|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | **FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES** | |
|  | | | | |
| **MAESTRIA**: Automatización para la  Industria 4.0 | | | | **ASIGNATURA**: APLICACIONES IIot |
| **NRO. PRÁCTICA**: | 1 | **TÍTULO PRÁCTICA**: Conexión de microprocesadores embebidos en la nube para el monitoreo y control | | |
| **OBJETIVO ALCANZADO:**  Desarrollar un sistema de monitoreo y control a través de la nube mediante microcontroladores embebidos.  **Objetivos específicos**  1. Conocer los sistemas embebidos ESP32 o Raspberry pi PICO  2. Establecer conexión a través del protocolo MQTT con plataformas IoT (Thingspeak, Adafruit, Firebase)  3. Verificar el funcionamiento mediante pruebas de conectividad. | | | | |
| **MARCO TEORICO** | | | | |
| **1. Requisitos y conocimientos previos**   1. Conocimientos de programación 2. Microcontroladores 3. Protocolo MQTT | | | | |
| **2. Equipos, instrumentos y software**     |  |  |  |  | | --- | --- | --- | --- | | **Descripción** | **Cantidad** | **Marca** | **Identificación / serie** | | Computadora S.O. Windows 10 u otra versión, 32 o 64 bits | 1 | - | - | | Microcontrolador (ESP32 o Raspberry Pi -PICO) | 1 | - | - | | Pulsante | 1 | - | - | | Potenciómetro |  |  |  | | Led |  |  |  | |  |  |  |  |     . | | | | |
| **3. Exposición**    **ESP32** es la denominación de una familia de chips SoC de bajo coste y consumo de energía, con tecnología Wi-Fi y Bluetooth de modo dual integrada. El ESP32 emplea un microprocesador Tensilica Xtensa LX6 en sus variantes de simple y doble núcleo e incluye interruptores de antena, balun de radiofrecuencia, amplificador de potencia, amplificador receptor de bajo ruido, filtros, y módulos de administración de energía. El ESP32 fue creado y desarrollado por Espressif Systems y es fabricado por TSMC utilizando su proceso de 40 nm..  Las características del ESP32 incluyen:  **Procesador:**   * CPU: microprocesador de 32-bit Xtensa LX6 de doble núcleo (o de un solo núcleo), operando a 160 o 240 MHz y rindiendo hasta 600 DMIPS * Co-procesador de ultra baja energía (ULP)   Memoria: 520 KiB SRAM  Conectividad inalámbrica:   * Wi-Fi: 802.11 b/g/n * Bluetooth: v4.2 BR/EDR y BLE   Interfaces periféricas:   * 12-bit SAR ADC de hasta 18 canales * 2 × 8-bit DACs * 10 × sensores de tacto (sensores capacitivos GPIOs) * 4 × SPI * 2 × interfaces I²S * 2 × interfaces I²C * 3 × UART * Controlador host SD/SDIO/CE-ATA/MMC/eMMC * Controlador esclavo SDIO/SPI * Interfaz Ethernet MAC con DMA dedicado y soporte para el protocolo IEEE 1588 Precision Time Protocol * Bus CAN 2.0 * Controlador remoto infrarrojo (TX/RX, hasta 8 canales) * Motor PWM * LED PWM (hasta 16 canales) * Sensor de efecto Hall * Pre-amplificador analógico de ultra baja potencia   Seguridad:   * Soporta todas las características de seguridad estándar de IEEE 802.11, incluyendo WFA, WPA/WPA2 y WAPI * Arranque seguro * Cifrado flash * 1024-bit OTP, hasta 768-bit para clientes * Criptografía acelerada por hardware: AES, SHA-2, RSA, criptografía de curva elíptica (ECC), generador de números aleatorios (RNG)   Administración de energía:   * Regulador interno de baja caída * Dominio de poder individual para RTC * Corriente de 5μA en modo de suspensión profundo * Despierta por interrupción de GPIO, temporizador, medidas de ADC, interrupción por sensor de tacto capacitivo * almacenamiento incorporado   El ESP32 incluye la siguiente memoria integrada:​  Almacenamiento Tamaño  SRAM 520 KiBit  Memoria flash 448 KiBit  NVRAM 16 KiBit Raspberry Pi Pico   La Raspberry Pi Pico es una tarjeta de desarrollo que cuenta con un microcontrolador construido en silicio, diseñado por Raspberry Pi. Es una placa de microcontrolador de bajo costo y alto rendimiento con interfaces digitales flexibles.  **Características:**   * El chip microcontrolador RP2040 diseñado por Raspberry Pi en el Reino Unido. * Procesador ARM Cortex M0+ de doble núcleo, reloj flexible que funciona hasta 133 MHz. * 264kB de SRAM, y 2MB de memoria Flash a bordo. * Pads con un troquelado que permite soldar la tarjeta directamente sobre tu pcb. * USB 1.1 con soporte para anfitrión o dispositivo. * Modo de funcionamiento de baja potencia (sleep) y modos de inactividad. * Programar arrastrando el archivo al almacenamiento masivo del USB. * 26 pines GPIO multifunción. * 2×SPI, 2×I2C, 2×UART, 3×12-bit ADC, 16×canales PWM controlables. * Reloj y temporizador preciso en el chip. * Sensor de temperatura. * Librerías de coma flotante en el microcontrolador. * 8 Máquinas de estado con IO programables (PIO) con soporte para uso personalizado de periféricos**.** | | | | |
| **ACTIVIDADES DESARROLLADAS** | | | | |
| **observamos que aparece como una nueva unidad de disco**    **Conectamos nuestro microcontrolador Raspberry Pico W(RPPw) y observamos que en Administrador de dispositivos aparece que el driver no está instalado.**    **Luego, descargamos el firmware de Python para RPPw.**  **<https://micropython.org/download/RPI_PICO_W/>**    Copiamos el firmware que descargamos, dentro de la unidad de disco RPI-RP2    Se reiniciará RPPw automáticamente. | | | | |
| **3.** instalación de IDE de desarrollo Thonny  <https://thonny.org/> | | | | |
| **5.** Plataforma Thonny    El dispositivo RPPw debe estar conectado al usb del computador.  En la esquina inferior derecha debe aparecer RPP w y luego hacemos clic sobre *Micropython (Rasberry Pi Pco – Broad CDC @ COM3)*    **Aparecerá en la Consola el siguiente mensaje**    **Esto nos indica que esta listo para trabajar.** | | | | |
| **Para instalar los paquetes necesarios** | | | | |
|  | | | | |
|  | | | | |
| **Programa  import network**  **import urequests as requests**  **import time**  **from machine import Pin**  **# Configuración WiFi**  **WIFI\_SSID = 'PaulCh'**  **WIFI\_PASSWORD = 'Paul1234'**  **# Configuración Firebase**  **FIREBASE\_URL = 'https://maestria-iiot-rppico-default-rtdb.firebaseio.com/'**  **FIREBASE\_API\_KEY = 'AIzaSyCG9om499BzFsSaqWuJ4k6l-uE9r0-xWic'**  **FIREBASE\_EMAIL = 'paulchasi@hotmail.com'**  **FIREBASE\_PASSWORD = 'Paul2179'**  **# Variables globales**  **id\_token = None**  **def connect\_wifi():**  **wlan = network.WLAN(network.STA\_IF)**  **wlan.active(True)**  **if not wlan.isconnected():**  **print("Conectando a WiFi...")**  **wlan.connect(WIFI\_SSID, WIFI\_PASSWORD)**  **for \_ in range(15):**  **if wlan.isconnected():**  **break**  **time.sleep(1)**  **if not wlan.isconnected():**  **raise RuntimeError("Error de conexión WiFi")**  **print("✅ WiFi conectado")**  **print("📶 IP:", wlan.ifconfig()[0])**  **def firebase\_auth():**  **global id\_token**  **auth\_url = f"https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key={FIREBASE\_API\_KEY}"**  **auth\_data = {**  **"email": FIREBASE\_EMAIL,**  **"password": FIREBASE\_PASSWORD,**  **"returnSecureToken": True**  **}**    **try:**  **response = requests.post(auth\_url, json=auth\_data)**  **if response.status\_code == 200:**  **id\_token = response.json()['idToken']**  **print("🔑 Autenticación exitosa con Firebase")**  **return True**  **print(f"❌ Error de autenticación (HTTP {response.status\_code}):", response.text)**  **return False**  **except Exception as e:**  **print("❌ Error en auth:", str(e))**  **return False**  **finally:**  **if 'response' in locals():**  **response.close()**  **def read\_firebase\_data():**  **url = f"{FIREBASE\_URL}random.json?auth={id\_token}" # Lee todo el nodo random**  **try:**  **response = requests.get(url)**  **if response.status\_code == 200:**  **data = response.json()**    **# Visualización mejorada**  **print("\n" + "═"\*35)**  **print("📊 DATOS DE FIREBASE")**  **print("═"\*35)**  **print(f"🔢 Data: {data.get('data', 'N/D')}")**  **print(f"🌡 Temperatura: {data.get('temperatura', 'N/D')}°C")**  **print("═"\*35)**    **return data**  **print(f"❌ Error de lectura (HTTP {response.status\_code}):", response.text)**  **return None**  **except Exception as e:**  **print("❌ Error de conexión:", str(e))**  **return None**  **finally:**  **if 'response' in locals():**  **response.close()**  **def main():**  **connect\_wifi()**    **if not firebase\_auth():**  **print("⚠️ No se puede continuar sin autenticación")**  **return**    **print("\n🔄 Iniciando monitoreo de Firebase...")**  **while True:**  **read\_firebase\_data()**  **time.sleep(10) # Lectura cada 3 segundos**  **if \_\_name\_\_ == "\_\_main\_\_":**  **main()** | | | | |
|  | | | | |
|  | | | | |

***Nombre de estudiante*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***Firma de estudiante*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**