UNIVERSIDAD DEL QUINDÍO ®

Res.MEN 014915 - 02 AGO 2022

RENOVACIÓN ACREDITACIÓN

Custom chips en Wokwi

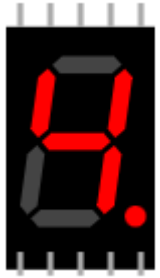
Ing. Luis Miguel Capacho V., M.Sc.

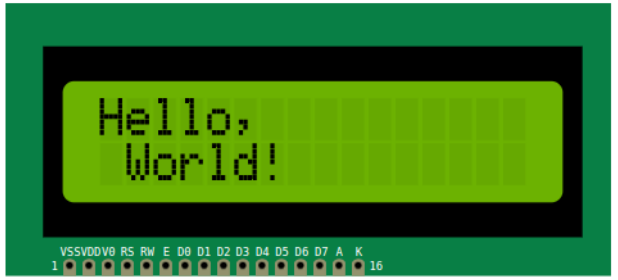
UNIQUINDÍO

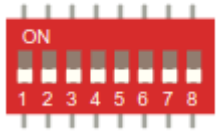
en conexión territorial

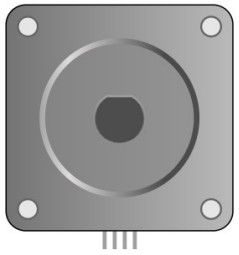
www.uniquindio.edu.co

Introducción









Los circuitos integrados personalizados o **Custom Chips**, permiten crear nuevos modelos de simulación y extender la funcionalidad de Wokwi.

Son escritos en C, pero pueden usar cualquier lenguaje que compile a WebAssembly. También pueden ser escritos en Verilog

Estructura de un custom chip

Archivos

JSON

El archivo JSON define las entradas y salidas del chip y los ajuste como nombre, autor y controles

```
{
  "name": ,
  "author": ,
  "pins": [ ],
  "controls": [ ]
}
```

Source code
C/Rust/Verilog

Código fuente con la funcionalidad del chip. Contiene las funciones y el estado del chip. Existe un conjunto API para acceder a diferentes características como GPIO, SPI, I2C, UART, temporización y Framebuffer.

<chip-name>.chip.json

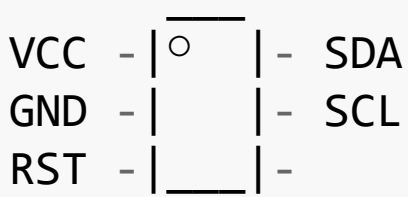
Propiedad	Tipo	Descripción
name	Cadena de caracteres	El nombre del chip. Es el nombre que será mostrado en el diagrama.
author	Cadena de caracteres	El nombre de quien creó el chip
pins	Arreglo de cadenas	El listado de pines del chip
controls	Arreglo de objetos	El listado de controles del chip (opcional)
display	Objeto	Configuración del display (opcional)

Pines

Listado de los pines del chip, empezando desde el pin número 1. Si se quiere saltar un pin, se puede usar una cadena vacía (“”) para el nombre del pin. Ejemplo:

```
"pins": ["VCC", "GND", "RST", "", "SCL", "SDA"],
```

Lo cual crearía un chip con cinco pines:



Controles

Los controles permiten interactuar con el chip mientras la simulación está corriendo.

Propiedad	Tipo	Descripción
id	Cadena de caracteres	El identificador del control escrito en formato camelCase.
label	Cadena de caracteres	El nombre del control que será mostrado al usuario
type	Cadena de caracteres	El tipo de control. Por ahora solo está disponible “range”
min	Número	El valor mínimo del control
max	Número	El valor máximo del control
step	Número	El cambio mínimo que se realiza al cambiar el control

Display

La propiedad display permite crear un chip con una pantalla o display para graficar o escribir texto, y de esta forma crear pantallas LCD, OLED personalizadas., o simplemente mostrar el estado del chip.

Propiedad	Tipo	Descripción
width	Número	El ancho del display en pixeles
height	Número	El alto del display en pixeles

Ejemplo:

```
"display": {
    "width": 128,
    "height": 64
 },
```

API (Application Programming Interfaces)

Actualmente, Wokwi cuenta con siete API para la creación de custom chips:

- **GPIO API**: Permite interactuar con la simulación usando pines digitales.
- **Analog API**: Permite interactuar con la simulación usando entradas y salidas analógicas.
- **Time API**: Un conjunto de funciones para la temporización. Incluye configuración de temporizadores para la generación de eventos cada tiempo específico.
- **UART, SPI e I2C API**: Permiten asociar al chip una interfaz de comunicación serial UART, SPI o I2C para la trasmisión de datos entre el chip y los elementos de la simulación usando el formato de cada interfaz.
- **Framebuffer API**: Permite la implementación de displays (LCD, OLED)

Atributos

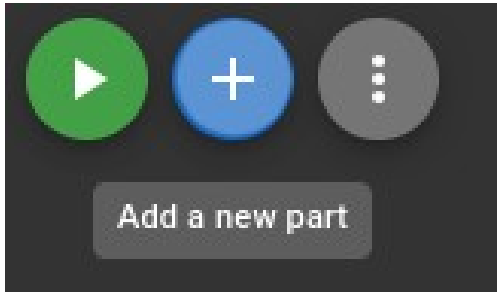
Los atributos son parámetros de entrada con los que se pueden cambiar características internas del chip en el archivo diagram.json. También permiten asociar los controles para obtener los datos durante la simulación.

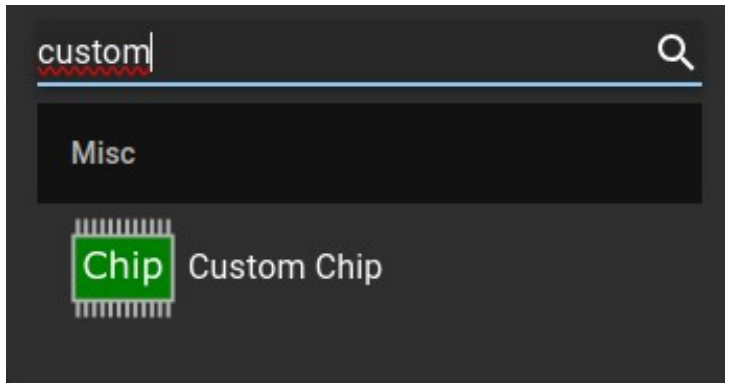
Se recomienda seguir las siguientes convenciones.

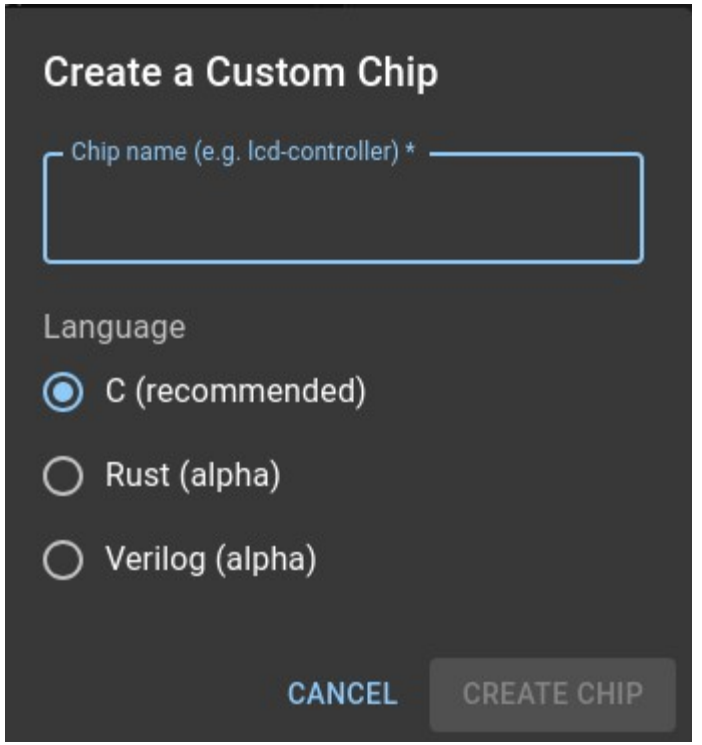
- Usar camelCase para los nombres de los atributos.
- Usar nombres en inglés.

Primeros pasos

Para crear un custom chip en Wokwi, se debe crear un proyecto o abrir uno existente y dar clic en el botón + para agregar un un nuevo componente. Seleccionar el lenguaje y asignar un nombre.



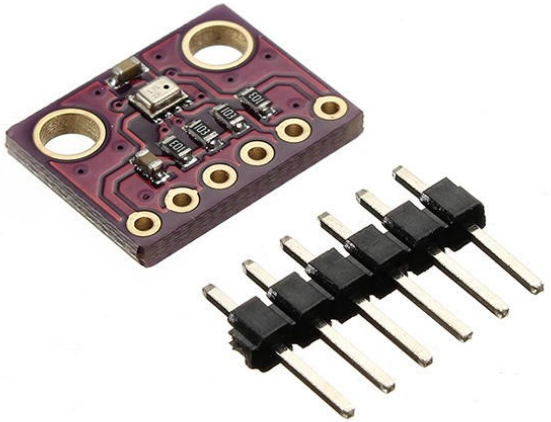


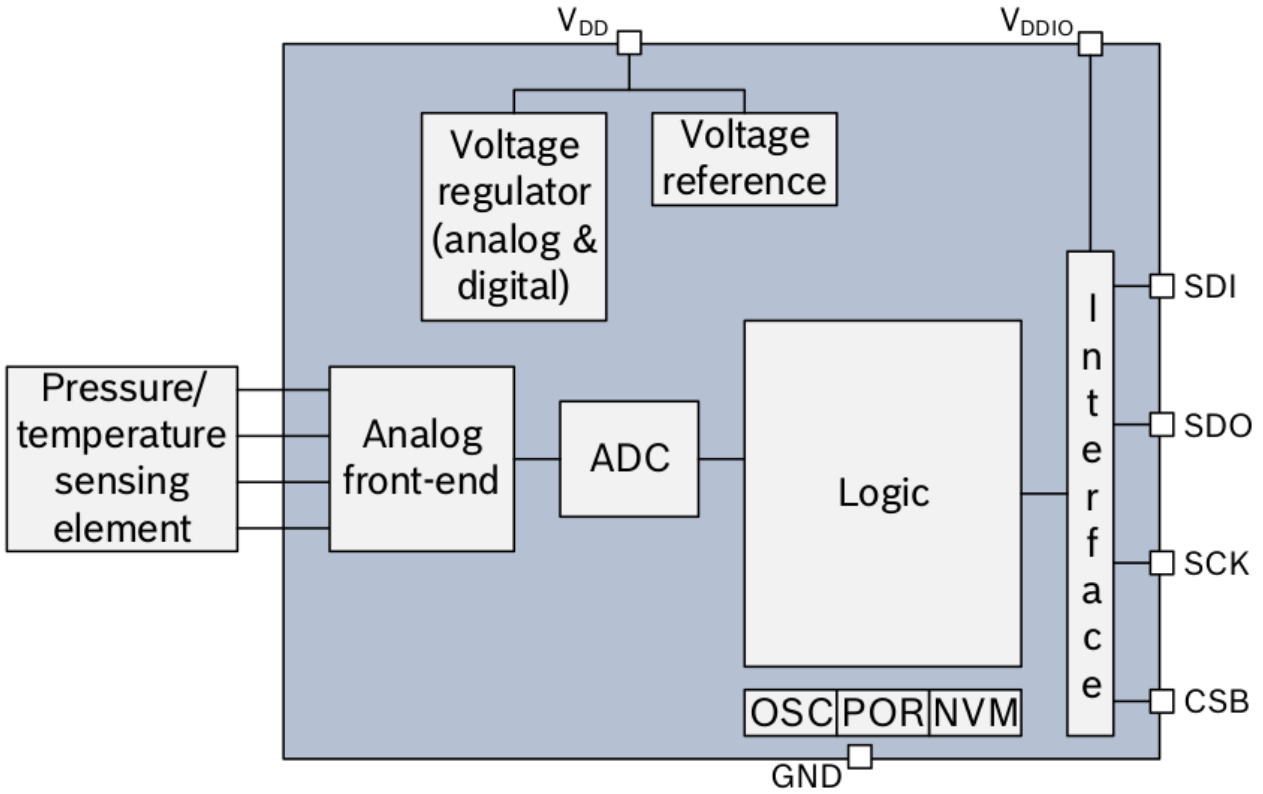


Wokwi crea una plantilla base para empezar a escribir el código usando el API de Wokwi para custom chip

```
#include "wokwi-api.h"
#include <stdio.h>
#include <stdlib.h>

typedef struct {
  // TODO: Put your chip variables here
} chip_state_t;

void chip_init() {
  chip_state_t *chip = malloc(sizeof(chip_state_t));

  // TODO: Initialize the chip, set up IO pins, create timers, etc.

  printf("Hello from custom chip!\n");
}
```

Sensor de presión y temperatura



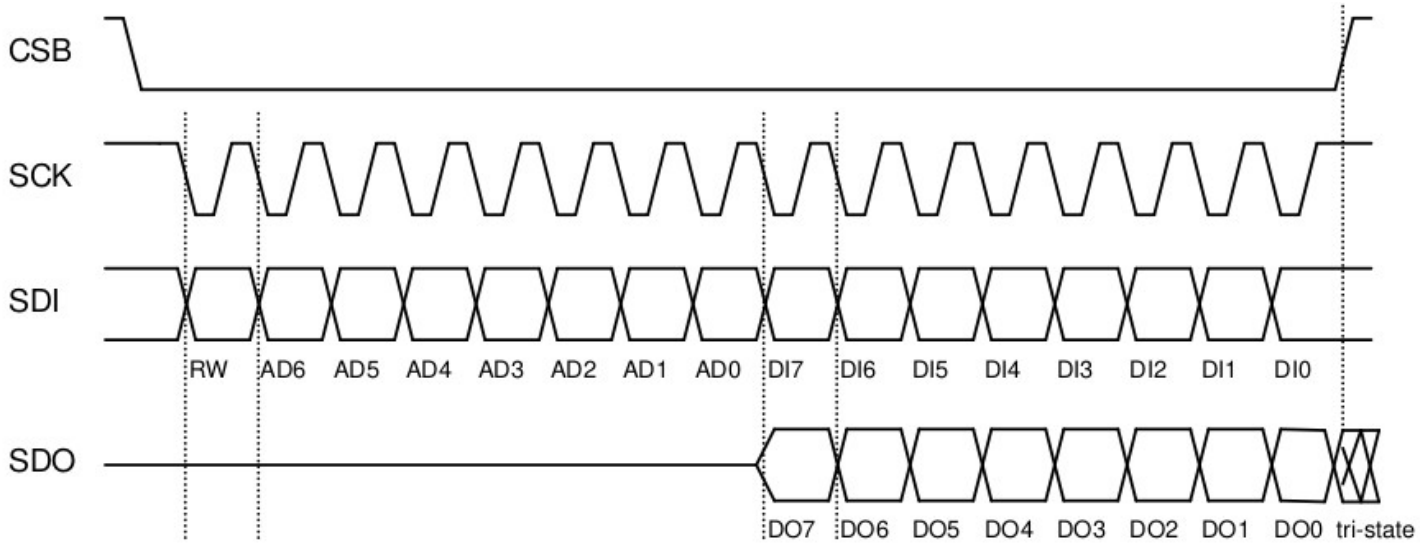


Sensor de presión y temperatura

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
temp_xlsb	0xFC	temp_xlsb<7:4>				0	0	0	0	0x00
temp_lsb	0xFB	temp_lsb<7:0>								0x00
temp_msb	0xFA	temp_msb<7:0>								0x80
press_xlsb	0xF9	press_xlsb<7:4>				0	0	0	0	0x00
press_lsb	0xF8	press_lsb<7:0>								0x00
press_msb	0xF7	press_msb<7:0>								0x80
config	0xF5	t_sb[2:0]				filter[2:0]			spi3w_en[0]	0x00
ctrl_meas	0xF4	osrs_t[2:0]				osrs_p[2:0]		mode[1:0]		0x00
status	0xF3					measuring[0]			im_update[0]	0x00
reset	0xE0	reset[7:0]								0x00
id	0xD0	chip_id[7:0]								0x58
calib25...calib00	0xA1…0x88	calibration data								individual

Registers:	Reserved registers	Calibration data	Control registers	Data registers	Status registers	Revision	Reset
Type:	do not write	read only	read / write	read only	read only	read only	write only

Sensor de presión y temperatura



	Control byte								Data byte								Data byte								
Start	RW	Register address (F6h)								Data register - address F6h								Data register - address F7h								Stop
CSB = 0	1	1	1	1	0	1	1	0		bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	CSB = 1

UNIVERSIDAD DEL QUINDÍO ®

ALTA CALIDAD

Res.MEN 014915 - 02 AGO 2022

CERTIFICADO POR SU COMPROMISO RESPONSABILIDAD SOCIAL FENALCO SOLIDARIO ®