Conexión a Ethernet

Para realizar una conexión por medio de Ethernet se deberá tener incorporadas las siguientes librerías cargadas en la carpeta ***Circuitpy:***

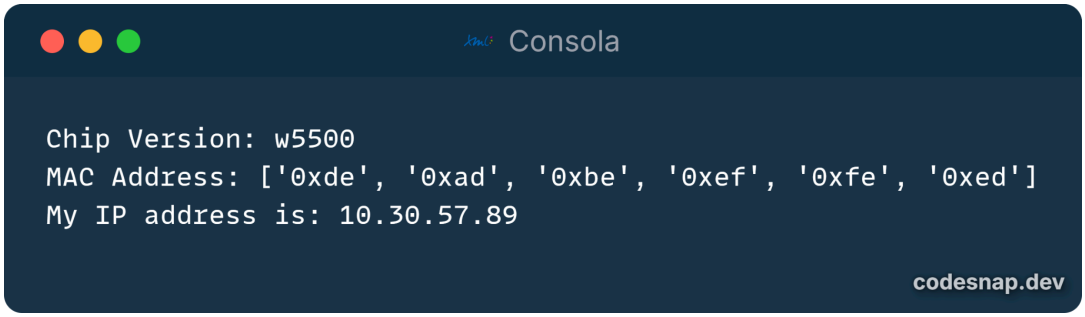
- ➔ **Adafruit_Bus_Device**
- ➔ **Adafruit_Wiznet5K**
- ➔ **Adafruit_Connection_Manager**
- ➔ **Adafruit_Requests**
- ➔ **Adafruit_Ticks**

Prueba inicial de conexión

code.py

```
# ================================
# IMPORTACIONES
# ================================
import time
import board
import busio
import digitalio

import adafruit_connection_manager
import adafruit_requests
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K


# ================================
# CONFIGURACIÓN DE HARDWARE
# ================================
def configurar_spi_cs():
    """Configura el bus SPI y el pin CS para el WIZnet5K"""
    cs = digitalio.DigitalInOut(board.GP9)
    spi_bus = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
    return spi_bus, cs


# ================================
# INICIALIZAR ETHERNET
# ================================
def inicializar_ethernet(spi_bus, cs):
    """Inicializa la interfaz Ethernet usando WIZNET5K"""
    return WIZNET5K(spi_bus, cs)


# ================================
# INICIALIZAR SESIÓN REQUESTS
# ================================
def crear_sesion_requests(eth):
    """Crea una sesión de requests usando adafruit_connection_manager"""
    pool = adafruit_connection_manager.get_radio_socketpool(eth)
    ssl_context = adafruit_connection_manager.get_radio_ssl_context(eth)
    return adafruit_requests.Session(pool, ssl_context)
```

```python
# ================================
# MOSTRAR INFORMACIÓN DE RED
# ================================
def mostrar_info_red(eth):
    """Muestra información del chip, MAC, IP, y resolución DNS"""
    print("Chip Version:", eth.chip)
    print("MAC Address:", [hex(i) for i in eth.mac_address])
    print("My IP address is:", eth.pretty_ip(eth.ip_address))


# ================================
# FUNCIÓN PRINCIPAL
# ================================
def main():
    print("Wiznet5k WebClient Test")

    # Configurar hardware
    spi_bus, cs = configurar_spi_cs()

    # Inicializar Ethernet
    eth = inicializar_ethernet(spi_bus, cs)

    # Crear sesión de requests
    requests = crear_sesion_requests(eth)

    # Mostrar información de red
    mostrar_info_red(eth)

    # eth._debug = True  # Descomentar  ver más logs


# ================================
# PUNTO DE ENTRADA
# ================================
if __name__ == "__main__":
    main()
```



Una vez que se tiene la conexión, se debe verificar su dirección, IP y el modelo del chip. Como siguiente paso se probará tanto request Get, Post, Service y MQTT.

Request GET

code.py

```
# ================================
# IMPORTACIONES
# ================================
import time
import board
import busio
import digitalio

import adafruit_connection_manager
import adafruit_requests
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K


# ================================
# CONSTANTES
# ================================
TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"


# ================================
# CONFIGURACIÓN DE HARDWARE
# ================================
def configurar_spi_cs():
    """Configura el bus SPI y el pin CS para el WIZnet5K"""
    cs = digitalio.DigitalInOut(board.GP9)
    spi_bus = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
    return spi_bus, cs
```

```python
# ================================
# INICIALIZACIÓN DE ETHERNET
# ================================
def inicializar_ethernet(spi_bus, cs):
    """Inicializa la interfaz Ethernet usando WIZNET5K"""
    return WIZNET5K(spi_bus, cs)


# ================================
# SESIÓN DE REQUESTS
# ================================
def crear_sesion_requests(eth):
    """Crea una sesión de requests usando adafruit_connection_manager"""
    pool = adafruit_connection_manager.get_radio_socketpool(eth)
    ssl_context = adafruit_connection_manager.get_radio_ssl_context(eth)
    return adafruit_requests.Session(pool, ssl_context)


# ================================
# MOSTRAR INFORMACIÓN DE RED
# ================================
def mostrar_info_red(eth):
    """Muestra información del chip, MAC, IP y resolución DNS"""
    print("Chip Version:", eth.chip)
    print("MAC Address:", [hex(i) for i in eth.mac_address])
    print("My IP address is:", eth.pretty_ip(eth.ip_address))
    hostname = "adafruit.com"
    ip = eth.pretty_ip(eth.get_host_by_name(hostname))
    print(f"IP lookup {hostname}: {ip}")


# ================================
# PETICIÓN HTTP
# ================================
def obtener_texto_url(requests, url):
    """Realiza una petición GET y muestra el texto"""
    print("Fetching text from", url)
    r = requests.get(url)
    print("-" * 40)
    print(r.text)
    r.close()
```

```python
# ================================
# FUNCIÓN PRINCIPAL
# ================================
def main():
    print("Wiznet5k WebClient Test")

    # Configuración e inicialización
    spi_bus, cs = configurar_spi_cs()
    eth = inicializar_ethernet(spi_bus, cs)
    requests = crear_sesion_requests(eth)

    # Información de red
    mostrar_info_red(eth)

    # Realizar petición HTTP
    obtener_texto_url(requests, TEXT_URL)

    # eth._debug = True  # Activar si necesitas ver más logs

    time.sleep(1)


# ================================
# PUNTO DE ENTRADA
# ================================
if __name__ == "__main__":
    main()
```

Consola

```
Fetching text from http://wifitest.adafruit.com/testwifi/index.html
----------------------------------------
This is a test of Adafruit WiFi!
If you can read this, its working :)
----------------------------------------
```

Request POST

code.py

```
# ================================
# IMPORTACIONES
# ================================
import time
import board
import busio
import digitalio

import adafruit_connection_manager
import adafruit_requests
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
# ================================
# CONSTANTES
# ================================
JSON_POST_URL = "http://httpbin.org/post"
DATA_A_ENVIAR = "31F"
# ================================
# CONFIGURACIÓN DE HARDWARE
# ================================
def configurar_spi_cs():
    """Configura el bus SPI y el pin CS para el WIZnet5K"""
    cs = digitalio.DigitalInOut(board.GP9)
    spi_bus = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
    return spi_bus, cs
# ================================
# INICIALIZAR ETHERNET
# ================================
def inicializar_ethernet(spi_bus, cs):
    """Inicializa la interfaz Ethernet usando WIZNET5K"""
    return WIZNET5K(spi_bus, cs)
# ================================
# SESIÓN DE REQUESTS
# ================================
def crear_sesion_requests(eth):
    """Crea una sesión de requests usando adafruit_connection_manager"""
    pool = adafruit_connection_manager.get_radio_socketpool(eth)
    ssl_context = adafruit_connection_manager.get_radio_ssl_context(eth)
    return adafruit_requests.Session(pool, ssl_context)
```

```python
# ================================
# PETICIÓN POST
# ================================
def enviar_dato_post(requests, url, data):
    """Envía un dato como POST a una URL y muestra la respuesta"""
    print(f"POSTing data to {url}: {data}")
    with requests.post(url, data=data) as response:
        print("-" * 40)
        json_resp = response.json()
        print("Data received from server:", json_resp["data"])
        print("-" * 40)
# ================================
# FUNCIÓN PRINCIPAL
# ================================
def main():
    print("Wiznet5k WebClient Test")

    # Inicialización de hardware y red
    spi_bus, cs = configurar_spi_cs()
    eth = inicializar_ethernet(spi_bus, cs)
    requests = crear_sesion_requests(eth)

    # Enviar POST
    enviar_dato_post(requests, JSON_POST_URL, DATA_A_ENVIAR)
# ================================
# PUNTO DE ENTRADA
# ================================
if __name__ == "__main__":
    main()
```

codesnap.dev

Consola

```
Wiznet5k WebClient Test
POSTing data to http://httpbin.org/post: 31F
----------------------------------------
Data received from server: 31F
----------------------------------------
```

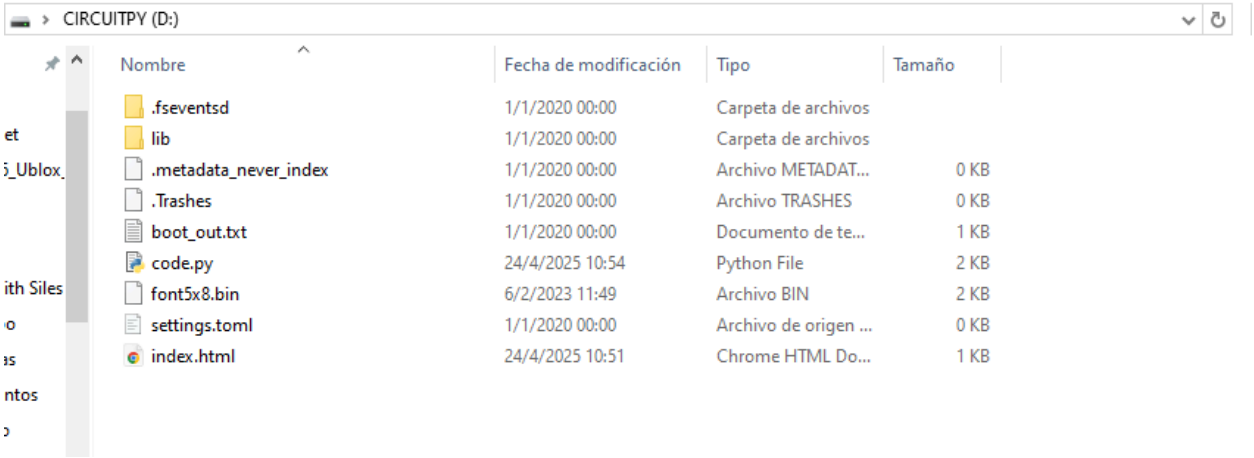
codesnap.dev

Conexión local server

El objetivo principal es permitir la creación de widgets personalizados utilizando HTML, CSS y JS. Se proporcionarán ejemplos de widgets con una estructura similar a la de Arduino, implementados en la placa Archi - archi Net.

Vinculación de archivo HTML

Cree un archivo llamado "index.html" y guárdelo en el directorio "D:/". Este archivo será invocado desde "code.py".



Prueba de archivo index.html

index.html

```
<!-- archivo: index.html -->
<!DOCTYPE html>
<html>
  <head><title>Hola</title></head>
  <body><h1>Hola desde archivo</h1></body>
</html>
```

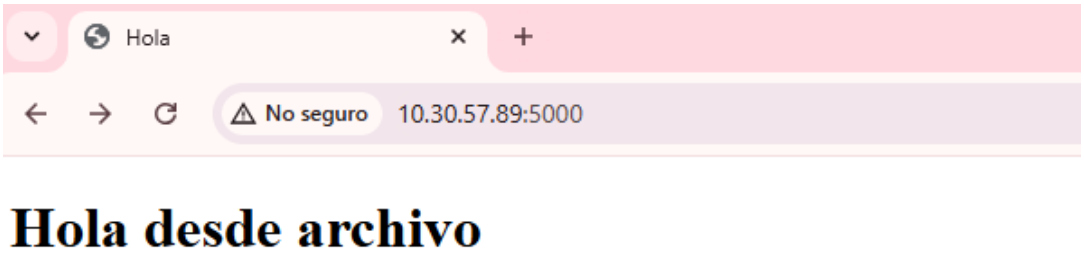
codesnap.dev

Código Archi

code.py

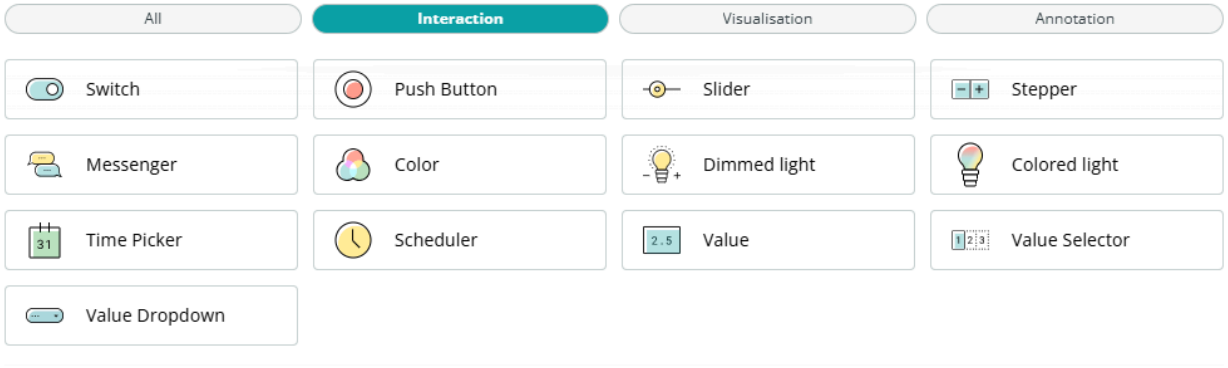
```python
import board
import busio
import digitalio
import adafruit_connection_manager
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_httpserver import Server, Request, Response, JSONResponse

# ==============================
# CONFIGURACIÓN DE ETHERNET
# ==============================
cs = digitalio.DigitalInOut(board.GP9)
spi_bus = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
eth = WIZNET5K(spi_bus, cs)
pool = socketpool.SocketPool(eth)
server = Server(pool, "/static", debug=True)

# ==============================
# RUTA HTML PRINCIPAL
# ==============================
@server.route("/")
def pagina_switch(request: Request):
    with open("/index.html", "r") as file:
        contenido_html = file.read()
    return Response(request, contenido_html.encode("utf-8"), content_type="text/html")

# ==============================
# INICIAR SERVIDOR
# ==============================
print("Iniciando servidor en:", eth.pretty_ip(eth.ip_address))
server.serve_forever(str(eth.pretty_ip(eth.ip_address)))
```

codesnap.dev

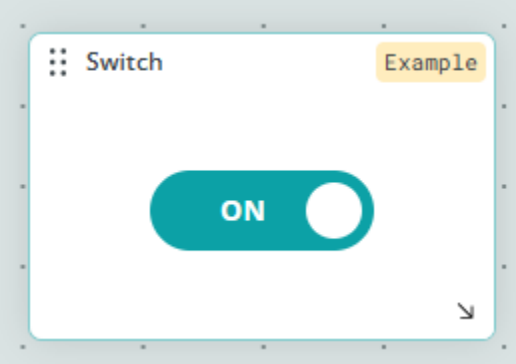


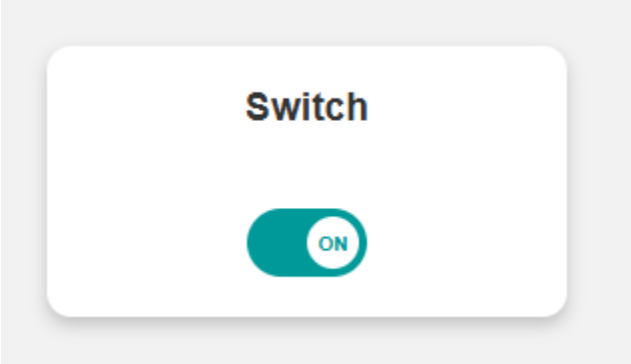
Widget de interaction

Arduino CLOUD



Widget Switch



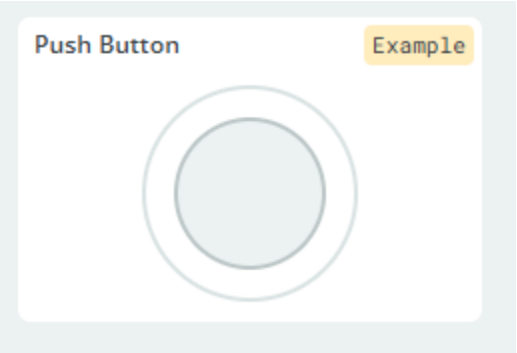


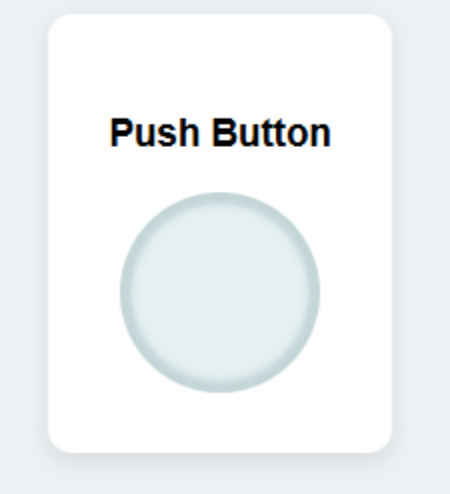
Se debe insertar el código switch en el archivo index.html. En cuanto al código, se agrega una función toggle.

code.py

```python
import board
import busio
import digitalio
import adafruit_connection_manager
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_httpserver import Server, Request, Response, JSONResponse

# ==============================
# CONFIGURACIÓN DE ETHERNET
# ==============================
cs = digitalio.DigitalInOut(board.GP9)
spi_bus = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
eth = WIZNET5K(spi_bus, cs)
pool = socketpool.SocketPool(eth)
server = Server(pool, "/static", debug=True)

# ==============================
# ESTADO DEL SWITCH
# ==============================
estado_switch = {"encendido": True}

# ==============================
# RUTA HTML PRINCIPAL
# ==============================
@server.route("/")
def pagina_switch(request: Request):
    with open("/index.html", "r") as file:
        contenido_html = file.read()
    return Response(request, contenido_html.encode("utf-8"), content_type="text/html")
# ==============================
# RUTA PARA CAMBIAR ESTADO
# ==============================
@server.route("/toggle", methods=["POST"])
def cambiar_estado(request: Request):
    datos = request.json()
    estado_switch["encendido"] = datos.get("encendido", False)
    print("Nuevo estado:", "ENCENDIDO" if estado_switch["encendido"] else "APAGADO")
    return JSONResponse(request, {"estado": estado_switch["encendido"]})

# ==============================
# INICIAR SERVIDOR
# ==============================
print("Iniciando servidor en:", eth.pretty_ip(eth.ip_address))
server.serve_forever(str(eth.pretty_ip(eth.ip_address)))
```

Push Button





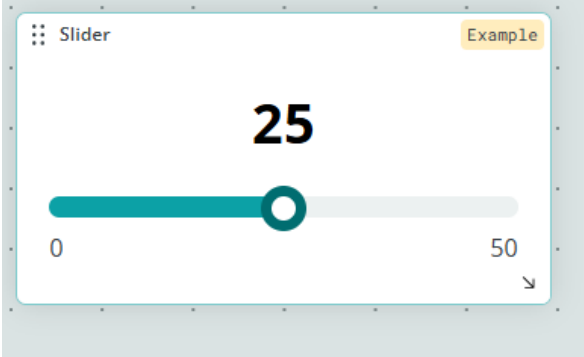
Se debe insertar el código push button en el archivo index.html. En cuanto al código, se agrega una función manejar pulsaciones.

code.py

```
# ============================================
# IMPORTACIÓN DE MÓDULOS NECESARIOS
# ============================================
import board
import busio
import digitalio

import adafruit_connection_manager
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_httpserver import Server, Request, Response, JSONResponse

# ============================================
# CONFIGURACIÓN DE HARDWARE Y RED ETHERNET
# ============================================

def configurar_ethernet():
    cs = digitalio.DigitalInOut(board.GP9)  # Pin CS del W5500
    spi_bus = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)  # SPI para W5500
    wiznet = WIZNET5K(spi_bus, cs)  # Crear objeto de red
    return wiznet

# ============================================
# CONFIGURACIÓN DEL SERVIDOR HTTP
# ============================================

def configurar_servidor(eth):
    pool = socketpool.SocketPool(eth)
    servidor = Server(pool, "/static", debug=True)
    return servidor
```

```python
# ============================================
# RUTA PRINCIPAL PARA SERVIR EL HTML
# ============================================

def definir_ruta_html(servidor):
    @servidor.route("/")
    def pagina_principal(request: Request):
        with open("/index.html", "r") as archivo:
            contenido = archivo.read()
        return Response(request, contenido.encode("utf-8"), content_type="text/html")

# ============================================
# RUTA PARA RECIBIR EL EVENTO DE PULSACIÓN
# ============================================

def definir_ruta_push(servidor):
    @servidor.route("/push", methods=["POST"])
    def manejar_pulsacion(request: Request):
        datos = request.json()
        if datos and datos.get("presionado", False):
            print("🔘 Botón presionado desde el navegador.")
        return JSONResponse(request, {"ok": True})

# ============================================
# FUNCIÓN PRINCIPAL
# ============================================

def main():
    print("Inicializando Ethernet...")
    eth = configurar_ethernet()

    print("Configurando servidor HTTP...")
    servidor = configurar_servidor(eth)

    # Registrar rutas
    definir_ruta_html(servidor)
    definir_ruta_push(servidor)

    # Iniciar servidor
    ip = eth.pretty_ip(eth.ip_address)
    print("🌐 Servidor iniciado en:", ip)
    servidor.serve_forever(str(ip))

# ============================================
# EJECUCIÓN DEL PROGRAMA
# ============================================
main()
```

Slider



code.py

```
# ==========================================
# IMPORTACIÓN DE MÓDULOS
# ==========================================
import board
import busio
import digitalio

import adafruit_connection_manager
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_httpserver import Server, Request, Response, JSONResponse

# ==========================================
# CONFIGURACIÓN ETHERNET
# ==========================================
def configurar_ethernet():
    cs = digitalio.DigitalInOut(board.GP9)
    spi = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
    eth = WIZNET5K(spi, cs)
    return eth

# ==========================================
# CONFIGURACIÓN SERVIDOR HTTP
# ==========================================
def configurar_servidor(eth):
    pool = socketpool.SocketPool(eth)
    servidor = Server(pool, "/static", debug=True)
    return servidor
```

```python
# ============================================
# RUTA: PÁGINA HTML
# ============================================
def definir_ruta_html(server):
    @server.route("/")
    def mostrar_pagina(request: Request):
        with open("/index.html", "r") as archivo:
            html = archivo.read()
        return Response(request, html.encode("utf-8"), content_type="text/html")

# ============================================
# RUTA: SLIDER (POST)
# ============================================
def definir_ruta_slider(server):
    @server.route("/slider", methods=["POST"])
    def manejar_slider(request: Request):
        datos = request.json()
        if datos and "valor" in datos:
            print(f"🎚️ Slider en: {datos['valor']}")
        return JSONResponse(request, {"ok": True})

# ============================================
# MAIN: INICIALIZACIÓN
# ============================================
def main():
    print("🔌 Inicializando Ethernet...")
    eth = configurar_ethernet()

    print("🖥️ Iniciando servidor...")
    server = configurar_servidor(eth)

    definir_ruta_html(server)
    definir_ruta_slider(server)

    ip = eth.pretty_ip(eth.ip_address)
    print("✅ Servidor activo en:", ip)
    server.serve_forever(str(ip))

main()
```

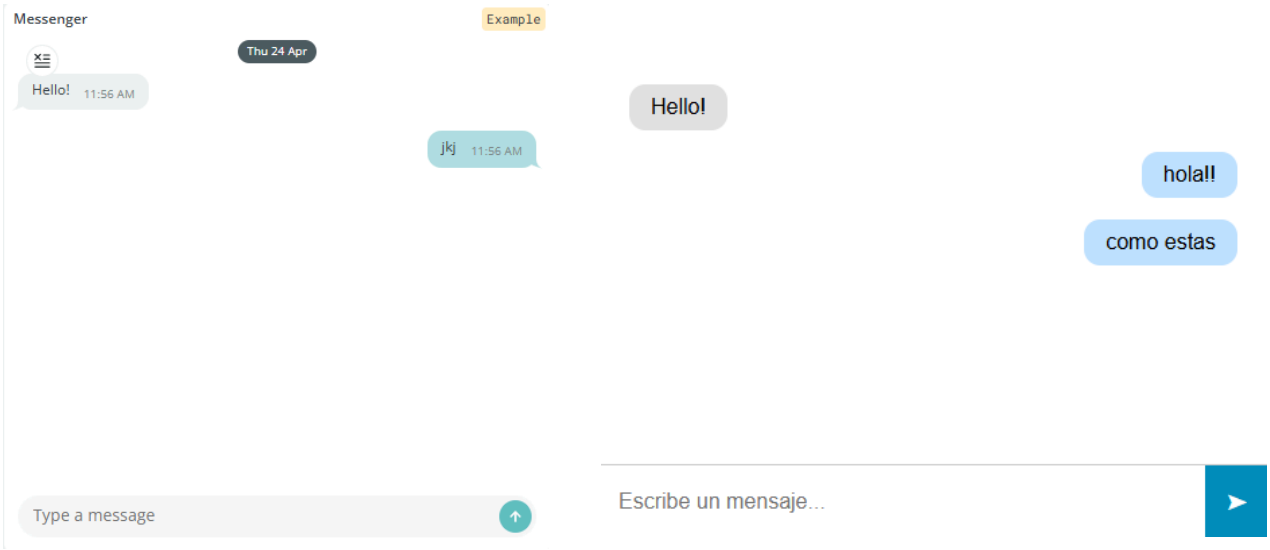
Stepper

code.py

```
# ============================================
# IMPORTACIÓN DE MÓDULOS
# ============================================
import board
import busio
import digitalio

from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_httpserver import Server, Request, Response, JSONResponse

# ============================================
# CONFIGURACIÓN DE ETHERNET
# ============================================
def configurar_ethernet():
    cs = digitalio.DigitalInOut(board.GP9)
    spi = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
    eth = WIZNET5K(spi, cs)
    return eth

# ============================================
# CONFIGURACIÓN DE SERVIDOR
# ============================================
def configurar_servidor(eth):
    pool = socketpool.SocketPool(eth)
    server = Server(pool, "/static", debug=True)
    return server
```

```python
# ============================================
# RUTA PRINCIPAL: HTML
# ============================================
def ruta_html(server):
    @server.route("/")
    def index(request: Request):
        with open("/index.html", "r") as archivo:
            contenido = archivo.read()
        return Response(request, contenido.encode("utf-8"), content_type="text/html")

# ============================================
# RUTA: VALOR DEL STEPPER
# ============================================
def ruta_stepper(server):
    @server.route("/stepper", methods=["POST"])
    def actualizar_valor(request: Request):
        datos = request.json()
        if datos and "valor" in datos:
            print(f"🔄 Stepper actualizado a: {datos['valor']}")
        return JSONResponse(request, {"ok": True})

# ============================================
# INICIO
# ============================================
def main():
    print("🌐 Configurando Ethernet...")
    eth = configurar_ethernet()

    print("🖥️ Iniciando servidor HTTP...")
    server = configurar_servidor(eth)

    ruta_html(server)
    ruta_stepper(server)

    ip = eth.pretty_ip(eth.ip_address)
    print("✅ Servidor corriendo en:", ip)
    server.serve_forever(str(ip))

main()
```

Messenger

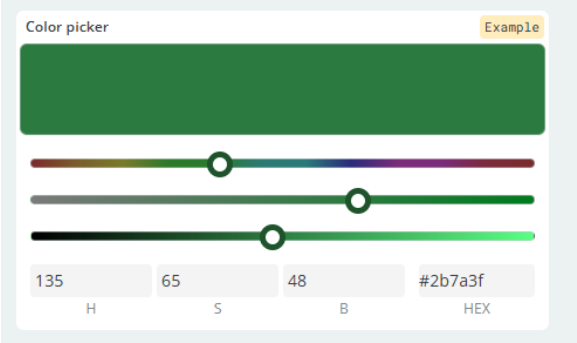


code.py

```
# ============================================
# IMPORTAR MÓDULOS
# ============================================
import board
import busio
import digitalio

from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_httpserver import Server, Request, Response, JSONResponse

# ============================================
# CONFIGURAR ETHERNET
# ============================================
def configurar_ethernet():
    cs = digitalio.DigitalInOut(board.GP9)
    spi = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
    eth = WIZNET5K(spi, cs)
    return eth

# ============================================
# CONFIGURAR SERVIDOR
# ============================================
def configurar_servidor(eth):
    pool = socketpool.SocketPool(eth)
    server = Server(pool, "/static", debug=True)
    return server
```

```python
# ============================================
# RUTAS
# ============================================
def rutas(server):
    @server.route("/")
    def index(request: Request):
        with open("/index.html", "r") as archivo:
            contenido = archivo.read()
        return Response(request, contenido.encode("utf-8"), content_type="text/html")

    @server.route("/mensaje", methods=["POST"])
    def recibir_mensaje(request: Request):
        datos = request.json()
        if datos and "texto" in datos:
            print("📥 Mensaje recibido:", datos["texto"])
        return JSONResponse(request, {"ok": True})

# ============================================
# FUNCIÓN PRINCIPAL
# ============================================
def main():
    eth = configurar_ethernet()
    server = configurar_servidor(eth)
    rutas(server)

    ip = eth.pretty_ip(eth.ip_address)
    print("🟢 Servidor escuchando en:", ip)
    server.serve_forever(str(ip))

main()
```

Color picker



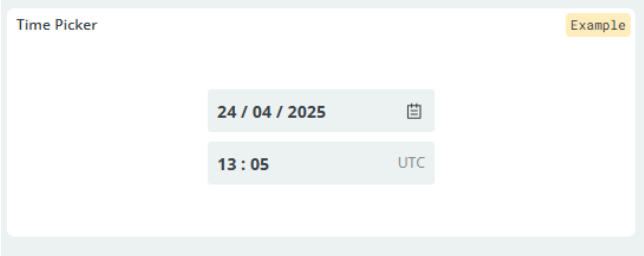
code.py

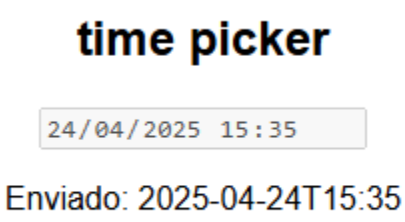
```python
# ======================== IMPORTACIÓN DE MÓDULOS ========================
import board
import busio
import digitalio
import neopixel
import adafruit_connection_manager
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_httpserver import Server, Request, Response, JSONResponse
from adafruit_pixel_framebuf import PixelFramebuffer


# ======================== BLOQUE: CONFIGURACIÓN DE MATRIZ ========================
def configurar_matriz():
    pixeles = neopixel.NeoPixel(board.NEOPIXEL, 64, brightness=0.2, auto_write=False)
    matriz = PixelFramebuffer(pixeles, 8, 8, alternating=False)
    return matriz


# ======================== BLOQUE: CONFIGURACIÓN DE RED ========================
def configurar_red():
    cs = digitalio.DigitalInOut(board.GP9)
    spi_bus = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
    eth = WIZNET5K(spi_bus, cs)
    pool = socketpool.SocketPool(eth)
    server = Server(pool, "/static", debug=True)
    return eth, server


# ======================== BLOQUE: RESPUESTA CON HTML ========================
def manejar_html(request: Request) -> Response:
    with open("/index.html", "r") as f:
        html = f.read()
    return Response(request, html.encode("utf-8"), content_type="text/html")
```

```python
# ======================== BLOQUE: CAMBIAR COLOR DE MATRIZ ========================
def aplicar_color_a_matriz(hex_color: str, matriz):
    r = int(hex_color[1:3], 16)
    g = int(hex_color[3:5], 16)
    b = int(hex_color[5:7], 16)

    for y in range(8):
        for x in range(8):
            matriz.pixel(x, y, (r, g, b))
    matriz.display()
    print(f"Color actualizado: {r}, {g}, {b}")


# ======================== BLOQUE: MANEJADOR DE RUTA COLOR ========================
def ruta_color(request: Request, matriz) -> JSONResponse:
    data = request.json()
    hex_color = data.get("color", "#000000")
    aplicar_color_a_matriz(hex_color, matriz)
    return JSONResponse(request, {"ok": True})


# ======================== BLOQUE PRINCIPAL ========================
def main():
    matriz = configurar_matriz()
    eth, server = configurar_red()

    # Ruta para cargar HTML
    @server.route("/")
    def root(request: Request):
        return manejar_html(request)

    # Ruta POST para aplicar color
    @server.route("/color", methods=["POST"])
    def post_color(request: Request):
        return ruta_color(request, matriz)

    print("Servidor en:", eth.pretty_ip(eth.ip_address))
    server.serve_forever(str(eth.pretty_ip(eth.ip_address)))


# ======================== INICIO DEL PROGRAMA ========================
main()
```


Los widgets "Dimmed Light" y "Colored Light" no se implementarán debido a su similitud con el slider y el selector de color.

Time Picker





code.py

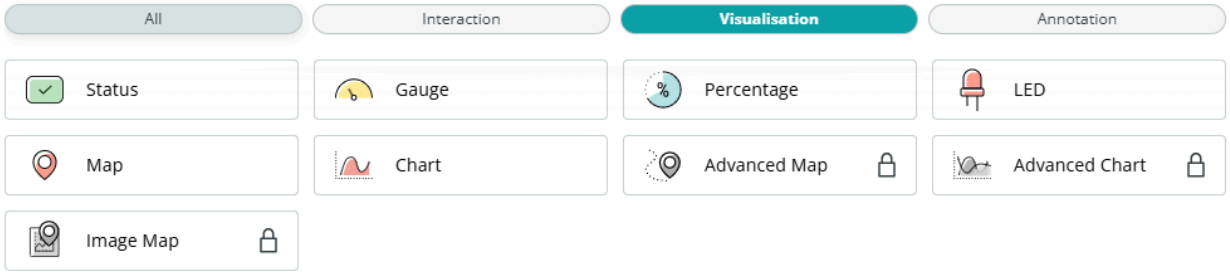
```python
# ======================== IMPORTACIONES ========================
import board
import busio
import digitalio
import neopixel
import adafruit_connection_manager
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
import adafruit_wiznet5k.adafruit_wiznet5k_socketpool as socketpool
from adafruit_httpserver import Server, Request, Response, JSONResponse
from adafruit_pixel_framebuf import PixelFramebuffer


# ======================== MATRIZ OPCIONAL ========================
def configurar_matriz():
    pixeles = neopixel.NeoPixel(board.NEOPIXEL, 64, brightness=0.2, auto_write=False)
    matriz = PixelFramebuffer(pixeles, 8, 8, alternating=False)
    return matriz


# ======================== RED Y SERVIDOR ========================
def configurar_red():
    cs = digitalio.DigitalInOut(board.GP9)
    spi_bus = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
    eth = WIZNET5K(spi_bus, cs)
    pool = socketpool.SocketPool(eth)
    server = Server(pool, "/static", debug=True)
    return eth, server


# ======================== CARGAR HTML ========================
def manejar_html(request: Request) -> Response:
    with open("/index.html", "r") as f:
        html = f.read()
    return Response(request, html.encode("utf-8"), content_type="text/html")


# ======================== PROCESAR FECHA Y HORA ========================
def manejar_hora(request: Request) -> JSONResponse:
    data = request.json()
    datetime_str = data.get("datetime", "")
    print("Fecha y hora recibidas:", datetime_str)
    return JSONResponse(request, {"ok": True})
```

```python
# ======================== PROGRAMA PRINCIPAL ========================
def main():
    matriz = configurar_matriz()  # Si no querés usar matriz, podés comentar esta línea
    eth, server = configurar_red()

    @server.route("/")
    def root(request: Request):
        return manejar_html(request)

    @server.route("/hora", methods=["POST"])
    def recibir_hora(request: Request):
        return manejar_hora(request)

    print("Servidor en:", eth.pretty_ip(eth.ip_address))
    server.serve_forever(str(eth.pretty_ip(eth.ip_address)))


# ========================= EJECUCIÓN =========================
main()
```


El planificador es similar al selector de tiempo, pero envía parámetros adicionales como duración, repetición y fin de recurrencia.

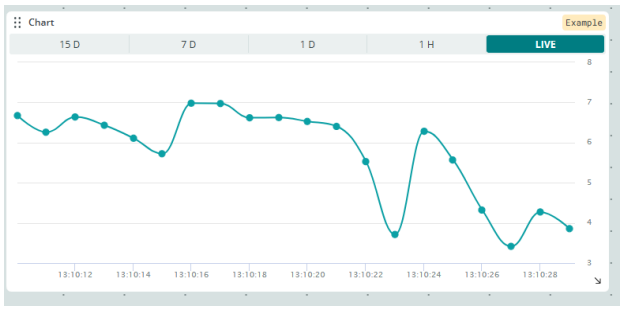
Lo mismo aplica para "Value", que es una vista del valor de una variable como hicimos anteriormente. También "Value Selector" es como tres botones en conjunto, y "Value Down" es igual, solo que varía el diseño.

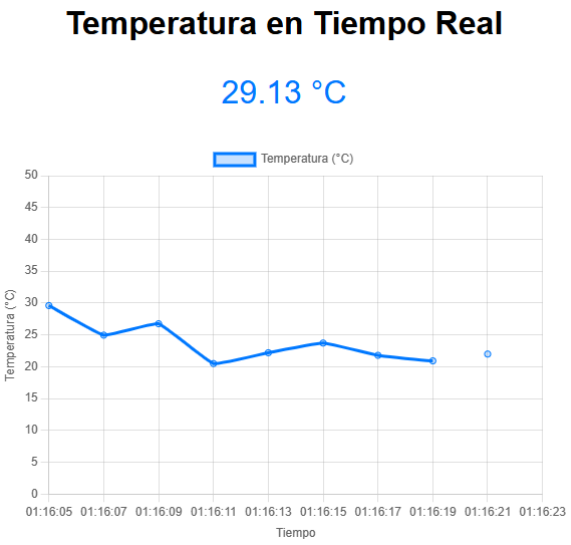
Widget de visualización



El widget de status consiste simplemente en enviar el valor post de una acción / variable como se ha visto anteriormente.

Widget Gauge

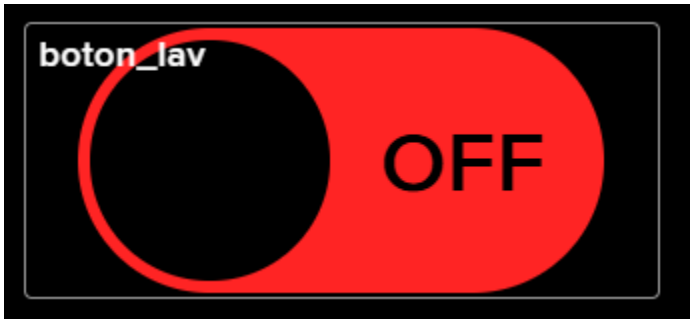




Se debe insertar el código del gráfico. Este código enviará un valor aleatorio en tiempo real y lo mostrará en el servicio.

Conexión desde MQTT (Broker local | Broker Adafruit)

Desde Adafruit io cloud, tienes la capacidad de crear tu propio dashboard y gestionar variables.



code.py

```python
import time
import adafruit_connection_manager
import board
import busio
from adafruit_wiznet5k.adafruit_wiznet5k import WIZNET5K
from digitalio import DigitalInOut
import adafruit_minimqtt.adafruit_minimqtt as MQTT

# --- Configuración del usuario ---
ADAFRUIT_AIO_USERNAME = "angieush"
ADAFRUIT_AIO_KEY = "aio_xChV08Gedt2VV8K7PMEsPX3BWktP"
FEED_BOTON = f"{ADAFRUIT_AIO_USERNAME}/feeds/boton"

# --- Funciones de conexión y MQTT ---
def inicializar_ethernet():
    cs = DigitalInOut(board.GP9)
    spi_bus = busio.SPI(board.GP10, MOSI=board.GP11, MISO=board.GP12)
    eth = WIZNET5K(spi_bus, cs)
    return eth

def crear_cliente_mqtt(eth):
    pool = adafruit_connection_manager.get_radio_socketpool(eth)
    ssl_context = adafruit_connection_manager.get_radio_ssl_context(eth)

    mqtt_client = MQTT.MQTT(
        broker="io.adafruit.com",
        username=ADAFRUIT_AIO_USERNAME,
        password=ADAFRUIT_AIO_KEY,
        is_ssl=False,  # Ethernet no usa SSL
        socket_pool=pool,
        ssl_context=ssl_context,
    )
```

```
    mqtt_client.on_connect = conectado
    mqtt_client.on_disconnect = desconectado
    mqtt_client.on_message = mensaje_recibido
    return mqtt_client

# --- Callbacks ---
def conectado(client, userdata, flags, rc):
    print(f"¡Conectado a Adafruit IO! Subscrito al feed: {FEED_BOTON}")
    client.subscribe(FEED_BOTON)

def desconectado(client, userdata, rc):
    print("Desconectado de Adafruit IO.")

def mensaje_recibido(client, topic, message):
    print(f"Mensaje recibido en {topic}: {message}")

# --- Bucle principal ---
def main():
    print("Inicializando Ethernet...")
    eth = inicializar_ethernet()

    print("Creando cliente MQTT...")
    mqtt_client = crear_cliente_mqtt(eth)

    print("Conectando a Adafruit IO...")
    mqtt_client.connect()

    print("Escuchando mensajes...")
    while True:
        mqtt_client.loop()
        time.sleep(1)

# --- Ejecutar programa principal ---
if __name__ == "__main__":
    main()
```