

Proyecto Final - Curso de Python UdeA.

Internet de las Cosas con la tarjeta de desarrollo ESP32 y MicroPython.

Leoncio Gómez
Medellín, Antioquia
lncgomz@gmail.com

RESUMEN

La disponibilidad de los sistemas en chip (SoC) así como su miniaturización, costo reducido y prestaciones cada vez más numerosas junto con la integración de diferentes lenguajes de programación de alto nivel para su manejo, ha significado un poderoso incentivo para el desarrollo de diferentes aplicaciones de internet de las cosas (IoT) que comprende la interconexión digital de objetos cotidianos con internet. En el contexto del curso de programación en Python ofrecido por la Universidad de Antioquia, se lleva a cabo un primer acercamiento a la implementación de esta tecnología de forma tal que se apliquen los conocimientos adquiridos durante este período lectivo; en este sentido, se creará un dispositivo que permita la comunicación, a través de la red WiFi, con cualquier periférico conectado a internet que soporte el protocolo HTTP, como resultado de esta interacción, será posible manejar remotamente un circuito electrónico que responda de manera dinámica a estas peticiones y las traduzca en un fenómeno físico concreto.

Keywords: IoT, MicroPython, ESP32, Python

1. INTRODUCCIÓN

1.1. Enunciado del Proyecto

Como proyecto final del curso de programación en Python, se sugirió la **Propuesta Internet de las Cosas** con el siguiente enunciado:

Desarrollar una aplicación que permita transmitir un mensaje por medio de colores. La aplicación debe comunicarse a través de un módulo ESP controlado por micropython y un led RGB al que se le enviara la información a través de un programa. La aplicación debe: Representar por colores la frase "Hola Mundo". La codificación de las letras que necesitara para su representación se muestra a continuación:

- La letra "A" Será Representada con el color blanco
- La "D" Será Representada con el color rojo
- La "H" Será Representada con el color verde
- La "L" Será Representada con el color azul
- La "M" Será Representada con el color amarillo
- La "N" Será Representada con el color violeta
- La "O" Será Representada con el color naranjado
- La "U" Será Representada con el color café

El espacio entre las palabras se representará cuando no se muestren colores durante 2 segundos.

2. IMPLEMENTACIÓN

2.1. Definición de Alcance

Durante esta etapa del proyecto se decidió ampliar el alcance del mismo, permitiendo la codificación en colores de todas las letras

del alfabeto inglés además de las mencionadas en el enunciado; del mismo modo, se optó por la inclusión de más funcionalidades como la codificación del mensaje a código Morse utilizando un buzzer pasivo y la presentación de cada una de las letras del texto a través de un display de 7 segmentos.

2.2. Materiales

Para la implementación del proyecto, se requirieron los siguientes materiales y equipos.

- 1 Tarjeta de desarrollo *Espressif ESP32-WROOM-32D*.
- 1 LED RGB de cátodo común.
- 1 Buzzer pasivo.
- 4 Resistencias de 220 Ω .
- 1 Display de 7 segmentos de cátodo común.
- 2 Protoboards pequeños.
- 18 conectores macho-macho.

2.3. Software

Como entorno integrado de desarrollo se optó por la utilización de **Thonny** en su versión 3.3.0 debido a su facilidad de uso y diseño intuitivo. El mismo es un software gratuito, distribuido bajo licencia MIT, integrado con **MicroPython** por defecto y disponible para los sistemas operativos más utilizados.

Del mismo modo, se empleó el programa **KiCAD** para realizar la esquematización del circuito de este proyecto.

Para el desarrollo del proyecto, se utilizará **Python 3.8**

2.4. Configuración del ESP32

Para configurar apropiadamente esta tarjeta de desarrollo, deben seguirse una serie de pasos detallados en la **documentación oficial** de MicroPython.

De manera resumida:

- **Descargar** el firmware correspondiente al modelo de la tarjeta que se esté utilizando. En este caso concreto, se escogió el archivo *esp32-idf3-20200902-v1.13.bin*.
- **Descargar** el script *esptool.py* ya sea directamente desde el repositorio o utilizando la herramienta *PIP* integrada en Python.
- Para lograr la comunicación entre el equipo y la ESP32, es necesario la utilización de un puerto serial, el cual no está integrado en la mayoría de los equipos más recientes; no obstante, puede utilizarse un convertidor USB a Puerto Serial o **instalar** el driver que permita la creación de un puerto COM virtual a partir de la conexión USB, esta última fue la opción escogida.

Una vez descargados los archivos indicados e instalando el puerto serial virtual, la configuración del ESP32 se reduce a la ejecución del siguiente código:

```
pip install esptool
```

```
esptool.py --port puertoESP32 erase_flash
```

```
esptool.py --chip esp32 --port puertoESP32 write_flash  
-z 0x1000 archivoFirmware.bin
```

Donde *puertoESP32* corresponde al puerto virtual generado después de instalar el driver y conectar la tarjeta al equipo (En este caso, corresponde a COM3) y *archivoFirmware* es el archivo binario que fue descargado anteriormente (esp32-idf3-20200902-v1.13.bin).

2.5. Conexión WiFi

Para permitir la conexión de la ESP32 con la red WiFi, es necesario descargar e importar, dentro de la memoria de la tarjeta, la librería correspondiente, una de las opciones más utilizadas en este caso es el archivo **wifmgr.py**. En la red se encuentran disponibles **tutoriales** que explican detalladamente como lograr una conexión efectiva a la red WiFi utilizando esta librería.

2.6. Comunicarse con la ESP32 a través de Internet

Una vez lograda la conexión de la tarjeta con la red WiFi, el próximo paso es personalizar la página de respuesta que recibe el usuario al conectarse a la IP estática de la misma (En este caso, 192.168.1.6); desde aquí, se podrán enviar parámetros al microcontrolador a través de formularios, que serán traducidos a peticiones GET o POST según la necesidad y recibidos para su posterior procesamiento empleando código Python.

En la figura 1 puede apreciarse la página diseñada como interfaz entre el periférico y la tarjeta ESP32, en la misma se muestra la información sobre su propósito y funcionamiento y se incluye un cuadro de texto, dos botones de opción (Radio Buttons) y un botón para enviar los datos del formulario (Submit); a nivel de código, el cuadro de texto está asociado al parámetro *msg* y corresponde al mensaje que se va a codificar, mientras que los botones de opción están agrupados y asociados a la variable *mode* que posteriormente le indicará al código Python el tipo de conversión que se desea realizar (A código RGB o a código Morse).

2.7. Componentes del Circuito

Llegados a este punto, la ESP32 es capaz de conectarse a la red WiFi, recibir conexiones y enviar una página web de respuesta a la cual pueden agregarse botones y formularios para que el usuario interactúe con ellos y envíe estos datos de vuelta a la tarjeta; sin embargo, se desea poder utilizar esta información para provocar algún tipo de respuesta (Encender una luz, emitir un sonido, activar un relé, etc.) en este sentido, se debe ensamblar el circuito con aquellos componentes que reaccionen de la forma que se necesita; para el caso concreto de este proyecto, se desea encender una luz de un determinado color en función de las letras de las palabras recibidas, emitir un sonido que tenga correspondencia biunívoca

Figura 1: Página web de respuesta después de una conexión exitosa a la IP estática de la tarjeta de desarrollo.

con estos caracteres y mostrar estas letras en algún tipo de indicador visual.

2.7.1. LED RGB. Los ledes RGB consisten en un led rojo, uno azul y otro verde. Ajustando independientemente cada uno de ellos, los ledes RGB son capaces de producir una amplia gama de colores. A diferencia de los ledes dedicados a un solo color, los ledes RGB no producen longitudes de ondas puras. Además, los módulos disponibles comercialmente no suelen estar optimizados para hacer mezclas suaves de color. [2]



Figura 2: Pin Out de un Led RGB Cátodo Común.

El ajuste del RGB se logra a través de la modulación de ancho de pulso, el cual nos permite obtener diferentes valores de voltaje a partir de una señal continua; de este modo, ajustando apropiadamente los valores de voltaje que alimentan los pines de este componente, pueden generarse los colores deseados.

2.7.2. *Buzzer*. Es un transductor electroacústico que produce un sonido o zumbido continuo o intermitente de un mismo tono (generalmente agudo). Sirve como mecanismo de señalización o aviso y se utiliza en múltiples sistemas, como en automóviles o en electrodomésticos, incluidos los despertadores. [4]



Figura 3: Símbolo eléctrico del Buzzer.

2.7.3. *Display 7 Segmentos*. El visualizador de 7 segmentos es un componente que se utiliza para la representación de caracteres (normalmente números) en muchos dispositivos electrónicos, debido en gran medida a su simplicidad. Aunque externamente su forma difiere considerablemente de un led típico, internamente están constituidos por una serie de leds con unas determinadas conexiones internas, estratégicamente ubicados de tal forma que forme un número '8'. [3]

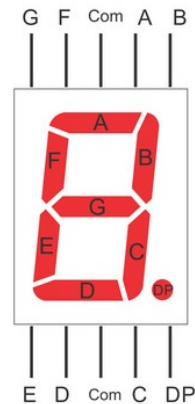


Figura 4: Pin Out del Display 7 Segmentos.

Cada uno de los segmentos que forman la pantalla están marcados con siete primeras letras del alfabeto ('a'-'g'), y se montan de forma que permiten activar cada segmento por separado, consiguiendo formar cualquier dígito numérico.

2.8. Ensamblaje del Circuito

La manera en la cual la ESP32 interactúa con los componentes electrónicos básicos es a través de sus periféricos [1], por el alcance de este proyecto, el énfasis estará puesto en las salidas digitales, toda vez que las señales de entrada del sistema están ingresando de manera inalámbrica, por medio de la conexión WiFi. En este sentido, primero se necesita definir donde irán conectados los componentes en la ESP32 ya que esta información será requerida al momento de implementar el código Python encargado de manipular estos componentes de la forma que se requiere. No existen muchas restricciones sobre cuales pines pueden utilizarse para conectar los componentes que se utilizarán en este proyecto, más allá de la comodidad para realizar la conexión y la posibilidad de funcionar como salida PWM. A continuación, se muestra la asignación de pines de la ESP32 para los 3 componentes a utilizar en este proyecto (Led RGB, Buzzer y Display de 7 Segmentos).

Cuadro 1: Conexiones de la ESP32 con Led RGB

PinOut	GPIO
R	25
G	26
B	27
-	GND

Cuadro 2: Conexiones de la ESP32 con Buzzer

PinOut	GPIO
+	33
-	GND

Cuadro 3: Conexiones de la ESP32 con Display 7 Segmentos Cátodo Común

PinOut	GPIO
a	15
b	2
c	4
d	5
e	18
f	19
g	21
dp	22
-	GND

Una vez decidida la disposición de las conexiones, el siguiente paso es implementarlas en el protoboard, siguiendo el esquema desarrollado para tal fin (Ver figura 5).

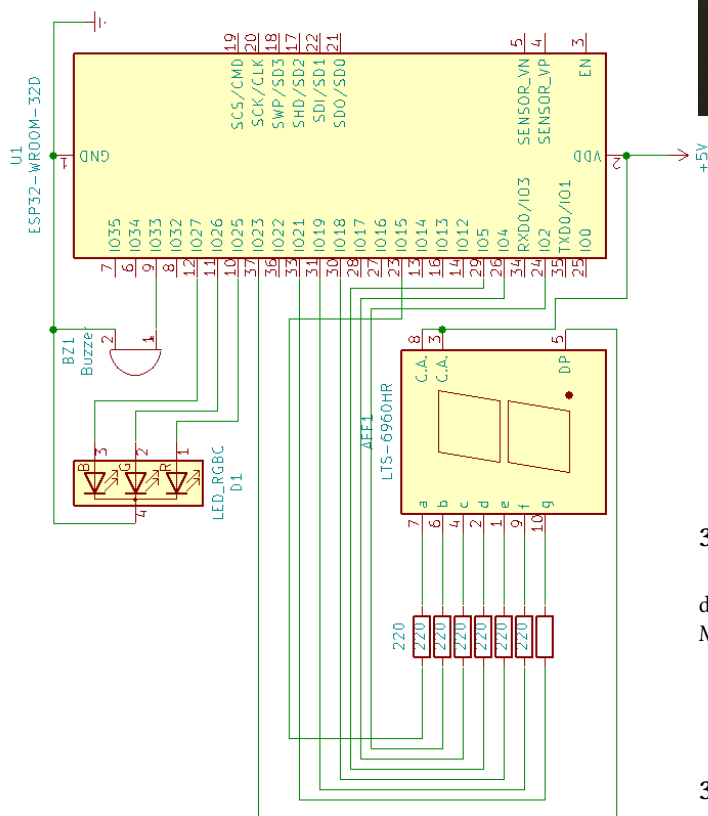


Figura 5: Esquemático del Proyecto IoT.

Un detalle de la implementación del circuito puede verse en la figura 6

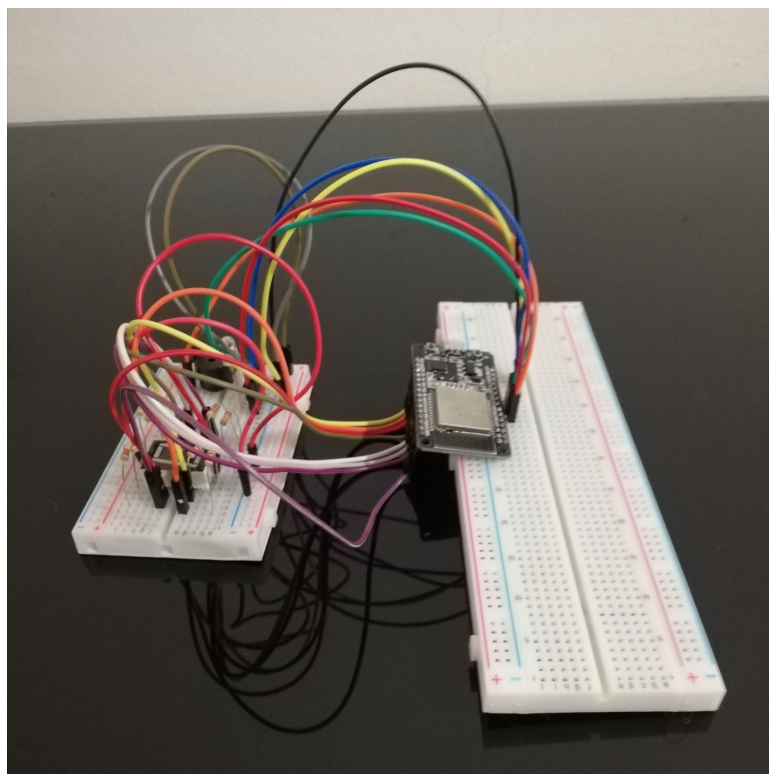


Figura 6: Circuito del Proyecto IoT.

3. FUNCIONAMIENTO

A continuación, se describe el funcionamiento del sistema en sus dos modalidades: Conversión a código RGB y Conversión a Código Morse.

3.1. Conversión a Código RGB

- Encender el ESP32 con las configuraciones y archivos descargados, puede utilizarse Thonny para poder visualizar los mensajes de depuración.
- Acceder, desde una laptop o teléfono inteligente, a la dirección IP estática del sistema, en este caso: **192.168.1.6**.
- En la pantalla principal 7, escribir el mensaje que se desee convertir y seleccionar la opción '*Convertir a Colores*'.





Proyecto IoT MicroPython UdeA

Introduzca el texto en el recuadro e indique si desea verlo convertido en **Código de Colores** o en **Código Morse** para una mejor experiencia, se recomienda utilizar solamente caracteres alfabéticos, omitiendo acentos y la letra Ñ

Escriba aquí su mensaje:

Hola Mundo

☒ Convertir a Colores
☐ Convertir a Morse

Enviar

Elaborado por: Leoncio Gómez - Diciembre 2020

Figura 7: Funcionamiento del Sistema: Conversión a código RGB (Colores)

- Presionar el botón 'Enviar'
- La página dinámica será recibida como respuesta (Ver figura 8), incluyendo información sobre la modalidad de conversión seleccionada y el resultado de la conversión tabulada, el código RGB correspondiente a cada letra es mostrado con el color que está representando.

Resultados

- Mensaje Enviado: Hola Mundo
- Modo de Conversión Seleccionado: Código RGB

Letra	Valor
H	[100, 100, 100]
O	[255, 0, 0]
L	[202, 62, 94]
A	[0, 0, 180]


M	[205, 145, 63]
U	[0, 154, 37]
N	[12, 75, 100]
D	[255, 200, 47]
O	[255, 0, 0]

Figura 8: Funcionamiento del Sistema: Conversión a código RGB (Colores), página dinámica de respuesta

- Inmediatamente después de recibir la página dinámica de respuesta, el Led RGB del circuito se encenderá consecutivamente mostrando los colores de cada una de las letras del mensaje enviado, al mismo tiempo, el display 7 segmentos mostrará la letra que se está representando en ese momento.

3.2. Conversión a Código Morse

- Encender el ESP32 con las configuraciones y archivos descargados, puede utilizarse Thonny para poder visualizar los mensajes de depuración.
- Acceder, desde una laptop o teléfono inteligente, a la dirección IP estática del sistema, en este caso: **192.168.1.6**.
- En la pantalla principal 7, escribir el mensaje que se desee convertir y seleccionar la opción 'Convertir a Morse'.



Proyecto IoT MicroPython UdeA

Introduzca el texto en el recuadro e indique si desea verlo convertido en **Código de Colores** o en **Código Morse** para una mejor experiencia, se recomienda utilizar solamente caracteres alfabéticos, omitiendo acentos y la letra Ñ

Escriba aquí su mensaje:

☐ Convertir a Colores
☒ Convertir a Morse

Elaborado por: Leoncio Gómez - Diciembre 2020

Figura 9: Funcionamiento del Sistema: Conversión a código Morse

Resultados

- Mensaje Enviado: Hola Mundo
- Modo de Conversión Seleccionado: Código Morse

Letra	Valor
H	[...]
O	[- -]
L	[. - .]
A	[. -]

M	[- -]
U	[. . -]
N	[- .]
D	[- . .]
O	[- - -]

Figura 10: Funcionamiento del Sistema: Conversión a código Morse, página dinámica de respuesta

- Presionar el botón 'Enviar'
- La página dinámica será recibida como respuesta (Ver figura 10), incluyendo información sobre la modalidad de conversión seleccionada y el resultado de la conversión tabulada, es decir, el código Morse correspondiente a cada letra.

- Inmediatamente después de recibir la página dinámica de respuesta, el buzzer del circuito se encenderá consecutivamente emitiendo pitidos largos y cortos, correspondientes a cada una de las letras del mensaje enviado, al mismo tiempo, el display 7 segmentos mostrará la letra que se está representando en ese momento.

ÍNDICE

Abstract	1
1. Introducción	1
1.1. Enunciado del Proyecto	1
2. Implementación	1
2.1. Definición de Alcance	1
2.2. Materiales	1
2.3. Software	1
2.4. Configuración del ESP32	1
2.5. Conexión WiFi	2
2.6. Comunicarse con la ESP32 a través de Internet	2
2.7. Componentes del Circuito	2
2.7.1. LED RGB	2
2.7.2. Buzzer	3
2.7.3. Display 7 Segmentos	3

2.8. Ensamblaje del Circuito	3
3. Funcionamiento	4
3.1. Conversión a Código RGB	4
3.2. Conversión a Código Morse	5
Índice	7
Referencias	7

REFERENCIAS

- [1] Random Nerd Tutorials. [n.d.]. ESP32 Pinout Reference. Website. URL: <https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>.
- [2] Wikipedia. [n.d.]. RGB. Website. URL: <https://es.wikipedia.org/wiki/LedRGB>.
- [3] Wikipedia. [n.d.]. Visualizador de siete segmentos. Website. URL: https://es.wikipedia.org/wiki/Visualizador_de_siete_segmentos.
- [4] Wikipedia. [n.d.]. Zumbador. Website. URL: <https://es.wikipedia.org/wiki/Zumbador>.