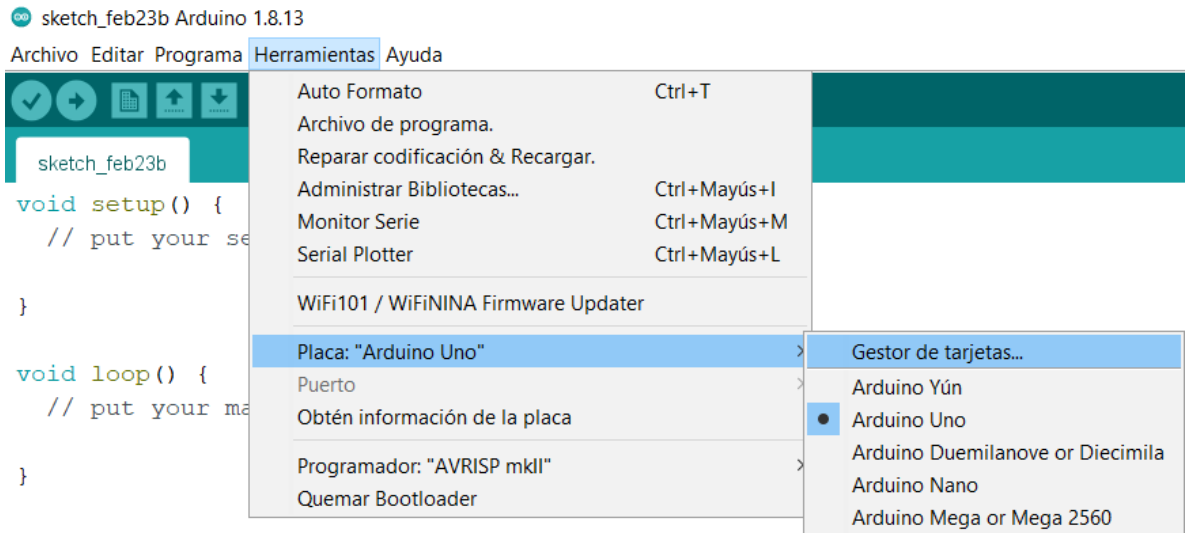


Figura 4. Acceso a la ventana del gestor de tarjetas.

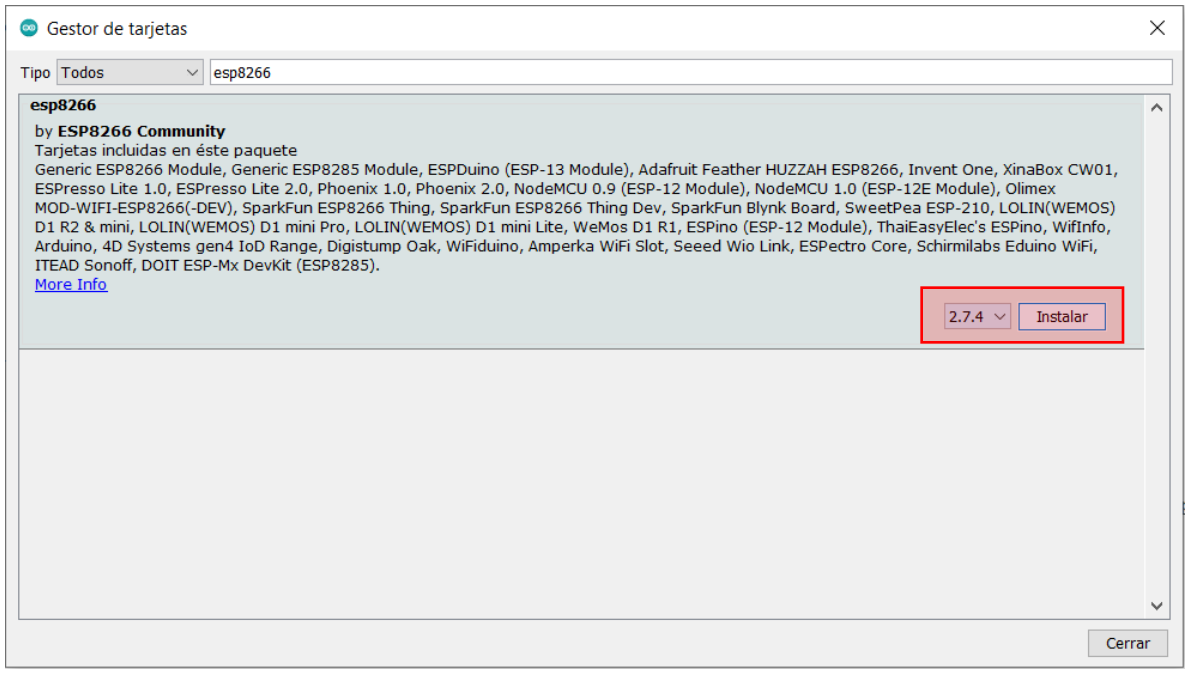


Figura 5. Instalación de placa ESP8266 desde el gestor de tarjetas.

Finalmente, se deberá verificar que la instalación haya sido completada con éxito, si es así deberá aparecer la leyenda “INSTALLED” al lado del paquete para la tarjeta ESP8266. (Figura 6)

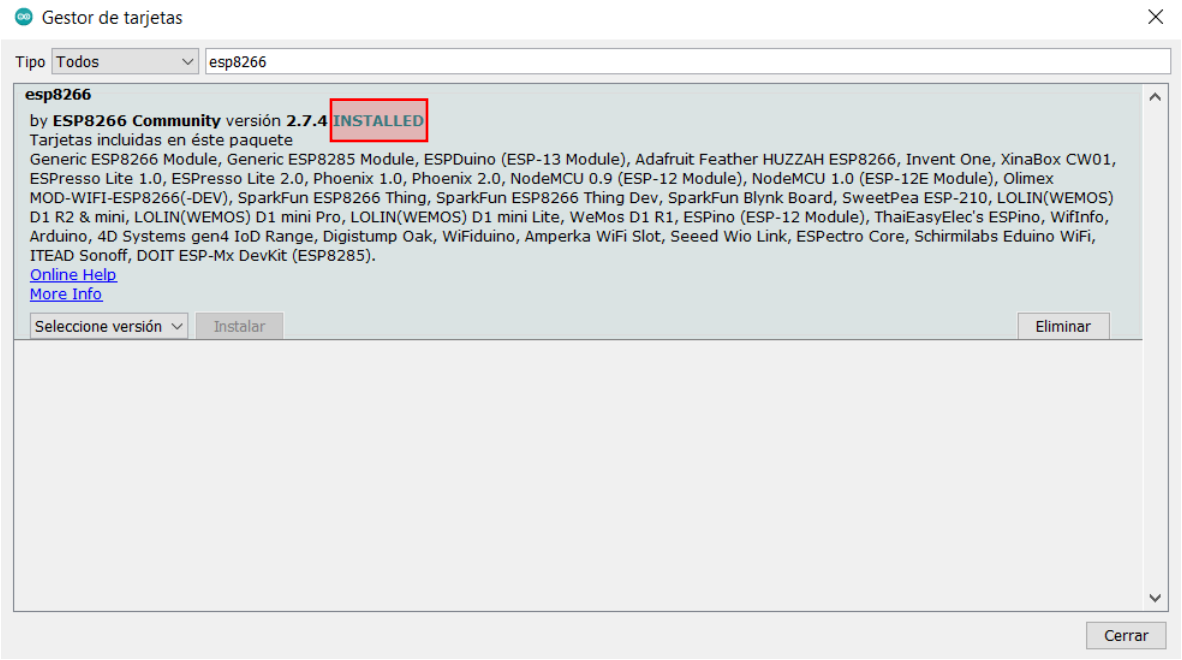


Figura 6. Verificación de instalación de placa ESP8266.

Finalizados los pasos anteriores se deberá reiniciar la aplicación de Arduino IDE. Con ello queda ya disponible el uso de la aplicación para programar tarjetas que utilizan el módulo de conexión Wi-Fi ESP8266. Por último, para poder iniciar a programar se deberá seleccionar la placa que se está utilizando desde la opción en la barra de menú → Herramientas → Placa → ESP8266 Boards. (Figura 7)

Para la elaboración de esta práctica se está utilizando la placa de desarrollo NodeMCU 1.0, por lo que fue esta la que se seleccionó de la lista de tarjetas, sin embargo, si se está utilizando alguna otra tarjeta que utilice el módulo ESP8266 se deberá seleccionar la tarjeta correspondiente de la lista.

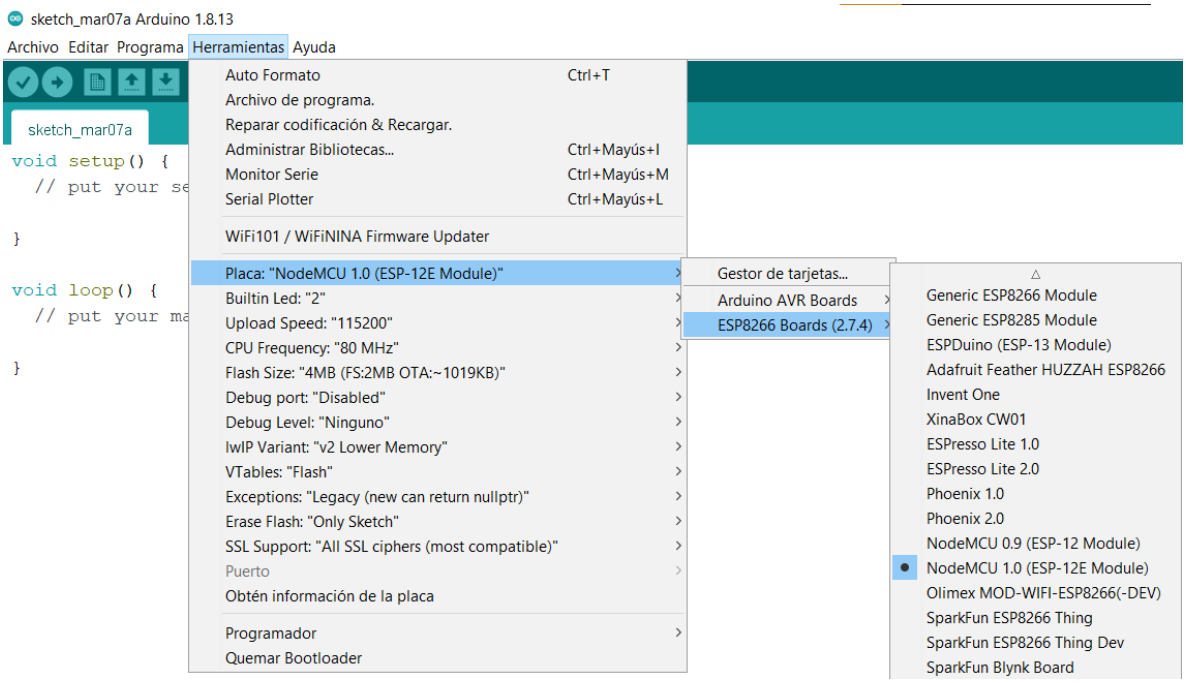


Figura 7. Selección de la placa.

c) Generación de una red local para la conexión

Antes de proceder con la programación de la tarjeta ESP8266 se generará la red local que se utilizará para la conexión entre la tarjeta y la computadora, además, en el caso de que se quieran utilizar varios dispositivos o módulos con conexión WiFi podrán conectarse todos ellos a esta misma red para lograr la comunicación.

Una forma muy sencilla para generar una red local es utilizando la propia herramienta que ofrece el sistema operativo Windows 10 para generar una "Zona con Cobertura". Esta opción permite generar una red para conectarse con varios dispositivos, e incluso permite compartir la conexión a internet a través de la computadora.

Para activar esta función deberá accederse a la ventana de Configuración → Red e Internet → Zona con cobertura inalámbrica móvil.

Desde esta ventana es posible activar o desactivar la red dando clic en el botón de la parte superior. Además, se puede configurar el nombre de la red y la contraseña que se utilizara para conectarse dando clic en el botón “Editar” localizado en la parte inferior. En este ejemplo se utilizará el nombre de red “MiRed” y la contraseña “123456789”, estos datos deberán recordarse posteriormente para programar la tarjeta ESP8266 para que se conecte a esta red.

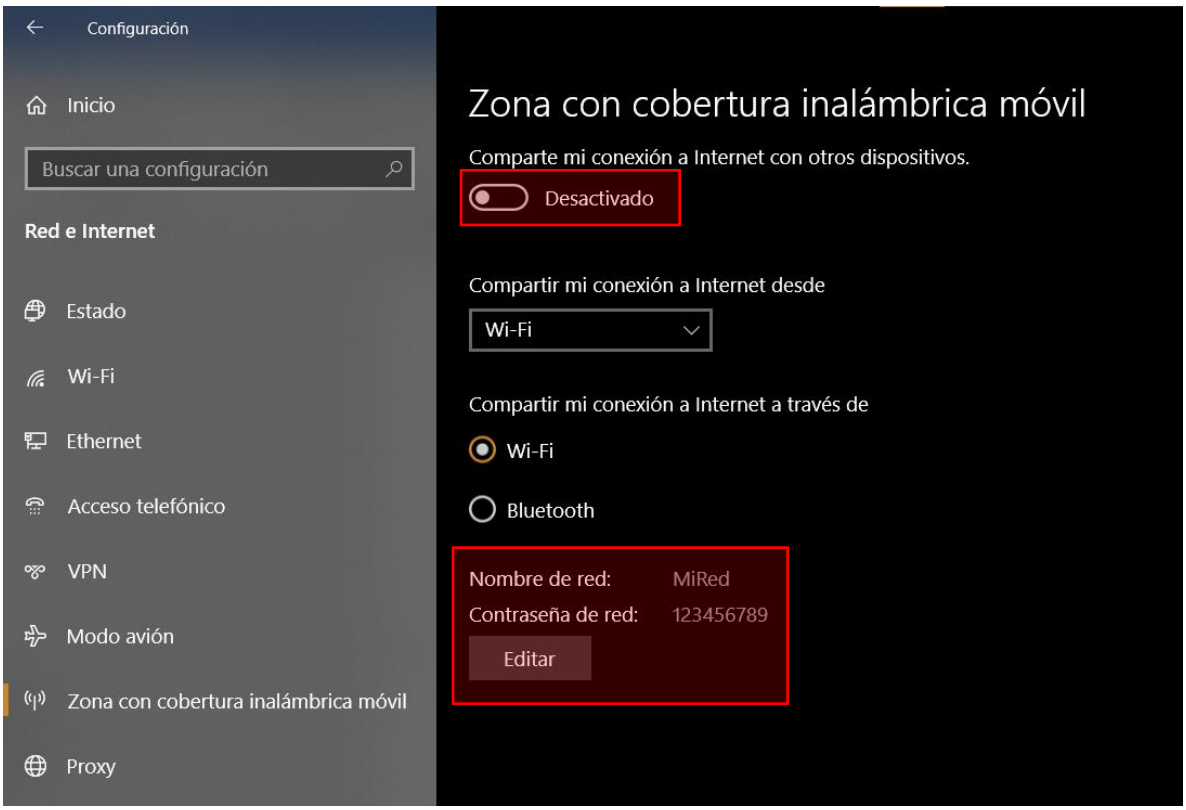


Figura 8. Configuración de Zona con Cobertura para la Red Local.

d) Programación de la tarjeta ESP8266

Para la programación de la tarjeta ESP8266 se considerará la conexión a la red local que configuramos anteriormente, la configuración necesaria para la recepción de datos mediante el protocolo UDP (que fue introducido en la práctica 1), y el uso de los GPIO de la placa para el control de elementos electrónicos como por ejemplo LEDs, motores, botones, etc.

Se tomará como base para el programa el ejemplo que puede encontrarse en la sección de ejemplos → Ejemplos para NodeMCU 1.0 →ESP8266WiFi → Udp.

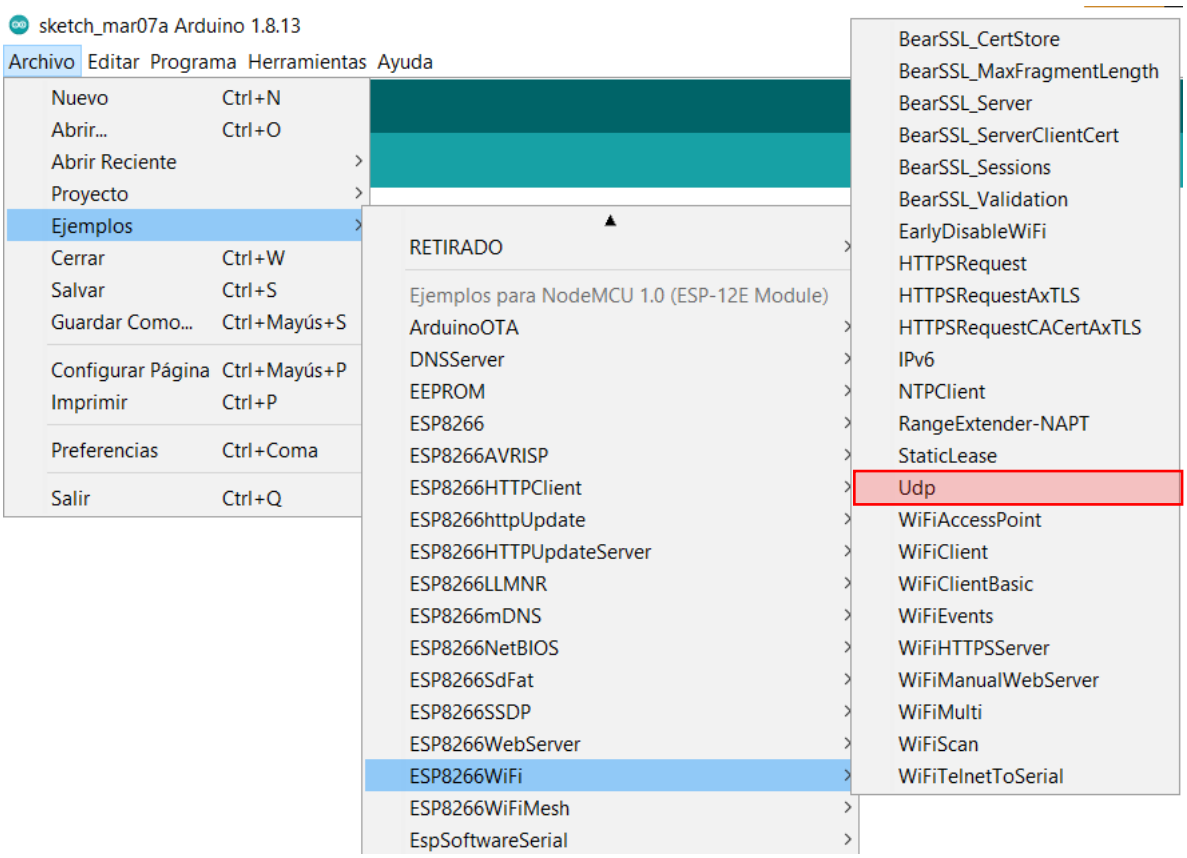


Figura 9. Ejemplo base para programación de ESP8266.

Al abrir este ejemplo se puede observar el código básico para lograr la conexión de la tarjeta a la red, así como el envío y recepción de datos mediante el protocolo UDP.

Lo primero que aparece en el código es la inclusión de las librerías necesarias para el funcionamiento del programa y la definición de variables que serán utilizadas por el programa para la conexión a la red o para la recepción o envío de datos.

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

#ifndef STASSID
#define STASSID "MiRed"
#define STAPSK  "123456789"
#endif

unsigned int localPort = 58625;      // local port to listen on

// buffers for receiving and sending data
char packetBuffer[UDP_TX_PACKET_MAX_SIZE + 1]; //buffer to hold incoming packet,
char  ReplyBuffer[] = "acknowledged\r\n";       // a string to send back

WiFiUDP Udp;
```

Las modificaciones importantes que deben realizarse en esta sección del código son el nombre de la red (STASSID), la contraseña de la red (STAPSK) y el puerto que se va a utilizar para la comunicación UDP (localPort).

La función setup() del programa es utilizada para realizar las configuraciones necesarias para su funcionamiento, esta función se ejecutará una vez al iniciar la tarjeta. Dentro de esta función se observa que se inicia la comunicación serial para poder mostrar información en el monitor serial, se inicia también la comunicación WiFi y se realiza la conexión con la red que se especificó. Mediante la comunicación serial se imprimirán “.” (puntos) mientras se esté intentando realizar la conexión y una vez que se haya realizado con éxito se imprimirán la dirección IP y el puerto que se está utilizando.

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(STASSID, STAPSK);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(500);
  }
  Serial.print("Connected! IP address: ");
  Serial.println(WiFi.localIP());
  Serial.printf("UDP server on port %d\n", localPort);
  Udp.begin(localPort);
}
```

La última sección del código corresponde a la función loop(), esta función es donde se programa lo que va a estar haciendo la tarjeta de manera constante mientras se ejecuta el programa. Una vez que se ha terminado de ejecutar la función setup(), se ejecuta de manera iterativa la función loop().

```
void loop() {
  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if (packetSize) {
    Serial.printf("Received packet of size %d from %s:%d\n    (to %s:%d, free heap = %d B)\n",
                  packetSize,
                  Udp.remoteIP().toString().c_str(), Udp.remotePort(),
                  Udp.destinationIP().toString().c_str(), Udp.localPort(),
                  ESP.getFreeHeap());

    // read the packet into packetBufffer
    int n = Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
    packetBuffer[n] = 0;
    Serial.println("Contents:");
    Serial.println(packetBuffer);

    // send a reply, to the IP address and port that sent us the packet we received
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    Udp.write(ReplyBuffer);
    Udp.endPacket();
  }

}
```

Tras haber generado la red que se va a utilizar y realizar las modificaciones correspondientes al programa para que se conecte a dicha red, se puede cargar el programa a la tarjeta para verificar su funcionamiento. Es importante recordar que se debe seleccionar el tipo de tarjeta que se está utilizando tal como se explicó en los pasos anteriores.

Además, se deberá seleccionar el puerto COM de la computadora al que se conectó la tarjeta, para esto es necesario conectar la tarjeta a la computadora mediante el cable USB. Por lo general, la computadora identificará el dispositivo y le asignará un puerto COM, la primera vez que se conecta la tarjeta a la computadora puede tomar un poco de tiempo en lo que se reconoce y se instalan los controladores necesarios para su funcionamiento. Ya con el dispositivo conectado se deberá seleccionar el puerto desde la Barra de Menú → Herramientas → Puerto → COM# (Figura 10)

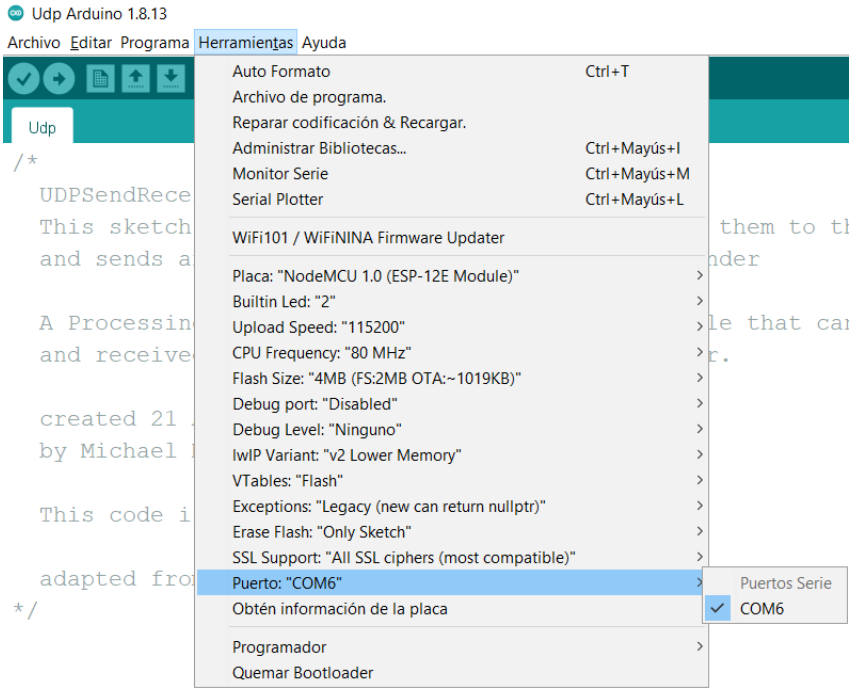


Figura 10. Selección del puerto COM.

Si no se tienen más dispositivos conectados, normalmente será el único puerto disponible para seleccionar. En caso de duda o que aparezcan más puertos disponibles y se desconozca cuál es el que corresponde a la tarjeta, la forma más sencilla de identificar el puerto es desconectando la tarjeta y revisando qué puerto es el que se agrega al conectarla de nuevo.

Una vez hecho esto se deberá cargar el programa haciendo clic en el botón para subir el programa (Figura 11). Al dar clic en este botón inicialmente se compilará el programa y a continuación se cargará en el dispositivo seleccionado, deberá esperarse a que termine este proceso. Una vez concluido se indicará en la parte inferior que el programa ya fue subido (Figura 12). En caso de presentarse algún mensaje de error, estos deberán resolverse y posteriormente volver a compilar y cargar el programa en la tarjeta.

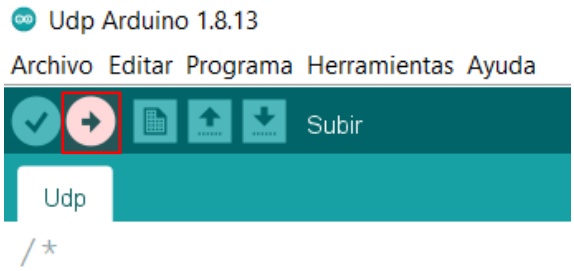


Figura 11. Botón para subir el programa a la tarjeta.

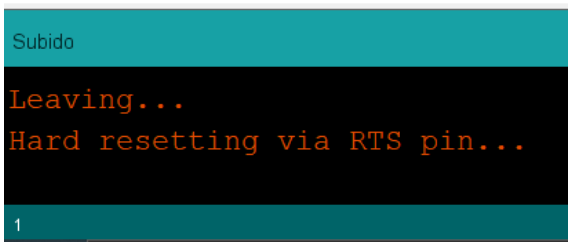


Figura 12. Indicación de que el programa terminó de subirse.

Finalizado este proceso ya estará cargado el programa en la tarjeta y se ejecutará toda vez que sea alimentada de manera correcta. Tras terminar de subir el programa se recomienda reiniciar la tarjeta ya sea cortando la alimentación (desconectándola de la computadora), o bien utilizando el botón de “reset” (RST) de la propia tarjeta.

Volviendo a conectar la tarjeta se puede verificar el funcionamiento del programa revisando el monitor serial del Arduino IDE, en este se muestran los mensajes que manda la tarjeta sobre su estado de conexión. Para visualizar esto se debe abrir el monitor serial dando clic en el botón que se ubica en la esquina superior derecha de la aplicación (Figura 13). También es importante seleccionar correctamente la velocidad de comunicación, de tal manera que coincida con la velocidad que está utilizando la tarjeta para el envío de los datos, de lo contrario los símbolos mostrados no corresponderán a los mensajes enviados por la tarjeta. Esta velocidad de comunicación se definió en el programa cargado en la tarjeta dentro de la función *setup()*, con la instrucción *Serial.begin(115200);* por lo que en este ejemplo deberá seleccionarse la velocidad como 115200 baudio en la parte inferior del monitor serial (Figura 14).

Figura 13. Botón para abrir el monitor serial.

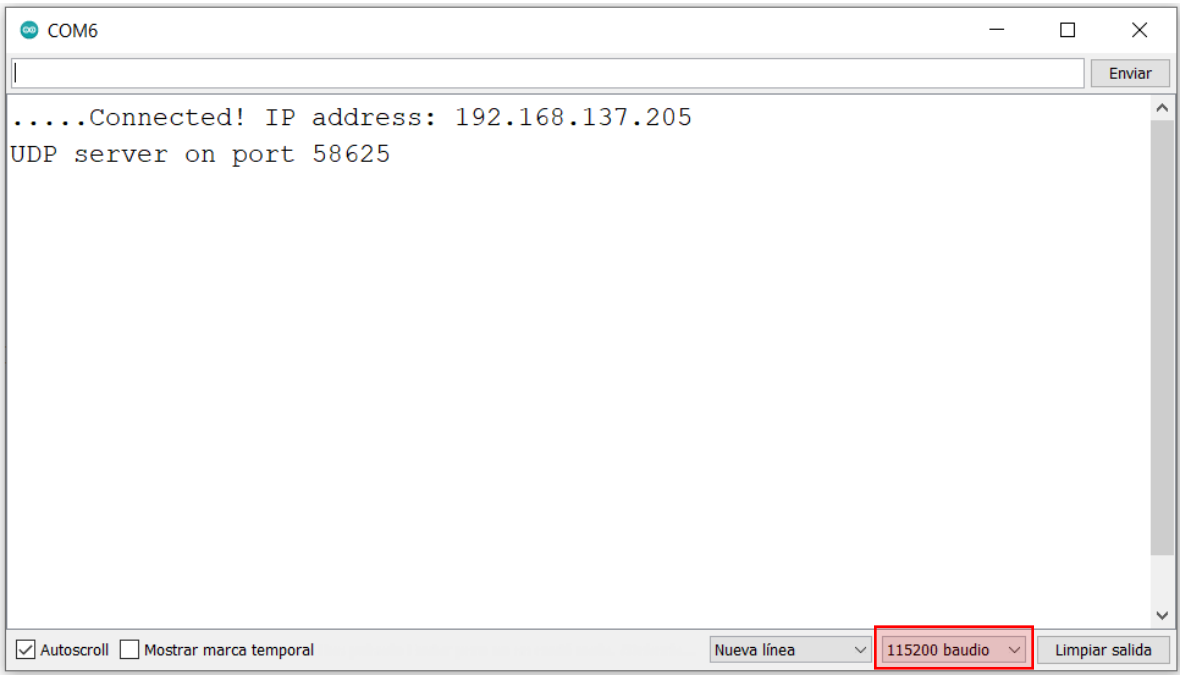


Figura 14. Monitor serial.

Como se mencionó anteriormente, lo que hace este programa es imprimir en el monitor serial la información sobre la conexión del dispositivo a la red inalámbrica. En la Figura 14 se muestra como se indica que la conexión fue realizada correctamente y se muestra también la dirección IP que fue asignada al dispositivo y el puerto que se está utilizando para la comunicación mediante el protocolo UDP. Esta información será importante recordarla para poder establecer la comunicación y enviar mensajes a dicho dispositivo desde otra aplicación o sistema.

También es posible verificar la conexión del dispositivo desde los ajustes de la red local, en donde ahora deberá mostrarse el dispositivo conectado a la red junto con su dirección IP y su dirección MAC. (Figura 15)

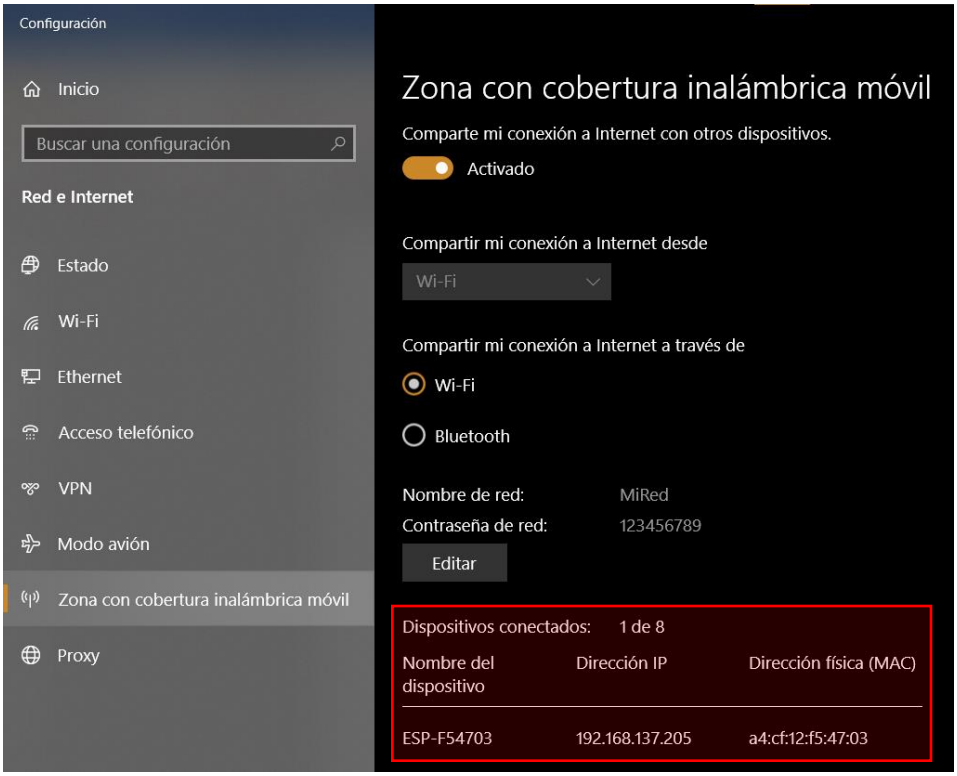


Figura 15. Configuración de la red con dispositivo conectado.

Antes de continuar con la práctica:

- Subir el programa del ejemplo a la tarjeta mediante el IDE de Arduino de acuerdo con el procedimiento explicado.
- Generar una red local para la comunicación con el dispositivo, deberán identificarse el nombre de la red y la contraseña para poder realizar la conexión.
- Conectar el dispositivo a la red e identificar su dirección IP y el puerto que se utilizará para la comunicación.

Envío de datos desde Simulink

A continuación, se realizará un programa en Simulink para enviar mensajes a la tarjeta mediante el protocolo UDP con la finalidad de poder controlarla de manera remota desde la computadora.

Para esto se puede generar un nuevo modelo en Simulink para enviar datos utilizando el bloque Packet Output (Figura 16).

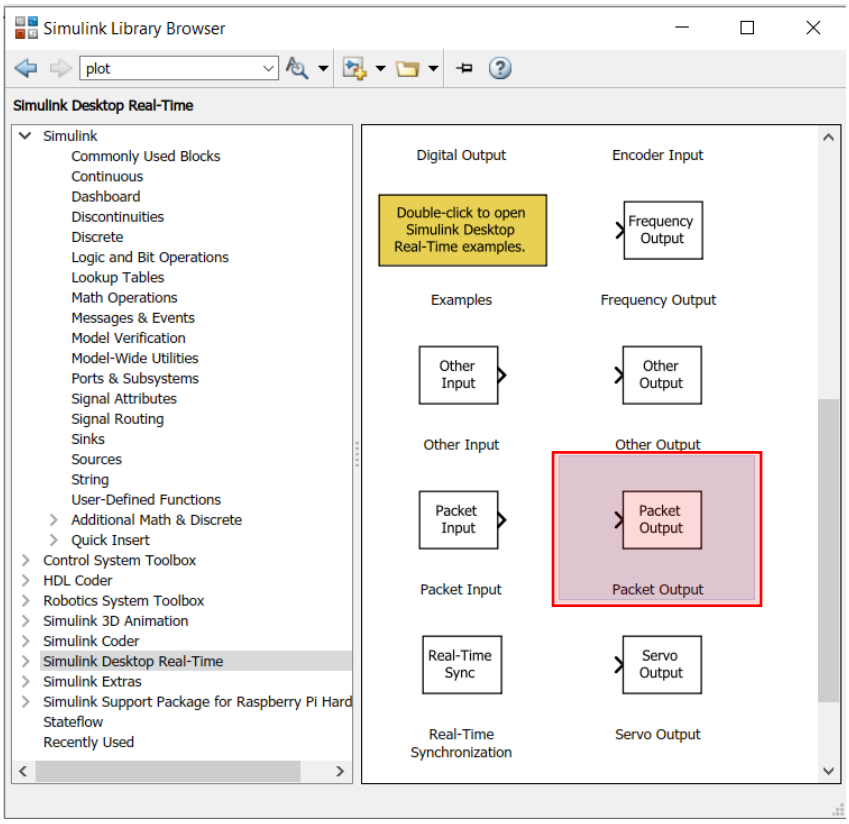


Figura 16. Bloque Packet Output para envío de datos.

Este bloque deberá configurarse de manera similar al bloque Packet Input que se utilizó en la práctica 1. Por ejemplo, para el envío de un solo dato entero cada segundo se puede utilizar la configuración mostrada en la Figura 17, misma que puede modificarse dando doble clic sobre el bloque.

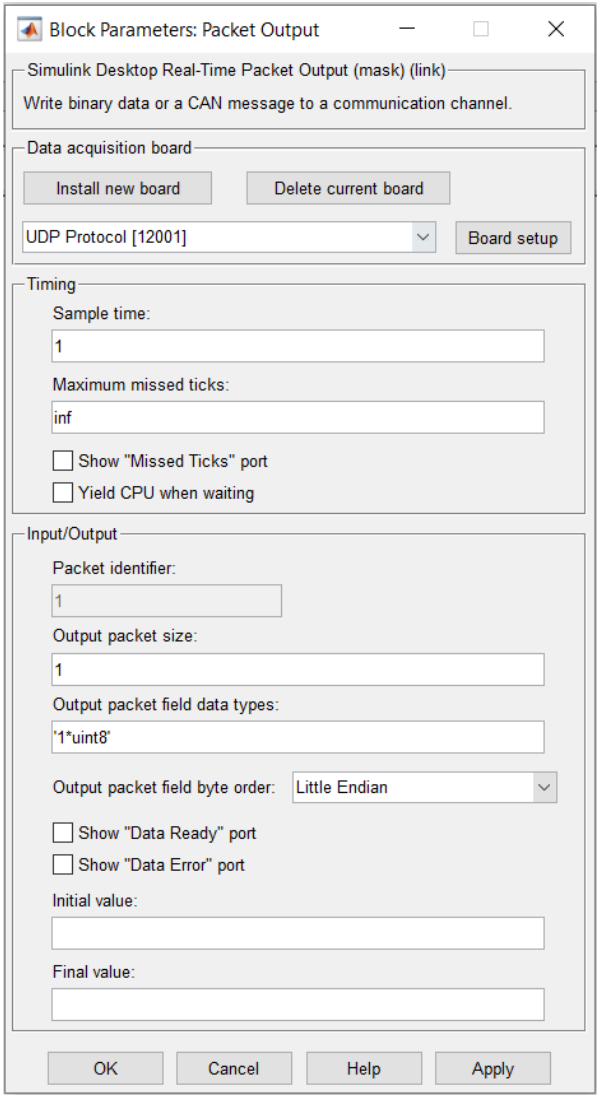


Figura 17. Parámetros de configuración del bloque Packet Output.

Aquí se observa que la configuración estableció el Sample Time con valor de 1, lo que quiere decir que la transmisión será cada segundo; además, el tamaño del paquete es de 1 y el tipo de dato enviado se estableció como “1*uint8”.

Hay que recordar también hacer la configuración de la dirección IP y del puerto para la comunicación, esto se realiza dando clic en el botón “Board setup”. Si no se tiene ninguna tarjeta instalada o no se quiere modificar otra se puede generar una nueva desde el botón “Install new board” → Standard Devices → UDP Protocol. (Como se explicó en la práctica 1)

Deberán introducirse los valores correspondientes a la dirección IP de la tarjeta ESP8266 y el puerto que se estará utilizando para la comunicación tal como fue establecido en el programa de la tarjeta. Tras modificar estos datos se puede probar la disponibilidad del dispositivo mediante el botón “Test” y finalmente dar clic en el botón “Ok” para guardar los cambios.

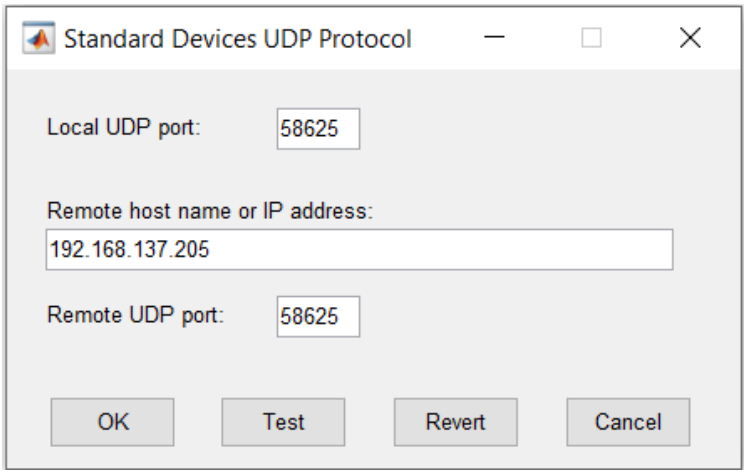


Figura 18. Configuración de la dirección IP y del puerto para comunicación.

Junto con este bloque se utilizará también un bloque de valor constante para generar el valor que se va a enviar a la tarjeta. Este bloque puede encontrarse en la biblioteca de bloques de Simulink en la sección “Commonly Used Blocks” bajo el nombre de “Constant” como se muestra en la Figura 19. La salida de este bloque deberá conectarse a la entrada del bloque “Packet Input”, de tal manera que este último utilice el valor de la constante para enviarlo mediante el protocolo UDP. (Figura 20)

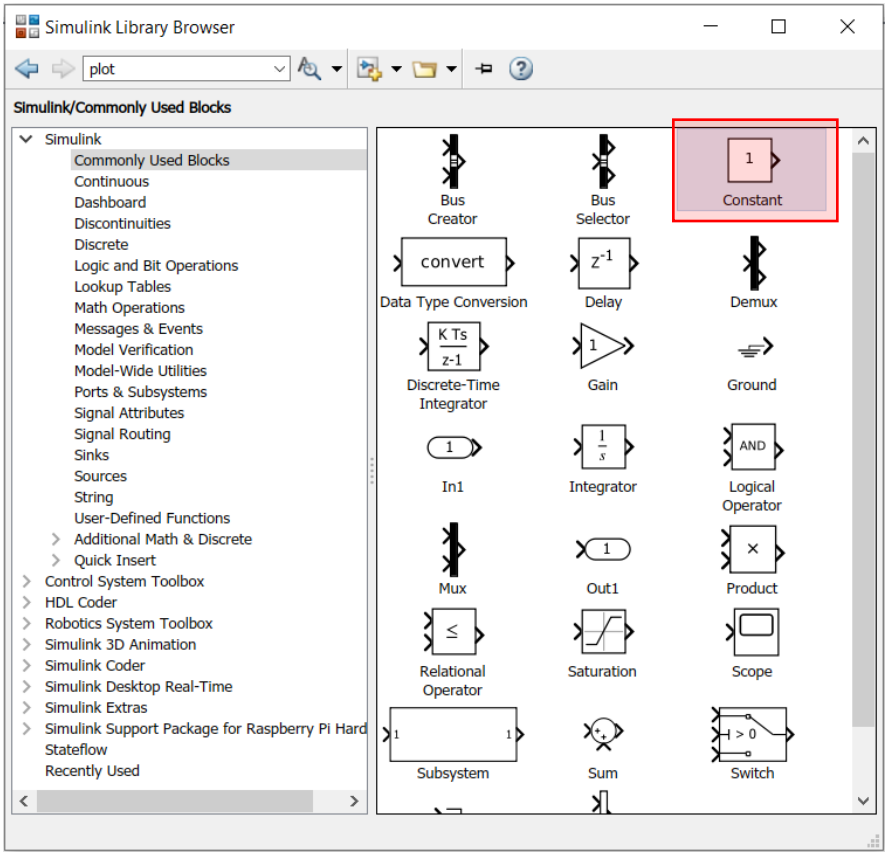


Figura 19. Bloque para generar un valor constante.

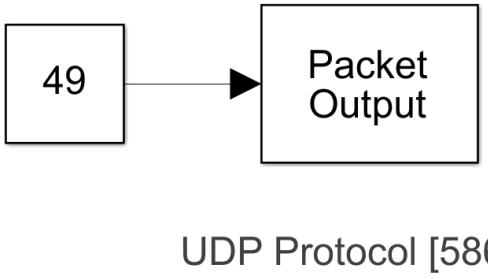


Figura 20. Conexión entre bloque "Constant" y bloque "Packet Output".

El valor de la constante puede modificarse haciendo doble clic en el bloque, con lo que se abre una ventana de parámetros como la mostrada en la figura 21. Cabe recordar que los datos se envían como bytes, por lo que, al leerlos en el monitor serial como caracteres, se va a representar el símbolo que corresponde a ese valor de acuerdo con el código ASCII. Es por eso por lo que aquí se utilizan los valores 48 y 49, que en código ASCII corresponden a '0' y '1'. Una tabla con el código ASCII puede encontrarse en la siguiente liga: https://elcodigoascii.com.ar/.

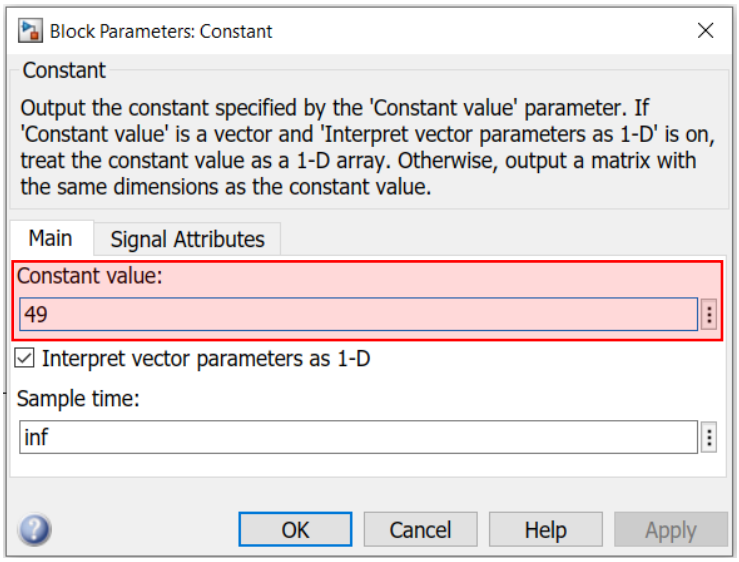


Figura 21. Parámetros de configuración para bloque "Constant".

Para hacer el cambio entre valores de manera sencilla se propone utilizar un elemento llamado "Rocker Switch" que se localiza en la biblioteca de bloques de Simulink en la sección Simulink → Dashboard (Figura 22). Este bloque funciona a manera de interruptor para modificar algún parámetro de otro bloque, en este caso se utilizará para alternar el valor del bloque "Constant" entre 48 y 49.

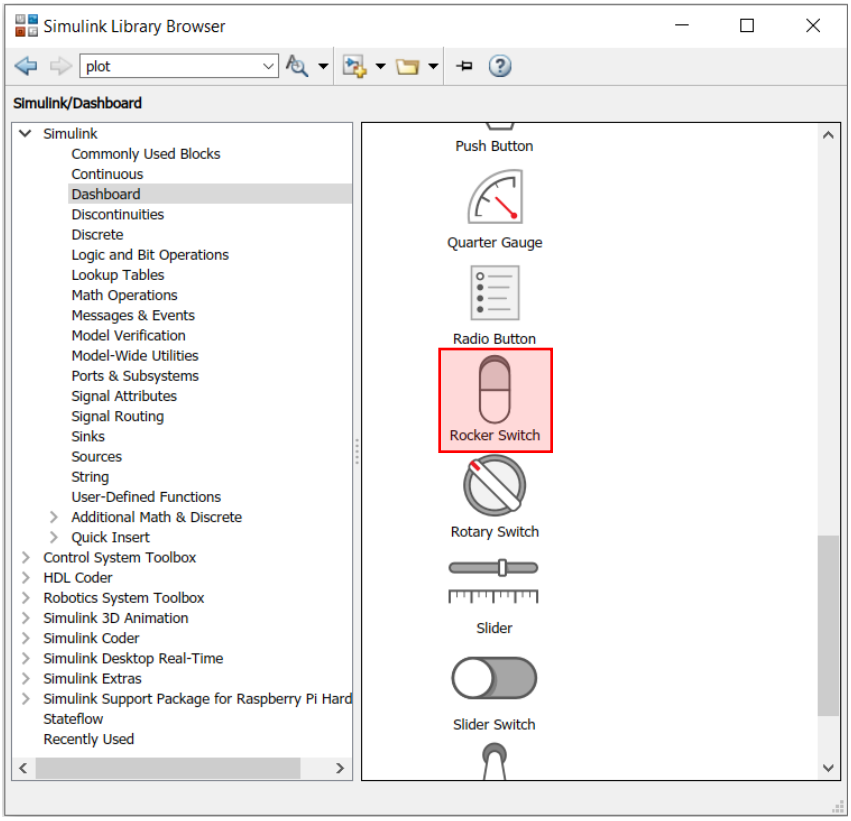


Figura 22. Bloque "Rocker Switch".

Al insertar este bloque en el modelo de Simulink se debe dar doble clic sobre él para acceder a la configuración de parámetros, con lo que se deberá abrir una nueva ventana como la que se muestra en la Figura 23.

Tras abrir esta ventana se deberá dar clic en el bloque que se quiere modificar con el interruptor, para este ejemplo se seleccionó el bloque del valor constante. A continuación, se debe habilitar la opción para conectar y se pueden establecer los valores que deberá tener cuando el interruptor esté arriba y cuando esté abajo.

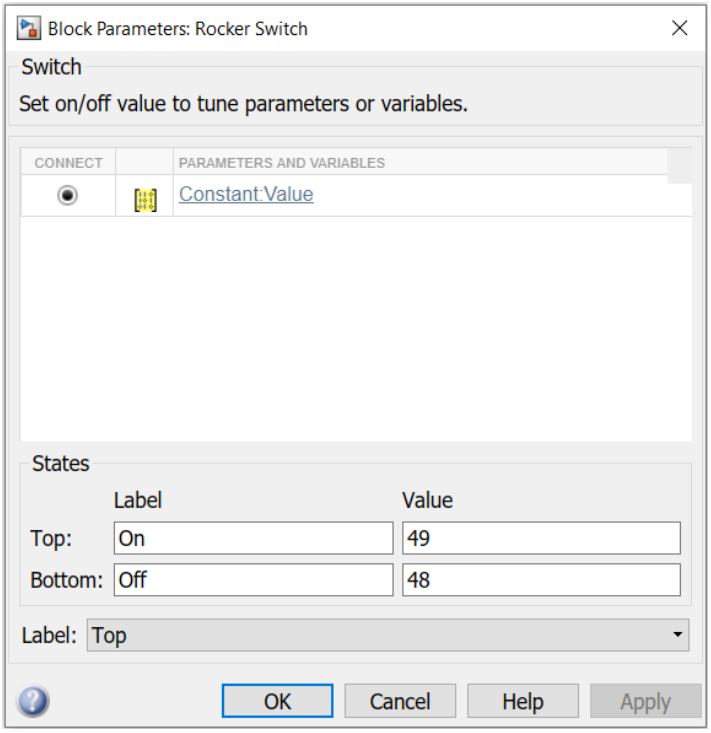


Figura 23. Configuración de parámetros del bloque "Rocker Switch".

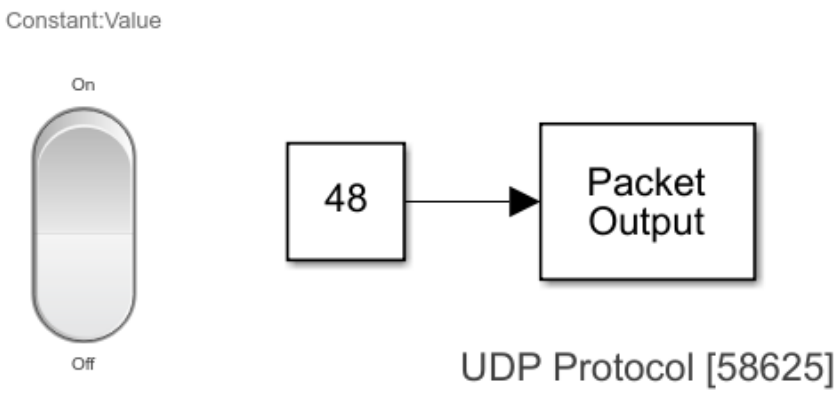


Figura 24. Modelo de Simulink para envío de datos por UDP.

Con esta configuración es posible correr la simulación de Simulink y, si todas las configuraciones fueron realizadas de manera correcta, comenzarán a enviarse los datos a la tarjeta ESP-8266 mediante la red. Para verificar el funcionamiento se puede acceder nuevamente al monitor serial, donde se desplegarán los datos recibidos en la tarjeta. Se debe verificar que al hacer clic en el interruptor el valor del dato enviado cambia conforme a los valores que se establecieron en la configuración de parámetros. Esto deberá visualizarse como se muestra en la Figura 25.

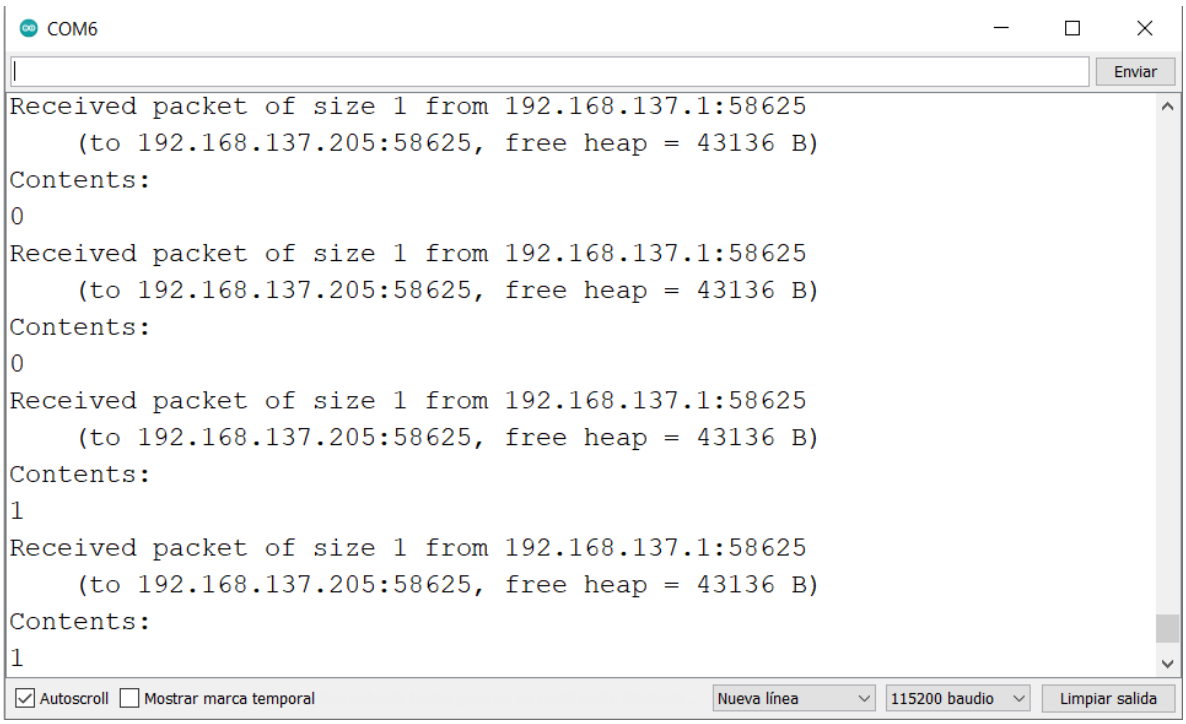


Figura 25. Verificación del envío de datos en el monitor serial.

Control de salidas de la tarjeta ESP-8266

Por último, se realizará el control de las salidas de la tarjeta ESP-8266, de tal manera que podamos utilizar los datos enviados vía WiFi para efectuar alguna acción con la tarjeta. Lo primero que debemos hacer es conocer los pines de la tarjeta, a continuación, se presentan los pines de la tarjeta NodeMCU, en caso de estar utilizando alguna otra tarjeta se deberá consultar el diagrama correspondiente.

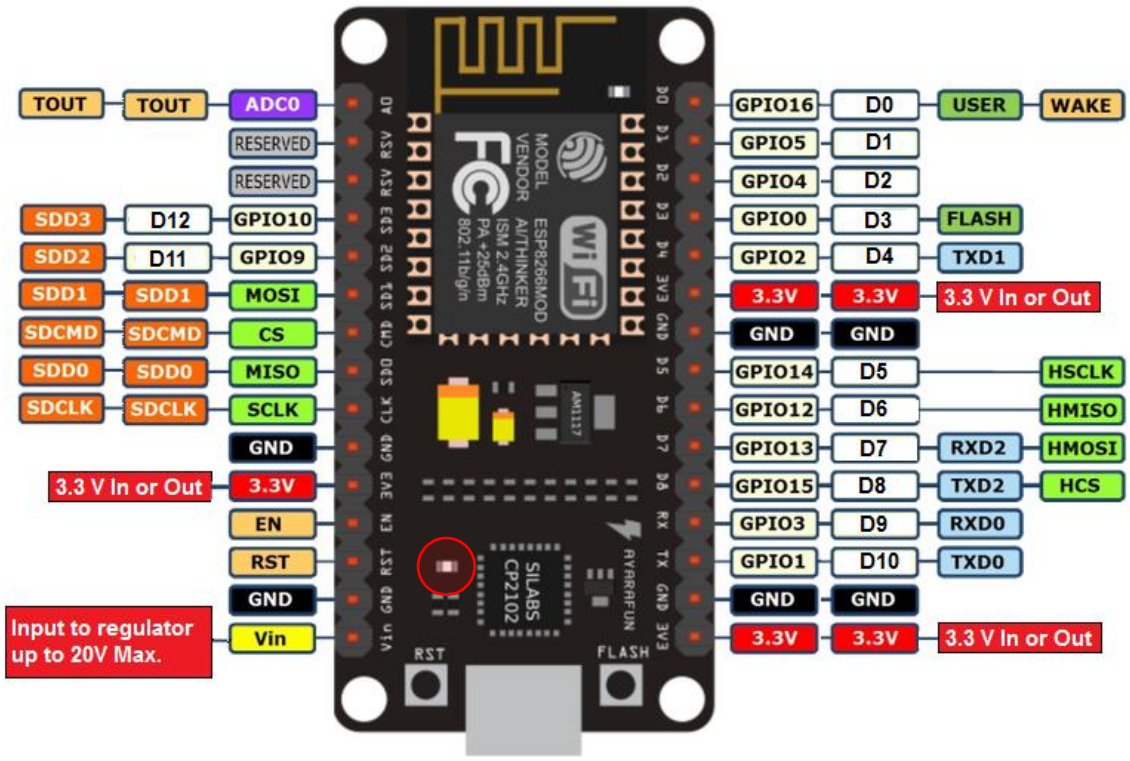


Figura 26. Diagrama de pines de la tarjeta NodeMCU ESP-8266.

Por facilidad, se utilizará primero el LED integrado que se encuentra conectado al pin D0, el cual se encuentra marcado por un círculo rojo en la Figura 26.

Sin embargo, es posible utilizar los demás pines, ya sea como salidas/entradas digitales, salidas PWM, entradas analógicas, comunicación, etc. Una introducción básica sobre cómo utilizar estos pines se puede encontrar en la página: https://www.luisllamas.es/guia-de-programacion-del-esp8266-en-entorno-arduino/.

Para poder controlar el LED integrado en la tarjeta se deberán hacer algunas modificaciones al código. Lo primero es incluir dentro de la función setup() la instrucción que configura el pin D0 como salida digital: pinMode(D0, OUTPUT);

En esta instrucción el primer parámetro (D0) indica qué pin es el que se está configurando, mientras que el segundo parámetro (OUTPUT) indica cómo se está configurando. En este caso OUTPUT indica que el pin se utilizará como salida digital, si se quisiera utilizar como entrada digital se puede configurar como INPUT.

Con esta modificación la función setup() queda de la siguiente forma:

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(STASSID, STAPSK);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(500);
  }
  Serial.print("Connected! IP address: ");
  Serial.println(WiFi.localIP());
  Serial.printf("UDP server on port %d\n", localPort);
  Udp.begin(localPort);

  pinMode(D0, OUTPUT);
}
```

Después, se deberá colocar dentro de la función loop() el código necesario para que se modifique la salida del pin D0 de acuerdo al dato que se recibió mediante la comunicación UDP. Si se recibió un '1' se deberá encender el LED, mientras que si se recibió un '0' se deberá apagar el LED. Para cambiar la salida digital del pin se utiliza la siguiente instrucción:

digitalWrite(D0, LOW);

digitalWrite(D0, HIGH);

En esta instrucción se indica el pin que se quiere modificar (D0) y el estado que se quiere asignar como salida a dicho pin (LOW o HIGH).

Una consideración importante es que el LED integrado en la tarjeta opera de forma invertida, es decir, si se asigna el pin D0 como HIGH el LED se apagará, mientras que si se asigna como LOW se encenderá.

Una forma de lograr controlar el LED con el dato recibido es modificando la función loop() de la siguiente manera:

```
void loop() {
  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if (packetSize) {
    Serial.printf("Received packet of size %d from %s:%d\n    (to %s:%d, free heap = %d B)\n",
                  packetSize,
                  Udp.remoteIP().toString().c_str(), Udp.remotePort(),
                  Udp.destinationIP().toString().c_str(), Udp.localPort(),
                  ESP.getFreeHeap());

    // read the packet into packetBufffer
    int n = Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
    packetBuffer[n] = 0;
    Serial.println("Contents:");
    Serial.println(packetBuffer);
```

```
    // send a reply, to the IP address and port that sent us the packet we received
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    Udp.write(ReplyBuffer);
    Udp.endPacket();

    if (packetBuffer[0] == '1'){
      digitalWrite(D0, LOW);
    }
    else if (packetBuffer[0] == '0'){
      digitalWrite(D0, HIGH);
    }
  }

}
```

Con este código se deberá ver reflejado en el LED el dato enviado desde el entorno de Simulink, por lo que ejecutando de nuevo la simulación se podrá encender y apagar el LED de la tarjeta directamente con el interruptor del modelo de Simulink.

Realiza las siguientes actividades para concluir con la práctica:

- Realiza el modelo de Simulink para enviar datos mediante el protocolo UDP a la tarjeta ESP-8266 y verifica que sean recibidos correctamente en el monitor serial.
- Realiza las modificaciones necesarias para poder encender y apagar el LED integrado de la tarjeta con el interruptor de Simulink.
- Controla 3 elementos electrónicos distintos con la tarjeta ESP-8266 de manera inalámbrica utilizando la comunicación UDP. Estos elementos pueden ser LEDs, motores DC, servomotores, pantallas LCD, relevadores, etc.
  Se recomienda verificar el funcionamiento de los dispositivos a utilizar para identificar la manera de controlarlos, además, pueden requerir elementos adicionales como fuente de alimentación y resistencias.

8. Resultados

Como resultados de esta práctica se deben entregar los programas realizados para módulo ESP-8266 y el modelo en Simulink, así como un video en el que se explica y se demuestra el funcionamiento.

9. Conclusiones

En esta práctica se revisaron los conocimientos básicos para lograr la comunicación inalámbrica vía WiFi utilizando el protocolo UDP. Se aprendió a programar una tarjeta de desarrollo con el módulo ESP-8266, que permite utilizar este tipo de comunicación inalámbrica para una infinidad de aplicaciones desde el encendido de LEDs hasta el control inalámbrico de robots móviles.

Bibliografía

- Documentación de Matlab/Simulink.
- Documentación de Arduino
- Programación de la tarjeta NodeMCU ESP-8266:
  - https://www.luisllamas.es/programar-esp8266-con-el-ide-de-arduino/
  - https://www.luisllamas.es/guia-de-programacion-del-esp8266-en-entorno-arduino/