UNIVERSIDAD ALFONSO X EL SABIO

ESCUELA POLITÉCNICA SUPERIOR

GRADUADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

MÓDULO DE DOMÓTICA CON ARDUINO EN RED DE IOT

ALUMNO: GOZALO DÍAZ DEL RÍO FONTÁN

N.P.: 121291

DIRECTOR DEL TRABAJO: MARIA ANTONIA SIMÓN

FECHA DE PRESENTACIÓN: 8 DE SEPTIEMBRE DE 2020

BREVE DESCRIPCIÓN:

DESARROLLO DE UN MÓDULO DE CONTROL REMOTO AUTOMÁTICO Y VERSÁTIL PARA APLICACIONES DE DOMÓTICA, ASÍ COMO DEL SISTEMA DE CONTROL Y GESTIÓN PARA ESTOS MÓDULOS MEDIANTE EL USO DE ARDUINO Y RASPBERRY PI. APLICACIONES DE IOT EN LA DOMÓTICA Y LA INFORMÁTICA INDUSTRIAL.

FIRMA DEL DIRECTOR **FIRMA DEL TUTOR** **FIRMA DEL ALUMNO**

1. OBJETIVOS Y ALCANCE DEL PROYECTO

En este proyecto se pretende diseñar un módulo controlador para ser empleado en ámbito de la domótica. Dicho módulo consistirá en una placa microcontroladora Arduino con un módulo de Wifi acoplado para permitir las comunicaciones inalámbricas. Estos módulos serán capaces de activar o desactivar actuadores en función de comandos originados en un ordenador central en la forma de una Raspberry Pi, así como de leer distintos sensores y comunicar esta lectura al ordenador.

En el consiguiente proyecto se realizará la programación y montaje de un módulo de Automatización mediante el uso de un Arduino Nano equipado con una antena de Wifi, así como la configuración de un Ordenador de Control.

El objetivo del proyecto es crear un módulo barato y accesible que permita controlar de forma inalámbrica cualquier dispositivo al que se conecte. Este control será posible realizarlo desde cualquier dispositivo capaz de comunicarse con una red Wifi.

El principal enfoque del proyecto es la versatilidad del producto. Se pretende que los módulos controladores puedan ser empleados para actuar sobre una gran variedad de dispositivos sin requerir alteraciones en su código. Para ello se pretende centralizar el control y la automatización en el ordenador central, minimizando la algoritmia contenida en los microcontroladores.

2. INTRODUCCIÓN

2.1 ESTUDIO DE CONCEPTO

Nos encontramos en una época en la que la automatización en el ámbito de la domótica está en pleno auge. En todos los comercios de electrónica se venden y promocionan los dispositivos "Smart-home", que prometen emular la existencia de un mayordomo virtual que encienda o apague luces, abra y cierre puertas, controle la temperatura, realice videovigilancia… Decenas de productos y funciones distintas se ponen a nuestra disposición para facilitarnos una interacción más automática con nuestro día a día.

Sin embargo, estas múltiples soluciones se encuentran siempre asociadas a una marca comercial, con Amazon Echo y Google Home a la cabeza del mercado actual, y requieren que todos los componentes del sistema pertenezcan a la misma familia. Con este proyecto se pretende crear un dispositivo que nos permita ejercer este nivel de control sobre aparatos y electrodomésticos "no inteligentes", sirviendo así de puente para permitir la automatización domótica sin necesidad de actualizar todos los aparatos ya presentes en casa.

2.2 INTERNET DE LAS COSAS

El Internet de las Cosas (Internet of Things, IoT) es un sistema de dispositivos de computación, controladores mecánicos o digitales y otros aparatos interconectados entre sí con identificadores únicos. Esta red de dispositivos cuenta con la capacidad de transmitir datos entre sí y realizar diversas tareas sin necesidad de interacción humana. El Internet de las Cosas nace de los sistemas empotrados, la necesidad de análisis en tiempo real del machine learning.

Actualmente las aplicaciones del Internet de las Cosas están siendo constantemente expandidas y exploradas por la Industria para todo tipo de aplicaciones. Gracias al IoT, los sistemas cableados tradicionales y los complejos y delicados autómatas, omnipresentes en ámbitos industriales, están empezando a desaparecer para ser sustituidos por dispositivos más modernos con capacidades de IoT, mucho más versátiles y modulares.

2.3 DOMÓTICA

La domótica (Unión del latín *domus*, casa, y *robótica)*, en su más básica definición, se trata de la implementación e integración de la tecnología en el diseño inteligente de un recinto cerrado. La domótica engloba una gran variedad de servicios y aplicaciones como la seguridad, la accesibilidad o el entretenimiento. El foco principal de la domótica es facilitar la vida doméstica proporcionando al usuario un nivel superior de interacción y control sobre su entorno mediante la instalación de dispositivos inteligentes que actúan de interfase entre humano y aplicación doméstica. Así, un frigorífico capaz de realizar pedidos al supermercado cuando un producto se acaba, un sistema de iluminación activado por voz o un termostato que se puede programar desde el teléfono móvil, independientemente de la distancia al hogar, todos entran bajo el concepto de la domótica.

2.3.1. Historia de la Domótica

Los inicios de la Domótica se pueden atribuir a la distribución de energía a todos los hogares a principios del siglo 20. Este avance permitió la pronta aparición de elementos como máquinas lavavajillas, lavadoras automáticas de ropa o calentadores de agua. Estos electrodomésticos que ahora consideramos parte esencial de un hogar moderno fueron máquinas revolucionarias que

permitieron automatizar tareas cotidianas y aminorar el esfuerzo humano necesario para llevarlas a cabo.

No fue hasta 1975 sin embargo que se desarrolló la primera tecnología de red de comunicación orientada hacia la domótica. Esta tecnología era el protocolo de comunicaciones X10, aún empleado en la industria moderna. Desarrollado en Escocia por Pico Electronics, X10 emplea las mismas líneas de corriente que la red eléctrica doméstica del interior del hogar para transmitir sus mensajes. Mediante picos de tensión en frecuencias electromagnéticas determinadas, este protocolo es capaz de transmitir instrucciones e información simple a una enorme variedad de dispositivos conectados a la red.

En la actualidad la domótica se está desviando hacia las comunicaciones inalámbricas, ya sean mediante infrarrojos, Bluetooth o wifi, y la aparición de nuevas tecnologías solo facilita el crecimiento del sector. Las estadísticas muestran que en Estados Unidos, a finales de 2012 1.5 millones de hogares tenían alguna forma de domótica instalada mientras que para el final de 2018, el número había aumentado hasta los 45 millones de viviendas. Este crecimiento exponencial sigue en pleno auge en todo el planeta y cada día surgen nuevas tecnologías para facilitar cada vez más las tareas del día a día.

2.3.2. Aplicaciones de la Domótica

Dentro de las innumerables aplicaciones que existen de la domótica, se discutirán las más presentes y habituales:

Calefacción y aire acondicionado (Control de temperatura)

Estos sistemas controlan la calefacción, refrigeración y ventilación del hogar. Ya sea a través de termostatos que han de ser configurados manualmente o de sistemas inteligentes completamente conectados a la red, estos sistemas se encuentran presentes en la gran mayoría de los hogares modernos. Los modelos más nuevos permiten monitorizar y controlar la climatización individual de cada habitación del hogar desde cualquier lugar empleando un dispositivo móvil.

Vigilancia y seguridad

Puede ser empleando cámaras, micrófonos, sensores de CO2 o cualquier otra cantidad de distintas tecnologías, pero la domótica está muy presente en los sistemas de seguridad del hogar. Desde cámaras de circuito cerrado que almacenan sus grabaciones en discos duros hasta sistemas de detección de movimiento y ocupación que transmiten sus datos en directo a la nube para ser accedidos desde un móvil, la domótica permite aumentar la seguridad del hogar.

Control de dispositivos para eficiencia energética

Esta aplicación trata de regular el consumo del hogar de manera que se reduzcan al máximo el coste y la contaminación generados. Un hogar equipado con un módulo fotovoltaico, por ejemplo, puede disponer de un sistema domotizado que restrinja y automatice el uso de electrodomésticos de gran consumo como una lavadora o una secadora a aquellas horas en las que el Sol genere más energía. De la misma manera, un automatismo puede contener programadas aquellas franjas horarias en los que la compañía eléctrica cobra menos por potencia consumida y aprovecharlos para activar funciones o aparatos que consumen más energía, manteniéndolos en reposo durante las franjas de mayor consumo. Sistemas como estos pueden llegar a reducir de forma sorprendente los costes energéticos del hogar y son enormemente compatibles con otros sistemas inteligentes de control del hogar.

Accesibilidad para personas de avanzada edad y personas discapacitadas

Una aplicación que está creciendo de forma muy acelerada cuanto más avanza la tecnología es la de aumentar la accesibilidad al hogar de las personas con movilidad reducida, problemas de visión u otras discapacidades. Estos dispositivos permiten a una persona con movilidad reducida subir o bajar escaleras sin dificultad, abrir o cerrar puertas y tantas otras tareas cotidianas. Un asistente del hogar con interfaz de voz puede facilitar las tareas diarias a una persona con discapacidad visual.

2.4 TECNOLOGÍAS Y LENGUAJES DE PROGRAMACIÓN

Para la realización de este proyecto se han empleado una variedad de tecnologías y lenguajes de programación, que se detallan a continuación:

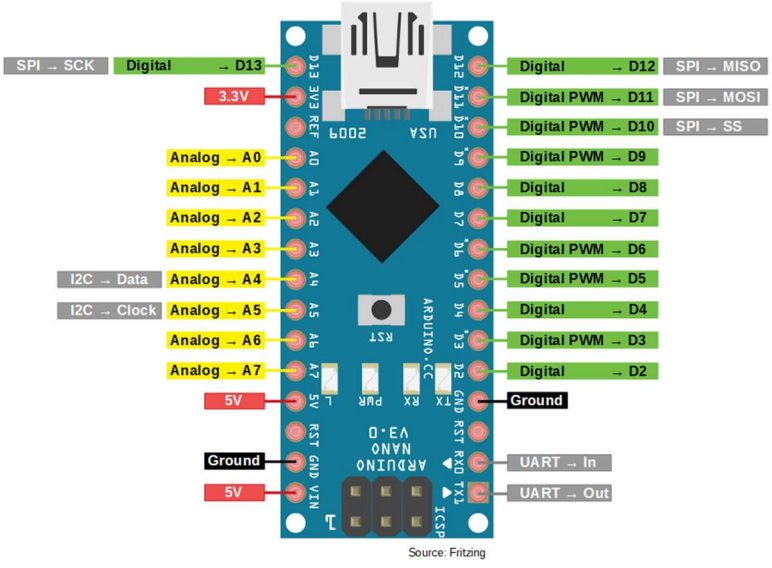
2.4.1. Arduino

Las placas de desarrollo Arduino son una colección de microcontroladores con una memoria flash y varios pines de entrada/salida asociados recogidos bajo una licencia de hardware libre. Su principal uso es en el ámbito de la robótica y la automatización y son programables a través de un software de licencia libre empleando el lenguaje C++. Una placa de Arduino puede leer y modificar el estado de tensión de sus pines tanto de forma discreta digital, con dos niveles lógicos, como de forma analógica, permitiendo leer y emitir valores continuos.

Se trata de una pieza de hardware barata, resistente, sencilla de programar, muy personalizable y enormemente versátil. Existen numerosos modelos con diferencias en el número de pines, la potencia del microcontrolador, el tamaño de la memoria y el número de pines. Para este proyecto se empleará el modelo Nano debido a su pequeño tamaño, pese a que el funcionamiento es idéntico si se desea emplear el modelo UNO o MEGA.

Arduino Nano

Entre las múltiples variedades de placas de desarrollo de Arduino, el Nano es la versión más reducida de tamaño que sin embargo mantiene todas las funcionalidades del modelo UNO, el más tradicional. El Nano cuenta con un controlador ATmega328P de 16MHz, con 1KB de EEPROM, 32 KB de memoria Flash y 2KB de SRAM. Para funcionar requiere alimentación entre 7 y 12 Voltios y opera sus entradas y salidas a 5V. Cuenta con 14 pines que pueden ejercer de entradas / salidas digitales (6 que permiten salidas PWM) y 8 analógicos. Puede emitir hasta 40mA de corriente por cada pin. Puede conectarse por USB a un ordenador para su programación y alimentación o puede ser alimentado a través de uno de sus pines. A través de otro de sus pines puede alimentar dispositivos a 3.3 V.

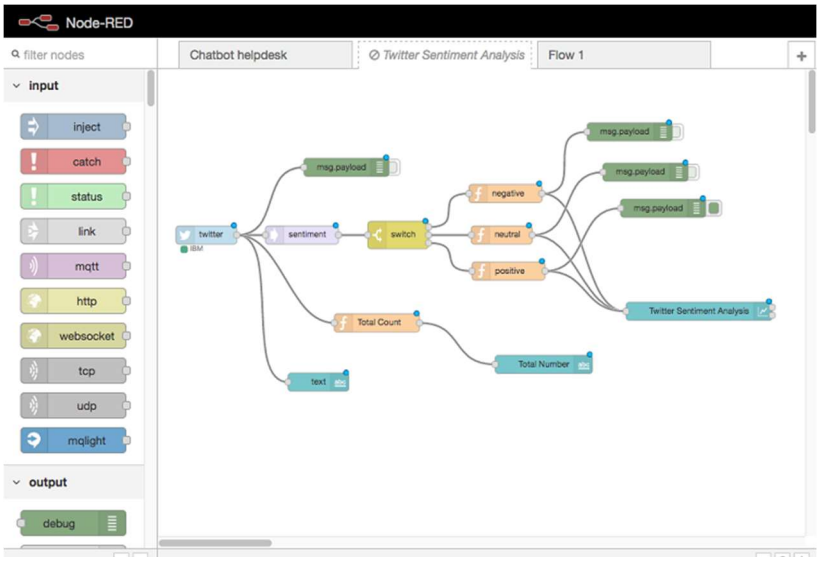


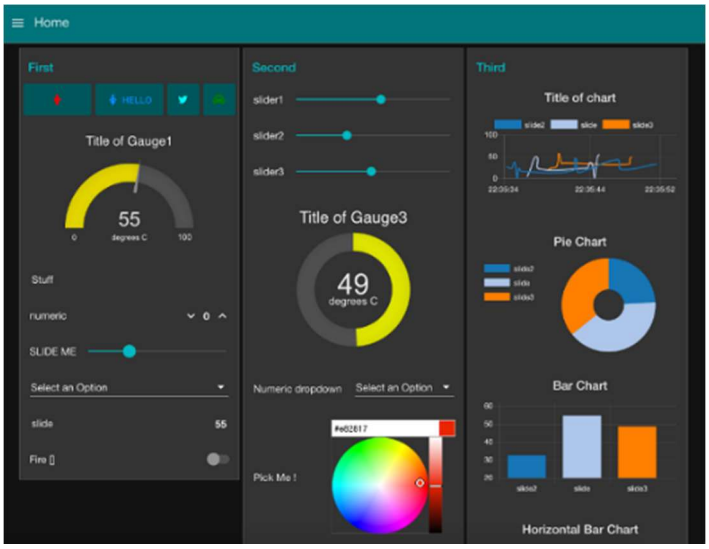
2.4.2. Raspberry Pi

Las Raspberry Pi son una serie de pequeños ordenadores de una sola placa. Sus aplicaciones son prácticamente ilimitadas y se encuentran presentes en aplicaciones industriales, educativas, investigadoras y en robótica, destacando gracias a su sorprendente potencia, versatilidad y portabilidad. Existen múltiples generaciones del producto, cada una con capacidades mejoradas. Para este proyecto se empleará una Raspberry Pi 4 Modelo B de 4 Gb de memoria, pese a que el proyecto se puede replicar a la perfección con cualquier modelo a partir de la Raspberry Pi 3.

Para su funcionamiento, como con cualquier ordenador, la Raspberry Pi requiere un Sistema Operativo. Este sistema se instala en una tarjeta SD que luego se inserta en la placa y actúa como disco duro del sistema. Hay numerosas imágenes distintas para Raspberry, dependiendo de la funcionalidad que se desee de ellas. Por su versatilidad y lo extendido de su uso, emplearemos Raspbian, una distribución de Debian adaptada a la Raspberry. Esto significa que, con muy pocas modificaciones, este proyecto puede funcionar con cualquier ordenador operando con una distribución de Linux, como por ejemplo Ubuntu.

2.4.3. Node-Red





Node-Red es una herramienta de programación diseñada por IBM para conectar dispositivos de hardware, APIs y servicios online dentro del ámbito del IoT. Consiste en una interfaz web alojada en un dispositivo que ejerce de anfitrión. En esta interfaz se pueden programar flujos de nodos que intercambian datos en forma de mensajes en formato JSON (JavaScript Object Notation). Al tratarse de una herramienta Open Source, cualquier usuario puede contribuir al proyecto sus propios nodos y funciones, por lo que el proyecto crece exponencialmente con su comunidad y es muy sencillo encontrar nodos y funciones personalizados para la aplicación que se desea. También cuanta con un nodo que permite programar funciones propias en lenguaje Javascript. Al

haber sido desarrollado con IBM, su implementación junto con nodos de MQTT lo convierte en la más versátil herramienta para el IoT.

2.4.4. MQTT

MQTT (Message Queing Telemetry Transport) es un protocolo de comunicaciones de red muy ligero basado en mecánicas de publicación-subscripción que transmite mensajes entre dispositivos, generalmente haciendo uso de conexiones TCP/IP (comunicaciones de red de Internet). Sin embargo, también está soportado por cualquier protocolo que ofrezca comunicaciones bidireccionales sin pérdidas. Su utilidad destaca en aplicaciones que requieran un código muy ligero o en sistemas con ancho de banda limitado.

El funcionamiento de MQTT es muy simple: los elementos que participan en la comunicación son un gestor de mensajes, el broker, y en número indeterminado de clientes. Los componentes de un mensaje MQTT son un tema y un contenido. Un cliente interactúa con el tema o temas deseados de dos maneras: puede **publicar** un mensaje en un tema concreto y puede **suscribirse** a uno o más temas. El gestor de mensajes se encarga de registrar los mensajes publicados en cada tema y distribuirlos de forma inmediata a todos los clientes suscritos a dicho tema. Existen funcionalidades que permiten asegurar distintas Calidades de Servicio (Quality of Service, QoS) para la comunicación:

- QoS 0: En modo QoS 0, el gestor envía los mensajes a los clientes suscritos en cuanto los recibe y después borra el mensaje. También se conoce como modo "Fire and Forget".
- QoS 1: En modo QoS 1, el gestor al recibir un mensaje, lo reenviará a los clientes suscritos y esperará una confirmación de mensaje recibido antes de borrarlo. De no recibir la confirmación, volverá a enviar el mensaje de forma periódica hasta recibir la confirmación. Este modo se conoce también como "At least Once", Al menos una vez.
- QoS 2: En modo QoS 2 el gestor se asegura de que el mensaje le llega al cliente esperando una confirmación por parte de este y luego vuelve a consultar con el cliente si la comunicación ha tenido éxito antes de borrar el mensaje. Este modo se conoce como "Once and Only Once", Una vez y sólo una vez. QoS 2 también permite emplear la persistencia de mensajes, un sistema mediante el cual el gestor siempre almacena un registro de todos los mensajes enviados a cada tema para que si un nuevo cliente se suscribe a ese tema reciba todos los mensajes que se ha "perdido" antes de su suscripción.

Para la aplicación que nos ocupa emplearemos QoS 2, ya que la información que transmitiremos no es de naturaleza crítica.

2.4.5. PuTTY, FileZilla

PuTTY es un cliente de SSH que permite acceder a la consola de comandos de una máquina de Ubuntu (En este caso el sistema Raspbian de la Raspberry Pi) y manejarla a través de una red a la que esté conectada. FileZilla permite la transferencia de archivos entre dos equipos empleando SSH. Ambos programas son necesarios para operar la Raspberry Pi.

3. MEMORIA DEL PROYECTO

3.1 MATERIALES Y PROGRAMAS EMPLEADOS

3.1.1. Hardware

Raspberry Pi 4 Modelo B (4Gb).

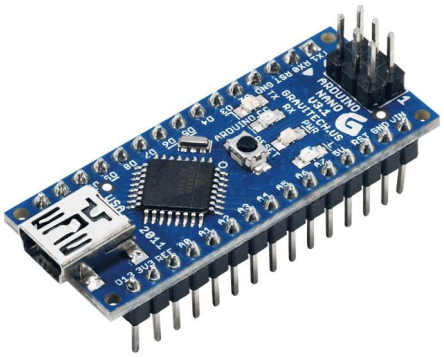
Puede ser sustituida por cualquier modelo igual o posterior a la Raspberry Pi 3. Para este proyecto se ha empleado la Pi 4 modelo B de 4 Gb de memoria RAM. Si se va a emplear el modelo 4 es recomendable acompañarlo de una carcasa con radiadores y ventilador incorporado, ya que es necesario facilitar la disipación de calor

Pantalla táctil LCD para Raspberry Pi (Opcional)

No es necesario disponer de pantalla LCD para la Raspberry, pero puede servir de otra interfaz más para interactuar con el sistema

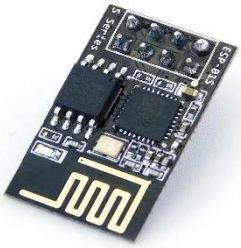
Arduino Nano

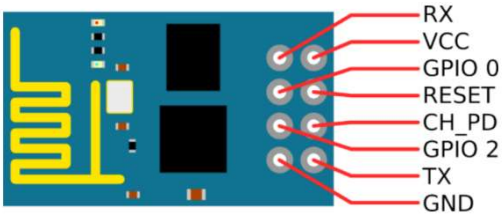
El modelo empleado de Arduino nano es el V3.0, que emplea un chip FTDI para comunicación serie con el ordenador. No debería haber ningún inconveniente usando un clon de Arduino (ELEGOO, por ejemplo), simplemente será necesario seleccionar el complemento correspondiente en la IDE de Arduino al escribir el programa.



Módulo de Wifi ESP8266 ES01

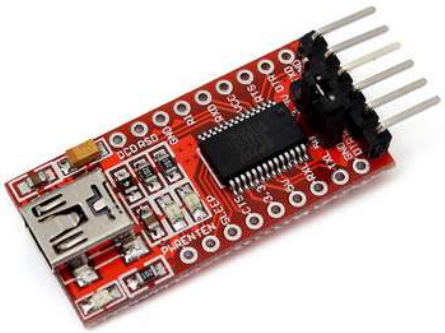
Existen varias versiones de este producto, pero parecen ser todas compatibles. Se puede emplear otro módulo de wifi compatible con Arduino si se desea, siempre que sea programable en C++ empleando el IDE de Arduino. Debido a que posee dos pines para comunicación serie, es necesario un adaptador para poder programarlo desde el ordenador. Algunas fuentes dicen que se puede emplear el propio Arduino para ello, pero no parece una técnica fiable o fácilmente replicable.



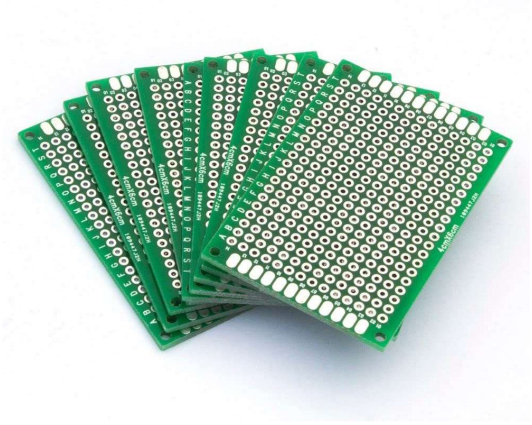


Conversor FTDI Mini USB a TTL Serie

Se trata simplemente de un adaptador que permite programar el ESP8266 desde el ordenador conectándolo con un cable USB.



Placas PCB perforadas para soldadura con estaño



Soldador, estaño, cables

Transformador de tensión 220V AC – 12 V DC

Convertidores de tensión 12V – 5V y 5V – 3.3V

Transistores y actuadores varios para demostración (Ventiladores, bombillas LED, etc.)

3.1.2. Software

Raspberry Pi

Para la Raspberry Pi, se empleará la versión más actualizada de **Raspbian**, una distribución gratuita basada en Debian (Ubuntu) y optimizada para la Raspberry Pi.

Ya incluido en esta distribución se encuentra **Node-Red**, pese a que, para mayor seguridad acerca de la compatibilidad de todas las partes del proyecto se reinstalará la versión más moderna.

También es necesario un bróker de MQTT. Se empleará mosquitto por su facilidad de instalación y gestión.

Si se desea emplear una pantalla táctil LCD para Raspberry Pi, también será necesario instalar los controladores de dicha pantalla, que permiten emplear los puertos de entrada / salida genéricos de la Raspberry (GPIO) para transmitir imagen a la pantalla y leer las señales introducidas con el puntero táctil.

ESP8266

Para poder programar y operar correctamente con el ESP8266, es necesario descargar las bibliotecas *ESP8266WiFi.h* y *PubSubClient.h* de C++ en el ordenador que se vaya a emplear para la programación. Esto puede hacerse muy fácilmente desde el gestor de librerías de la IDE de Arduino.

PC

La principal herramienta que se necesita en el PC es la IDE de Arduino. Las siglas IDE, en inglés significan “Integrated Development Enviroment” (Entorno integrado de Desarrollo) y se refieren a un software que permita programar código en uno o varios lenguajes de programación distintos. La IDE de Arduino es un entorno gratuito proporcionado por la propia empresa Arduino Software a través de su página web, y permite programar código en C y C++ para luego grabarlo en

microcontroladores de Arduino o similares. Esta herramienta permite ampliar su administrador de placas para escribir programas en más microcontroladores que los predeterminados, por lo que también hace posible programar el módulo ESP8266.

Otras dos herramientas necesarias para el PC son programas de comunicación SSH (Secure Shell) con la Raspberry. SSH es un protocolo de comunicación criptográfico permite acceder a la consola de comandos del sistema Raspbian en la raspberry, así como a su estructura de carpetas permitiendo al usuario tanto enviar órdenes y comandos como transferir archivos, todo desde el PC. Los dos programas de SSH a emplear son PuTTY y FileZilla, el primero permite la conexión con la consola de comandos mientras que el segundo permite realizar transferencias de archivos.

3.2 FASES DEL PROYECTO

A continuación, se detallarán las fases del proyecto para su replicación y los requisitos para el desarrollo de las mismas. Cada fase va precedida de un código que indica a que parte del proyecto pertenece:

RP – Raspberry Pi, TH – Desarrollo teórico, ESP – Módulo Wifi esp8266, AN – Arduino NANO, POT – Potencia

3.3 PROGRAMACIÓN

3.3.1. Raspberry Pi

RP1 – Raspbian

Requisito: Ninguno

En esta fase se instala el sistema operativo Raspbian en la Raspberry Pi.

Del sitio web de Raspberry se descarga el software Raspberry Pi Imager. Una vez descargado y habiendo insertado una tarjeta SD formateada en el ordenador, se ejecuta el programa y se elige grabar en la tarjeta el sistema NOOBS (New Out Of the Box Software), que permite instalar una variedad de imágenes.

Una vez grabado el sistema NOOBS, se debe conectar la Raspberry a un monitor y un teclado, que emplearemos para navegar por el menú de NOOBS hasta instalar Raspbian lite. Una vez instalado debemos acceder a las opciones de configuración del sistema mediante el comando `sudo raspi-config` y activar la comunicación SSH en las opciones de Interfaz.

RP2 – Wifi

Requisito: RP1

En esta fase la Raspberry Pi se configura como punto de acceso Wifi, generando la red "RaspiHouse".

Para facilitar este proceso, se ha escrito un programa que contiene los comandos bash necesarios para realizar la configuración. Solo es necesario transferir mediante FileZilla el archivo wifiAPconfig.sh al directorio de usuario de la Raspberry y ejecutar desde ese directorio el comando `sudo bash wifiAPconfig.sh.`

Si se desea configurar el SSID y la contraseña de la red wifi, se puede editar el ejecutable con `sudo nano wifiAPconfig.sh` y cambiarlos en las líneas:

```
sudo echo -e "ssid=RaspiHouse" >> /etc/hostapd/hostapd.conf
```

```
sudo echo -e "wpa_passphrase=MyIoTProject" >> /etc/hostapd/hostapd.conf
```

RP3 – MQTT

Requisito: RP1

En esta fase se instala y configura mosquitto, el software gestor de mensajes de MQTT que emplearemos. Simplemente es necesario ejecutar los siguientes comandos con la Raspberry conectada a internet:

```
sudo apt-get update
sudo apt-get install mosquitto
```

RP4 – Node-Red

Requisito: RP3

En esta fase se instala Node-Red junto con los servicios en segundo plano que permiten añadir nodos nuevos y gestionar el funcionamiento del servicio. Simplemente es necesario ejecutar los siguientes comandos con la raspberry conectada a internet:

```
sudo bash < (curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

sudo systemctl enable nodered.service
sudo npm install node-red-dashboard
```

Por último es necesario transferir mediante FileZilla el archivo flows.json al directorio /home/pi/.nodered de la raspberry.

En cuanto se haya completado esta fase, ya se puede acceder a Node-Red a través de cualquier dispositivo conectado a la red Wifi de la Raspberry, accediendo al puerto 1880 de la dirección IP de la Raspberry (https://192.168.4.1:1880) en la barra de navegación del navegador). Para acceder a la aplicación se emplea la dirección https://192.168.4.1:1880/ui

3.3.2. Módulo ESP8266

ESP1 – Programación

Requisito: RP4

Se programará el módulo ESP8266 empleando el IDE de Arduino. Para ello se debe montar el siguiente circuito con el ESP8266 y el adaptador de FTDI a TTL.

Una vez montado, desde la IDE de Arduino se debe configurar el módulo. Para ello se accede al menú Archivo -> Preferencias y en el campo de gestor de otras placas se añade la línea:

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

Reiniciamos el IDE de Arduino y ahora en “Herramientas” seleccionamos como tarjeta el módulo genérico de ESP8266. Conectamos el cable al adaptador de FTDI y escribimos el código contenido en el archivo espCode.ino. En este momento se determina el identificador

```
// MQTT Configuration
const String clientId = "ESP1";
const char* inTopic = "raspihouse/espin/1";
const char* outTopic = "raspihouse/espout/1";
```

único de este módulo ESP: en el código, bajo "MQTT Configuration" se debe especificar el identificador que se le asignará al módulo (Ver imagen).

Se observa en la imagen que el nombre del cliente que se ha proporcionado es ESP1 y que para la comunicación MQTT, se suscribirá al tema *raspihouse/espin/1* y publicará en *raspihouse/espout/2*. Estos valores pueden ser alterados como se desee.

3.3.3. Arduino Nano

AN1 – Programación

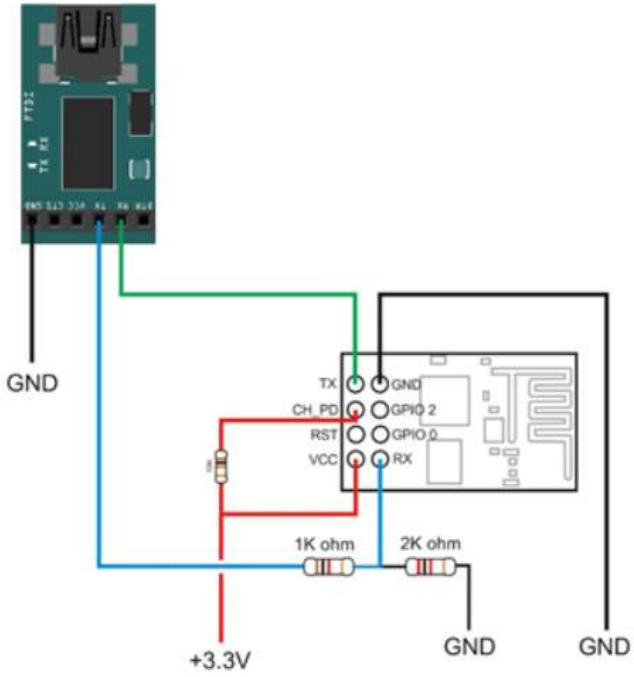
Requisito: ESP1

Se programará el Arduino Nano haciendo uso de la IDE de Arduino. Simplemente es necesario conectarlo al PC, seleccionar la placa Nano en el selector de placas y escribir en él el código arduinoCode.ino

3.4 IMPLEMENTACIÓN DEL CIRCUITO

3.4.1. Circuito de programación del módulo ESP826-01

El primer circuito que es necesario montar se trata del circuito para programar el módulo ESP8266. Este es un circuito provisional, ya que solo se emplea en la fase ESP1.



Circuito de programación de ESP8266-01 con adaptador FTDI a SSL

Según algunas fuentes, es posible que sea necesario conectar el pin GPIO 02 del ESP8266 a GND para activar el modo de programación. Esto no fue necesario en el prototipo de este proyecto, pero puede ser relevante para modelos ligeramente distintos del módulo ESP.

4. PLIEGO DE CONDICIONES

Pese a que el proyecto está enfocado a la versatilidad y su aplicación pretende no estar condicionada por ningún factor externo, siendo posible desplegarlo en cualquier entorno sin dificultades, existen unas mínimas condiciones necesarias para su implementación. En este apartado se especificarán esas condiciones, así como se harán notar las precauciones especiales que deben ser tomadas a la hora de montarlo.

4.1 CONECTIVIDAD

Para el correcto funcionamiento del sistema, es necesario configurar los diferentes componentes del sistema para que se comuniquen entre sí a través de Wifi. En este Proyecto se emplea una red de Wifi generada por la Raspberry Pi para efectuar las comunicaciones, esto requiere que todos los componentes del proyecto se encuentren dentro del alcance de esta red. Dado que para ello es casi indispensable que todos se encuentren en la misma habitación, no se recomienda la implementación del proyecto a distancias mayores.

En el Anexo 3 se puede encontrar una forma de solventar esta limitación: en vez de emplear una red generada por la Raspberry, los módulos individuales y la Raspberry emplean una red Wifi preexistente (una red doméstica que alcance las habitaciones del hogar).

A la hora de cambiar la configuración de redes del sistema Raspbian, se recomienda tener un mínimo conocimiento de su funcionamiento. Esto es porque existe el riesgo de bloquear ciertas comunicaciones que no nos permitieran volver a acceder al sistema, requiriendo un formateo y nueva instalación de Raspbian.

4.2 REFRIGERACIÓN

La Raspberry Pi empleada es la Pi 4. Este modelo ofrece muy altas prestaciones para su tamaño y coste, pero también presenta problemas de recalentamiento. Si se pretende emplear el modelo 4, es recomendable añadir radiadores y hasta un pequeño ventilador a la Raspberry. La mayoría de carcasas (incluida la que ejerce de soporte para la pantalla LCD) vienen de serie con este ventilador y radiadores para ser instalados.

4.3 MONTAJE

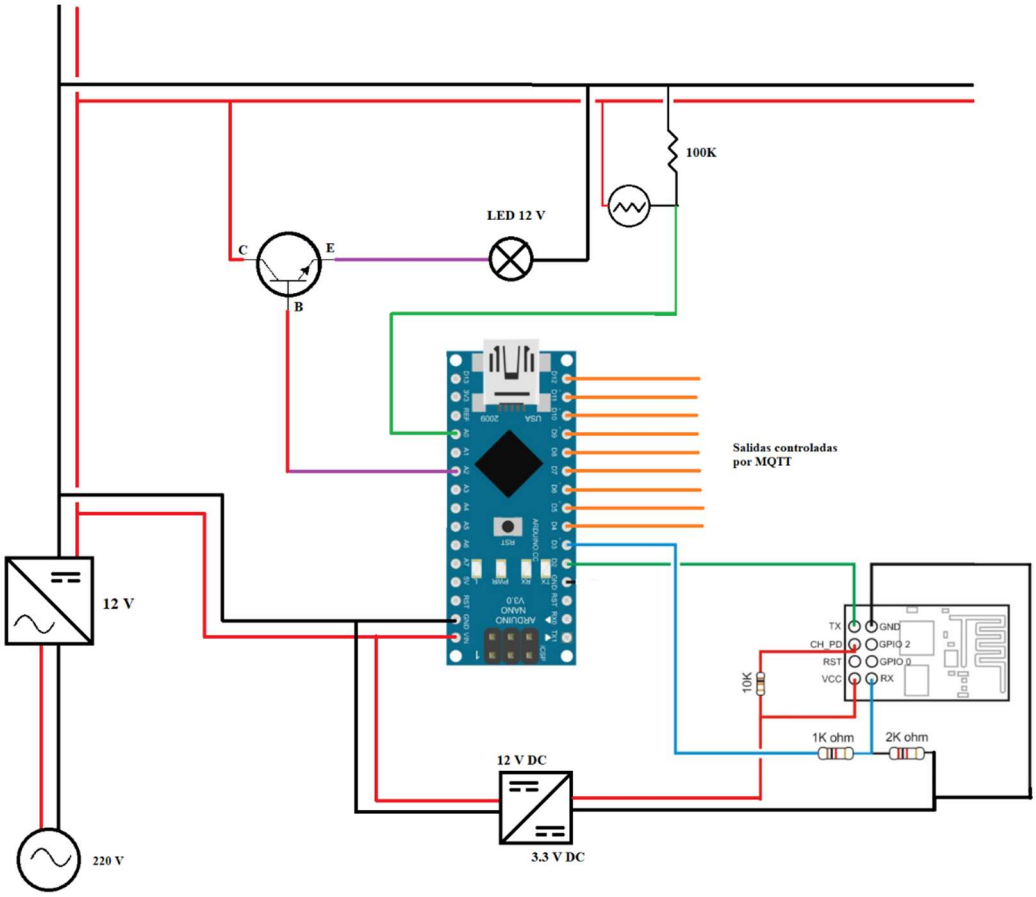
El montaje de los componentes del proyecto requiere que se realice la soldadura de los componentes a las PCBs. Esta soldadura, si no se realiza con extremo cuidado y atención puede llegar a dañar los componentes o a crear cortocircuitos. Especialmente el Arduino y el módulo ESP8266 pueden dañarse si la soldadura entra en contacto con la propia placa que contiene el microcontrolador

5. PRESUPUESTO

• Arduino Nano	10.99 € / 3 uds
• Raspberry Pi 4 4Gb con carcasa, tarjeta SD y accesorios	76.49 €
- Solamente la Raspberry Pi sin accesorios	(52.00 €)
• Módulo ESP8266-01	10.81 € / 5uds
• Pantalla táctil LCD para Raspberry Pi (incluye carcasa + ventilación)	28.99 €
• Placas PCB (distintos tamaños)	11.49 €/ 30 uds
• Soldador, puntas, estaño y cables	11.99 €
• Transistores	12.00 € / 8 uds
• Transformador 220 AC – 12 DC	14.99 €
• Adaptador FTDI / SSL por USB	7.60 € / 3 uds
• Convertidores buck 12 V - 3.3V	3.72 € / 10 uds
• Convertidores buck 12V – 5V	5.23 € / 5 uds

6. PLANOS

El montaje del proyecto es muy simple: solamente es necesario conectar el Arduino con el módulo ESP8266 de la siguiente manera:



7. ANEXOS

7.1 PROGRAMAS

7.1.1. Módulo ESP

Aquí se muestra el código programado en el módulo ESP8266-01. En este ejemplo, los temas de MQTT son raspihouse/espin/2 para la subscripción (identificando este módulo como el módulo 2 dentro de la red) y raspihouse/espout/2 para la publicación de mensajes.

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

const bool DEBUG = false;

// Wifi configuration
const char* ssid = "RaspiHouse";
const char* password = "MyIoTProject";
const char* mqtt_server = "192.168.4.1";

// MQTT Configuration
const String clientId = "ESP2";
const char* inTopic = "raspihouse/espin/2";
const char* outTopic = "raspihouse/espout/2";

// Client Objects
WiFiClient espClient;
PubSubClient client(espClient);

//Message buffers
#define MSG_BUFFER_SIZE  (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;

void setup_wifi() {
  delay(10);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
}}

void callback(char* topic, byte* payload, unsigned int length) {
  if(DEBUG){
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
  }
  payload[length] = '\0';
  Serial.println((char*)payload);
  payload = '\0';}

void reconnect() {
  // Loop until we're reconnected
```

```
  while (!client.connected()) {
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      client.publish(outTopic, "Connected");
      if(DEBUG){
        Serial.println("Connected");
      }
      client.subscribe(inTopic);
    }
    else {
      delay(5000);
 }}}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  if(DEBUG){
    Serial.print("Setup coplete");
  }
}

unsigned long lastBlink = millis();
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  if (Serial.available()){
    String str = Serial.readString();
    int len = str.length();
    char payload[len];
    str.toCharArray(payload, len);
    client.publish(outTopic, payload);
  }

  // Blink the LED every 3 seconds
  if (millis() - lastBlink > 1500) {
    lastBlink = millis();
    if(DEBUG){
      Serial.print("...");
    }
    digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
}}
```

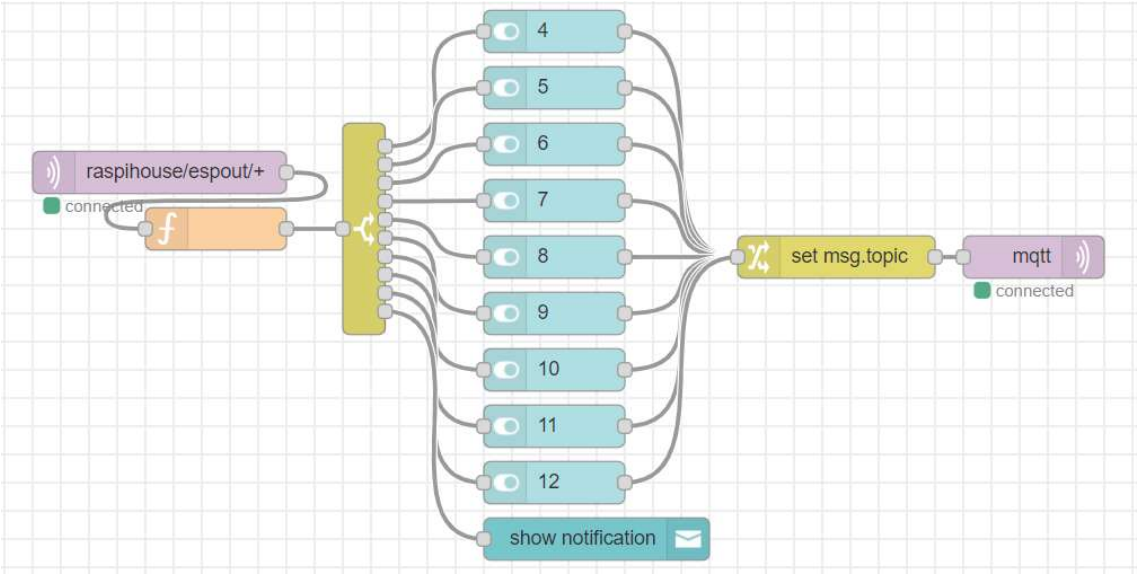
7.1.2. Arduino Nano

En el siguiente código se ha programado el Arduino Nano para configurar sus salidas digitales como controladas mediante MQTT a través del módulo ESP8266, y además, empleando dos de sus puertos analógicos, se han conectado una fotorresistencia y una lámpara LED. Cuando la fotorresistencia detecte valores bajos de luz ambiente, el LED se encenderá hasta que la fotorresistencia vuelva a detectar luz durante más de 5 segundos. Cada vez que el LED se enciende o se apaga, el cambio se notifica por MQTT para poder reflejarlo en Node-Red.

```
#include <SoftwareSerial.h>

SoftwareSerial esp(2, 3); // RX, TX

unsigned long lastBlink = 0;
String state;
int lamp_timer;

void setup() {
  digitalWrite(LED_BUILTIN, HIGH);
  for(int i = 4; i < 12; i++){
    pinMode(i, OUTPUT);
  }
  Serial.begin(9600);
  esp.begin(9600);
  digitalWrite(LED_BUILTIN, LOW);
}

void loop (){
  if (esp.available()){
    int pin = esp.parseInt();
    if(pin < 13 && pin > 1){
      digitalWrite(pin, !digitalRead(pin));
      state = digitalRead(pin) == HIGH ? " ON" : " OFF";
      Serial.println(String(pin) + state);
      esp.println(String(pin) + state);
    }
  }
  // Blink the LED every 3 seconds
  if (millis() - lastBlink > 1500) {
    lastBlink = millis();
    digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
  }

  // Insert custon code below
  // For example, make synchronous calls to read a sensor

  // Photorresistor connected to pin A0 triggers a lamp controlled by pin A1
  if(digitalRead(A0) < 30 && digitalRead(A2) == LOW){
    lamp_timer = millis();
    digitalWrite(A2, HIGH);
    Serial.println("Lamp ON");
    esp.println("Lamp ON");
  }
  if(millis() - lamp_timer > 5000 && digitalRead(A2) >30 && digitalRead(A0) == HIGH){
```

```
    digitalWrite(A2, LOW);
    Serial.println("Lamp OFF");
    esp.println("Lamp OFF");
  }
}
```

7.1.3. Node-Red



El código de Node-Red se almacena en format JSON (JavaScript Object Notation). Se incluye a continuación:

[{"id":"dc679b91.be9ab8","type":"tab","label":"Flow 1","disabled":false,"info":""},{"id":"85f9a98d.4abfa8","type":"mqtt out","z":"dc679b91.be9ab8","name":"","topic":"","qos":"","retain":"","broker":"ba42d1a.b19673","x":750,"y":300,"wires":[]},{"id":"8530674d.835968","type":"mqtt in","z":"dc679b91.be9ab8","name":"","topic":"raspihouse/espout/+","qos":"2","datatype":"auto","broker":"ba42d1a.b19673","x":130,"y":240,"wires":[["66d47305.b5869c"]]},{"id":"8c6ffe13.f36bf","type":"change","z":"dc679b91.be9ab8","name":"","rules":[{"t":"set","p":"topic","pt":"msg","to":"raspihouse/espin/2","tot":"str"}],"action":"","property":"","from":"","to":"","reg":false,"x":610,"y":300,"wires":[["85f9a98d.4abfa8"]]},{"id":"81704e22.e1187","type":"ui_switch","z":"dc679b91.be9ab8","name":"","label":"4","tooltip":"","group":"c5fd532c.51f91","order":3,"width":0,"height":0,"passthru":false,"decouple":"false","topic":"","style":"","onvalue":"4","onvalueType":"num","onicon":"","oncolor":"","offvalue":"4","offvalueType":"num","officon":"","offcolor":"","x":410,"y":140,"wires":[["8c6ffe13.f36bf"]]},{"id":"c592b061.d3c2f","type":"ui_switch","z":"dc679b91.be9ab8","name":"","label":"5","tooltip":"","group":"c5fd532c.51f91","order":3,"width":0,"height":0,"passthru":false,"decouple":"false","topic":"","style":"","onvalue":"5","onvalueType":"num","onicon":"","oncolor":"","offvalue":"5","offvalueType":"num","officon":"","offcolor":"","x":410,"y":180,"wires":[["8c6ffe13.f36bf"]]},{"id":"82196b2.7506698","type":"ui_switch","z":"dc679b91.be9ab8","name":"","label":"6","tooltip":"","group":"c5fd532c.51f91","order":3,"width":0,"height":0,"passthru":false,"decouple":"false","topic":"","style":"","onvalue":"6","onvalueType":"num","onicon":"","oncolor":"","offvalue":"6","offvalueType":"num","officon":"","offcolor":"","x":410,"y":220,"wires":[["8c6ffe13.f36bf"]]},{"id":"a337cedd.65e5d","type":"ui_switch","z":"dc679b91.be9ab8","name":"","label":"7","tooltip":"","group":"c5fd532c.51f91","order":3,"width":0,"height":0,"passthru":false,"decouple":"false","topic":"","style":"","onvalue":"7","onvalueType":"num","onicon":"","oncolor":"","offvalue":"7","offvalueType":"num","officon":"","offcolor":"","x":410,"y":260,"wires":[["8c6ffe13.f36bf"]]},{"id":"3eeb1

d2c.bb36a2","type":"ui_switch","z":"dc679b91.be9ab8","name":"","label":"8","tooltip":"","gro
up":"c5fd532c.51f91","order":3,"width":0,"height":0,"passthru":false,"decouple":"false","topic"
:"","style":"","onvalue":"8","onvalueType":"num","onicon":"","oncolor":"","offvalue":"8","offv
alueType":"num","officon":"","offcolor":"","x":410,"y":300,"wires":[["8c6ffe13.f36bf"]]},{"id"
:"3d0ea542.6761fa","type":"ui switch","z":"dc679b91.be9ab8","name":"","label":"9","tooltip":
"","group":"c5fd532c.51f91","order":3,"width":0,"height":0,"passthru":false,"decouple":"false",
"topic":"","style":"","onvalue":"9","onvalueType":"num","onicon":"","oncolor":"","offvalue":"9
","offvalueType":"num","officon":"","offcolor":"","x":410,"y":340,"wires":[["8c6ffe13.f36bf"]]
},{"id":"3d4a207e.21e9b","type":"ui switch","z":"dc679b91.be9ab8","name":"","label":"10","t
ooltip":"","group":"c5fd532c.51f91","order":3,"width":0,"height":0,"passthru":false,"decouple":
"false","topic":"","style":"","onvalue":"10","onvalueType":"num","onicon":"","oncolor":"","off
value":"10","offvalueType":"num","officon":"","offcolor":"","x":410,"y":380,"wires":[["8c6ffe
13.f36bf"]]},{"id":"6c1224d9.41316c","type":"ui_switch","z":"dc679b91.be9ab8","name":"","l
abel":"11","tooltip":"","group":"c5fd532c.51f91","order":3,"width":0,"height":0,"passthru":fals
e,"decouple":"false","topic":"","style":"","onvalue":"11","onvalueType":"num","onicon":"","on
color":"","offvalue":"11","offvalueType":"num","officon":"","offcolor":"","x":410,"y":420,"wir
es":[["8c6ffe13.f36bf"]]},{"id":"58509412.b91f6c","type":"ui_switch","z":"dc679b91.be9ab8",
"name":"","label":"12","tooltip":"","group":"c5fd532c.51f91","order":3,"width":0,"height":0,"p
assthru":false,"decouple":"false","topic":"","style":"","onvalue":"12","onvalueType":"num","on
icon":"","oncolor":"","offvalue":"12","offvalueType":"num","officon":"","offcolor":"","x":410,
"y":460,"wires":[["8c6ffe13.f36bf"]]},{"id":"66d47305.b5869c","type":"function","z":"dc679b
91.be9ab8","name":"","func":"board = msg.topic.split('/')[2];\npayload =
msg.payload.split('\\n');\n\nfor(var i in payload){\n   noti = msg.payload.split(' ')\n   msg.pin =
noti[0];\n   \n   if(noti[1] == \"ON\"){msg.payload = true}\n   else if(noti[1] ==
\"OFF\"){msg.payload = false}\n   else{return null}\n   \n   \n   if(noti[0] == 'Lamp'){\n
msg.payload = \"Lamp is \" + noti [1];\n   }\n
node.send(msg);\n}","outputs":1,"noerr":0,"x":170,"y":280,"wires":[["355f27a3.3e1298"]]},{"i
d":"355f27a3.3e1298","type":"switch","z":"dc679b91.be9ab8","name":"","property":"pin","pro
pertyType":"msg","rules":[{"t":"eq","v":"4","vt":"str"},{"t":"eq","v":"5","vt":"str"},{"t":"eq","v
":"6","vt":"str"},{"t":"eq","v":"7","vt":"str"},{"t":"eq","v":"8","vt":"str"},{"t":"eq","v":"9","vt":
"str"},{"t":"eq","v":"10","vt":"str"},{"t":"eq","v":"11","vt":"str"},{"t":"eq","v":"12","vt":"str"},
{"t":"eq","v":"Lamp","vt":"str"}],"checkall":"true","repair":false,"outputs":10,"x":275,"y":280,"
wires":[["81704e22.e1187"],["c592b061.d3c2f"],["82196b2.7506698"],["a337cedd.65e5d"],["3
eeb1d2c.bb36a2"],["3d0ea542.6761fa"],["3d4a207e.21e9b"],["6c1224d9.41316c"],["58509412.
b91f6c"],["1799bcd5.62f6d3"]],"l":false},{"id":"1799bcd5.62f6d3","type":"ui_toast","z":"dc67
9b91.be9ab8","position":"bottom
right","displayTime":"10","highlight":"","sendall":true,"outputs":0,"ok":"OK","cancel":"","raw"
:false,"topic":"","name":"","x":440,"y":500,"wires":[]},{"id":"ba42d1a.b19673","type":"mqtt-
broker","z":"","name":"localhost","broker":"localhost","port":"1883","clientid":"nodered","usetl
s":false,"compatmode":false,"keepalive":"60","cleansession":true,"birthTopic":"","birthQos":"0
","birthPayload":"","closeTopic":"","closeQos":"0","closePayload":"","willTopic":"","willQos":
"0","willPayload":""},{"id":"c5fd532c.51f91","type":"ui_group","z":"","name":"Default","tab":
"404f7dbe.807134","order":1,"disp":true,"width":"6","collapse":false},{"id":"404f7dbe.807134
","type":"ui_tab","z":"","name":"Home","icon":"dashboard","disabled":false,"hidden":false}]

7.2 ALTERNATIVA AL PUNTO DE ACCESO EN RASPBERRY PI

La red Wifi generada por la Raspberry Pi no presenta un alcance muy grande. Si se desea desplegar este proyecto a distancias mayores que una habitación, es necesario configurar la raspberry para que se conecte a la red Wifi doméstica con una dirección IP estática. Para ello, la manera más sencilla es conectando la Raspberry a un monitor, ratón y teclado. Una vez encendida, en la parte superior del escritorio se encuentran los ajustes de red. Simplemente es necesario seleccionar la interfaz que se esté empleando para la conexión wifi (wlan0) o cableada (eth0) y

en el campo de dirección IP introducir una IP válida y libre de la red 192.168.0.0/16 (Por ejemplo 192.168.1.123).

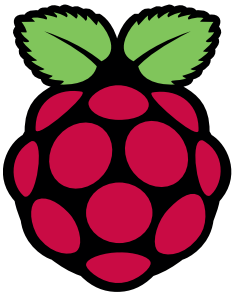
Hecho esto y habiendo comprobado que, tras un reinicio, la Raspberry efectúa correctamente la conexión a Internet, es necesario modificar el programa a introducir en el módulo ESP8266 de la forma que se indica a continuación:

PUNTO DE ACCESO	RED WIFI DOMÉSTICA
// Wifi configuration	// Wifi configuration
const char* ssid = "RaspiHouse";	const char* ssid = “<SSID>”;
const char* password = "MyIoTProject";	const char* password = “<Password>”;
const char* mqtt_server = "192.168.4.1";	const char* mqtt_server = "192.168.1.123";

7.3 HOJAS DE DATOS TÉCNICOS

A continuación se adjuntan las hojas de datos de la Raspberry Pi 4 Modelo B, Arduino Nano V3.0 y ESP 8266-01

DATASHEET



Raspberry Pi 4 Model B

Release 1

June 2019

Table 1: Release History

Release	Date	Description
1	21/06/2019	First release

The latest release of this document can be found at `https://www.raspberrypi.org`

Contents

List of Figures

List of Tables

1 Introduction

The Raspberry Pi 4 Model B (Pi4B) is the first of a new generation of Raspberry Pi computers supporting more RAM and with siginficantly enhanced CPU, GPU and I/O performance; all within a similar form factor, power envelope and cost as the previous generation Raspberry Pi 3B+.

The Pi4B is avaiable with either 1, 2 and 4 Gigabytes of LPDDR4 SDRAM.

2 Features

2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
  - Up to 6x UART
  - Up to 6x I2C
  - Up to 5x SPI
  - 1x SDIO interface
  - 1x DPI (Parallel RGB Display)
  - 1x PCM
  - Up to 2x PWM channels
  - Up to 3x GPCLK outputs

2.3 Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained
  - Recent Linux kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Availability of GPU functions using standard APIs

3 Mechanical Specification

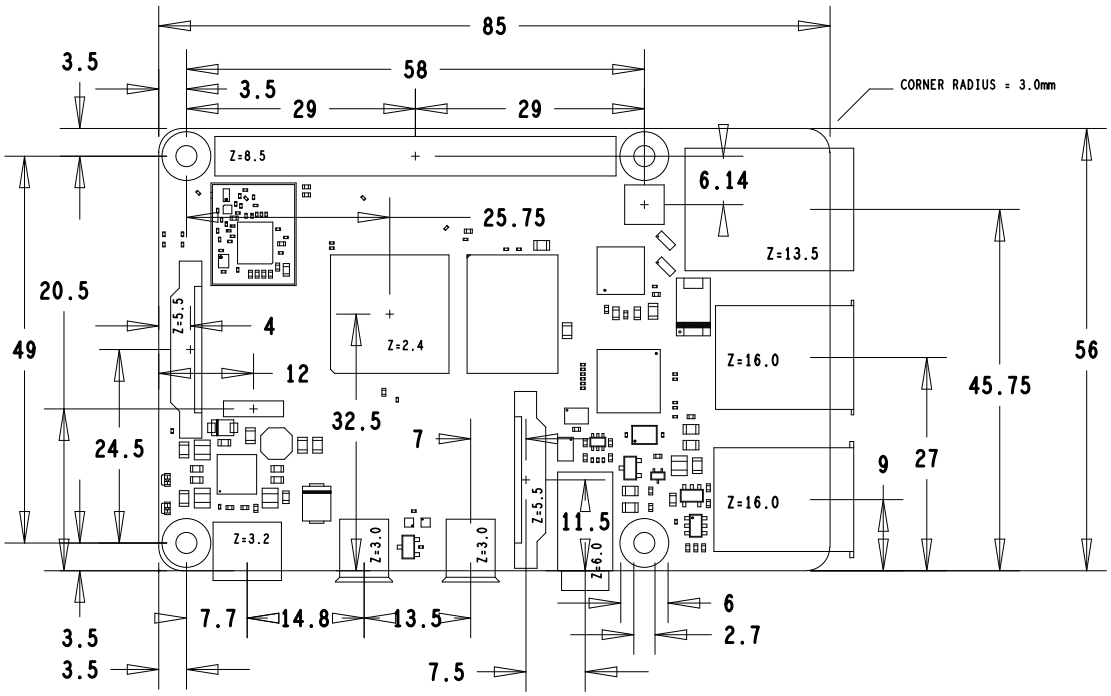


Figure 1: Mechanical Dimensions

4 Electrical Specification

Caution! Stresses above those listed in Table 2 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD_IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 3.3V	-	-	TBD	V
V_{IH}	Input high voltage[a]	VDD_IO = 3.3V	TBD	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	TBD	μA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 3.3V, IOL = -2mA	-	-	TBD	V
V_{OH}	Output high voltage[b]	VDD_IO = 3.3V, IOH = 2mA	TBD	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 3.3V, VO = 0.4V	TBD	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 3.3V, VO = 2.3V	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	kΩ
R_{PD}	Pulldown resistor	-	TBD	-	TBD	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	TBD	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	TBD	-	ns

[a] Default drive strength, CL = 5pF, VDD_IO = 3.3V

Table 4: Digital I/O Pin AC Characteristics

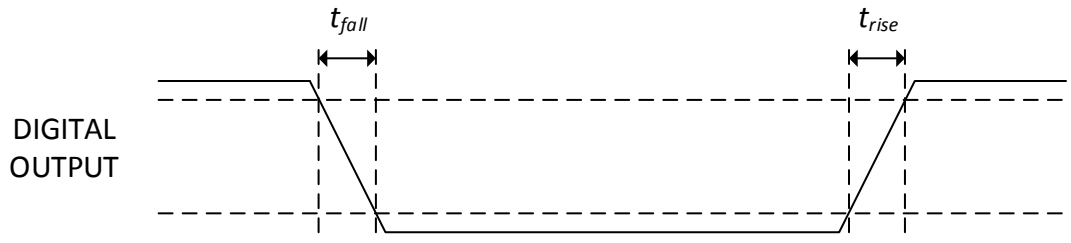


Figure 2: Digital IO Characteristics

4.1 Power Requirements

The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

5 Peripherals

5.1 GPIO Interface

The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

5.1.1 GPIO Pin Assignments

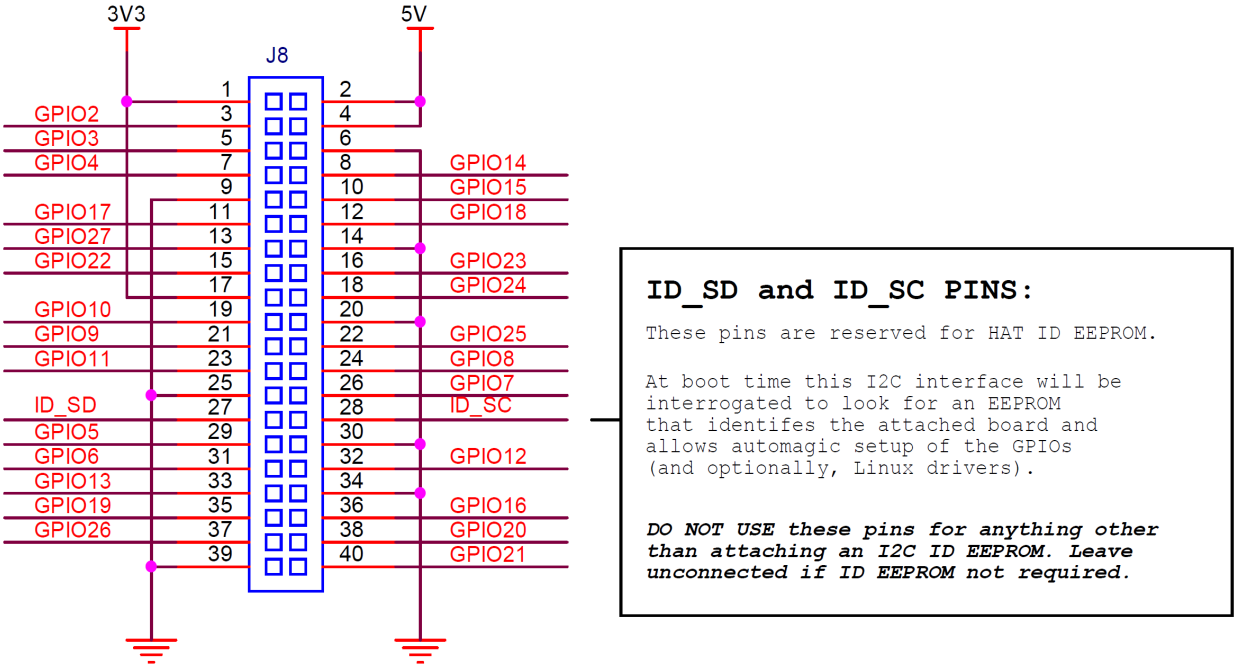


Figure 3: GPIO Connector Pinout

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI.

In addition to the standard peripheral options found on legacy Pis, extra I2C, UART and SPI peripherals have been added to the BCM2711 chip and are available as further mux options on the Pi4. This gives users much more flexibility when attaching add-on hardware as compared to older models.

5.1.2 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPI_D2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPI_D4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPI_D5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPI_D6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPI_D7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

Table 5 details the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the BCM2711 Peripherals Specification document which can be downloaded from the hardware documentation section of the website.

5.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.

5.1.4 SD/SDIO Interface

The Pi4B has a dedicated SD card socket which suports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins.

5.2 Camera and Display Interfaces

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

5.3 USB

The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

5.4 HDMI

The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

5.5 Audio and Composite (TV Out)

The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack.

The analog audio output can drive 32 Ohm headphones directly.

5.6 Temperature Range and Thermals

The recommended ambient operating temperature range is 0 to 50 degrees Celcius.

To reduce thermal output when idling or under light load, the Pi4B reduces the CPU clock speed and voltage. During heavier load the speed and voltage (and hence thermal output) are increased. The internal governor will throttle back both the CPU speed and voltage to make sure the CPU temperature never exceeds 85 degrees C.

The Pi4B will operate perfectly well without any extra cooling and is designed for sprint performance - expecting a light use case on average and ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high termperature at full performance, further cooling may be needed.

6 Availability

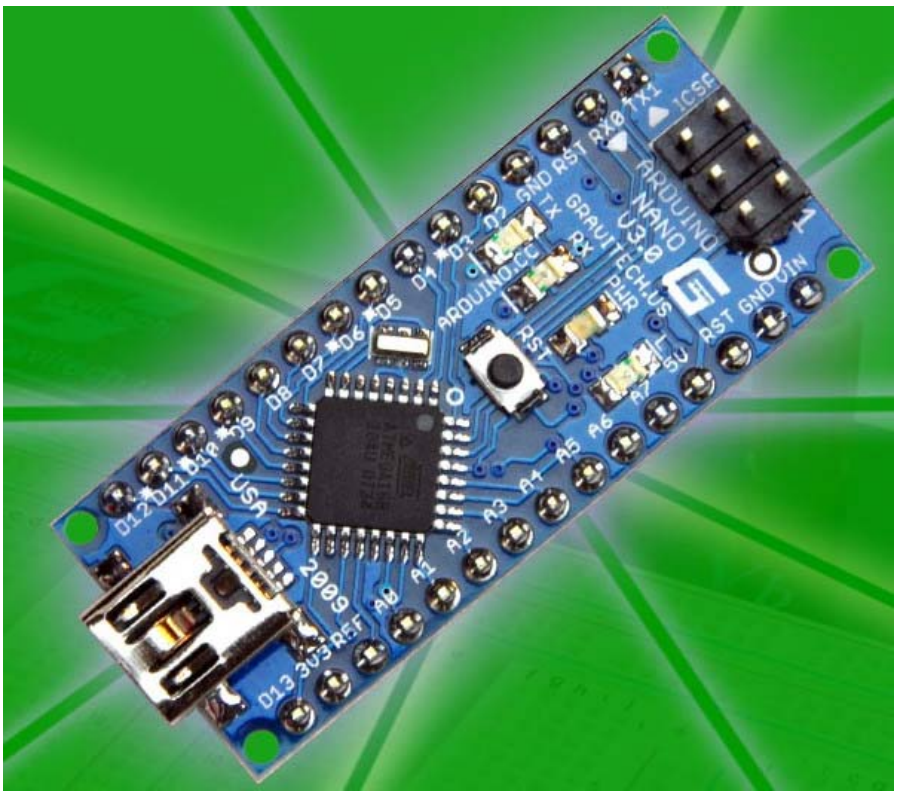
Raspberry Pi guarantee availability Pi4B until at least January 2026.

7 Support

For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.

Arduino Nano (V3.0)

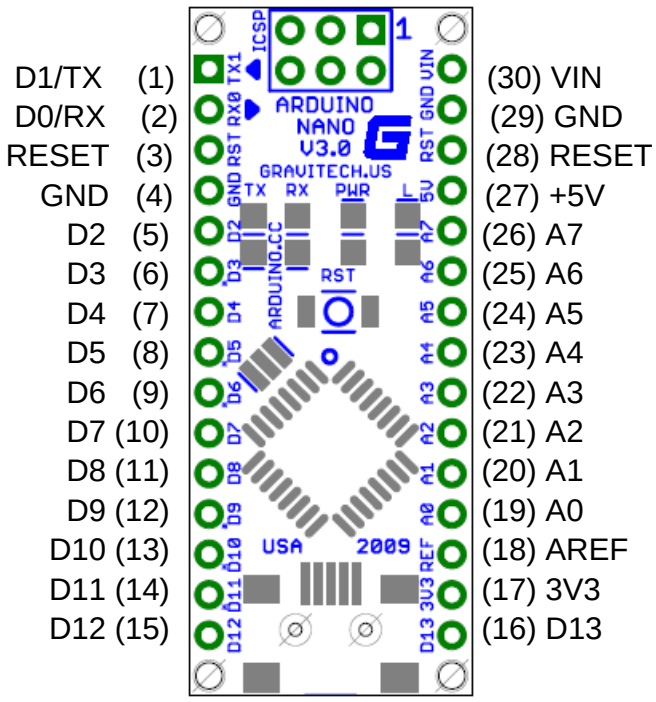
User Manual



More information:

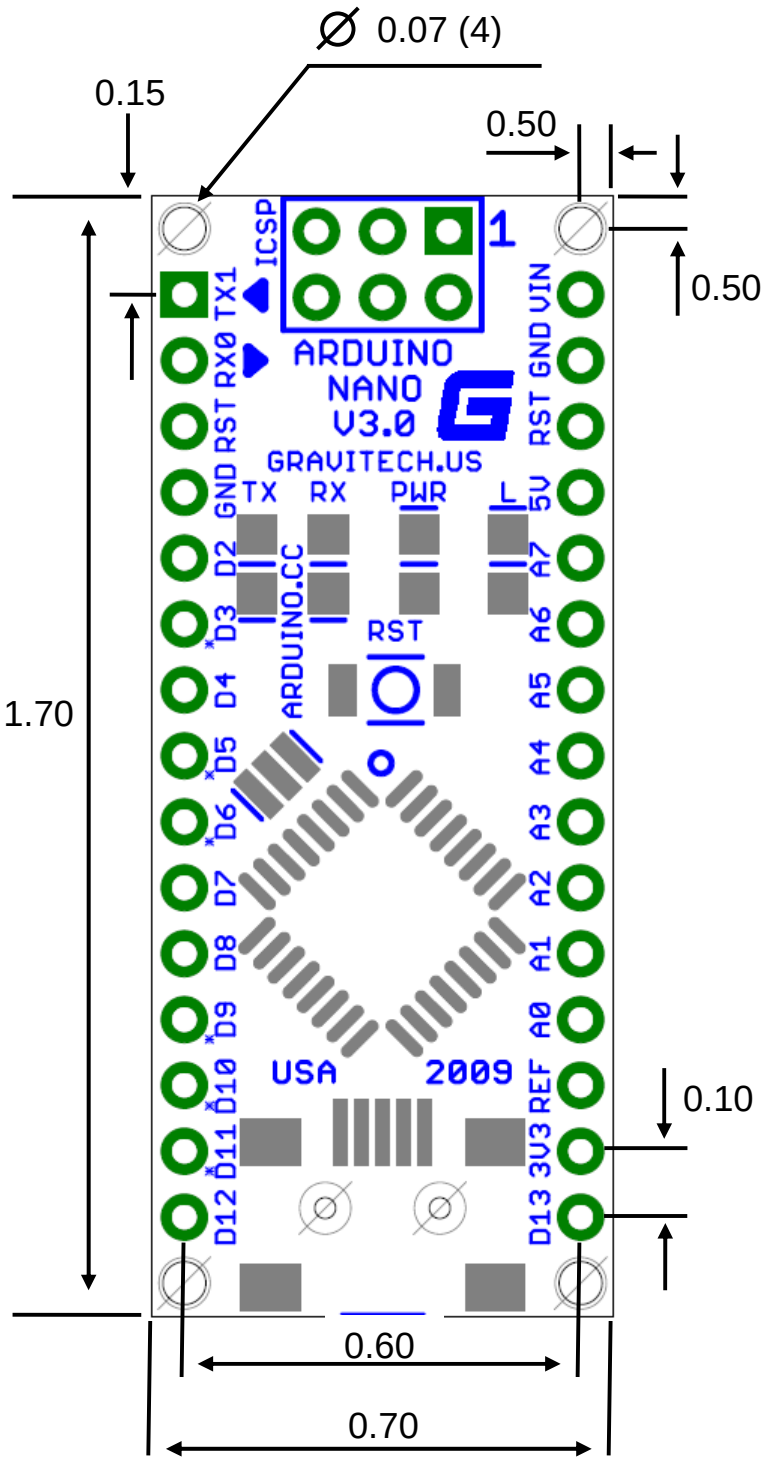
www.arduino.cc Rev 3.0

Arduino Nano Pin Layout



Pin No.	Name	Type	Description
1-2, 5-16	D0-D13	I/O	Digital input/output port 0 to 13
3, 28	RESET	Input	Reset (active low)
4, 29	GND	PWR	Supply ground
17	3V3	Output	+3.3V output (from FTDI)
18	AREF	Input	ADC reference
19-26	A0-A7	Input	Analog input channel 0 to 7
27	+5V	Output or Input	+5V output (from on-board regulator) or +5V (input from external power supply)
30	VIN	PWR	Supply voltage

Arduino Nano Mechanical Drawing



ESP8266EX

Datasheet

Version 6.5
Espressif Systems

About This Guide

This document introduces the specifications of ESP8266EX.

Release Notes

Date	Version	Release Notes
2015.12	V4.6	Updated Chapter 3.
2016.02	V4.7	Updated Section 3.6 and Section 4.1.
2016.04	V4.8	Updated Chapter 1.
2016.08	V4.9	Updated Chapter 1.
2016.11	V5.0	Added Appendix II “Learning Resources”.
2016.11	V5.1	Changed the power consumption during Deep-sleep from 10 μA to 20 μA in Table 5-2.
2016.11	V5.2	Changed the crystal frequency range from “26 MHz to 52 MHz” to “24 MHz to 52 MHz” in Section 3.3.
2016.12	V5.3	Changed the minimum working voltage from 3.0 V to 2.5 V.
2017.04	V5.4	Changed chip input and output impedance from 50 Ω to 39 + j6 Ω.
2017.10	V5.5	Updated Chapter 3 regarding the range of clock amplitude to 0.8 V ~ 1.5 V.
2017.11	V5.6	Updated VDDPST from 1.8 V ~ 3.3 V to 1.8 V ~ 3.6 V.
2017.11	V5.7	• Corrected a typo in the description of SDIO_DATA_0 in Table 2-1; • Added the testing conditions for the data in Table 5-2.
2018.02	V5.8	• Updated Wi-Fi protocols in Section 1.1; • Updated description of the integrated Tensilica processor in 3.1.

Date	Version	Release Notes
2018.09	V5.9	• Update document cover; • Added a note for Table 1-1; • Updated Wi-Fi key features in Section 1.1; • Updated description of the Wi-Fi function in 3.5; • Updated pin layout diagram; • Fixed a typo in Table 2-1; • Removed Section AHB and AHB module; • Restructured Section Power Management; • Fixed a typo in Section UART; • Removed description of transmission angle in Section IR Remote Control; • Other optimization (wording).
2018.11	V6.0	• Added an SPI pin in Table 4-2; • Updated the diagram of packing information.
2019.08	V6.1	Removed description of the GPIO function in Section 4.1.
2019.08	V6.2	Updated notes on CHIP_EN in Section 5.1
2019.12	V6.3	Add feedback links.
2020.04	V6.4	• Removed the description of “Antenna diversity”; • Updated the feedback links.
2020.07	V6.5	• Updated the description of HSPI in Section 4.3; • Updated links in Appendix.

Documentation Change Notification

Espressif provides email notifications to keep customers updated on changes to technical documentation. Please subscribe at https://www.espressif.com/en/subscribe.

Certification

Download certificates for Espressif products from https://www.espressif.com/en/certificates.

Table of Contents

1. Overview

Espressif's ESP8266EX delivers highly integrated Wi-Fi SoC solution to meet users' continuous demands for efficient power usage, compact design and reliable performance in the Internet of Things industry.

With the complete and self-contained Wi-Fi networking capabilities, ESP8266EX can perform either as a standalone application or as the slave to a host MCU. When ESP8266EX hosts the application, it promptly boots up from the flash. The integrated high-speed cache helps to increase the system performance and optimize the system memory. Also, ESP8266EX can be applied to any microcontroller design as a Wi-Fi adaptor through SPI/SDIO or UART interfaces.

ESP8266EX integrates antenna switches, RF balun, power amplifier, low noise receive amplifier, filters and power management modules. The compact design minimizes the PCB size and requires minimal external circuitries.

Besides the Wi-Fi functionalities, ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor and on-chip SRAM. It can be interfaced with external sensors and other devices through the GPIOs. Software Development Kit (SDK) provides sample codes for various applications.

Espressif Systems' Smart Connectivity Platform (ESCP) enables sophisticated features including:

- Fast switch between sleep and wakeup mode for energy-efficient purpose;
- Adaptive radio biasing for low-power operation
- Advance signal processing
- Spur cancellation and RF co-existence mechanisms for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation

1.1. Wi-Fi Key Features

- 802.11 b/g/n support
- 802.11 n support (2.4 GHz), up to 72.2 Mbps
- Defragmentation
- 2 x virtual Wi-Fi interface
- Automatic beacon monitoring (hardware TSF)
- Support Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode

1.2. Specifications

Table 1-1. Specifications

Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4 GHz ~ 2.5 GHz (2400 MHz ~ 2483.5 MHz)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: –91 dbm (11 Mbps)
		802.11 g: –75 dbm (54 Mbps)
		802.11 n: –72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5 V ~ 3.6 V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	–40 °C ~ 125 °C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

📖 ***Note:***

The TX power can be configured based on the actual user scenarios.

1.3. Applications

- Home appliances
- Home automation
- Smart plugs and lights
- Industrial wireless control
- Baby monitors
- IP cameras
- Sensor networks
- Wearable electronics
- Wi-Fi location-aware devices
- Security ID tags
- Wi-Fi position system beacons

2. Pin Definitions

Figure 2-1 shows the pin layout for 32-pin QFN package.

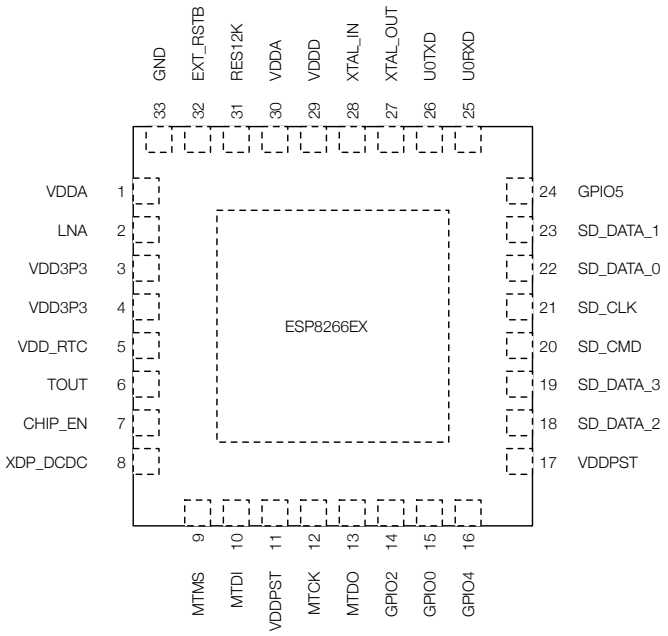


Figure 2-1. Pin Layout (Top View)

Table 2-1 lists the definitions and functions of each pin.

Table 2-1. ESP8266EX Pin Definitions

Pin	Name	Type	Function
1	VDDA	P	Analog Power 2.5 V ~ 3.6 V
2	LNA	I/O	RF antenna interface Chip output impedance = 39 + j6 Ω. It is suggested to retain the π-type matching network to match the antenna.
3	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V
4	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V
5	VDD_RTC	P	NC (1.1 V)
6	TOUT	I	ADC pin. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously.
7	CHIP_EN	I	Chip Enable High: On, chip works properly Low: Off, small current consumed

Pin	Name	Type	Function
8	XPD_DCDC	I/O	Deep-sleep wakeup (need to be connected to EXT_RSTB); GPIO16
9	MTMS	I/O	GPIO 14; HSPI_CLK
10	MTDI	I/O	GPIO 12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
12	MTCK	I/O	GPIO 13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO 15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART TX during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 20 Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200 Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200 Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200 Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200 Ω); SPI_MISO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200 Ω); SPI_MOSI; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART Rx during flash programming; GPIO3
26	U0TXD	I/O	UART TX during flash programming; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 2.5 V ~ 3.6 V
30	VDDA	P	Analog Power 2.5 V ~ 3.6 V
31	RES12K	I	Serial connection with a 12 kΩ resistor and connect to the ground
32	EXT_RSTB	I	External reset signal (Low voltage level: active)

📖 ***Note:***

1. *GPIO2, GPIO0, and MTDO are used to select booting mode and the SDIO mode;*
2. *U0TXD should not be pulled externally to a low logic level during the powering-up.*

3. Functional Description

The functional diagram of ESP8266EX is shown as in Figure 3-1.

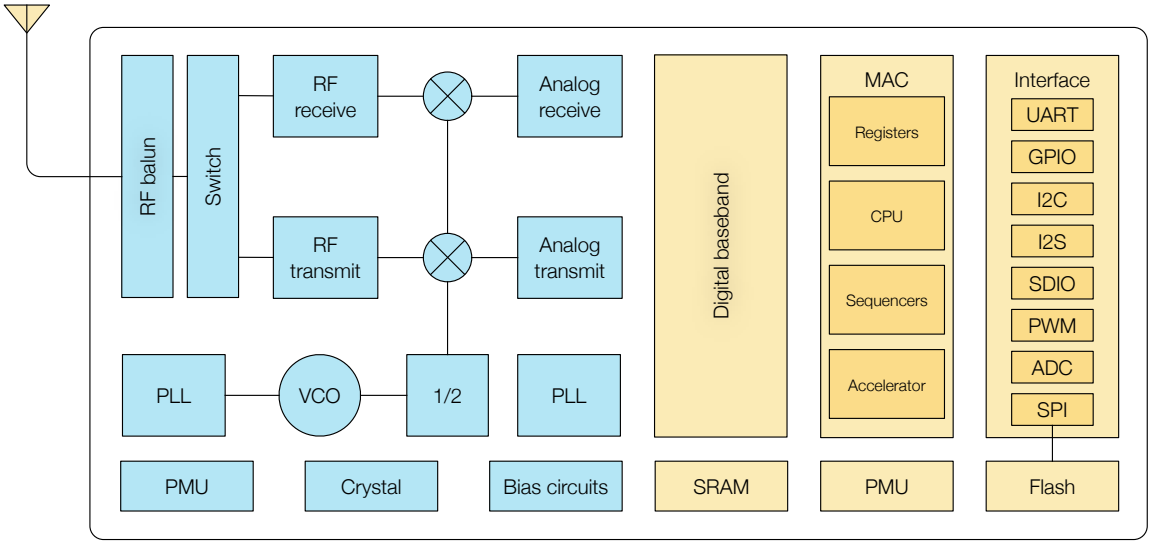


Figure 3-1. Functional Block Diagram

3.1. CPU, Memory, and Flash

3.1.1. CPU

The ESP8266EX integrates a Tensilica L106 32-bit RISC processor, which achieves extra-low power consumption and reaches a maximum clock speed of 160 MHz. The Real-Time Operating System (RTOS) and Wi-Fi stack allow 80% of the processing power to be available for user application programming and development. The CPU includes the interfaces as below:

- Programmable RAM/ROM interfaces (iBus), which can be connected with memory controller, and can also be used to visit flash.
- Data RAM interface (dBus), which can connected with memory controller.
- AHB interface which can be used to visit the register.

3.1.2. Memory

ESP8266EX Wi-Fi SoC integrates memory controller and memory units including SRAM and ROM. MCU can access the memory units through iBus, dBus, and AHB interfaces. All memory units can be accessed upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK, SRAM space available to users is assigned as below.

- RAM size < 50 kB, that is, when ESP8266EX is working under the Station mode and connects to the router, the maximum programmable space accessible in Heap + Data section is around 50 kB.
- There is no programmable ROM in the SoC. Therefore, user program must be stored in an external SPI flash.

3.1.3. External Flash

ESP8266EX uses external SPI flash to store user programs, and supports up to 16 MB memory capacity theoretically.

The minimum flash memory of ESP8266EX is shown below:

- OTA disabled: 512 kB at least
- OTA enabled: 1 MB at least

> ⚠ **Notice:**
>
> *SPI mode supported: Standard SPI, Dual SPI and Quad SPI. The correct SPI mode should be selected when flashing bin files to ESP8266. Otherwise, the downloaded firmware/program may not be working properly.*

3.2. Clock

3.2.1. High Frequency Clock

The high frequency clock on ESP8266EX is used to drive both transmit and receive mixers. This clock is generated from internal crystal oscillator and external crystal. The crystal frequency ranges from 24 MHz to 52 MHz.

The internal calibration inside the crystal oscillator ensures that a wide range of crystals can be used, nevertheless the quality of the crystal is still a factor to consider to have reasonable phase noise and good Wi-Fi sensitivity. Refer to Table 3-1 to measure the frequency offset.

Table 3-1. High Frequency Clock Specifications

Parameter	Symbol	Min	Max	Unit
Frequency	FXO	24	52	MHz
Loading capacitance	CL	-	32	pF
Motional capacitance	CM	2	5	pF
Series resistance	RS	0	65	Ω

Parameter	Symbol	Min	Max	Unit
Frequency tolerance	ΔFXO	–15	15	ppm
Frequency vs temperature (–25 °C ~ 75 °C)	ΔFXO,Temp	–15	15	ppm

3.2.2. External Clock Requirements

An externally generated clock is available with the frequency ranging from 24 MHz to 52 MHz. The following characteristics are expected to achieve good performance of radio.

Table 3-2. External Clock Reference

Parameter	Symbol	Min	Max	Unit
Clock amplitude	VXO	0.8	1.5	Vpp
External clock accuracy	ΔFXO,EXT	–15	15	ppm
Phase noise @1-kHz offset, 40-MHz clock	-	-	–120	dBc/Hz
Phase noise @10-kHz offset, 40-MHz clock	-	-	–130	dBc/Hz
Phase noise @100-kHz offset, 40-MHz clock	-	-	–138	dBc/Hz

3.3. Radio

ESP8266EX radio consists of the following blocks.

- 2.4 GHz receiver
- 2.4 GHz transmitter
- High speed clock generators and crystal oscillator
- Bias and regulators
- Power management

3.3.1. Channel Frequencies

The RF transceiver supports the following channels according to IEEE802.11 b/g/n standards.

Table 3-3. Frequency Channel

Channel No.	Frequency (MHz)	Channel No.	Frequency (MHz)
1	2412	8	2447
2	2417	9	2452
3	2422	10	2457
4	2427	11	2462

Channel No.	Frequency (MHz)	Channel No.	Frequency (MHz)
5	2432	12	2467
6	2437	13	2472
7	2442	14	2484

3.3.2. 2.4 GHz Receiver

The 2.4 GHz receiver down-converts the RF signals to quadrature baseband signals and converts them to the digital domain with 2 high resolution high speed ADCs. To adapt to varying signal channel conditions, RF filters, automatic gain control (AGC), DC offset cancelation circuits and baseband filters are integrated within ESP8266EX.

3.3.3. 2.4 GHz Transmitter

The 2.4 GHz transmitter up-converts the quadrature baseband signals to 2.4 GHz, and drives the antenna with a high-power CMOS power amplifier. The function of digital calibration further improves the linearity of the power amplifier, enabling a state of art performance of delivering +19.5 dBm average TX power for 802.11b transmission and +18 dBm for 802.11n (MSC0) transmission.

Additional calibrations are integrated to offset any imperfections of the radio, such as:

- Carrier leakage
- I/Q phase matching
- Baseband nonlinearities

These built-in calibration functions reduce the product test time and make the test equipment unnecessary.

3.3.4. Clock Generator

The clock generator generates quadrature 2.4 GHz clock signals for the receiver and transmitter. All components of the clock generator are integrated on the chip, including all inductors, varactors, loop filters, linear voltage regulators and dividers.

The clock generator has built-in calibration and self test circuits. Quadrature clock phases and phase noise are optimized on-chip with patented calibration algorithms to ensure the best performance of the receiver and transmitter.

3.4. Wi-Fi

ESP8266EX implements TCP/IP and full 802.11 b/g/n WLAN MAC protocol. It supports Basic Service Set (BSS) STA and SoftAP operations under the Distributed Control Function (DCF). Power management is handled with minimum host interaction to minimize active-duty period.

3.4.1. Wi-Fi Radio and Baseband

The ESP8266EX Wi-Fi Radio and Baseband support the following features:

- 802.11 b and 802.11 g
- 802.11 n MCS0-7 in 20 MHz bandwidth
- 802.11 n 0.4 µs guard-interval
- up to 72.2 Mbps of data rate
- Receiving STBC 2 x 1
- Up to 20.5 dBm of transmitting power
- Adjustable transmitting power

3.4.2. Wi-Fi MAC

The ESP8266EX Wi-Fi MAC applies low-level protocol functions automatically, as follows:

- 2 × virtual Wi-Fi interfaces
- Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode
- Request To Send (RTS), Clear To Send (CTS) and Immediate Block ACK
- Defragmentation
- CCMP (CBC-MAC, counter mode), TKIP (MIC, RC4), WEP (RC4) and CRC
- Automatic beacon monitoring (hardware TSF)
- Dual and single antenna Bluetooth co-existence support with optional simultaneous receive (Wi-Fi/Bluetooth) capability

3.5. Power Management

ESP8266EX is designed with advanced power management technologies and intended for mobile devices, wearable electronics and the Internet of Things applications.

The low-power architecture operates in the following modes:

- Active mode: The chip radio is powered on. The chip can receive, transmit, or listen.
- Modem-sleep mode: The CPU is operational. The Wi-Fi and radio are disabled.
- Light-sleep mode: The CPU and all peripherals are paused. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
- Deep-sleep mode: Only the RTC is operational and all other part of the chip are powered off.

Table 3-4. Power Consumption by Power Modes

Power Mode	Description	Power Consumption
Active (RF working)	Wi-Fi TX packet	Please refer to Table 5-2.
	Wi-Fi RX packet	
Modem-sleep①	CPU is working	15 mA
Light-sleep②	-	0.9 mA
Deep-sleep③	Only RTC is working	20 uA
Shut down	-	0.5 uA

📖 ***Notes:***

① ***Modem-sleep*** *mode is used in the applications that require the CPU to be working, as in PWM or I2S applications. According to 802.11 standards (like U-APSD), it shuts down the Wi-Fi Modem circuit while maintaining a Wi-Fi connection with no data transmission to optimize power consumption. E.g., in DTIM3, maintaining a sleep of 300 ms with a wakeup of 3 ms cycle to receive AP's Beacon packages at interval requires about 15 mA current.*

② *During* ***Light-sleep*** *mode, the CPU may be suspended in applications like Wi-Fi switch. Without data transmission, the Wi-Fi Modem circuit can be turned off and CPU suspended to save power consumption according to the 802.11 standards (U-APSD). E.g. in DTIM3, maintaining a sleep of 300 ms with a wakeup of 3ms to receive AP's Beacon packages at interval requires about 0.9 mA current.*

③ *During* ***Deep-sleep*** *mode, Wi-Fi is turned off. For applications with long time lags between data transmission, e.g. a temperature sensor that detects the temperature every 100 s, sleeps for 300 s and wakes up to connect to the AP (taking about 0.3 ~ 1 s), the overall average current is less than 1 mA. The current of 20 μA is acquired at the voltage of 2.5 V.*

4. Peripheral Interface

4.1. General Purpose Input/Output Interface (GPIO)

ESP8266EX has 17 GPIO pins which can be assigned to various functions by programming the appropriate registers.

Each GPIO PAD can be configured with internal pull-up or pull-down (XPD_DCDC can only be configured with internal pull-down, other GPIO PAD can only be configured with internal pull-up), or set to high impedance. When configured as an input, the data are stored in software registers; the input can also be set to edge-trigger or level trigger CPU interrupts. In short, the IO pads are bi-directional, non-inverting and tristate, which includes input and output buffer with tristate control inputs.

These pins, when working as GPIOs, can be multiplexed with other functions such as I2C, I2S, UART, PWM, and IR Remote Control, etc.

4.2. Secure Digital Input/Output Interface (SDIO)

ESP8266EX has one Slave SDIO, the definitions of which are described as Table 4-1, which supports 25 MHz SDIO v1.1 and 50 MHz SDIO v2.0, and 1 bit/4 bit SD mode and SPI mode.

Table 4-1. Pin Definitions of SDIOs

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SDIO_CLK
SDIO_DATA0	22	IO7	SDIO_DATA0
SDIO_DATA1	23	IO8	SDIO_DATA1
SDIO_DATA_2	18	IO9	SDIO_DATA_2
SDIO_DATA_3	19	IO10	SDIO_DATA_3
SDIO_CMD	20	IO11	SDIO_CMD

4.3. Serial Peripheral Interface (SPI/HSPI)

ESP8266EX has two SPIs.

- One general Slave/Master SPI
- One general Slave/Master HSPI

Functions of all these pins can be implemented via hardware.

4.3.1. General SPI (Master/Slave)

Table 4-2. Pin Definitions of SPIs

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SPICLK
SDIO_DATA0	22	IO7	SPIQ/MISO
SDIO_DATA1	23	IO8	SPID/MOSI
SDIO_DATA_2	18	IO9	SPIHD
SDIO_DATA_3	19	IO10	SPIWP
U0TXD	26	IO1	SPICS1
GPIO0	15	IO0	SPICS2
SDIO_CMD	20	IO11	SPICS0

📖 ***Note:***

SPI mode can be implemented via software programming. The clock frequency is 80 MHz at maximum when working as a master, 20 MHz at maximum when working as a slave.

4.3.2. HSPI (Master/Slave)

Table 4-3. Pin Definitions of HSPI

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	HSPICLK
MTDI	10	IO12	HSPIQ/MISO
MTCK	12	IO13	HSPID/MOSI
MTDO	13	IO15	HPSICS

📖 ***Note:***

SPI mode can be implemented via software programming. The clock frequency is 20 MHz at maximum.

4.4. I2C Interface

ESP8266EX has one I2C, which is realized via software programming, used to connect with other microcontrollers and other peripheral equipments such as sensors. The pin definition of I2C is as below.

Table 4-4. Pin Definitions of I2C

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	I2C_SCL
GPIO2	14	IO2	I2C_SDA

Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized via software programming, and the clock frequency is 100 kHz at maximum.

4.5. I2S Interface

ESP8266EX has one I2S data input interface and one I2S data output interface, and supports the linked list DMA. I2S interfaces are mainly used in applications such as data collection, processing, and transmission of audio data, as well as the input and output of serial data. For example, LED lights (WS2812 series) are supported. The pin definition of I2S is shown in Table 4-5.

Table 4-5. Pin Definitions of I2S

I2S Data Input			
Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	I2SI_DATA
MTCK	12	IO13	I2SI_BCK
MTMS	9	IO14	I2SI_WS
MTDO	13	IO15	I2SO_BCK
U0RXD	25	IO3	I2SO_DATA
GPIO2	14	IO2	I2SO_WS

4.6. Universal Asynchronous Receiver Transmitter (UART)

ESP8266EX has two UART interfaces UART0 and UART1, the definitions are shown in Table 4-6.

Table 4-6. Pin Definitions of UART

Pin Type	Pin Name	Pin Num	IO	Function Name
UART0	U0RXD	25	IO3	U0RXD
	U0TXD	26	IO1	U0TXD
	MTDO	13	IO15	U0RTS
	MTCK	12	IO13	U0CTS
UART1	GPIO2	14	IO2	U1TXD
	SD_D1	23	IO8	U1RXD

Data transfers to/from UART interfaces can be implemented via hardware. The data transmission speed via UART interfaces reaches 115200 x 40 (4.5 Mbps).

UART0 can be used for communication. It supports flow control. Since UART1 features only data transmit signal (TX), it is usually used for printing log.

> 📖 ***Note:***
>
> *By default, UART0 outputs some printed information when the device is powered on and booting up. The baud rate of the printed information is relevant to the frequency of the external crystal oscillator. If the frequency of the crystal oscillator is 40 MHz, then the baud rate for printing is 115200; if the frequency of the crystal oscillator is 26 MHz, then the baud rate for printing is 74880. If the printed information exerts any influence on the functionality of the device, it is suggested to block the printing during the power-on period by changing (U0TXD, U0RXD) to (MTDO, MTCK).*

4.7. Pulse-Width Modulation (PWM)

ESP8266EX has four PWM output interfaces. They can be extended by users themselves. The pin definitions of the PWM interfaces are defined as below.

Table 4-7. Pin Definitions of PWM

Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	PWM0
MTDO	13	IO15	PWM1
MTMS	9	IO14	PWM2
GPIO4	16	IO4	PWM3

The functionality of PWM interfaces can be implemented via software programming. For example, in the LED smart light demo, the function of PWM is realized by interruption of the timer, the minimum resolution reaches as high as 44 ns. PWM frequency range is adjustable from 1000 μs to 10000 μs, i.e., between 100 Hz and 1 kHz. When the PWM

frequency is 1 kHz, the duty ratio will be 1/22727, and a resolution of over 14 bits will be achieved at 1 kHz refresh rate.

4.8. IR Remote Control

ESP8266EX currently supports one infrared remote control interface. For detailed pin definitions, please see Table 4-8 below.

Table 4-8. Pin Definitions of IR Remote Control

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	IR TX
GPIO5	24	IO 5	IR Rx

The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are supported by this interface. The frequency of modulated carrier signal is 38 kHz, while the duty ratio of the square wave is 1/3. The transmission range is around 1m which is determined by two factors: one is the maximum current drive output, the other is internal current-limiting resistance value in the infrared receiver. The larger the resistance value, the lower the current, so is the power, and vice versa.

4.9. ADC (Analog-to-Digital Converter)

ESP8266EX is embedded with a 10-bit precision SAR ADC. TOUT (Pin6) is defined as below:

Table 4-9. Pin Definition of ADC

Pin Name	Pin Num	Function Name
TOUT	6	ADC Interface

The following two measurements can be implemented using ADC (Pin6). However, they cannot be implemented at the same time.

- Measure the power supply voltage of VDD3P3 (Pin3 and Pin4).

Hardware Design	TOUT must be floating.
RF Initialization Parameter	The 107th byte of ***esp_init_data_default.bin*** (0 ~ 127 bytes), `vdd33_const` must be set to `0xFF`.
RF Calibration Process	Optimize the RF circuit conditions based on the testing results of VDD3P3 (Pin3 and Pin4).
User Programming	Use `system_get_vdd33` instead of `system_adc_read`.

- Measure the input voltage of TOUT (Pin6).

Hardware Design	The input voltage range is 0 to 1.0 V when TOUT is connected to external circuit.
RF Initialization Parameter	The value of the 107th byte of ***esp_init_data_default.bin*** (0 ~ 127 bytes), `vdd33_const` must be set to the real power supply voltage of Pin3 and Pin4. The unit and effective value range of `vdd33_const` is 0.1 V and 18 to 36, respectively, thus making the working power voltage range of ESP8266EX between 1.8 V and 3.6 V.
RF Calibration Process	Optimize the RF circuit conditions based on the value of `vdd33_const`. The permissible error is ± 0.2 V.
User Programming	Use `system_adc_read` instead of `system_get_vdd33`.

📖 ***Notes:***

esp_init_data_default.bin *is provided in SDK package which contains RF initialization parameters (0 ~ 127 bytes). The name of the 107th byte in* ***esp_init_data_default.bin*** *is* `vdd33_const`, *which is defined as below:*

- *When vdd33_const = 0xff, the power voltage of Pin3 and Pin4 will be tested by the internal self-calibration process of ESP8266EX itself. RF circuit conditions should be optimized according to the testing results.*
- *When 18 = < vdd33_const = < 36, ESP8266EX RF Calibration and optimization process is implemented via (vdd33_const/10).*
- *When vdd33_const < 18 or 36 < vdd33_const < 255, vdd33_const is invalid. ESP8266EX RF Calibration and optimization process is implemented via the default value 3.3 V.*

5. Electrical Specifications

5.1. Electrical Characteristics

Table 5-1. Electrical Characteristics

Parameters		Conditions	Min	Typical	Max	Unit
Operating Temperature Range		-	–40	Normal	125	°C
Maximum Soldering Temperature		IPC/JEDEC J-STD-020	-	-	260	°C
Working Voltage Value		-	2.5	3.3	3.6	V
I/O	V_{IL}	-	–0.3	-	0.25 V_{IO}	V
	V_{IH}		0.75 V_{IO}		3.6	
	V_{OL}	-	-	-	0.1 V_{IO}	
	V_{OH}		0.8 V_{IO}		-	
	I_{MAX}	-	-	-	12	mA
Electrostatic Discharge (HBM)		TAMB = 25 °C	-	-	2	KV
Electrostatic Discharge (CDM)		TAMB = 25 °C	-	-	0.5	KV

Notes on CHIP_EN:

The figure below shows ESP8266EX power-up and reset timing. Details about the parameters are listed in Table 5-2.

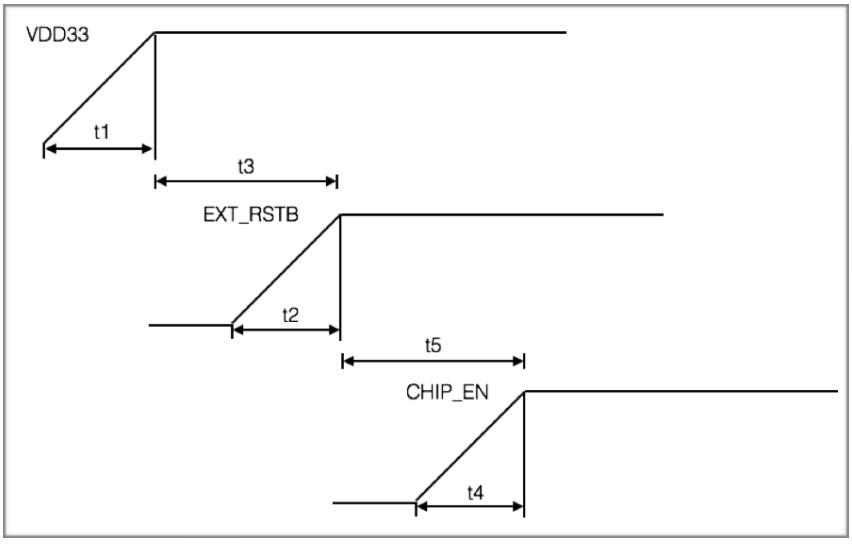


Figure 5-1. ESP8266EX Power-up and Reset Timing

Table 5-2. Description of ESP8266EX Power-up and Reset Timing Parameters

	Description	Min	Max	Unit
t1	The rise-time of VDD33	10	2000	µs
t2	The rise-time of EXT_RSTB	0	2	ms
t3	EXT_RSTB goes high after VDD33	0.1	-	ms
t4	The rise-time of CHIP_EN	0	2	ms
t5	CHIP_EN goes high after EXT_RSTB	0.1	-	ms

5.2. RF Power Consumption

Unless otherwise specified, the power consumption measurements are taken with a 3.0 V supply at 25 °C of ambient temperature. All transmitters' measurements are based on a 50% duty cycle.

Table 5-3. Power Consumption

Parameters	Min	Typical	Max	Unit
TX802.11 b, CCK 11 Mbps, P_{OUT} = +17 dBm	-	170	-	mA
TX802.11 g, OFDM 54Mbps, P_{OUT} = +15 dBm	-	140	-	mA
TX802.11 n, MCS7, P_{OUT} = +13 dBm	-	120	-	mA
Rx802.11 b, 1024 bytes packet length, –80 dBm	-	50	-	mA
Rx802.11 g, 1024 bytes packet length, –70 dBm	-	56	-	mA
Rx802.11 n, 1024 bytes packet length, –65 dBm	-	56	-	mA

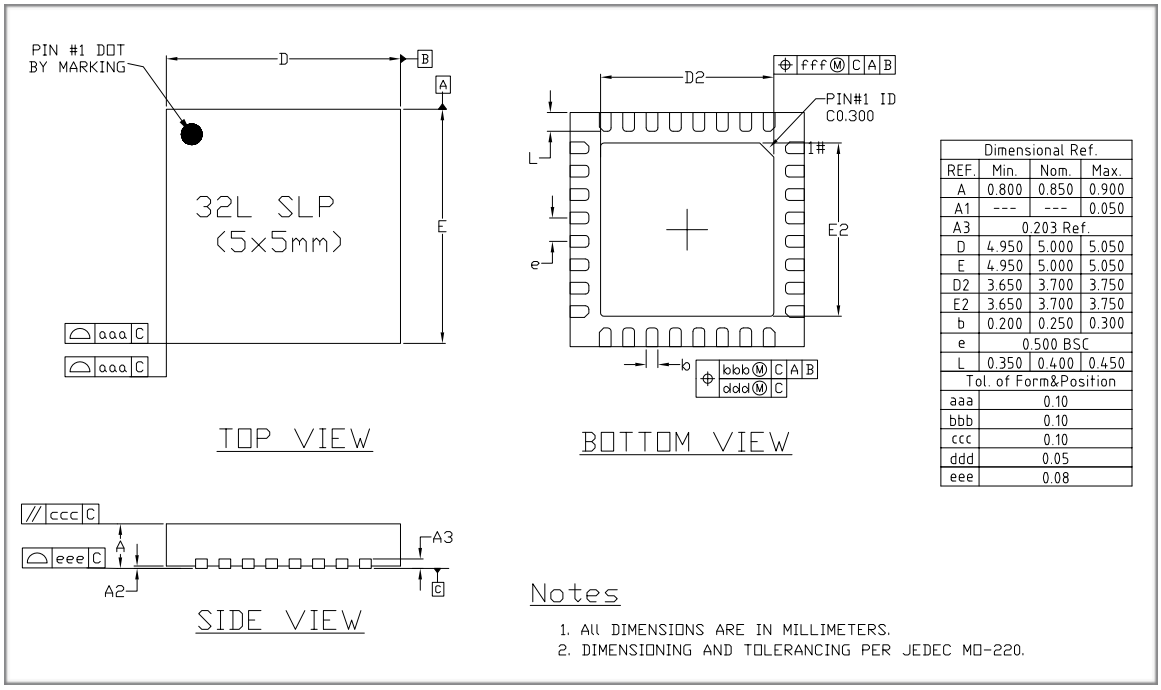
5.3. Wi-Fi Radio Characteristics

The following data are from tests conducted at room temperature, with a 3.3 V power supply.

Table 5-3. Wi-Fi Radio Characteristics

Parameters	Min	Typical	Max	Unit
Input frequency	2412	-	2484	MHz
Output impedance	-	39 + j6	-	Ω
Output power of PA for 72.2 Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1 Mbps	-	–98	-	dBm
CCK, 11 Mbps	-	–91	-	dBm
6 Mbps (1/2 BPSK)	-	–93	-	dBm
54 Mbps (3/4 64-QAM)	-	–75	-	dBm
HT20, MCS7 (65 Mbps, 72.2 Mbps)	-	–72	-	dBm
Adjacent Channel Rejection				
OFDM, 6 Mbps	-	37	-	dB
OFDM, 54 Mbps	-	21	-	dB
HT20, MCS0	-	37	-	dB
HT20, MCS7	-	20	-	dB

6. Package Information



Dimensional Ref.			
REF.	Min.	Nom.	Max.
A	0.800	0.850	0.900
A1	---	---	0.050
A3	0.203 Ref.		
D	4.950	5.000	5.050
E	4.950	5.000	5.050
D2	3.650	3.700	3.750
E2	3.650	3.700	3.750
b	0.200	0.250	0.300
e	0.500 BSC		
L	0.350	0.400	0.450
Tol. of Form&Position			
aaa	0.10		
bbb	0.10		
ccc	0.10		
ddd	0.05		
eee	0.08		

Figure 6-1. ESP8266EX Package

I. Appendix - Pin List

For detailed pin information, please see *ESP8266 Pin List*.

- Digital Die Pin List
- Buffer Sheet
- Register List
- Strapping List

> 📖 ***Notes:***
>
> - *INST_NAME refers to the IO_MUX REGISTER defined in* ***eagle_soc.h****, for example MTDI_U refers to PERIPHS_IO_MUX_MTDI_U.*
> - *Net Name refers to the pin name in schematic.*
> - *Function refers to the multifunction of each pin pad.*
> - *Function number 1 ~ 5 correspond to FUNCTION 0 ~ 4 in SDK. For example, set MTDI to GPIO12 as follows.*
>   - `#define FUNC_GPIO12  3 //defined in eagle_soc.h`
>   - `PIN_FUNC_SELECT(PERIPHS_IO_MUX_MTDI_U,FUNC_GPIO12)`

II. Appendix - Learning Resources

II.1. Must-Read Documents

- *ESP8266 Quick Start Guide*

  Description: This document is a quick user guide to getting started with ESP8266. It includes an introduction to the ESP-LAUNCHER, instructions on how to download firmware to the board and run it, how to compile the AT application, as well as the structure and debugging method of RTOS SDK. Basic documentation and other related resources for the ESP8266 are also provided.
- *ESP8266 SDK Getting Started Guide*

  Description: This document takes ESP-LAUNCHER and ESP-WROOM-02 as examples of how to use the ESP8266 SDK. The contents include preparations before compilation, SDK compilation and firmware download.
- *ESP8266 Pin List*

  Description: This link directs you to a list containing the type and function of every ESP8266 pin.
- *ESP8266 Hardware Design Guideline*

  Description: This document provides a technical description of the ESP8266 series of products, including ESP8266EX, ESP-LAUNCHER and ESP-WROOM.
- *ESP8266 Technical Reference*

  Description: This document provides an introduction to the interfaces integrated on ESP8266. Functional overview, parameter configuration, function description, application demos and other pieces of information are included.
- *ESP8266 Hardware Resources*

  Description: This zip package includes manufacturing BOMs, schematics and PCB layouts of ESP8266 boards and modules.
- *FAQ*

II.2. Must-Have Resources

- ESP8266 SDKs

  Description: This webpage provides links both to the latest version of the ESP8266 SDK and the older ones.
- ESP8266 Tools

Description: This webpage provides links to both the ESP8266 flash download tools and the ESP8266 performance evaluation tools.

- ESP8266 Apps
- ESP8266 Certification and Test Guide
- ESP8266 BBS
- ESP8266 Resources

Espressif IOT Team

www.espressif.com

8. BIBLIOGRAFÍA

Arduino Project Hub: https://create.arduino.cc/projecthub/

Things Board IoT Platform: https://thingsboard.io/docs/samples/arduino/

Arduino User Forum: https://arduino.stackexchange.com/

Conexión y Programación de ESP8266: https://iot-guider.com/esp8266-nodemcu/program-esp8266-arduino-using-ftdi/

Librería WifiESP para ESP8266: https://github.com/bportaluri/WiFiEsp

Librería PubSubClient para MQTT: https://github.com/knolleary/pubsubclient/

Manual módulo ESP8266-01: https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf

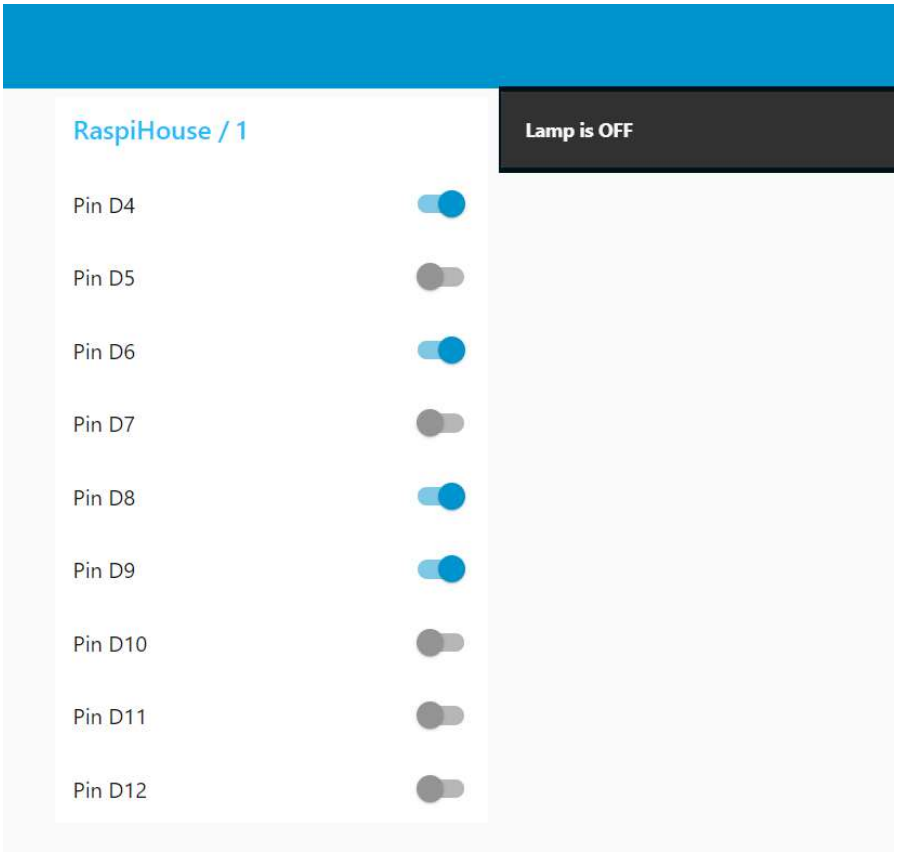
Stack Overflow. Comunidad de programadores: https://stackoverflow.com/

Node-Red: https://nodered.org/docs

9. CONCLUSIÓN

Con este proyecto se ha logrado diseñar un módulo de domótica versátil y escalable, creando una red de IoT a la que se pueden conectar un número indefinido de dispositivos adicionales. Gracias a la utilidad de Node-Red, puede programarse lógica y funciones adicionales sin necesidad de reprogramar el Arduino o el módulo ESP.

La aplicación final toma la forma de uno o más módulos de domótica siendo controlados por la misma instancia de Node-Red desde cualquier dispositivo con conectividad Wifi (teléfono móvil, Tablet PC, Raspberry Pi…) y a través de una interfaz sencilla y personalizable.



El prototipo cumple con los objetivos de escalabilidad y versatilidad propuestos. Cada módulo cuenta con 9 pines de entrada/salida digitales completamente controlables desde la interfaz de aplicación, así como de 8 puertos de entrada/salida analógica que permiten añadir funcionalidades programadas en el Arduino. Esta asignación de puertos digitales para la aplicación y analógicos para funciones programadas permite que no haya interferencias o choques entre ambas.