.UBAfiuba
FACULTAD DE INGENIERÍA

Protocolos de Comunicación de Sistemas Embebidos

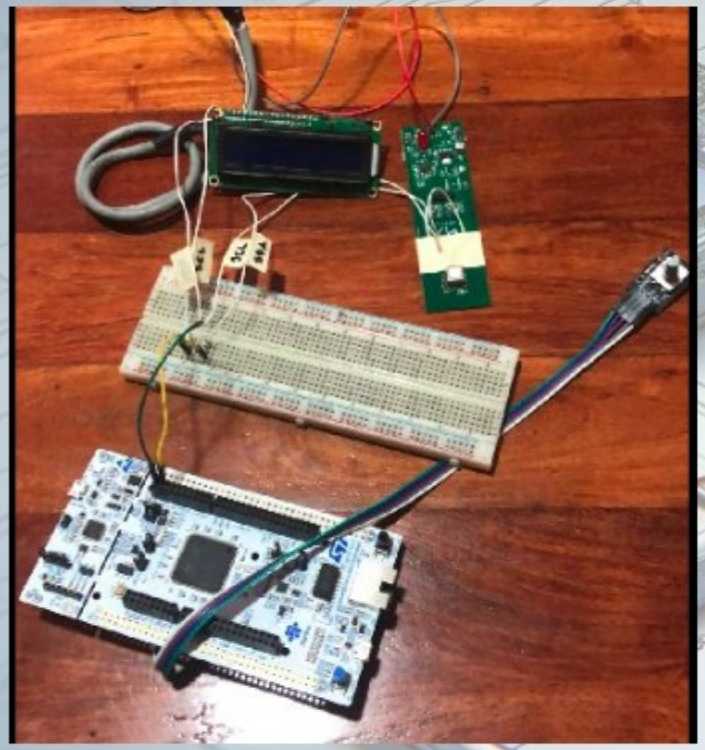
Docente - Israel Pavelek - PCSE - CESE

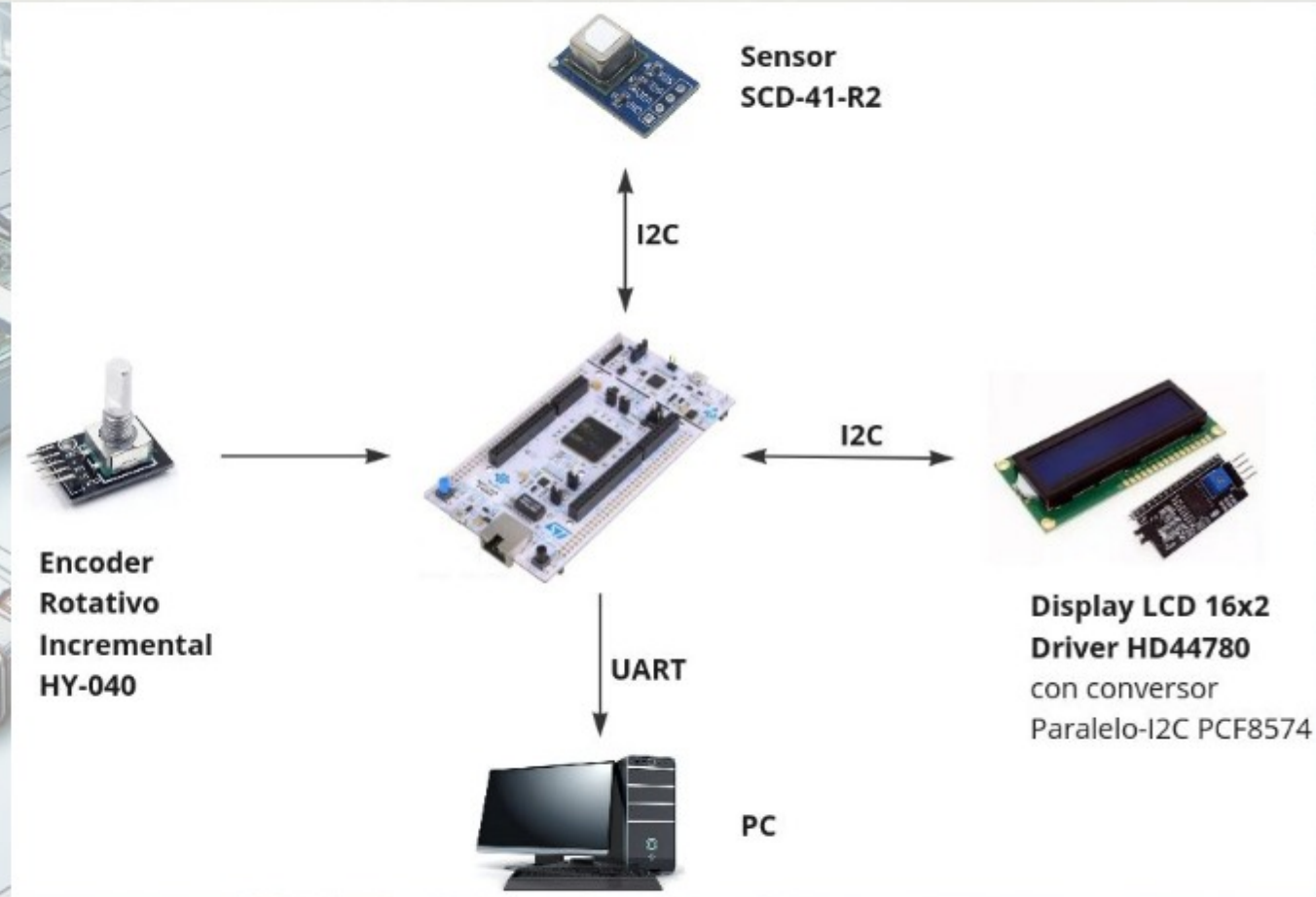
Alumno - Podoroska Iván

GITHUB - https://github.com/ivanpodo/PdM_workspace/tree/main/tp-final

.UBAfiuba
FACULTAD DE INGENIERÍA

TEMA



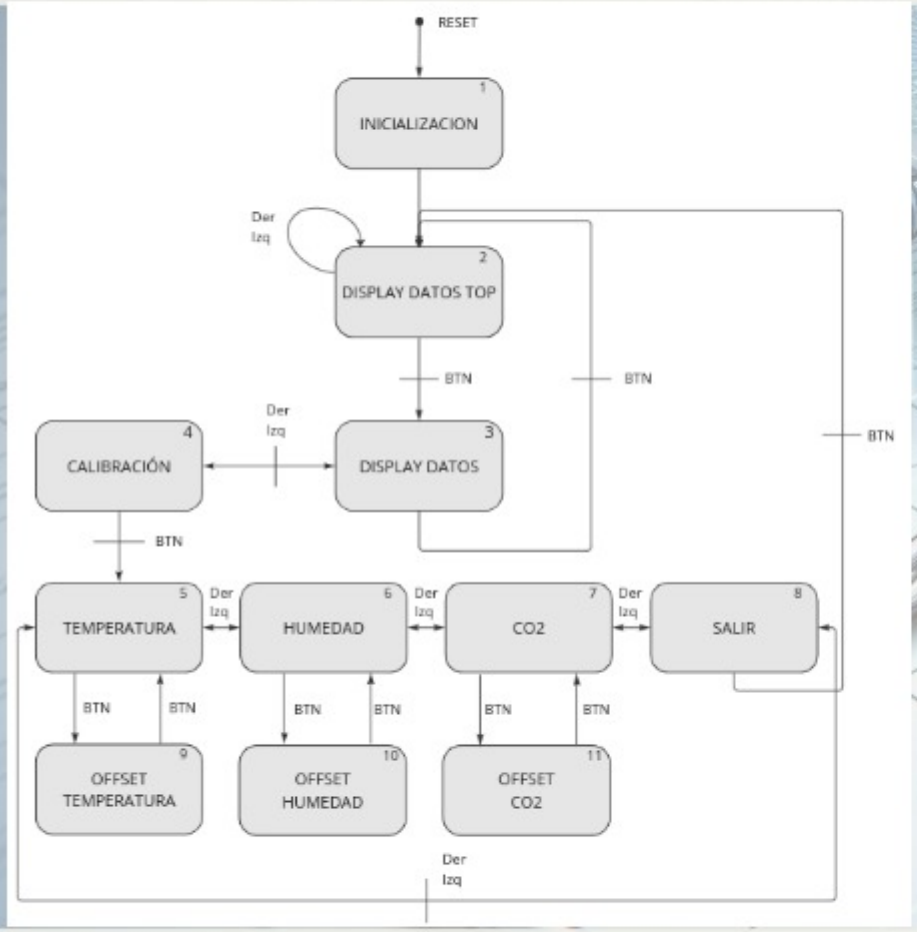


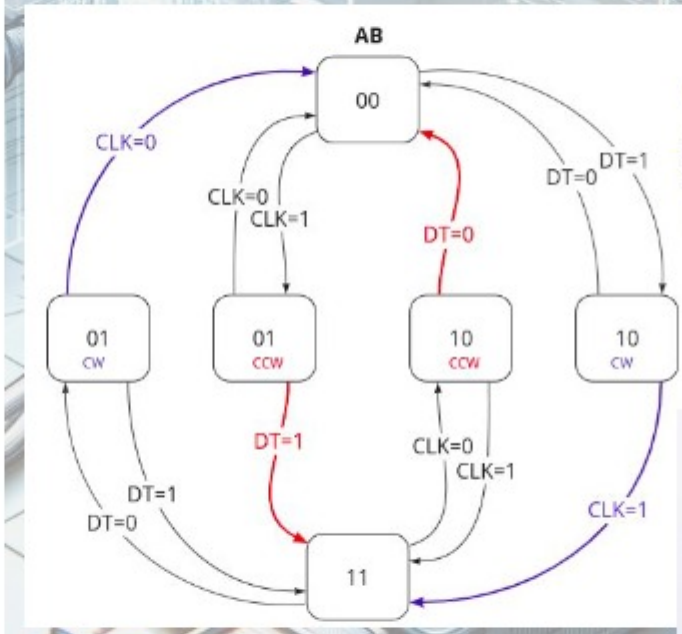
Docente - Israel Pavelek - PCSE - CESE

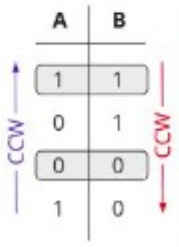
Alumno - Podoroska Iván

.UBAfiuba
FACULTAD DE INGENIERÍA

FUNCIONAMIENTO

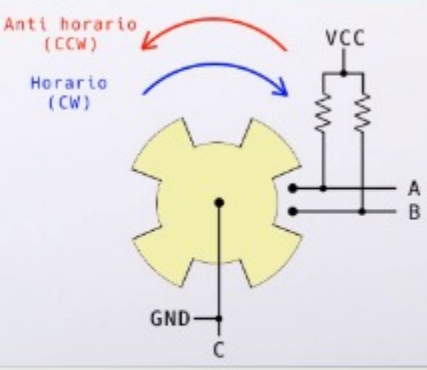







```
GPIO_InitStruct.Pin = (GPIO_PIN_10 | GPIO_PIN_11 | GPIO_PIN_12);
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;

HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```



Docente - Israel Pavelek - PCSE - CESE

Alumno - Podoroska Iván

.UBAfiuba
FACULTAD DE INGENIERÍA

CONFIGURACION

UART

- **Longitud de palabra** - 8 bits
- **Chequeo de paridad** - desactivado
- **Bit de stop** - 1
- **Baud Rate** - 115200
- **Modo** - dual Tx/Rx
- **Oversampling** x 16

```
huart_.Instance          = UART_INSTANCE;
huart_.Init.BaudRate     = UART_BAUDRATE;
huart_.Init.Parity       = UART_PARITY_NONE;
huart_.Init.StopBits     = UART_STOPBITS_1;
huart_.Init.WordLength   = UART_WORDLENGTH_8B;
huart_.Init.Mode         = UART_MODE_TX_RX;
huart_.Init.HwFlowCtl    = UART_HWCONTROL_NONE;
huart_.Init.OverSampling = UART_OVERSAMPLING_16;
```

I2C

- **Direccionamiento** - 7 bits
- **Doble dirección** - desactivado
- **Frecuencia** - 100 kHz
- **Streching** - desactivado
- **Duty cycle** - 2/1
- **Llamada general** - desactivada

```
hi2c1.Instance             = I2C1;
hi2c1.Init.ClockSpeed      = I2C_CLOCK_RATE;
hi2c1.Init.DutyCycle       = I2C_DUTYCYCLE_2;
hi2c1.Init.OwnAddress1     = 0;
hi2c1.Init.AddressingMode  = I2C_ADDRESSINGMODE_7BIT;
hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
hi2c1.Init.OwnAddress2     = 0;
hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
hi2c1.Init.NoStretchMode   = I2C_NOSTRETCH_DISABLE;
```

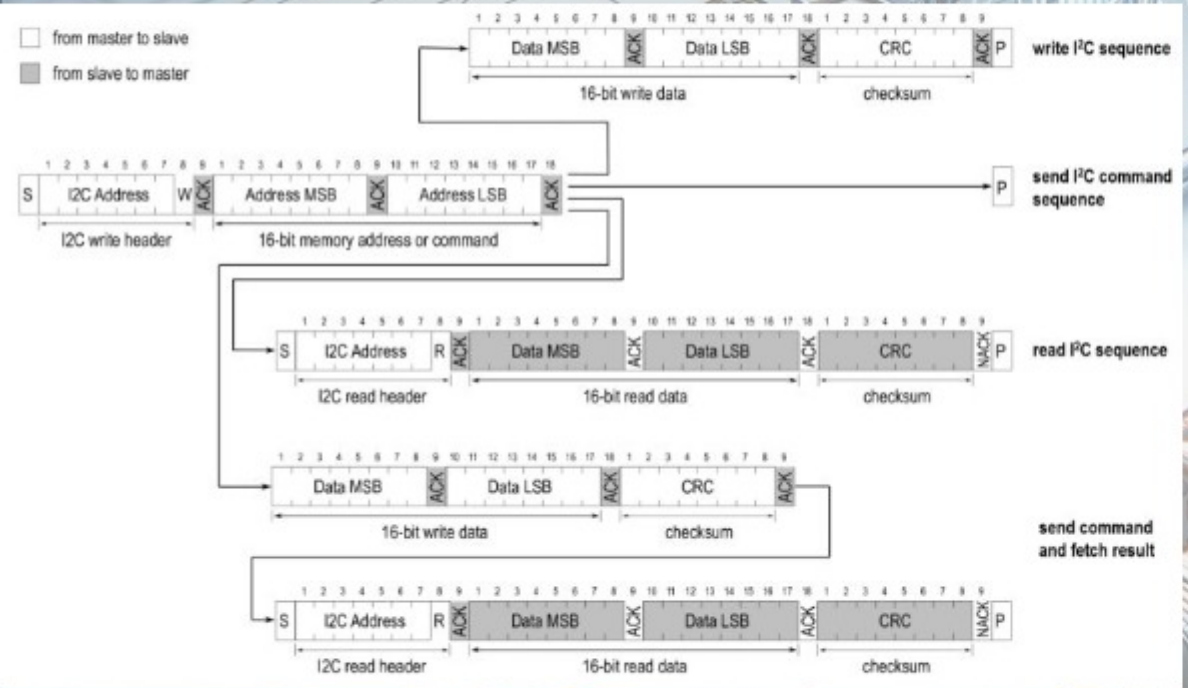
Docente - Israel Pavelek - PCSE - CESE

Alumno - Podoroska Iván

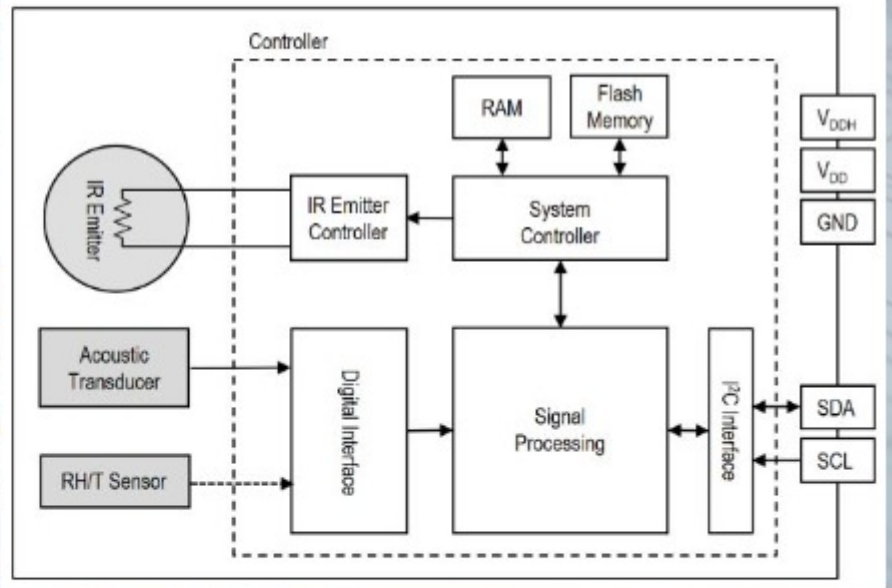
.UBAfiuba
FACULTAD DE INGENIERÍA

SCD41

SCD4x	Hex. Code
I²C address	0x62



Functional Block Diagram



Docente - Israel Pavelek - PCSE - CESE

Alumno - Podoroska Iván

.UBAfiuba
FACULTAD DE INGENIERÍA

SCD41

3.4 SCD4x Command Overview

An overview of the available SCD4x commands can be found in **Table 9.** A detailed description for each command can be found in the following sections.

Domain	Command	Hex. Code	I²C sequence type (see Section 3.3)	Execution	
				time [ms]	During meas.
Basic Commands Section 3.5	start_periodic_measurement	0x21b1	send command	-	no
	read_measurement	0xec05	read	1	yes
	stop_periodic_measurement	0x3f86	send command	500	yes
On-chip output signal compensation Section 3.6	set_temperature_offset	0x241d	write	1	no
	get_temperature_offset	0x2318	read	1	no
	set_sensor_altitude	0x2427	write	1	no
	get_sensor_altitude	0x2322	read	1	no
	set_ambient_pressure	0xe000	write	1	yes
	get_ambient_pressure	0xe000	read	1	yes
Field calibration Section 3.7	perform_forced_recalibration	0x362f	send command and fetch result	400	no
	set_automatic_self_calibration_enabled	0x2416	write	1	no
	get_automatic_self_calibration_enabled	0x2313	read	1	no
Low power periodic measurement mode Section 3.8	start_low_power_periodic_measurement	0x21ac	send command	-	no
	get_data_ready_status	0xe4b8	read	1	yes

3.11 Checksum Calculation

The 8-bit CRC checksum transmitted after each data word is generated by a CRC algorithm. Its properties are displayed in **Table 37**. The CRC covers the contents of the two previously transmitted data bytes. To calculate the checksum only these two previously transmitted data bytes are used. Note that command words are not followed by CRC.

Property	Value
Name	CRC-8
Width	8 bit
Protected Data	read and/or write data
Polynomial	0x31 ($x^8 + x^5 + x^4 + 1$)
Initialization	0xff
Reflect input	False
Reflect output	False
Final XOR	0x00
Examples	CRC (0xbeef) = 0x92

Example code (C/C++)

```
#define CRC8_POLYNOMIAL 0x31
#define CRC8_INIT 0xff

uint8_t sensirion_common_generate_crc(const uint8_t* data, uint16_t count) {
    uint16_t current_byte;
    uint8_t crc = CRC8_INIT;
    uint8_t crc_bit;
    /* calculates 8-Bit checksum with given polynomial */
    for (current_byte = 0; current_byte < count; ++current_byte) {
        crc ^= (data[current_byte]);
        for (crc_bit = 8; crc_bit > 0; --crc_bit) {
            if (crc & 0x80)
                crc = (crc << 1) ^ CRC8_POLYNOMIAL;
            else
                crc = (crc << 1);
        }
    }
    return crc;
}
```

Table 37: I²C CRC properties

Docente - Israel Pavelek - PCSE - CESE

Alumno - Podoroska Iván

.UBAfiuba
FACULTAD DE INGENIERÍA

DEMOSTRACIÓN

VIDEO DEMOSTRATIVO: https://drive.google.com/file/d/1c6SptZLK5JwAlFcHeT-6YWO50yyMYHFK/view?usp=drive_link

DEMO UART: https://drive.google.com/file/d/1jQVafHF2vWL_g4ZWpDJu-ALKsj6rms3a/view?usp=drive_link

Docente - Israel Pavelek - PCSE - CESE

Alumno - Podoroska Iván