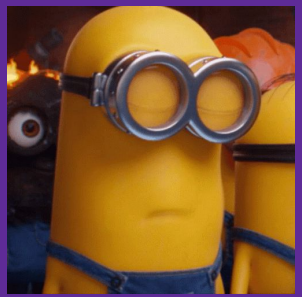
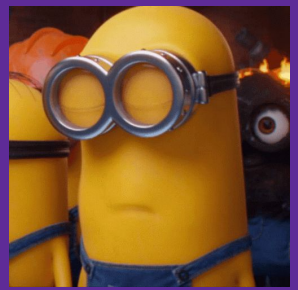
Comunicaciones I2C vs SPI.

Curso robótica II: Día 3





Comunicación SPI e I2C

- Protocolos más comunes junto con UART.
- Comunicación en serie.
- Comunicación entre elementos de una PCB/Equipo.
- Distancias cortas (máximo 1m)
- Se trabajan con lectura/escritura con registros.
- Muy común en sensores/actuadores.

Ejemplo mapa registros BMI160

Read/write	read only	write only	reserved

Register Address	Register Name	Default Value	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x7E	CMD	0x00	cmd							
0x7D	-	-	reserved							
0x7C	-	-	reserved							
0x7B	STEP_CONF_1	0x03	reserved				step_cnt_en	step_conf_10_8		
0x7A	STEP_CONF_0	0x15	step_conf_7_0							
0x79	STEP_CNT_1	0x00	step_cnt_15_8							
0x78	STEP_CNT_0	0x00	step_cnt_7_0							
0x77	OFFSET_6	0x00	gyr_off_en	acc_off_en	off_gyr_z_9_8		off_gyr_y_9_8		off_gyr_x_9_8	
0x76	OFFSET_5	0x00	off_gyr_z_7_0							
0x75	OFFSET_4	0x00	off_gyr_y_7_0							
0x74	OFFSET_3	0x00	off_gyr_x_7_0							
0x73	OFFSET_2	0x00	off_acc_z							
0x72	OFFSET_1	0x00	off_acc_y							
0x71	OFFSET_0	0x00	off_acc_x							
0x70	NV_CONF	0x00	reserved				u_spare_0	i2c_wdt_en	i2c_wdt_sel	spi_en
0x6F	-	-	reserved							
0x6E	-	-	reserved							

Más ejemplos

Table 7. ADT7411 Registers

RD/WR Address	Name	Power-on Default
00h	Interrupt Status 1	00h
01h	Interrupt Status 2	00h
02h	Reserved	
03h	Internal Temperature and V_{DD} LSBs	00h
04h	External Temperature and AIN1 to AIN4 LSBs	00h
05h	AIN5 to AIN8 LSBs	00h
06h	V_{DD} MSBs	xxh
07h	Internal Temperature MSBs	00h
08h	External Temperature MSBs/AIN1 MSBs	00h
09h	AIN2 MSBs	00h
0Ah	AIN3 MSBs	00h
0Bh	AIN4 MSBs	00h
0Ch	AIN5 MSBs	00h
0Dh	AIN6 MSBs	00h
0Eh	AIN7 MSBs	00h
0Fh	AIN8 MSBs	00h
10h-17h	Reserved	
18h	Control Configuration 1	00h
19h	Control Configuration 2	00h
1Ah	Control Configuration 3	00h

Más ejemplos

Table 7. ADT7411 Registers

RD/WR Address	Name
00h	Interrupt Status 1
01h	Interrupt Status 2
02h	Reserved
03h	Internal Temperature and V_{DD} LS
04h	External Temperature and AIN1 to A
05h	AIN5 to AIN8 LSBs
06h	V_{DD} MSBs
07h	Internal Temperature MSBs
08h	External Temperature MSBs/AIN1
09h	AIN2 MSBs
0Ah	AIN3 MSBs
0Bh	AIN4 MSBs
0Ch	AIN5 MSBs
0Dh	AIN6 MSBs
0Eh	AIN7 MSBs
0Fh	AIN8 MSBs
10h-17h	Reserved
18h	Control Configuration 1
19h	Control Configuration 2
1Ah	Control Configuration 3

8 REGISTER MAP

The following table lists the register map for the IAM-20380.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Accessible (writable) in Sleep Mode	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00	00	SELF_TEST_X_GYRO	R/W	N	XG_ST_DATA[7:0]							
01	01	SELF_TEST_Y_GYRO	R/W	N	YG_ST_DATA[7:0]							
02	02	SELF_TEST_Z_GYRO	R/W	N	ZG_ST_DATA[7:0]							
13	19	XG_OFFS_USRH	R/W	N	X_OFFS_USR [15:8]							
14	20	XG_OFFS_USRL	R/W	N	X_OFFS_USR [7:0]							
15	21	YG_OFFS_USRH	R/W	N	Y_OFFS_USR [15:8]							
16	22	YG_OFFS_USRL	R/W	N	Y_OFFS_USR [7:0]							
17	23	ZG_OFFS_USRH	R/W	N	Z_OFFS_USR [15:8]							
18	24	ZG_OFFS_USRL	R/W	N	Z_OFFS_USR [7:0]							
19	25	SMPLRT_DIV	R/W	N	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	N	-	FIFO_MODE	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	N	XG_ST	YG_ST	ZG_ST	FS_SEL [1:0]		-	FCHOICE_B[1:0]	
1E	30	LP_MODE_CFG	R/W	N	GYRO_CYCLE	G_AVGCFG[2:0]			-			
23	35	FIFO_EN	R/W	N	TEMP_FIFO_EN	XG_FIFO_EN	YG_FIFO_EN	ZG_FIFO_EN	-	-	-	-
36	54	FSYNC_INT	R/C	N	FSYNC_INT	-	-	-	-	-	-	-
37	55	INT_PIN_CFG	R/W	Y	INT_LEVEL	INT_OPEN	LATCH_INT_EN	INT_RD_CLEAR	FSYNC_INT_LEVEL	FSYNC_INT_MODE_EN	-	-
38	56	INT_ENABLE	R/W	Y	-			FIFO_OFLOW_EN	-	GDRIVE_INT_EN	-	DATA_RDY_INT_EN
3A	58	INT_STATUS	R/C	N	-			FIFO_OFLOW_INT	-	GDRIVE_INT	-	DATA_RDY_INT

Ejemplo Registro STATUS (BMI160)

2.11.6 Register (0x1B) STATUS

ADDRESS 0x1B
RESET 0b00000000
MODE R
DESCRIPTION Reports sensor status flags.
DEFINITION

Bit	Acronym	Definition
7	drdy_acc	Data ready (DRDY) for accelerometer in register
6	drdy_gyr	Data ready (DRDY) for gyroscope in register
5	drdy_mag	Data ready (DRDY) for magnetometer in register
4	nvm_rdy	NVM controller status
3	foc_rdy	FOC completed
2	mag_man_op	'0' indicates no manual magnetometer interface operation '1' indicates a manual magnetometer interface operation triggered via MAG_IF[2] or MAG_IF[3]
1	gyr_self_test_ok	'0' when gyroscope self-test is running or failed. '1' when gyroscope self-test completed successfully.

Drdy_*: gets reset when one byte of the register for sensor * is read.

Nvm_rdy: status of NVM controller: ´0´ → NVM write operation is in progress; ´1´ → NVM is ready to accept a new write trigger

foc_rdy: Fast offset compensation completed

Ejemplo registro ACC_CONF (BMI160)

2.11.11 Register (0x40) ACC_CONF

ADDRESS 0x40
RESET 0b00101000
MODE RW
DESCRIPTION Sets the output data rate, the bandwidth, and the read mode of the acceleration sensor.
DEFINITION

Name		Register (0x40) ACC_CONF		
Bit	7	6	5	4
Read/Write	R/W	R/W	R/W	R/W
Reset Value	0	0	1	0
Content	acc_us	acc_bwp		
Bit	**3**	**2**	**1**	**0**
Read/Write	R/W	R/W	R/W	R/W
Reset Value	1	0	0	0
Content	acc_odr			

acc_us: undersampling parameter. The undersampling parameter is typically used in low power mode

acc_bwp: bandwidth parameter determines filter configuration (acc_us=0) and averaging for undersampling mode (acc_us=1). For details see chapter 2.2.4.

acc_odr: define the output data rate in Hz is given by $100/2^{8\text{-val(acc\_odr)}}$. The output data rate is independent of the power mode setting for the sensor

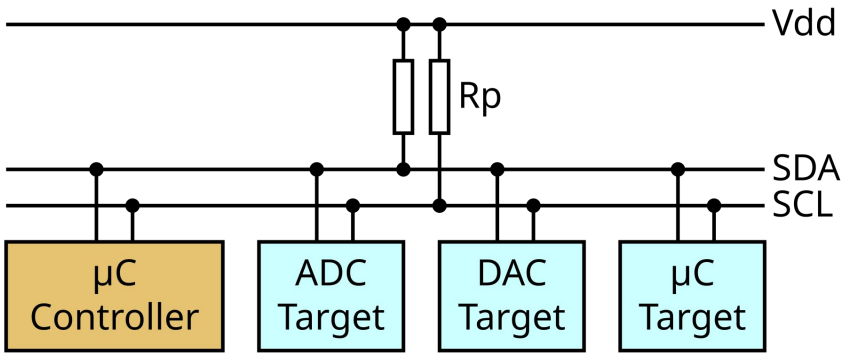
acc_odr	Output data rate in Hz
0b0000	Reserved
0b0001	25/32
0b0010	25/16
...	
0b1000	100
...	
0b1011	800
0b1100	1600
0b1101-0b1111	Reserved

When *acc_us* is set to '0' and the accelerometer is in low-power mode, it will change to normal mode. If the *acc_us* is set to '0' and an command to enter low-power mode is send to the Register (0x7E) CMD, this command is ignored.

Inter-Integrated Circuit (I2C)

Características I2C

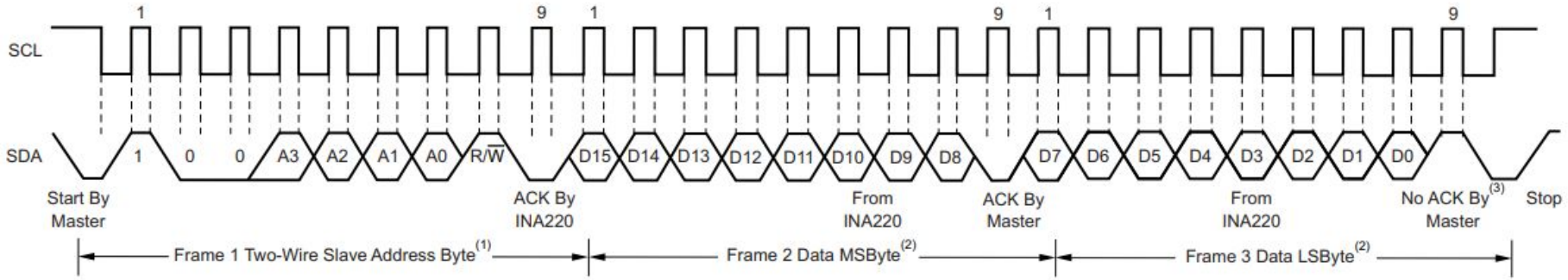
- Comunicación half-duplex
- Solo se necesitan **2 pines**:
  - SDA: Datos
  - SCL: Reloj
- Se necesitan resistencias de pull-up. Su valor limita la velocidad.
- Velocidades:
  - Standard: 100 kbits/s
  - Fast: 400 kbits/s
  - Existen más, pero el ESP32 no las soporta
- Los dispositivos tienen direcciones (generalmente configurables)
- Meten el Read/Write en el byte de address.



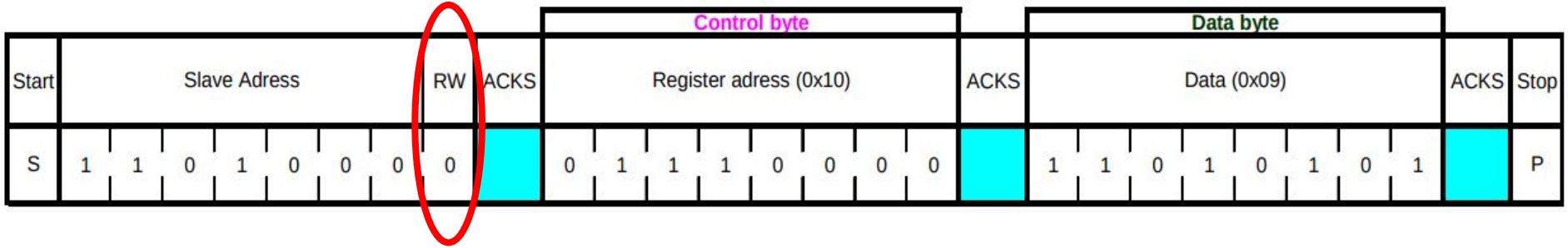
I²C modes

Mode[3]	Maximum speed	Maximum capacitance	Drive	Direction
Standard mode (Sm)	100 kbit/s	400 pF	Open drain*	Bidirectional
Fast mode (Fm)	400 kbit/s	400 pF	Open drain*	Bidirectional
Fast mode plus (Fm+)	1 Mbit/s	550 pF	Open drain*	Bidirectional
High-speed mode (Hs)	1.7 Mbit/s	400 pF	Open drain*	Bidirectional
High-speed mode (Hs)	3.4 Mbit/s	100 pF	Open drain*	Bidirectional
Ultra-fast mode (UFm)	5 Mbit/s	?	Push–pull	Unidirectional

Señales I2C


SCL
SDA
1
9
1
9
1
9
1
0
0
A3
A2
A1
A0
R/W
D15
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
Start By Master
ACK By INA220
From INA220
ACK By Master
From INA220
No ACK By Master(3)
Stop
Frame 1 Two-Wire Slave Address Byte(1)
Frame 2 Data MSByte(2)
Frame 3 Data LSByte(2)

Ejemplo I2C del bmi160



A tener en cuenta durante el diseño

- El valor de la resistencia de pull-up limita la velocidad:
  - A mayor resistencia, menos velocidad.
  - A menor resistencia, más velocidad.
- Elegir la resistencia entre 2k2Ω y 10kΩ.
- Para resolver problemas:
  - Ejecutar escáner de I2C en velocidades bajas.
  - Probar velocidades de I2C más bajas.
    - Si funciona, bajar el valor de las resistencias y volver a subir la velocidad
    - Sino, seguramente sea otra cosa.

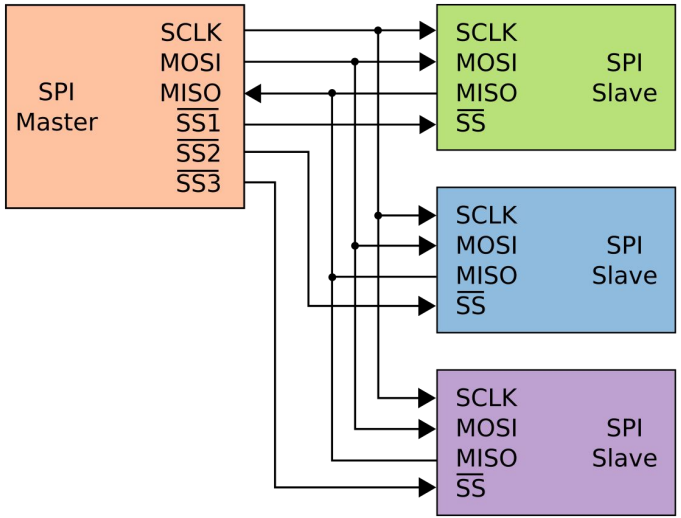
Ejemplos de chips I2C

- **VL53l0x**: Sensor de distancia ToF
- **TMP100**: Sensor de temperatura
- **AHT-20:** Sensor de humedad y temperatura
- **TCA9548A:** Multiplexor I2C
- **MCP23016**: Expansor I2c

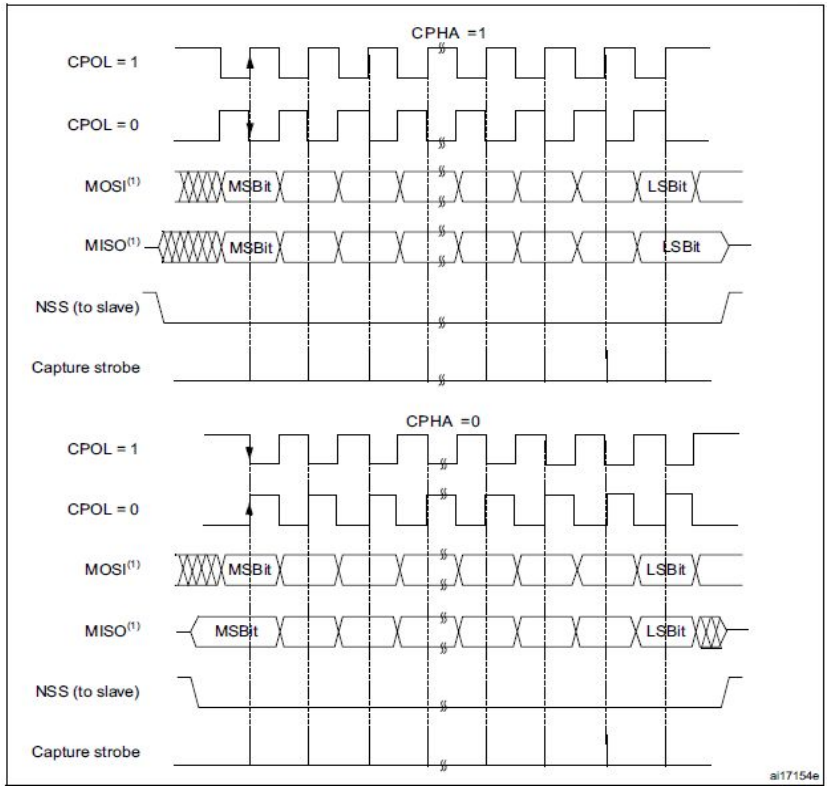
Serial Peripheral Interface (SPI)

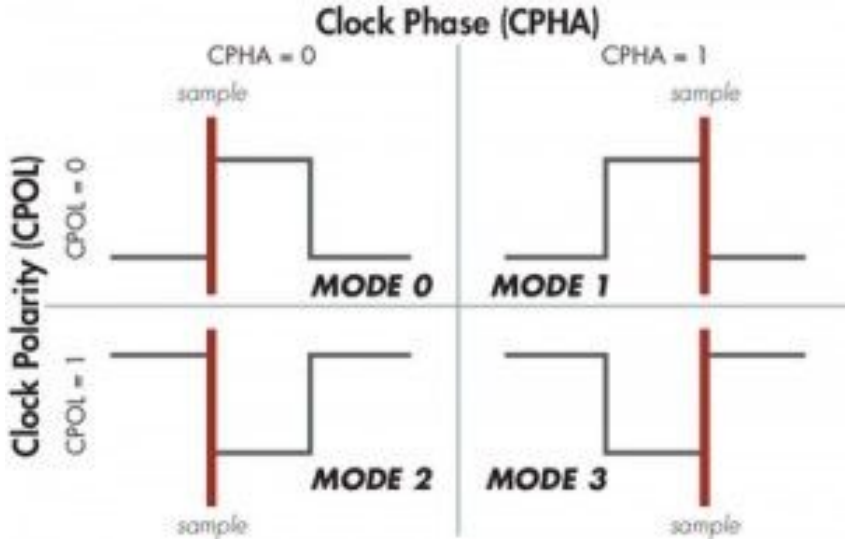
Características SPI

- Comunicación full-duplex
- Se necesitan 3 pines + 1x dispositivo:
  - MOSI: Master Out Slave In
  - MISO: Master In Slave Out
  - SCLK: Reloj
  - CS: Chip Select
- Velocidades hasta cientos de MHz
- Solo puede haber un CS encendido a la vez.
- Se puede configura el CPAH y CPOL dando 4 configuraciones:
  - CPAH = 0, CPOL = 0 (modo 0)
  - CPAH = 0, CPOL = 1
  - CPAH = 1, CPOL = 0
  - CPAH = 1, CPOL = 1 (modo 3)

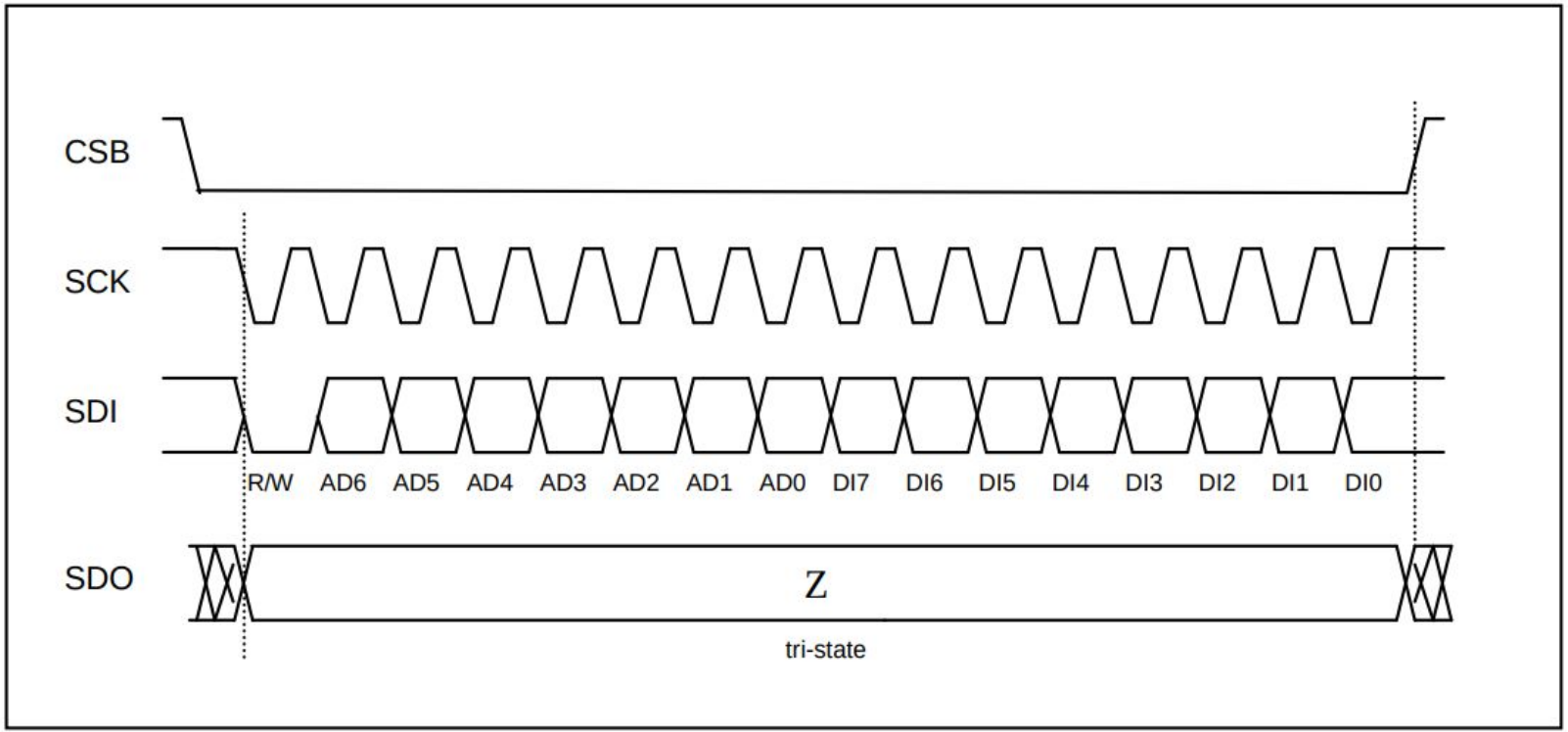


Modos SPI

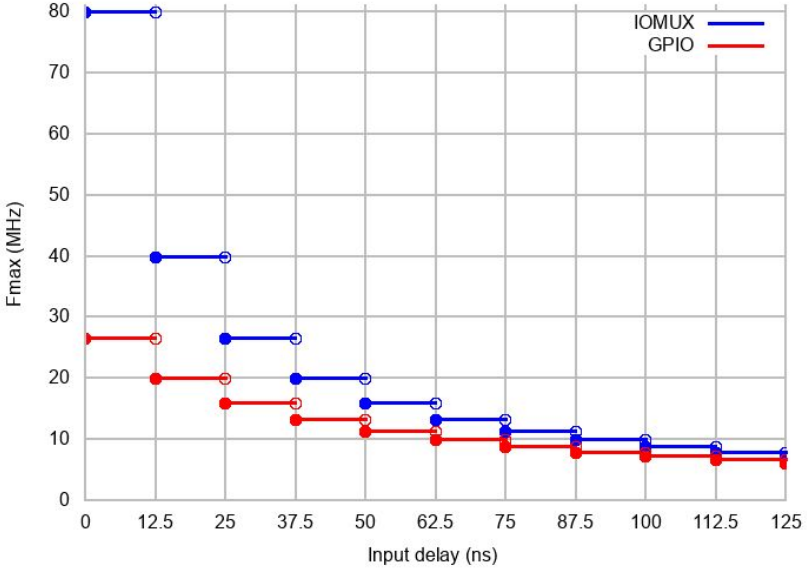

CPHA =1
CPOL = 1
CPOL = 0
MOSI(1)
MSBit
LSBit
MISO(1)
MSBit
LSBit
NSS (to slave)
Capture strobe
CPHA =0
CPOL = 1
CPOL = 0
MOSI(1)
MSBit
LSBit
MISO(1)
MSBit
LSBit
NSS (to slave)
Capture strobe
ai17154e


Clock Phase (CPHA)
CPHA = 0
CPHA = 1
sample
sample
Clock Polarity (CPOL)
CPOL = 0
CPOL = 1
MODE 0
MODE 1
MODE 2
MODE 3
sample
sample

Ejemplo SPI BMI160



Límites de velocidad para el SPI en ESP32



The IO_MUX pins for SPI buses are given below.

Pin Name	SPI 2 (GPIO Number)	SPI 3 (GPIO Number)
CS0 [1]	15	5
SCLK	14	18
MISO	12	19
MOSI	13	23
QUADWP	2	22
QUADHD	4	21

[1] : Only the first Device attached to the bus can use the CS0 pin.

Ejemplos de chips SPI

- **ICM-40627:** Imu
- **BMP384**: Sensor de presión
- Memorias Flash (Usando QSPI)
- **HIH6030-000-001**: Sensor de humedad
- **PCF2123**: RTC

I2C vs SPI (En ESP32)

	I2C	SPI
Velocidad	Hasta 400 kbps	Hasta 80 MHz
Número de pines	2	3 + 1x (nº Dispositivos)
Componentes externos	Resistencias pull-up	Nada
Interfaces disponibles	2 en el ESP32	3 en el ESP32

No usar el SPI1

Al ataqueerr

ESP-IDF: SPI Configuración

- spi_bus_config_t:
  - Configuramos el SPI común, pines generales.
  - Se usa en: spi_bus_initialize
- spi_device_interface_config_t:
  - Configuración de la comunicación.
  - Hay uno por cada dispositivo que tengas conectado.
  - Se usa en: spi_bus_add_device
- spi_device_handle_t:
  - Tiene que haber uno por cada dispositivo conectado.
  - Se usa para hacer la acción en un dispositivo concreto.

```c
#include <driver/spi_master.h>

typedef struct {
    int mosi_io_num;
    int miso_io_num;
    int sclk_io_num;
    /* ... */
} spi_bus_config_t;


typedef struct {
    uint8_t command_bits;
    uint8_t address_bits;
    uint8_t mode;
    int clock_speed_hz;
    int spics_io_num;
    /* ... */
} spi_device_interface_config_t;
```

ESP-IDF: SPI Transmisión.

- spi_transaction_t:
  - Configuramos la transferencia, datos y longitud.
  - Se usa en: spi_device_polling_transmit
- Flags:
  - SPI_TRANS_USE_RXDATA
  - SPI_TRANS_USE_TXDATA

```
struct spi_transaction_t {
    uint32_t flags;
    size_t length;
    size_t rxlength;
    union {
        const void *tx_buffer;
        uint8_t tx_data[4];
    };
    union {
        void *rx_buffer;
        uint8_t rx_data[4];
    };
} ;
```