Internet de las Cosas y sistemas embebidos Smartband reminder

Mateo Scozzina, Pablo Ángel Toledo, Santiago Enrique Roatta

Facultad de Tecnología Informática, Universidad Abierta Interamericana (UAI), Rosario, Argentina
{Scozzina}mateoscozzina@outlook.com
{Toledo}pablotoledo1994@hotmail.com
{Roatta}santiago.roatta@gmail.com

Resumen. Este trabajo presenta un problema de Ingeniería de la asignatura Sistemas de Hardware de la carrera Ingeniería en Sistemas Informáticos de la Universidad Abierta Interamericana (UAI). Es una actividad obligatoria necesaria para regularizar la asignatura para la cual destinamos tres semanas dentro del curso 2021. Si bien esto no es un verdadero trabajo de investigación pues no aporta conocimiento nuevo, hemos decidido presentarlo respetando el formato habitual de los "papers científicos", con resumen, introducción, desarrollo, conclusiones y referencias. Primero, presentamos una breve introducción a los sistemas embebidos e Internet de las Cosas. A continuación, se describe el módulo TTGO que usamos en nuestro desarrollo. Finalmente, se muestra un prototipo de nuestro desarrollo: un recetario o recordatorio electrónico para automatizar el recordatorio a aquellas personas que necesitan tomar una medicina de manera periódica y permanente.

Palabras clave. SE, IoT, TTGO, Smartband, Medicina.

1 Introducción

El trabajo consiste en explorar las características del módulo TTGO y desarrollar una aplicación que nos muestre la relevancia que tienen actualmente los sistemas embebidos en nuestra vida cotidiana. Los **sistemas embebidos (SE)** son sistemas basados en un microprocesador que contienen hardware y software diseñados y optimizados específicamente para resolver un problema de una manera eficiente. El hardware que se utiliza para el desarrollo de un SE se diseña buscando administrar el tamaño, costo, rendimiento y el uso de la energía (este último principalmente en sistemas que funcionen a batería sin estar conectados a la red eléctrica). En su arquitectura, los SE contienen un microprocesador que ejecuta las operaciones a cierta velocidad, controladas por una señal de reloj. Esta señal de reloj está condicionada por los recursos internos y la máxima frecuencia de operación, y se mide en MHz (megahertz) [1].

Un **microcontrolador** se considera un SE, ya que en un solo chip se puede encontrar un sistema computarizado completo que cuenta con un microprocesador, unidades de memoria, unidades de entrada/salida y periféricos. **Los SE no son equivalentes a las computadoras de propósito general** como notebooks o PCs de escritorio, pues tienen recursos limitados y están desarrollados exclusivamente para resolver una tarea específica, como sucede por ejemplo en las alarmas contra robos, los sistemas de frenos ABS de los automóviles o los controles de climatización [2].

El concepto de **IoT** (Internet of Things, "Internet de las Cosas") basa su concepto en la esencia de la posibilidad de conectar los objetos que utilizamos en la vida cotidiana a Internet, siendo de gran importancia para marcar la primera evolución real de Internet, logrando que sea sensorial. IoT consiste en desarrollar un SE que tenga la capacidad de conectar el dispositivo a Internet a través de una dirección IP que le permitirá interactuar con el mundo exterior enviando y recibiendo datos [3], [4].

En cuanto al **módulo LILYGO TTGO T-Display ESP32 Development Board**, el mismo consiste en una placa de desarrollo que, a diferencia de otras, cuenta con características de hardware adicionales para desarrollar sistemas enfocados al IoT. El módulo ofrece una simple conexión de sensores y actuadores a Internet mediante WiFi o Bluetooth, además de que cuenta con un display integrado. El desarrollo del software puede ser realizado en diversos lenguajes de programación como C, C++, Python, MicroPython, entre muchos otros lenguajes de desarrollo.

2 Análisis

2.1 Características del datasheet del módulo TTGO, modos de comunicación con el mundo exterior y aplicaciones sugeridas

El módulo **LILYGO TTGO T-Display ESP32 Development Board** es una tarjeta de desarrollo basada en ESP32 que cuenta con pantalla a color de 1,14 pulgadas, interfaz para carga de baterías de litio, un botón de reset, WiFi y Bluetooth. Sirve para desarrollar sistemas enfocados al IoT (interactuar con el mundo exterior), desplegando información en la pantalla que tiene incorporada y ofreciendo la conexión de una batería de litio que alimente la placa y se pueda cargar mediante su puerto USB-C.

En la tabla 1, se detallan las características técnicas del módulo [5].

Tabla 1. Características del módulo LILYGO TTGO T-Display ESP32 Development Board.

HARDWARE SPECIFICATIONS	
Chipset	ESPRESSIF-ESP32 240MHz Xtensa® single-/dual-core 32-bit LX6 microprocessor
FLASH	QSPI flash 4MB/16MB
SRAM	520 kB SRAM

Modular interface	UART, SPI, SDIO, I2C, LED PWM, TV PWM, I2S, IRGPIO, ADC, capacitor touch sensor, DACLNA preamplifier
Display	IPS ST7789V 1.14 Inch
Working voltage	2.7V-4.2V
Working current	About 67MA
Sleep current	About 350uA
Working temperature range	-40℃ ~ +85℃
Size&Weight	51.52*25.04*8.54mm (7.81g)
Power Supply	USB 5V/1A
Charging current	500mA
Battery	3.7V lithium battery
JST Connector	2Pin 1.25mm
USB	Type-C
WIFI	
Standard	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC (esp32 chip)
Protocol	802.11 b/g/n (802.11n, speed up to150Mbps) A-MPDU and A-MSDU polymerization, support 0.4μS Protection interval
Frequency range	2.4GHz~2.5GHz(2400M~2483.5M)
Transmit Power	22dBm
BLUETOOTH	
Protocol	Meet Bluetooth v4.2BR/EDR and BLE standard
Radio frequency	With -97dBm sensitivity NZIF receiver Class-1, Class-2&Class-3 emitter AFH
SOFTWARE SPECIFICATIONS	
Wi-Fi Mode	Station/SoftAP/SoftAP+Station/P2P
Security mechanism	WPA/WPA2/WPA2-Enterprise/WPS
Encryption Type	AES/RSA/ECC/SHA
Firmware upgrade	UART download/OTA (Through network/host to download and write firmware)
Software Development	Support cloud server development/SDK for user firmware development
Networking protocol	IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT

User Configuration	AT + Instruction set, cloud server, Android/iOSapp
OS	FreeRTOS

Continuando con las características que ofrece el módulo TTGO, en la figura 1 se aprecian los pines que lo componen:

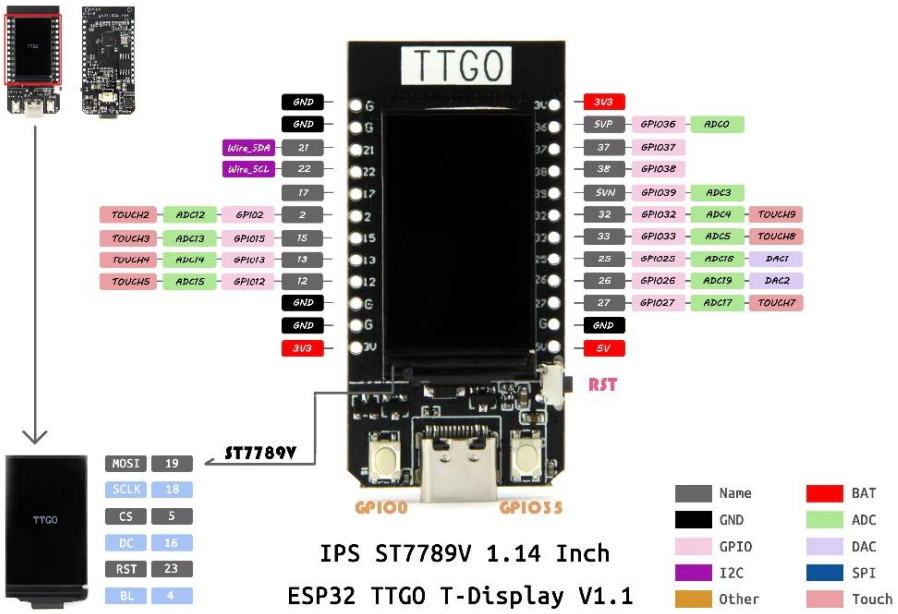

Fig. 1. Pines del módulo LILYGO TTGO T-Display ESP32 Development Board.

En cuanto a su funcionalidad, algunos ejemplos de aplicaciones que se pueden desarrollar con el módulo TTGO y que muestran relevancia en nuestra vida cotidiana son [6]:

- Calculadora básica: se puede añadir un teclado numérico con 4 botones que sirvan para las operaciones básicas y tener una calculadora compacta.
- Monitor de temperatura y humedad: se puede añadir al módulo un sensor DHT11 o DHT22 para tener en tiempo real los valores de temperatura y humedad del ambiente y transmitir los datos por WiFi o Bluetooth.
- Reloj despertador compacto: añadiendo un parlante, se puede generar un pequeño reloj despertador y que tenga muy bajo consumo de batería.
- Consola de videojuegos: en Internet se pueden encontrar varios proyectos en los cuales se desarrollan videojuegos simples que pueden correr en la placa.

2.2 Características del entorno de desarrollo Thonny

Thonny es un IDE (Integrated Development Environment, "Entorno de Desarrollo Integrado") de Python catalogado como uno de los 5 mejores para este lenguaje. Fue desarrollado en el Instituto de Ciencias de la Computación en la Universidad de Tartu, Estonia, adoptando un enfoque diferente ya que su depurador está diseñado específicamente para aprender y enseñar programación [7]. Posee una interfaz de usuario básica y simple que los principiantes pueden entender fácilmente, y permite ejecutar programas paso a paso usando Ctrl + F5 sin necesidad de puntos de interrupción. Evalúa expresiones con diferentes colores y representa llamadas a funciones para facilitar su comprensión [8].

El editor de Thonny le permite detectar fácilmente errores de sintaxis como paréntesis y comillas sin cerrar. Puede resaltar las apariciones de variables, por lo que no repite el mismo nombre y también ayuda a detectar otros errores. Las variables se presentan en base a un modelo simplificado, pero también puede cambiar a otros modelos realistas. El programa funciona en Windows, macOS y Linux. Viene equipado con Python v3.x incorporado, por lo que necesita un instalador simple para comenzar. Se puede instalar a través del administrador de paquetes del sistema operativo en Debian, Raspberry Pi, Ubuntu y Fedora [9].

Thonny ha recibido críticas favorables de las comunidades de educación en ciencias de la computación y Python. Ha sido una herramienta recomendada en varios MOOC de programación (Massive Open Online Course, "Curso masivo abierto en línea"), y desde Junio de 2017 se ha incluido de forma predeterminada en la distribución oficial del sistema operativo Raspberry Pi OS de Raspberry Pi [10].

En cuanto al lenguaje utilizado en el proyecto, **Micropython** fue creado en base de Python 3.4 orientado al uso de microcontroladores. Apunta a desarrollar un script de manera más rápida que la convencional con otros lenguajes. Uno de los grandes beneficios que posee es que al ser un derivado de Python cuenta con una gran cantidad de librerías y una gigante comunidad que conoce el lenguaje que puede ayudar a resolver cualquier cuestión que se presente. Además, la curva de aprendizaje es mucho más veloz debido a que se trata de un lenguaje más sencillo [11].

La mayor dificultad que conlleva implementar el código son los limitados recursos de memorias RAM y ROM, y al ser un lenguaje apropiado para ello, se utiliza mucho en ámbitos de SE facilitando a programadores tanto avanzados como principiantes a ejecutar su idea con gran velocidad [12], [13].

En las figuras 2 y 3, se muestran una imagen del módulo y los accesorios que se necesitaron para realizar el proyecto:

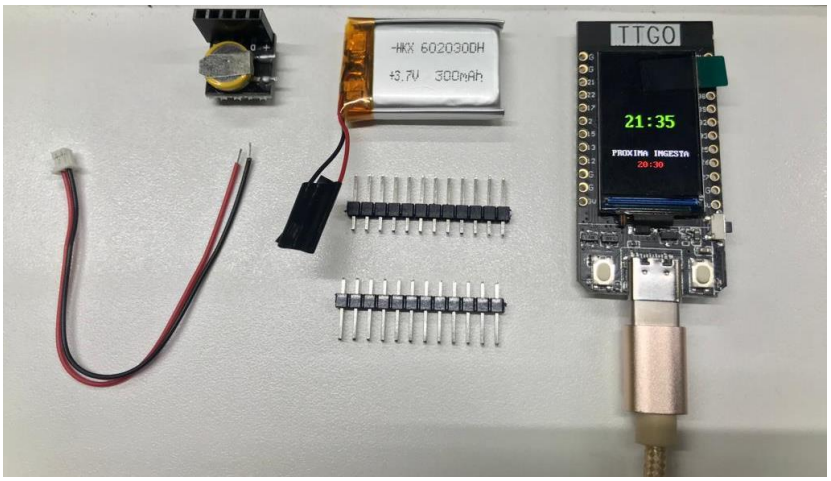

Fig. 2. Módulo TTGO, batería de 300 mAh y módulo RTC.

Para familiarizarnos con el IDE Thonny, experimentamos primero desarrollando un simple reloj en el módulo TTGO para chequear el funcionamiento del display:

Fig. 3. Prueba del display del módulo TTGO.

Se observa en la figura 4 el código utilizado para realizar la prueba del display:

main.py

```
from machine import Pin, SPI
import st7789
#Importa fuentes
import vga1_bold_16x32 as font1
import vga2_8x16 as font2
spi = SPI(1, baudrate=30000000, polarity=1, phase=1, sck=Pin(18), mosi=Pin(19))
display = st7789.ST7789(
    spi,
    135,
    240,
    reset=Pin(23, Pin.OUT),
    cs=Pin(5, Pin.OUT),
    dc=Pin(16, Pin.OUT),
    backlight=Pin(4, Pin.OUT),
    rotation=0)
display.init ()

medicina = "IBUPROFENO"
centrarmedicina = (134-(len(medicina)*8))/2

#display.text (font2, "TOMAR", int((134-(len("TOMAR")*8))/2), 40, st7789.RED)
#display.text (font2, medicina, int(centrarmedicina), 60, st7789.RED)

display.text (font1, "21:35", int((134-(len("XX:XX")*16))/2), 104, st7789.GREEN)
display.text (font2, "PROXIMA INGESTA", int((134-(len("PROXIMA INGESTA")*8))/2), 170, st7789.WHITE)
display.text (font2, "20:30", int((134-(len("XX:XX")*8))/2), 190, st7789.RED)

#display.hline (0, 119, 239, st7789.color565(225,225,225))
#display.vline (66, 0, 239, st7789.color565(225,225,225))
```

Fig. 4. Código para prueba de display en MicroPython sobre el IDE Thonny.

3 Desarrollo

Se pensó en el desarrollo de una smartband (pulsera inteligente) que provea a las personas que deben tomar una medicina con frecuencia la tranquilidad de tener automatizado el recordatorio de la ingesta, la cual opera gracias a una aplicación desarrollada en MicroPython cuya principal función es generar la alerta mencionada.

El objetivo de esta pulsera es tener una función precisa y concreta, y ser totalmente independiente de conexiones continuas ya que está pensada principalmente para pacientes de avanzada edad que quizás no tengan afinidad con la tecnología, simplemente deberán colocarse la pulsera y luego dejar que ella haga su trabajo. Cabe destacar que además de la alternativa de que el médico sea quien configure la smartband, también es factible que un paciente afín a la tecnología y al uso de una aplicación en un dispositivo Android, pueda autoconfigurar su recetario.

La misma se crea en base al módulo LILYGO TTGO ya que cuenta con las características necesarias de hardware para poder llevar a cabo este proyecto, como el display integrado, WiFi y el módulo de carga de la batería. Para proteger la placa y que tenga buena estética, es necesario el diseño de una carcasa hecha con una impresora 3D en base a plástico FLEX ya que éste tiene la capacidad de soportar flexión y podrá amoldarse con facilidad al brazo del paciente que la utiliza. Además, se requiere de la adición de un lector de tarjetas de memoria para poder colocar una Micro SD y alma-

cenar los parámetros que necesiten configurarse, al igual que un módulo RTC que marque la hora para que no sean borrados los datos en el caso que se apague. Para la notificación que debe recibir el paciente es necesario adicionar un parlante que pueda alertar cuando se cumpla la hora programada de la ingesta. Y, por otro lado, basados en el consumo que demanda la placa, se considera también una batería de 300 mAh para que alimente la misma y logre aumentar considerablemente su autonomía [5].

El profesional de la salud podrá configurar, a través de una aplicación en su dispositivo Android, el recetario personalizado de la medicina que deberá tomar el paciente. El médico se enlazará con la smartband a través de su dispositivo vía WiFi por única vez para enviarle la configuración a la pulsera, y de allí en más la pulsera será totalmente independiente para respetar a diario su configuración y alertar con notificaciones al paciente cuando llegue la hora de tomar la medicina. En caso de que se quiera cambiar la configuración del recetario, el paciente que así lo requiera deberá recurrir nuevamente al profesional para que el mismo realice el procedimiento de configuración, o podrá realizarlo él mismo si es apto para manejar la aplicación por su cuenta.

En cuento a la configuración de la ingesta por parte del profesional, se muestra en la figura 5 el diagrama de flujo que se utilizó para diseñar la aplicación:

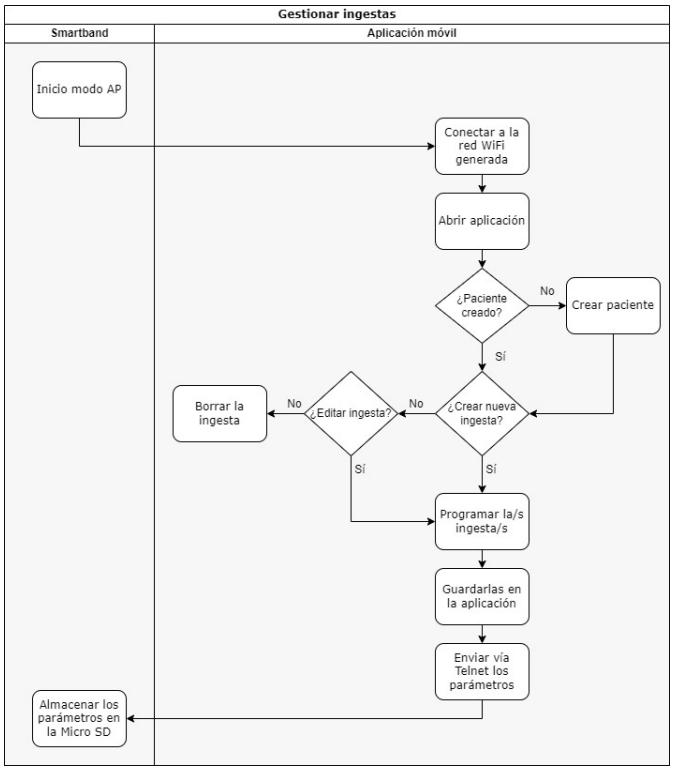

Fig. 5. Diagrama de flujo para gestionar ingestas.

Por último, en la figura 6 pueden verse las pantallas de la aplicación Android que utilizará el médico:

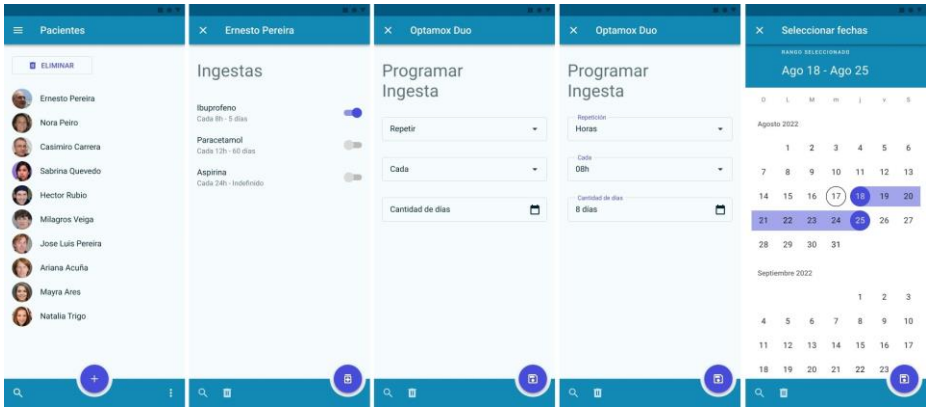

Fig. 6. Vistas de la aplicación móvil.

4 Conclusiones

A nivel mundial suceden cada año millones de errores en la toma de medicamentos, de los cuales miles terminan de manera trágica, siendo las personas de avanzada edad las de mayor susceptibilidad. Con el objetivo de contrarrestarlo, se desarrolló un prototipo de una smartband que permite estandarizar la posología y la forma de administración de cualquier fármaco de manera sencilla y práctica aumentando considerablemente el grado de adherencia farmacológica, y garantizando una interacción más amigable entre el profesional de la salud, los medicamentos y los pacientes.

En cuanto a la complejidad del trabajo, el tiempo destinado para el mismo fue de tres semanas y fue suficiente para terminar el prototipo, pero no para lograr una instancia comercializable del equipo. Al iniciar la investigación se intentó avanzar con el proyecto probando programar software no abierto y se corroboró que no se puede debido a que el mismo no cuenta con las librerías necesarias para lograr los objetivos. Debido a ello se decidió avanzar con el módulo TTGO ya que los lenguajes que se utilizan para programar esta placa son muy populares y cuentan con una gran cantidad de librerías, y la inmensa comunidad que los utiliza ofrece mucha ayuda a través de Internet para resolver problemáticas que plantean los programadores.

La elección del módulo TTGO para realizar el proyecto se hizo también debido a que cuenta con un display integrado y a que provee una gran facilidad para interactuar con Internet a través de sensores y actuadores, vía WiFi o Bluetooth, siendo en resumen una placa de desarrollo ideal para programar sistemas enfocados al IoT.

El prototipo desarrollado tiene entre sus ventajas más importantes el fundamento de que el equipo puede ser realizado con un muy bajo costo y ello está directamente rela-

cionado con la gran accesibilidad que tendrá el mismo para poder ser adquirido por la sociedad, además de su gran practicidad, comodidad y facilidad de uso. Los próximos pasos apuntan a convertir este prototipo en un sistema 100% operativo y comercializable, para lo cual se buscará optimizar la electrónica del equipo analizando la posibilidad de lograr una mayor duración de la batería y de confeccionar una pulsera más liviana para lograr aún una mayor comodidad al usarla.

Además, existe la posibilidad de que esta tecnología pueda fomentar la inclusión social a través de una propuesta al Gobierno para que se incluya este producto dentro de un programa de salud. De esta manera, aquellas personas que no estén en condiciones de adquirirlo también podrían contar con este dispositivo y se lograría implementar con mayor alcance social un sistema que ayude a las personas a mejorar su calidad de vida, y principalmente a los adultos mayores.

Referencias

1. Benito Úbeda Miñarro: Sistemas embebidos. Apunte de profesor del Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia (2009) 2-4.
2. Sergio Salas Arriarán: Todo sobre sistemas embebidos. Arquitectura, programación y diseño de aplicaciones prácticas con el PIC18F. Universidad Peruana de Ciencias Aplicadas, 1ra edición (2015) 38–39.
3. Secretaría de Regulación de Tecnologías de la Información y las Comunicaciones: Internet de las Cosas. Trabajo de investigación publicado por el Ministerio de Comunicaciones, Presidencia de la Nación (2018) 3-4.
4. Dave Evans: Internet de las Cosas. Cómo la próxima evolución de Internet lo cambia todo. Informe técnico de Cisco (2011) 5-7.
5. Espressif Systems: ESP32 Series. Datasheet publicado por Espressif Systems (2022) 8-56.
6. Shenzhen Xin Yuan Electronic Technology Co.: LILYGO TTGO T-Display ESP32 Development Board. Sitio web de LILYGO (2020).
7. Aivar Annamaa: Learn to code with Thonny. Sitio web de Fedora Magazine (2018).
8. Ciberninjas: Thonny, un IDE de Python específico para principiantes y Raspberry Pi. Sitio web de Ciberninjas (2020).
9. Durga Prasad Acharya: Los 11 mejores IDE de Python para potenciar el desarrollo y la depuración. Sitio web de Geekflare (2021).
10. Wikipedia: Thonny. Sitio web de Wikipedia (2022).
11. 330ohms: ¿Por qué aprender MicroPython te asegura un mejor futuro como programador? Sitio web de 330ohms (2021).
12. Damien George: MicroPython. Sitio web de MicroPython (2018).
13. Damien P. George, Paul Sokolovsky y contribuyentes: Quick reference for the ESP32. Sitio web de MicroPython (2022).