

Universidad de Costa Rica

Laboratorio de Microcontroladores

Laboratorio #4:
STM32: GPIO, ADC, comunicaciones e IoT

Prof. MSc. Marco Villalta Fallas

Marco Vásquez Ovares - B17032

Junior Ruiz Sánchez - B97026

Grupo: 01

I Ciclo 2025

Índice

1. Introducción	6
2. Nota teórica	7
2.1. STM32F429 Discovery kit	7
2.1.1. Diagrama de pines	8
2.1.2. Descripción de pines	10
2.1.3. Diagrama de bloques	10
2.1.4. Características eléctricas	10
2.2. Microcontrolador STM32F429ZIT6	11
2.2.1. Diagrama de pines	13
2.2.2. Descripción de pines	14
2.2.3. Diagrama de bloques	14
2.2.4. Características eléctricas	14
2.3. Sensor MEMS L3GD20	15
2.3.1. Diagrama de pines	16
2.3.2. Descripción de pines	17
2.3.3. Diagrama de bloques	18
2.3.4. Características eléctricas	18
2.3.5. Registros	20
2.4. Pantalla LCD/TFT ILI9341	21
2.4.1. Diagrama de pines	22
2.4.2. Descripción de pines	22
2.4.3. Diagrama de bloques	25
2.4.4. Características eléctricas	25
2.5. Biblioteca LibOpenCM3	27
2.6. Comunicación UART/USART	28
2.7. Protocolo de Comunicación SPI (Serial Peripheral Interface)	29
2.7.1. SPI read	29
2.7.2. SPI write	30
2.8. MQTT (Message Queuing Telemetry Transport)	31
2.9. Internet de las Cosas (IoT)	32
2.10. Diseño del circuito	33
2.10.1. Lista de componentes	34
2.10.2. Información adicional	35
3. Desarrollo	37
3.1. Análisis del firmware	37
3.1.1. Diagrama de flujo del programa	37
3.1.2. Firmware	38
3.1.3. Inclusión de Bibliotecas	38
3.1.4. Macros de configuración del giroscopio	38
3.1.5. Flags globales	39
3.1.6. Prototipos de funciones	39
3.1.7. Función <code>spi_setup()</code>	39
3.1.8. Función <code>button_setup()</code>	39

3.1.9. Función <code>led_setup()</code>	39
3.1.10. Función <code>adc_setup()</code> y <code>read_adc_naiive()</code>	39
3.1.11. Función <code>seismograph_setup()</code>	40
3.1.12. Función <code>main()</code>	40
3.1.13. Función <code>seismograph()</code>	40
3.2. Análisis electrónico	42
3.2.1. Lectura del nivel de la batería	42
3.3. Análisis de resultados	45
3.3.1. Circuito con divisor de tensión	45
3.3.2. Circuito con la batería ON	45
3.3.3. Circuito con la batería OFF	46
3.3.4. Circuito con la comunicación serial ON	47
3.3.5. Circuito con la comunicación serial OFF	48
3.3.6. IoT con batería ON	49
3.3.7. IoT con batería OFF	50
3.3.8. Datos en la terminal	51
4. Conclusiones y Recomendaciones	52
Bibliografía	53
5. Apéndices	53
5.1. Repositorio Git	53
5.2. Archivo <code>.sh</code>	54
5.3. Archivo <code>get_Data.py</code>	55
5.4. Archivo <code>IoT.py</code>	56
5.5. Archivo <code>thingsboard.py</code>	58
5.5.1. Archivo <code>.csv</code>	60

Índice de figuras

1.	STM32F429 Discovery board	7
2.	Código de colores para entender figuras 3 y 4 [3]	8
3.	Columna de pines P1, correspondiente a la sección de 5V [3]	9
4.	Columna de pines P2, correspondiente a la sección de 3V [3]	9
5.	Pines reservados para botón de usuario y leds [3]	10
6.	Hardware block diagram	10
7.	STM32F429ZIT6 package	11
8.	Diagrama de la página 47 del manual STM32F4	13
9.	STM32F429ZIT6 block diagram	14
10.	Pin connection	16
11.	Pin description	17
12.	Block diagram	18
13.	Electrical characteristics	18
14.	Absolute maximum ratings	19
15.	Register address map	20
16.	Descripción de pines	22
17.	Descripción de pines	23
18.	Descripción de pines	24
19.	Descripción de pines	24
20.	Block Diagram	25
21.	Absolute Maximum Ratings	25
22.	General DC Characteristics	26
23.	SPI - Read and write protocol	29
24.	SPI read protocol	29
25.	SPI write protocol	30
26.	Diseño final, parte 1	33
27.	Diseño final, parte 2	34
28.	Diagrama de flujo de seismograph	37
29.	Esquemático de un divisor de tensión ideal	42
30.	Esquemático de divisor de tensión simulado para obtener 5V a la salida	43
31.	Medición de la tensión de salida con la ayuda de un multímetro	44
32.	Circuito con divisor de tensión	45
33.	Circuito con la batería ON	46
34.	Circuito con la batería OFF	47
35.	Circuito con la comunicación serial ON	48
36.	Circuito con la comunicación serial OFF	49
37.	Aceleraciones X, Y, Z y nivel de batería (ON)	50
38.	Sismógrafo y nivel de batería (ON)	50
39.	Aceleraciones X, Y, Z y nivel de batería (OFF)	50
40.	Sismógrafo y nivel de batería (OFF)	51
41.	Datos en la terminal	51
42.	Archivo .csv con la información	60

Índice de tablas

- | | |
|---------------------------------------|----|
| 1. Lista de componentes | 34 |
|---------------------------------------|----|

1. Introducción

Resumen

Este proyecto describe el desarrollo de un sismógrafo implementado con la tarjeta de desarrollo STM32F429. El periférico principal es el giroscopio de 3 ejes L3GD20; a continuación, los datos se muestran en la pantalla TFT ILI9341. Para adaptar la tensión de la batería, se emplea un divisor resistivo que reduce el voltaje antes de que el valor sea leído por el ADC del microcontrolador. El cálculo del divisor se comprobó por simulación y con mediciones de multímetro, garantizando una lectura segura y estable.

El firmware fue escrito en C mediante la biblioteca libre libopencm3, con la cual se configura cada periférico y protocolo de comunicación de la tarjeta. El programa inicializa el reloj, habilita SPI para leer el giroscopio, y usa UART para enviar en tiempo real las lecturas a un equipo anfitrión. Un botón de usuario permite activar o desactivar esta transmisión serie sin reiniciar el sistema.

Para enlazar el dispositivo con la nube se desarrolló un script en Python. El código escucha el puerto UART, empaqueta la telemetría y la publica mediante MQTT en la plataforma ThingsBoard. De esta forma, las aceleraciones y el nivel de batería se visualizan como gráficas históricas y paneles en tiempo real.

Los ensayos de laboratorio confirmaron la correcta reducción de tensión, la estabilidad de la lectura del giroscopio y la integridad de los datos transmitidos a través de la red. En conjunto, la combinación de UART, SPI y MQTT, junto con libopencm3, demuestra una ruta accesible y eficiente para construir aplicaciones IoT que miden, procesan y comparten información en tiempo real.

2. Nota teórica

2.1. STM32F429 Discovery kit

La STM32F429 Discovery Kit es una placa de desarrollo diseñada por STMicroelectronics para facilitar la evaluación y prototipado de aplicaciones con el microcontrolador STM32F429ZI. Este microcontrolador pertenece a la familia STM32F4 y está basado en un núcleo ARM Cortex-M4 de alto rendimiento con unidad de punto flotante (FPU), ideal para tareas que requieren procesamiento intensivo y control en tiempo real.



Figura 1: STM32F429 Discovery board

Características generales de la placa:

- Microcontrolador principal
 - Modelo: STM32F429ZI
 - Arquitectura: ARM Cortex-M4 de 32 bits
 - Frecuencia de operación: Hasta 180 MHz
 - Memoria interna:
 - 2 MB de Flash
 - 256 KB de SRAM
 - Unidad de punto flotante (FPU): Incluida, útil para cálculos matemáticos complejos
- Pantalla y capacidades gráficas

- Pantalla LCD TFT de 2.4 pulgadas a color (QVGA, 320x240)
- Pantalla táctil resistiva
- Controlador gráfico Chrom-ART Accelerator™ (DMA2D), para renderizado rápido de gráficos
- Conectividad y periféricos
 - Interfaces de comunicación:
 - USB OTG (On-The-Go)
 - USART, UART
 - SPI
 - I2C
 - CAN
 - Sensor MEMS integrado:
 - Acelerómetro LIS3DSH, útil para detectar movimiento o vibraciones
 - LEDs y botones:
 - LEDs de usuario y de estado
 - Botón de usuario y botón de reinicio (RESET)
- Expansión y depuración
 - ST-LINK/V2-1 integrado:
 - Programación y depuración sin necesidad de hardware adicional
 - Conectores de expansión:
 - Pines accesibles tipo Arduino para conectar módulos adicionales o shields
 - GPIOs: Amplia cantidad de pines digitales y analógicos disponibles para uso general
- Alimentación
 - Vía USB mini-B o por fuente externa (típicamente 5V)
 - Reguladores integrados para suministrar 3.3V al microcontrolador y periféricos

2.1.1. Diagrama de pines

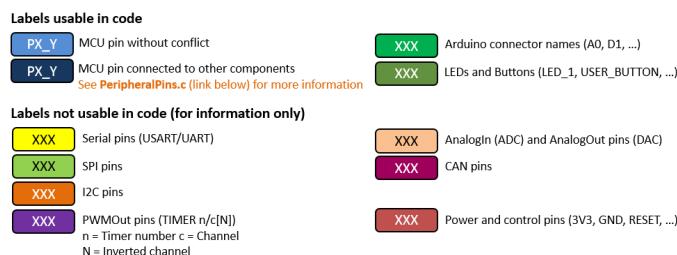


Figura 2: Código de colores para entender figuras 3 y 4 [3]

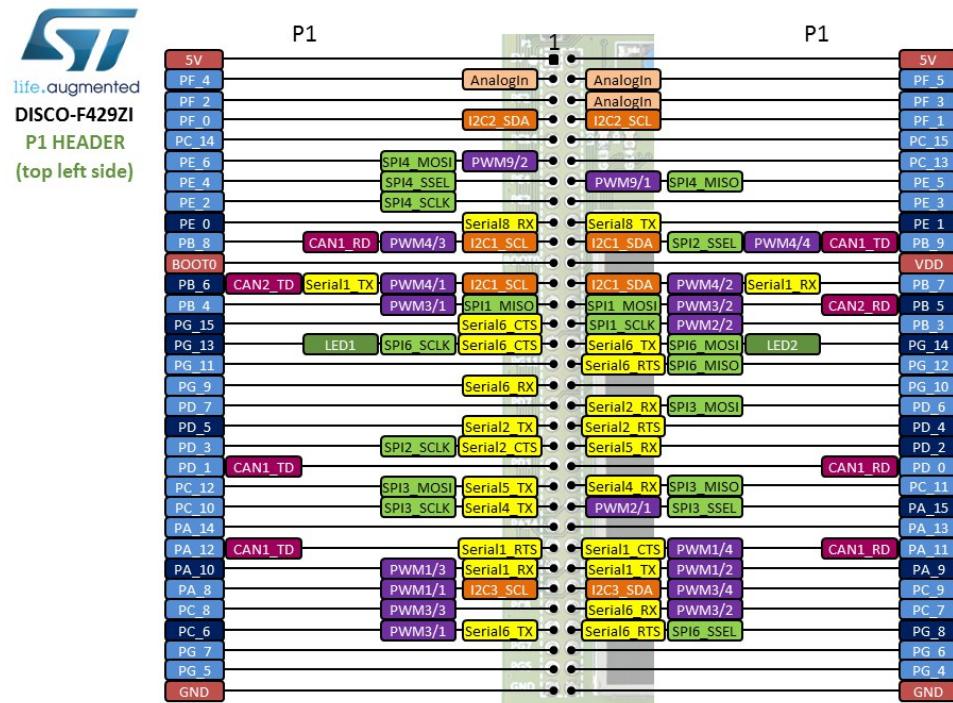


Figura 3: Columna de pines P1, correspondiente a la sección de 5V [3]

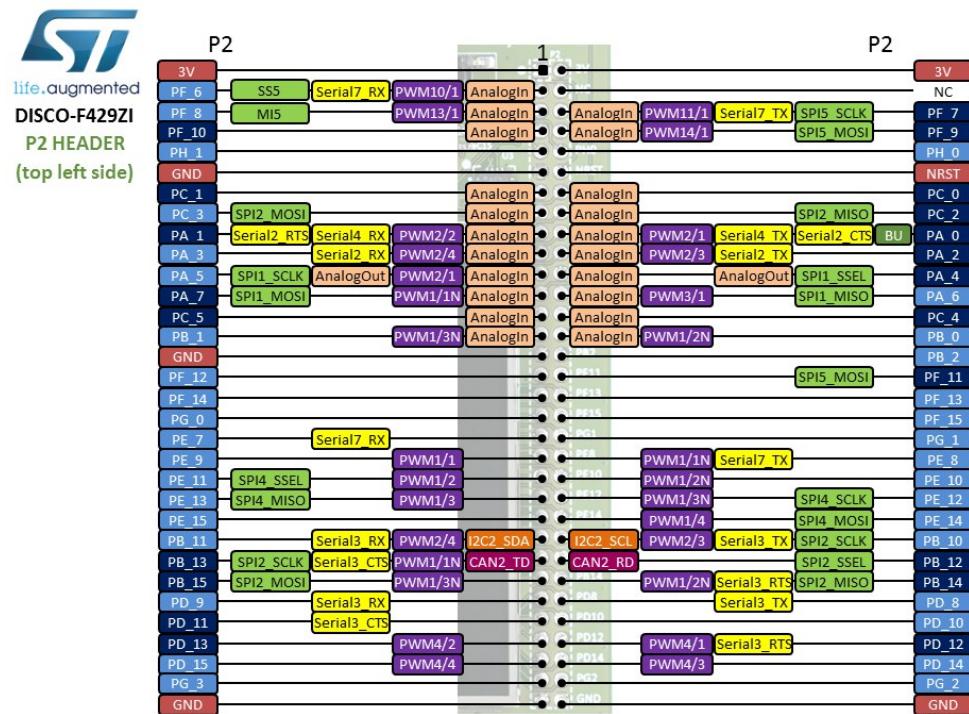


Figura 4: Columna de pines P2, correspondiente a la sección de 3V [3]

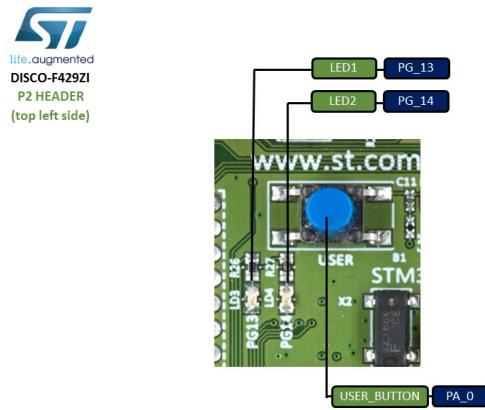


Figura 5: Pines reservados para botón de usuario y leds [3]

2.1.2. Descripción de pines

2.1.3. Diagrama de bloques

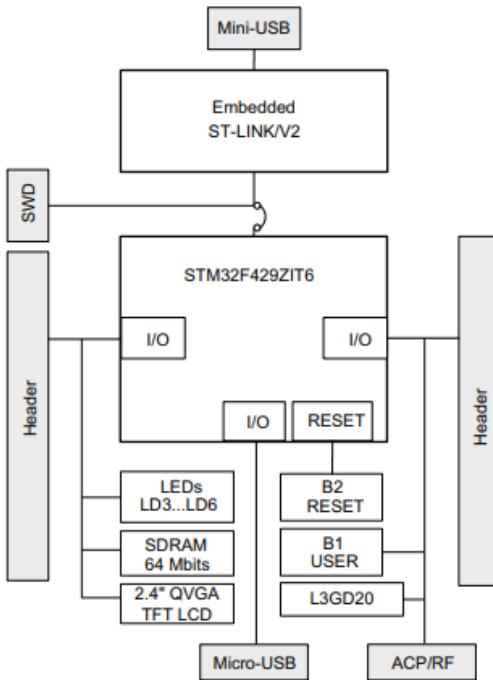


Figura 6: Hardware block diagram

2.1.4. Características eléctricas

2.2. Microcontrolador STM32F429ZIT6

El STM32F429ZIT6 es un microcontrolador de alto rendimiento de la serie STM32F4 de STMicroelectronics, basado en la arquitectura ARM Cortex-M4. Está diseñado para aplicaciones embebidas que requieren gran capacidad de procesamiento, gráficos, comunicaciones avanzadas y control en tiempo real. Es especialmente popular en sistemas que combinan procesamiento digital de señales con interfaces gráficas y periféricos complejos.



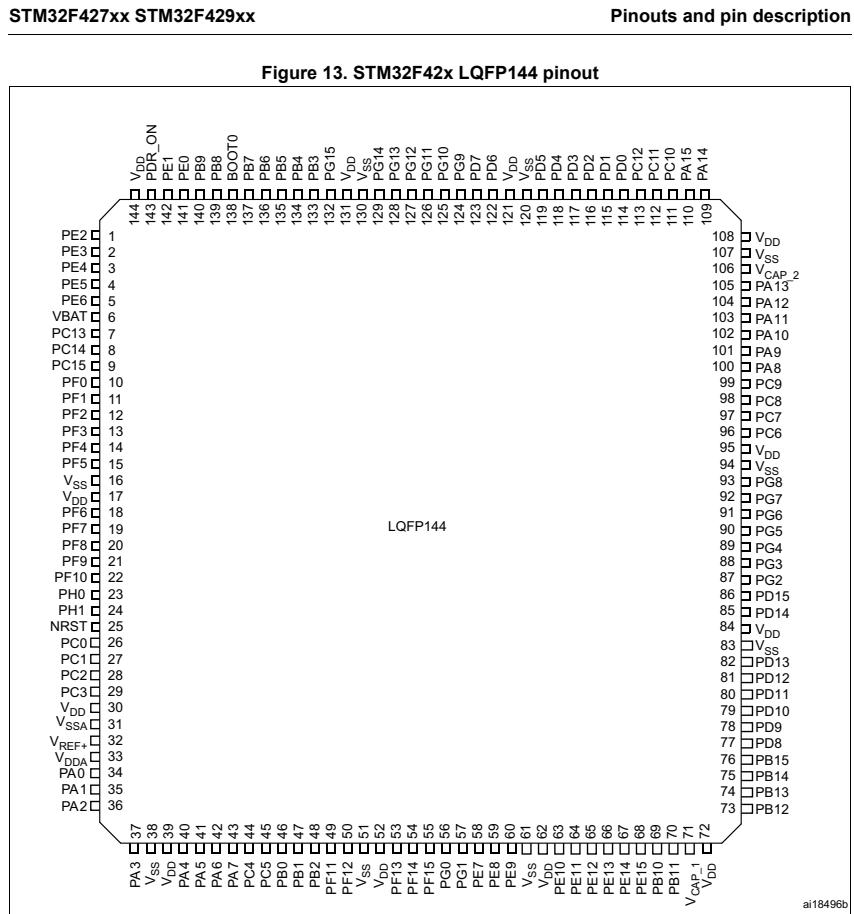
Figura 7: STM32F429ZIT6 package

Características generales del STM32F429ZIT6:

- Arquitectura y rendimiento
 - Núcleo: ARM Cortex-M4 de 32 bits con FPU (Unidad de Punto Flotante)
 - Frecuencia máxima: 180 MHz
 - Ciclo de instrucción: 225 DMIPS (Dhrystone MIPS)
 - Unidad DSP (Digital Signal Processing): Instrucciones SIMD y saturación aritmética
- Memoria
 - Memoria Flash interna: 2 MB
 - SRAM interna: 256 KB (dividida en diferentes bloques para acceso rápido y compartido)
 - Memoria externa: Soporte para interfaces como FSMC para memorias SRAM, NOR, NAND, SDRAM
- Periféricos
 - ADC: Tres ADCs de 12 bits (hasta 24 canales y 2.4 MSPS)
 - DAC: Dos canales DAC de 12 bits
 - Timers: Hasta 17 temporizadores (PWM, Input Capture, Output Compare, Encoder interface)
 - RTC (Reloj en tiempo real): Con alimentación independiente y soporte para backup
- Interfaces de comunicación
 - USART / UART: Hasta 6

- SPI / I2S: Hasta 3 SPI y 3 I2S
 - I2C: Tres interfaces
 - CAN: Dos controladores CAN 2.0B
 - USB: USB OTG FS y OTG HS (con PHY externo)
 - SDIO: Para tarjetas SD/MMC
 - Ethernet: MAC 10/100 con soporte IEEE 1588 (requiere PHY externo)
- Capacidades gráficas
 - Controlador gráfico Chrom-ART Accelerator™ (DMA2D)
 - Controlador LTDC (LCD-TFT Display Controller): Para pantallas externas RGB
 - GPIO y funciones adicionales
 - Número de pines GPIO: Hasta 140 pines de propósito general (dependiendo del encapsulado)
 - Interrupciones externas: Hasta 16 líneas EXTI
 - Encapsulado: LQFP-144 (el "Z.^{en}" el nombre indica 144 pines)
 - Consumo y operación
 - Voltaje de operación: 1.8V a 3.6V
 - Modos de bajo consumo: Sleep, Stop, Standby
 - Temperatura de operación: -40 °C a +105 °C (grado industrial)

2.2.1. Diagrama de pines



1. The above figure shows the package top view.



Figura 8: Diagrama de la página 47 del manual STM32F4

2.2.2. Descripción de pines

2.2.3. Diagrama de bloques

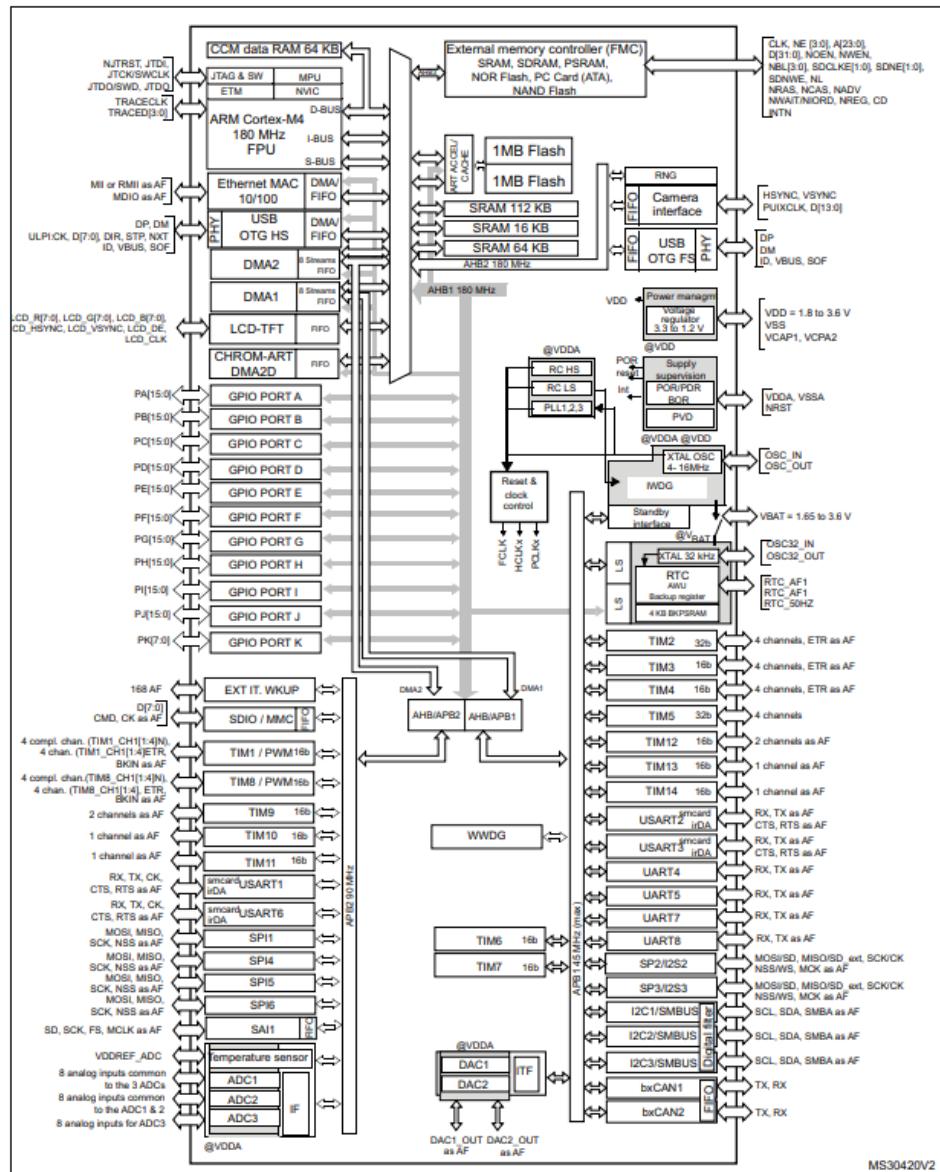


Figura 9: STM32F429ZIT6 block diagram

2.2.4. Características eléctricas

2.3. Sensor MEMS L3GD20

El L3GD20 es un giroscopio digital de tres ejes fabricado por STMicroelectronics, diseñado para medir la velocidad angular en los ejes X, Y y Z. Este sensor MEMS (Micro-Electro-Mechanical Systems) es ampliamente utilizado en sistemas de navegación, estabilización, detección de movimiento y aplicaciones de realidad aumentada, entre otros. Está integrado en la placa STM32F429 Discovery Kit y se comunica mediante una interfaz digital SPI o I2C.

Características generales del L3GD20:

- Medición y rendimiento
 - Tipo de sensor: Giroscopio MEMS de 3 ejes
 - Rangos de medición seleccionables: ± 250 , ± 500 , ± 2000 dps (grados por segundo)
 - Resolución de salida: 16 bits (por eje)
 - Sensibilidad típica:
 - ± 250 dps $\rightarrow 8.75$ mdps/LSB
 - ± 500 dps $\rightarrow 17.5$ mdps/LSB
 - ± 2000 dps $\rightarrow 70$ mdps/LSB
- Interfaces de comunicación
 - Digital: SPI (hasta 10 MHz) o I2C (hasta 400 kHz)
 - Interfaz seleccionable por hardware o software
- Rendimiento dinámico
 - Tasa de datos de salida (ODR): Hasta 800 Hz
 - Filtros:
 - Filtro paso bajo digital configurable
 - Filtro paso alto interno para reducción de ruido en ciertas aplicaciones
 - Ruido angular típico: 0.03 dps/ \sqrt{Hz}
 - Desviación de offset: <10 dps
- Características eléctricas
 - Voltaje de operación: 2.4V – 3.6V
 - Consumo típico: 6.1 mA en modo activo, <5 μ A en modo de suspensión
 - Modo de bajo consumo: Incluido, con reinicio rápido
- Funciones integradas
 - Interrupciones programables: Por umbral de velocidad angular, actividad/inactividad, FIFO lleno, etc.
 - FIFO: Buffer interno de 32 niveles para reducción de carga en el microcontrolador

- Detección de eventos: Movimiento, giros rápidos, etc.
- Encapsulado y montaje
 - Encapsulado: LGA-16 (4x4x1 mm)
 - Montaje: Superficie (SMT)

2.3.1. Diagrama de pines

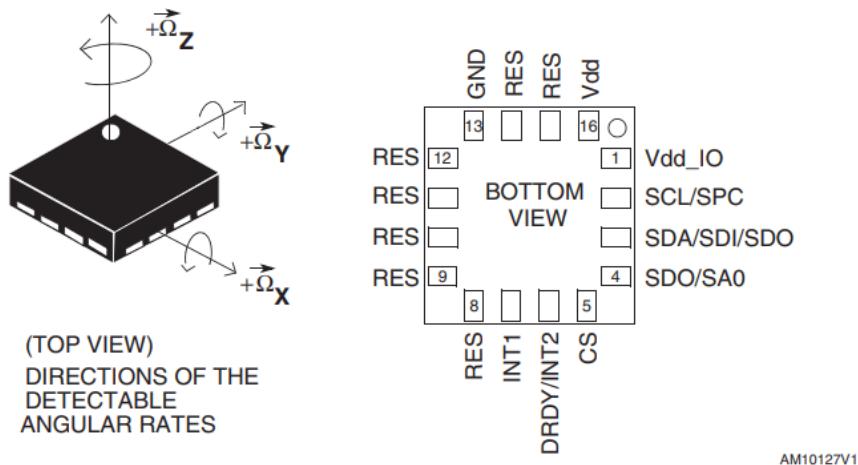


Figura 10: Pin connection

2.3.2. Descripción de pines

Pin#	Name	Function
1	Vdd_IO ⁽¹⁾	Power supply for I/O pins
2	SCL SPC	I ² C serial clock (SCL) SPI serial port clock (SPC)
3	SDA SDI SDO	I ² C serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)
4	SDO SA0	SPI serial data output (SDO) I ² C less significant bit of the device address (SA0)
5	CS	I ² C/SPI mode selection (1: SPI idle mode / I ² C communication enabled; 0: SPI communication mode / I ² C disabled)
6	DRDY/INT2	Data ready/FIFO interrupt (Watermark/Overrun/Empty)
7	INT1	Programmable interrupt
8	Reserved	Connect to GND
9	Reserved	Connect to GND
10	Reserved	Connect to GND
11	Reserved	Connect to GND
12	Reserved	Connect to GND
13	GND	0 V supply
14	Reserved	Connect to GND with ceramic capacitor ⁽²⁾
15	Reserved	Connect to Vdd
16	Vdd ⁽³⁾	Power supply

1. 100 nF filter capacitor recommended.
2. 1 nF min value must be guaranteed under 11 V bias condition.
3. 100 nF plus 10 μ F capacitors recommended.

Figura 11: Pin description

2.3.3. Diagrama de bloques

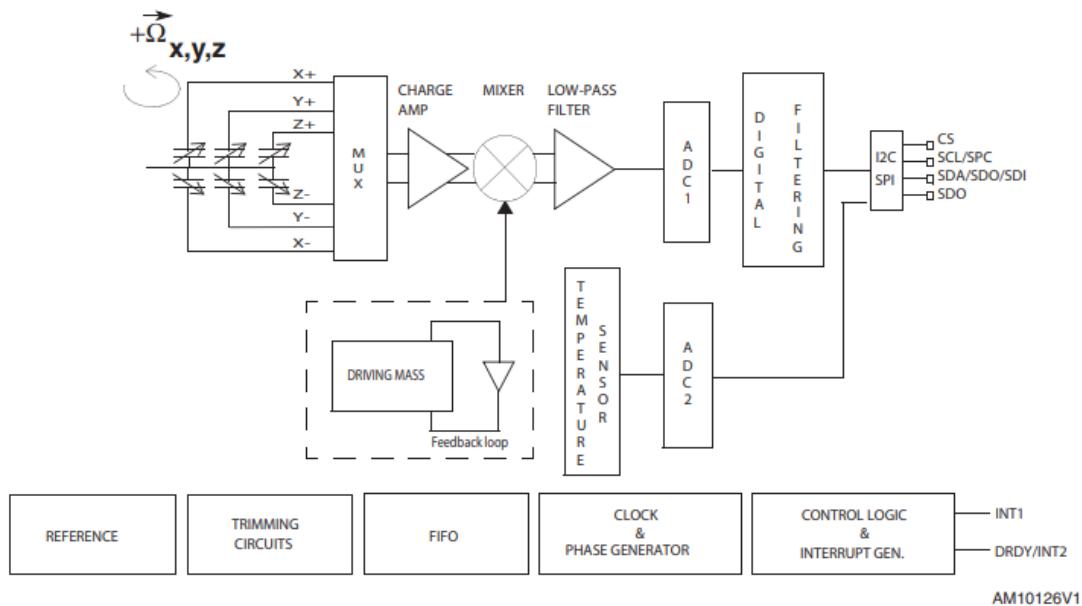


Figura 12: Block diagram

2.3.4. Características eléctricas

@ Vdd =3.0 V, T=25 °C unless otherwise noted.

Table 5. Electrical characteristics ⁽¹⁾

Symbol	Parameter	Test condition	Min.	Typ. ⁽²⁾	Max.	Unit
Vdd	Supply voltage		2.4	3.0	3.6	V
Vdd_IO	I/O pins supply voltage ⁽³⁾		1.71		Vdd+0.1	V
Idd	Supply current			6.1		mA
IddSL	Supply current in sleep mode ⁽⁴⁾	Selectable by digital interface		2		mA
IddPdn	Supply current in power-down mode	Selectable by digital interface		5		µA
VIH	Digital high level input voltage		0.8*Vdd_I_O			V
VIL	Digital low level input voltage			0.2*Vdd_I_O		V
Top	Operating temperature range		-40		+85	°C

1. The product is factory calibrated at 3.0 V.
2. Typical specifications are not guaranteed.
3. It is possible to remove Vdd maintaining Vdd_IO without blocking the communication busses; in this condition the measurement chain is powered off.
4. Sleep mode introduces a faster turn-on time relative to power-down mode.

Figura 13: Electrical characteristics

Symbol	Ratings	Maximum value	Unit
Vdd	Supply voltage	-0.3 to 4.8	V
T _{STG}	Storage temperature range	-40 to +125	°C
Sg	Acceleration g for 0.1 ms	10,000	g
ESD	Electrostatic discharge protection	2 (HBM)	kV
		1.5 (CDM)	kV
		200 (MM)	V
V _{in}	Input voltage on any control pin (CS, SCL/SPC, SDA/SDI/SDO, SDO/SA0)	-0.3 to V _{dd_IO} +0.3	V

Note: Supply voltage on any pin should never exceed 4.8 V



This is a mechanical shock sensitive device, improper handling can cause permanent damage to the part



This is an ESD sensitive device, improper handling can cause permanent damage to the part

Figura 14: Absolute maximum ratings

2.3.5. Registros

Name	Type	Register address		Default
		Hex	Binary	
Reserved	-	00-0E	-	-
WHO_AM_I	r	0F	000 1111	11010100
Reserved	-	10-1F	-	-
CTRL_REG1	rw	20	010 0000	00000111
CTRL_REG2	rw	21	010 0001	00000000
CTRL_REG3	rw	22	010 0010	00000000
CTRL_REG4	rw	23	010 0011	00000000
CTRL_REG5	rw	24	010 0100	00000000
REFERENCE	rw	25	010 0101	00000000
OUT_TEMP	r	26	010 0110	output
STATUS_REG	r	27	010 0111	output
OUT_X_L	r	28	010 1000	output
OUT_X_H	r	29	010 1001	output
OUT_Y_L	r	2A	010 1010	output
OUT_Y_H	r	2B	010 1011	output
OUT_Z_L	r	2C	010 1100	output
OUT_Z_H	r	2D	010 1101	output
FIFO_CTRL_REG	rw	2E	010 1110	00000000
FIFO_SRC_REG	r	2F	010 1111	output
INT1_CFG	rw	30	011 0000	00000000
INT1_SRC	r	31	011 0001	output
INT1_TSH_XH	rw	32	011 0010	00000000
INT1_TSH_XL	rw	33	011 0011	00000000
INT1_TSH_YH	rw	34	011 0100	00000000
INT1_TSH_YL	rw	35	011 0101	00000000
INT1_TSH_ZH	rw	36	011 0110	00000000
INT1_TSH_ZL	rw	37	011 0111	00000000
INT1_DURATION	rw	38	011 1000	00000000

Figura 15: Register address map

2.4. Pantalla LCD/TFT ILI9341

La pantalla LCD/TFT usada en la placa STM32F429 Discovery Kit está basada en el controlador gráfico ILI9341, un controlador popular fabricado por Ilitek para pantallas a color tipo TFT (Thin-Film Transistor). Esta pantalla ofrece una interfaz visual sencilla y versátil para sistemas embebidos, ideal para mostrar texto, gráficos, interfaces gráficas de usuario (GUI), datos de sensores, etc.

Características generales de la pantalla con ILI9341

- Pantalla

- Tipo: TFT LCD (Thin-Film Transistor Liquid Crystal Display)
- Tamaño: 2.4 pulgadas (en la STM32F429-DISC1)
- Resolución: 240 x 320 píxeles (QVGA)
- Color: 65,536 colores (modo RGB565 – 16 bits)
- Orientación: Vertical u horizontal (configurable por software)
- Retroiluminación: LED blanca, controlable desde el microcontrolador

- Controlador gráfico ILI9341

- Color por píxel: 16 bits (5 bits rojo, 6 bits verde, 5 bits azul – formato RGB565)
- RAM interna: 172.8 KB de GRAM para almacenamiento de imagen
- Funciones integradas:
 - Scroll vertical
 - Rotación de imagen
 - Inversión de color
 - Sleep mode / Standby
 - Control de brillo por software (mediante PWM externo)
- Velocidad de actualización: Hasta 60 FPS (dependiendo del bus y configuración)

- Interfaz de comunicación

- Soporte para múltiples interfaces:
 - SPI (Serial Peripheral Interface)
 - 8080/6800 parallel (8/16 bits)
 - MIPI DSI (en otras variantes)
- En la STM32F429 Discovery, la pantalla está conectada mediante una interfaz paralela FSMC o controlada por el LTDC (LCD-TFT Display Controller) del microcontrolador.

- Voltaje y consumo

- Voltaje típico: 2.8V a 3.3V
- Consumo: 5-10 mA (sin retroiluminación), hasta 40 mA con retroiluminación

2.4.1. Diagrama de pines

2.4.2. Descripción de pines

Power Supply Pins			
Pin Name	I/O	Type	Descriptions
VDDI	I	P	Low voltage power supply for interface logic circuits (1.65 ~ 3.3 V)
VDDI_LED	I		Power supply for LED driver interface. (1.65 ~ 3.3 V) If LED driver is not used, fix this pin at VDDI.
VCI	I	Analog Power	High voltage power supply for analog circuit blocks (2.5 ~ 3.3 V)
Vcore	O	Digital Power	Regulated Low voltage level for interface circuits Connect a capacitor for stabilization. Don't apply any external power to this pad
VSS3	I	I/O Ground	System ground level for I/O circuits.
VSS	I	Digital Ground	System ground level for logic blocks
VSSA	I	Analog Ground	System ground level for analog circuit blocks Connect to VSS on the FPC to prevent noise.
VSSC	I	Analog Ground	System ground level for analog circuit blocks Connect to VSS on the FPC to prevent noise

Interface Logic Signals				Descriptions				
Pin Name	I/O	Type	- Select the MCU interface mode				DB Pin in use	
			IM3	IM2	IM1	IM0	Register/Content	GRAM
IM[3:0]	I	(VDDI/VSS)	0	0	0	0	80 MCU 8-bit bus interface I	D[7:0] D[7:0]
			0	0	0	1	80 MCU 16-bit bus interface I	D[7:0] D[15:0]
			0	0	1	0	80 MCU 9-bit bus interface I	D[7:0] D[8:0]
			0	0	1	1	80 MCU 18-bit bus interface I	D[7:0] D[17:0]
			0	1	0	1	3-wire 9-bit data serial interface I	SDA: In/OUT
			0	1	1	0	4-wire 8-bit data serial interface I	SDA: In/OUT
			1	0	0	0	80 MCU 16-bit bus interface II	D[8:1] D[17:10], D[8:1]
			1	0	0	1	80 MCU 8-bit bus interface II	D[17:10] D[17:10]
			1	0	1	0	80 MCU 18-bit bus interface II	D[8:1] D[17:0]
			1	0	1	1	80 MCU 9-bit bus interface II	D[17:10] D[17:9]
			1	1	0	1	3-wire 9-bit data serial interface II	SDI: In SDO: Out
			1	1	1	0	4-wire 8-bit data serial interface II	SDI: In SDO: Out

MPU Parallel interface bus and serial interface select
If use RGB Interface must select serial interface.
*: Fix this pin at VDDI or VSS.

Figura 16: Descripción de pines

RESX	I	MCU (VDDI/VSS)	This signal will reset the device and must be applied to properly initialize the chip. Signal is active low.
EXTC	I	MCU (VDDI/VSS)	Extended command set enable. Low: extended command set is discarded. High: extended command set is accepted. Please connect EXTC to VDDI to read/write extended registers (RB0h~RCFh, RE0h~RFFh)
CSX	I	MCU (VDDI/VSS)	Chip select input pin ("Low" enable). This pin can be permanently fixed "Low" in MPU interface mode only. * note1.2
D/CX (SCL)	I	MCU (VDDI/VSS)	This pin is used to select "Data or Command" in the parallel interface or 4-wire 8-bit serial data interface. When DCX = '1', data is selected. When DCX = '0', command is selected. This pin is used serial interface clock in 3-wire 9-bit / 4-wire 8-bit serial data interface. <i>If not used, this pin should be connected to VDDI or VSS.</i>
RDX	I	MCU (VDDI/VSS)	8080- I /8080- II system (RDX): Serves as a read signal and MCU read data at the rising edge. <i>Fix to VDDI level when not in use.</i>
WRX (D/CX)	I	MCU (VDDI/VSS)	- 8080- I /8080- II system (WRX): Serves as a write signal and writes data at the rising edge. - 4-line system (D/CX): Serves as command or parameter select. <i>Fix to VDDI level when not in use.</i>
D[17:0]	I/O	MCU (VDDI/VSS)	18-bit parallel bi-directional data bus for MCU system and RGB interface mode <i>Fix to VSS level when not in use</i>
SDI/SDA	I/O	MCU (VDDI/VSS)	When IM[3] : Low, Serial in/out signal. When IM[3] : High, Serial input signal. The data is applied on the rising edge of the SCL signal. <i>If not used, fix this pin at VDDI or VSS.</i>
SDO	O	MCU (VDDI/VSS)	Serial output signal. The data is outputted on the falling edge of the SCL signal. If not used, open this pin
TE	O	MCU (VDDI/VSS)	Tearing effect output pin to synchronize MPU to frame writing, activated by S/W command. When this pin is not activated, this pin is low. If not used, open this pin.
DOTCLK	I	MCU (VDDI/VSS)	Dot clock signal for RGB interface operation. <i>Fix to VDDI or VSS level when not in use.</i>
VSYNC	I	MCU (VDDI/VSS)	Frame synchronizing signal for RGB interface operation. <i>Fix to VDDI or VSS level when not in use.</i>
HSYNC	I	MCU (VDDI/VSS)	Line synchronizing signal for RGB interface operation. <i>Fix to VDDI or VSS level when not in use.</i>
DE	I	MCU (VDDI/VSS)	Data enable signal for RGB interface operation. <i>Fix to VDDI or VSS level when not in use.</i>

Figura 17: Descripción de pines

LCD Driver Input/Output Pins			
Pin Name	I/O	Type	Descriptions
S720~S1	O	Source	Source output signals.. Leave the pin to open when not in use.
G320~G1	O	Gate	Gate output signals. Leave the pin to open when not in use.
DDVDH	O	Power Stabilizing capacitor	Output voltage of 1st step up circuit (2 x VCl). Input voltage to 2nd step up circuit. Generated power output pad for source driver block. Connect this pad to the capacitor for stabilization.
VGH	O	Power Stabilizing capacitor	Power supply for the gate driver. Adjust the VGH level with the BT[2:0] bits. Connect this pad with a stabilizing capacitor.
VGL	O	Power Stabilizing capacitor	Power supply for the gate driver. Adjust the VGL level with the BT[2:0] bits. Connect this pad with a stabilizing capacitor.
VCL	O	Power Stabilizing capacitor	Power supply for VCOML. VCL = 0~ - VCI Connect this pad with a stabilizing capacitor.
C11P, C11M C12P, C12M	P	Stabilizing capacitor	Connect the charge-pumping capacitor for generating DDVDH level.
C21P, C21M C22P, C22M	P	Stabilizing capacitor	Connect the charge-pumping capacitor for generating VGH, VGL level.
GVDD	O		High reference voltage for grayscale voltage generator. Internal register can be used to adjust the voltage.
VCOM	O		Power supply pad for the TFT- display counter electrode. Charge recycling method is used with VCI and VSSA voltage. Connect this pad to the TFT-display counter electrode.
LEDPWM	O		Output pin for PWM (Pulse Width Modulation) signal of LED driving. If not used, open this pad.
LEDON	O		Output pin for enabling LED driving. If not used, open this pad.

Test Pins			
Pin Name	I/O	Type	Descriptions
DUMMY	-	Open	Input pads used only for test purpose at IC-side. During normal operation, leave these pads open.
INT_TEST1 INT_TEST2	-	Open	Input pads used only for test purpose at IC-side. During normal operation, leave these pads open.

Figura 18: Descripción de pines

Liquid crystal power supply specifications Table

No.	Item	Description
1	TFT Source Driver	720 pins (240 x RGB)
2	TFT Gate Driver	320 pins
3	TFT Display's Capacitor Structure	Cst structure only (Cs on Common)
4	Liquid Crystal Drive Output	S1 ~ S720 V0 ~ V63 grayscales G1 ~ G320 VGH - VGL VCOM VCOMH - VCOML: Amplitude = electronic volumes
5	Input Voltage	VDDI 1.65V ~ 3.30V VCI 2.50V ~ 3.30V
6	Liquid Crystal Drive Voltages	DDVDH 4.5V ~ 5.8V VGH 10.0V ~ 18.0V VGL -5.0V ~ -10.0V VCL -1.5V ~ -2.5V VGH - VGL Max. 28.0V DDVDH VCI x2, VGH VCI x6, x7 VGL VCI x-3, x-4, VCL VCI x-1
7	Internal Step-up Circuits	

Figura 19: Descripción de pines

2.4.3. Diagrama de bloques

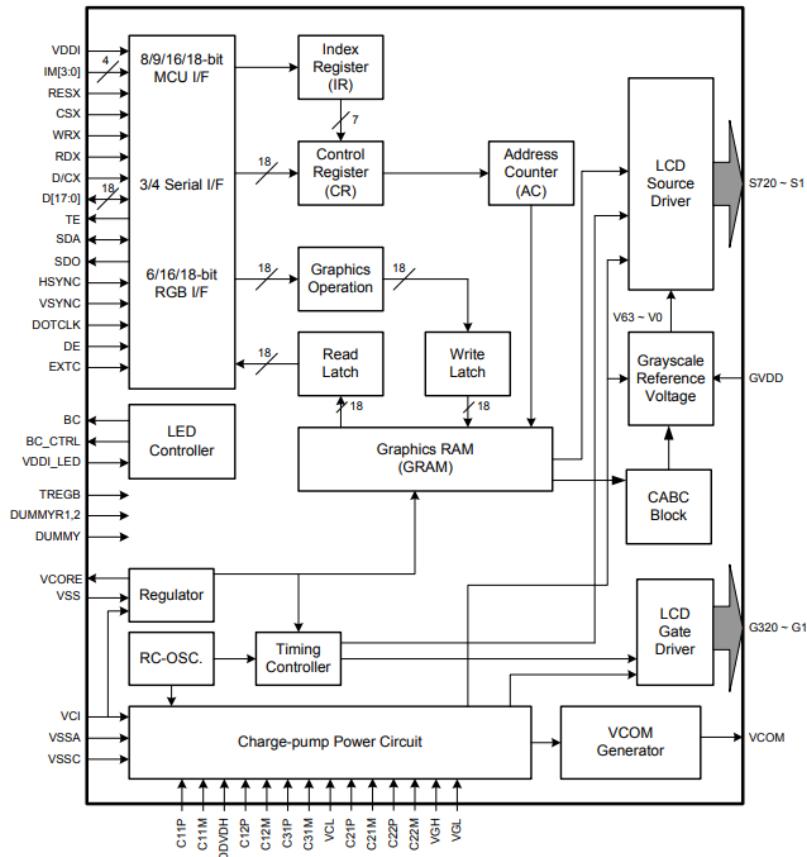


Figura 20: Block Diagram

2.4.4. Características eléctricas

Item	Symbol	Unit	Value
Supply voltage	VCI	V	-0.3 ~ +4.6
Supply voltage (Logic)	VDDI	V	-0.3 ~ +4.6
Supply voltage (Digital)	VCORE	V	-0.3 ~ +2.0
Driver supply voltage	VGH-VGL	V	-0.3 ~ +28.0
Logic input voltage range	VIN	V	-0.3 ~ VDDI + 0.3
Logic output voltage range	VO	V	-0.3 ~ VDDI + 0.3
Operating temperature	Topr	°C	-40 ~ +85
Storage temperature	Tstg	°C	-55 ~ +110

Note: If the absolute maximum rating of even one of the above parameters is exceeded even momentarily, the quality of the product may be degraded. Absolute maximum ratings, therefore, specify the values exceeding which the product may be physically damaged. Be sure to use the product within the range of the absolute maximum ratings.

Figura 21: Absolute Maximum Ratings

Item	Symbol	Unit	Condition	Min.	Typ.	Max.	Note
Power and Operation Voltage							
Analog Operating Voltage	VCI	V	Operating voltage	2.5	2.8	3.3	Note2
Logic Operating Voltage	VDDI	V	I/O supply voltage	1.65	2.8	3.3	Note2
Digital Operating voltage	VCORE	V	Digital supply voltage	-	1.5	-	Note2
Gate Driver High Voltage	VGH	V	-	10.0	-	18.0	Note3
Gate Driver Low Voltage	VGL	V	-	-10.0	-	-5.0	Note3
Driver Supply Voltage	-	V	VGH-VGL	15	-	28	Note3
Current consumption during standby mode	IST	μA	VCI=2.8V , Ta=25 °C	-	-	100	-
Input and Output							
Logic High Level Input Voltage	VIH	V	-	0.7*VDDI	-	VDDI	Note1,2,3
Logic Low Level Input Voltage	VIL	V	-	VSS	-	0.3*VDDI	Note1,2,3
Logic High Level Output Voltage	VOH	V	IOL=-1.0mA	0.8*VDDI	-	VDDI	Note1,2,3
Logic Low Level Output Voltage	VOL	V	IOL=1.0mA	VSS	-	0.2*VDDI	Note1,2,3
Logic High Level Input Current	IIH	uA	-	-	-	1	Note1,2,3
Logic Low Level input Current	IIL	uA	-	-1	-	-	Note1,2,3
Logic Input Leakage Current	ILEA	uA	VIN=VDDI or VSS	-0.1	-	+0.1	Note1,2,3
VCOM Operation							
VCOM High Voltage	VCOMH	V	Ccom=12nF	2.5	-	5.0	Note3
VCOM Low Voltage	VCOML	V	Ccom=12nF	-2.5	-	0.0	Note3
VCOM Amplitude Voltage	VCOMA	V	VCOMH-VCOML	4.0	-	5.5	Note3
Source Driver							
Source Output Range	Vsout	V	-	0.1	-	DDVDH-0.1	Note4
Gamma Reference Voltage	GVDD	V	-	3.0	-	5.0	Note3
Output Deviation Voltage (Source Output channel)	Vdev	mV	Sout>=4.2V Sout<=0.8V 4.2V>Sout>0.8V	-	-	20 15 -	Note4
Output Offset Voltage	VOFSET	mV	-	-	-	35	Note7
Booster Operation							
1 st Booster (VClx2) Voltage	DDVDH	V	-	4.95 (Note 5)	-	5.8 (Note 6)	Note3
1 st Booster (VClx2 Drop Voltage	VClx2 drop	%	loading=1mA	-	-	5	Note3
Liner Range	Vliner	V	-	0.2	-	DDVDH-0.2	

Figura 22: General DC Characteristics

2.5. Biblioteca LibOpenCM3

La biblioteca LibOpenCM3 es una biblioteca de código abierto desarrollada para facilitar el acceso y control de periféricos en microcontroladores basados en la arquitectura ARM Cortex-M. Está escrita en lenguaje C y sigue un enfoque de bajo nivel, pero ofrece una capa de abstracción estandarizada que simplifica el desarrollo de firmware sin depender de bibliotecas propietarias como las de los fabricantes (por ejemplo, STM32 HAL o STM32Cube).

Su diseño modular, portátil y transparente la hace especialmente atractiva para proyectos educativos, de investigación, o para desarrolladores que desean tener un control total sobre el hardware y el código generado.

Características principales

- Compatibilidad con múltiples familias de microcontroladores:
 - STM32 (F0, F1, F2, F3, F4, F7, L0, L1, L4)
 - LPC, EFM32, GD32, entre otros Cortex-M
 - Totalmente compatible con la familia STM32F429 (usada en la STM32F429 Discovery Kit)
- Código abierto y comunitario
 - Licencia: LGPL v3 (permite uso comercial bajo condiciones)
 - Comunidad activa y disponible en GitHub
 - Puede ser auditada, modificada y optimizada fácilmente
- Acceso directo al hardware
 - Cada periférico se maneja por medio de registros mapeados en memoria
 - Las funciones son explícitas y transparentes (por ejemplo, `gpio_set`, `timer_set_prescaler`, `uart_send_blocking`, etc.)
- Documentación en código
 - Los archivos fuente están ampliamente comentados
 - La estructura modular permite aprender cómo funciona cada periférico internamente

2.6. Comunicación UART/USART

La interfaz UART (Universal Asynchronous Receiver-Transmitter) permite el intercambio asíncrono de datos entre circuitos integrados sin un reloj compartido; USART extiende esa idea y añade la modalidad sincrónica mediante un reloj externo. Tal flexibilidad explica su presencia en controladores para drones, paneles fotovoltaicos, módulos Bluetooth LE, comunicación entre microcontroladores y pc, entre otras utilidades. ¿Por qué se prefiere en tales contextos? Porque la lógica requerida se reduce a un par de registros y una línea de interrupción, con consumo de potencia modesto. No obstante, la total ausencia de señal de sincronía obliga a cuidar tolerancias de baudios y capacidades de la capa física, sobre todo cuando la temperatura o la longitud del bus varía. Esa dualidad, baja carga de hardware frente al cuidado en la temporización, ha consolidado a UART y USART como cimientos de la ingeniería embebida.

2.7. Protocolo de Comunicación SPI (Serial Peripheral Interface)

El SPI (Serial Peripheral Interface) es un protocolo de comunicación sincrónica que permite la transferencia rápida de datos entre un maestro y uno o varios dispositivos esclavos. Desde un punto de vista teórico, SPI se caracteriza por ser full-duplex y utilizar múltiples líneas (MOSI, MISO, SCK y SS) para establecer una conexión determinística y eficiente. Su estructura de comunicación basada en relojes compartidos lo convierte en una solución ideal para la interacción con periféricos de alta velocidad como memorias flash, sensores, pantallas LCD, entre otros. El modelo maestro-esclavo en SPI refleja una arquitectura jerárquica que, aunque simple, requiere una configuración precisa del hardware para garantizar una transmisión confiable.

La interfaz serial interactúa con el mundo exterior a través de 4 cables: CS, SPC, SDI y SDO.

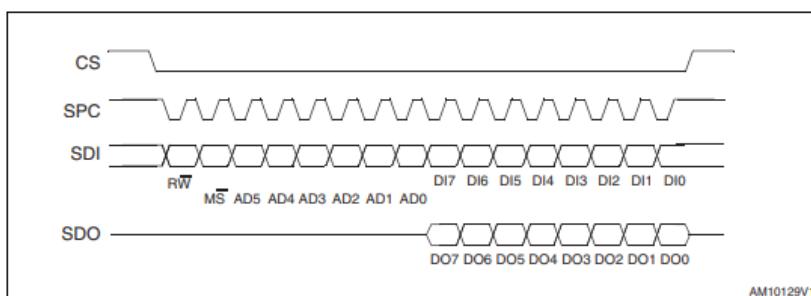
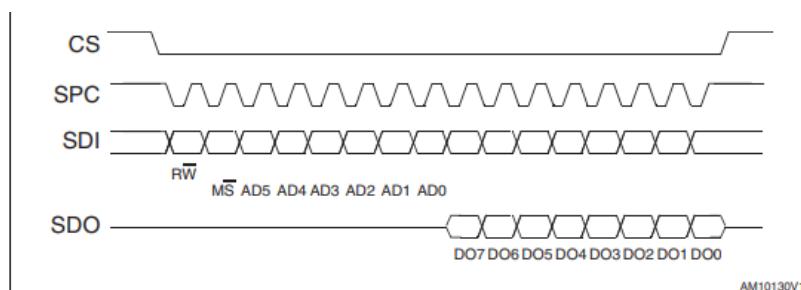


Figura 23: SPI - Read and write protocol

2.7.1. SPI read

El comando de lectura SPI se ejecuta con 16 pulsos de reloj. El comando de lectura de múltiples bytes se ejecuta añadiendo bloques de 8 pulsos de reloj al anterior.



The SPI read command is performed with 16 clock pulses. The multiple byte read command is performed by adding blocks of 8 clock pulses to the previous one.

bit 0: READ bit. The value is 1.

bit 1: MS bit. When 0 do not increment address; when 1 increment address in multiple reading.

bit 2-7: address AD(5:0). This is the address field of the indexed register.

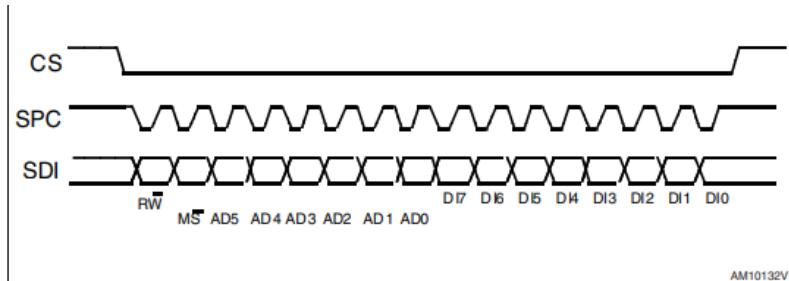
bit 8-15: data DO(7:0) (read mode). This is the data that will be read from the device (MSb first).

bit 16-... : data DO(...-8). Further data in multiple byte reading.

Figura 24: SPI read protocol

2.7.2. SPI write

El comando de escritura SPI se ejecuta con 16 pulsos de reloj. El comando de escritura de múltiples bytes se ejecuta añadiendo bloques de 8 pulsos de reloj al anterior.



The SPI Write command is performed with 16 clock pulses. The multiple byte write command is performed by adding blocks of 8 clock pulses to the previous one.

bit 0: WRITE bit. The value is 0.

bit 1: \bar{MS} bit. When 0, do not increment address; when 1, increment address in multiple writing.

bit 2-7: address AD(5:0). This is the address field of the indexed register.

bit 8-15: data DI(7:0) (write mode). This is the data that will be written to the device (MSb first).

bit 16-... : data DI(...-8). Further data in multiple byte writing.

Figura 25: SPI write protocol

2.8. MQTT (Message Queuing Telemetry Transport)

MQTT (Message Queuing Telemetry Transport): MQTT es un protocolo de mensajería ligero basado en el modelo de publicación/suscripción, diseñado específicamente para comunicaciones eficientes y eficaces en redes con recursos limitados, como ocurre en entornos de IoT (Internet of Things). Se basa en un intermediario central conocido como el *broker*, que recibe y recoge todos los mensajes y los distribuye únicamente a quienes están suscritos a cada “topic”. De este modo, emisores y receptores quedan totalmente desacoplados y pueden operar de forma asíncrona e independiente, lo que garantiza fiabilidad en la entrega de mensajes y asegura la transmisión con mínimo retraso. Esto favorece escalabilidad y crecimiento futuro. [2]

Las principales características son:

- **Paquetes mínimos:** solo dos bytes de control, lo cual reduce notablemente el consumo de ancho de banda y de energía.
- **Niveles de QoS:**
 - **QoS 0:** entrega “como máximo una vez” (sin confirmación).
 - **QoS 1:** entrega “al menos una vez” (con acuse de recibo).
 - **QoS 2:** entrega “exactamente una vez” (intercambio en cuatro pasos).
- **Desacoplamiento total:** no es necesario que el publicador conozca a los suscriptores ni viceversa.
- **Bajo consumo:** se minimiza el uso de batería, lo cual ayuda a prolongar su duración.

Gracias a estas particularidades propias del protocolo, se ha convertido por defecto en un estándar en domótica y telemetría, donde cada bit cuenta y cada dato es importante.

2.9. Internet de las Cosas (IoT)

El Internet de las Cosas (IoT) describe la interconexión digital de equipos físicos capaces de capturar, procesar y compartir información por medio de una red. Desde la perspectiva académica, se concibe como una prolongación de la computación presente en todas partes: objetos comunes reciben sensores, circuitos lógicos y módulos de radio para integrarse en sistemas inteligentes. Debido a las restricciones de cómputo, memoria y energía, las arquitecturas se diseñan con bloques ligeros; ejemplos habituales son microcontroladores de 32 bits con conjunto de instrucciones reducido, radios Wi-Fi de bajo consumo y sensores MEMS de tres ejes. En producción se combinan protocolos como MQTT y pasarelas en la nube para habilitar aplicaciones de medición de energía en tiempo real, mantenimiento predictivo en líneas de ensamblaje y gestión de riego agrícola según datos de humedad. El campo presenta retos de seguridad, interoperabilidad y gestión de grandes volúmenes de datos, factores que lo mantienen en el centro de la investigación multidisciplinaria.

2.10. Diseño del circuito

A continuación se presenta el diseño final del circuito:



Figura 26: Diseño final, parte 1

Elaboración propia



Figura 27: Diseño final, parte 2
Elaboración propia

2.10.1. Lista de componentes

Tabla 1: Lista de componentes

Componente	Cantidad	Precio (CRC)
STM32F429 Discovery kit	1	17336 [4]
3.9kΩ	1	200
4.7kΩ	1	200
Batería 9V	1	2300

2.10.2. Información adicional

GPIOs (General Purpose Input/Output)

Los GPIOs del STM32F429ZI son pines que pueden configurarse como entrada o salida digital para múltiples funciones. Algunas características clave:

- Puertos disponibles: GPIOA a GPIOK (no todos están disponibles externamente en el Discovery Kit).
- Configuración por software: Cada pin puede configurarse como:
 - Entrada digital
 - Salida digital
 - Función alternativa (periféricos: SPI, I2C, UART, etc.)
 - Entrada analógica (para ADC)
- Resistencia interna:
 - Pull-up
 - Pull-down
 - Sin resistencia
- Velocidad configurable:
 - Baja, media, alta y muy alta (para adaptarse a diferentes requisitos de interferencia/consumo)
- Voltaje de operación:
 - 3.3 lógicos (no 5V tolerant por defecto en todos los pines)
 - Algunos pines son tolerantes a 5V, pero no todos, especialmente los usados como entrada analógica no lo son.

Convertidores Analógicos a Digitales (ADC)

El STM32F429ZI posee 3 ADCs independientes (ADC1, ADC2 y ADC3), cada uno con múltiples canales:

Especificaciones principales:

- Resolución: 12 bits
 - Valores posibles: 0 a 4095
- Voltaje de referencia (V_{REF}): típicamente 3.3V (V_{DD} del micro)

- Entradas múltiples: hasta 16 canales por ADC (pueden usarse en modo independiente o simultáneo)
- Tasa de muestreo: hasta 2.4 MSPS (millones de muestras por segundo)
- Modos de funcionamiento:
 - Modo escaneo (para leer varios canales)
 - Modo continuo o discontinuo
 - Trigger por software o por eventos (timer, etc.)
- Conversión directa: puede leer sensores, voltajes de entrada, etc.

3. Desarrollo

3.1. Análisis del firmware

3.1.1. Diagrama de flujo del programa

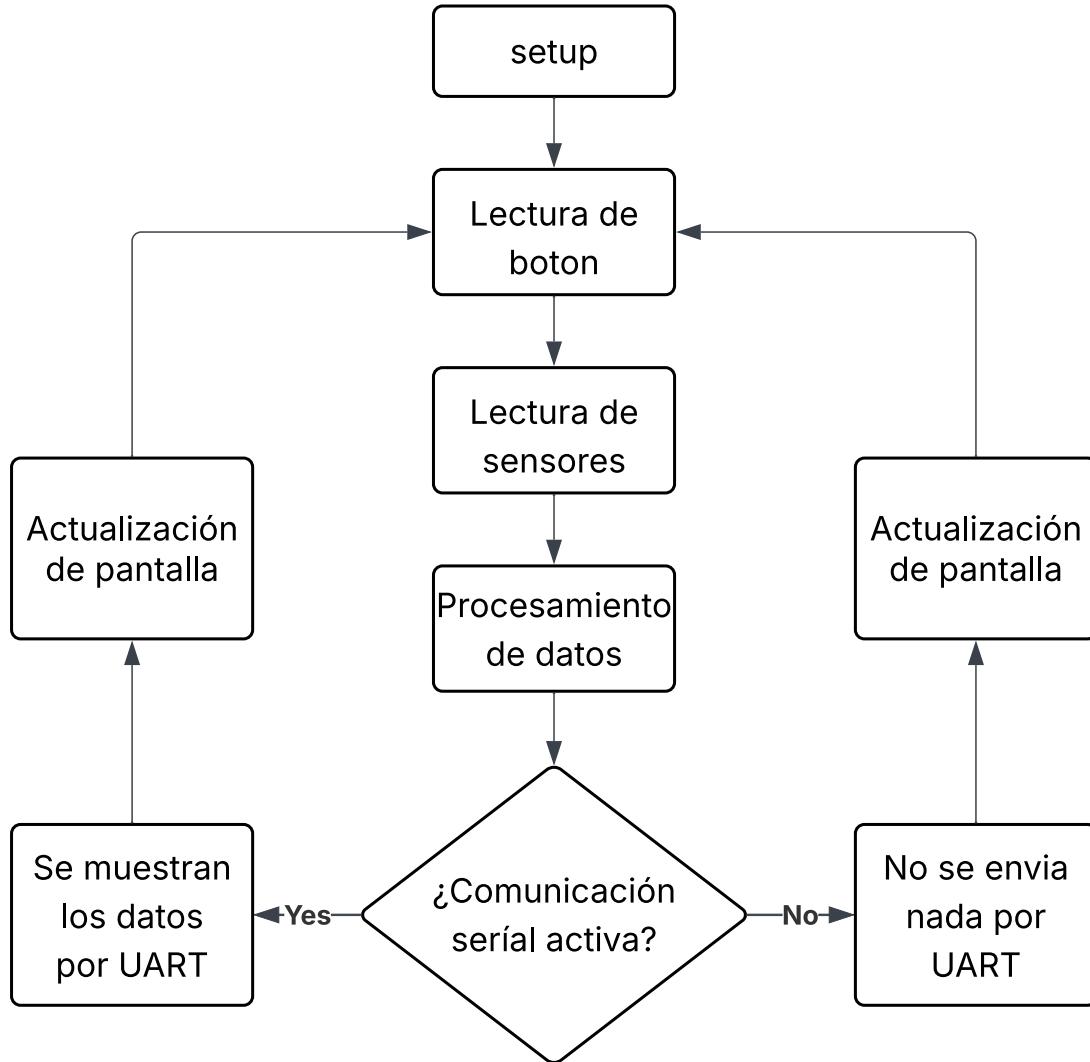


Figura 28: Diagrama de flujo de **seismograph**
Elaboración propia

- **Setup:** configuración de reloj, consola UART, SPI, GPIOs, ADC, giroscopio, SDRAM y LCD.
- **Lectura de botón:** lectura del estado del pin GPIOA0 para alternar la variable EN_comms.
- **Lectura de sensores:** lectura de los ejes X, Y y Z desde el giroscopio y del nivel de batería mediante ADC.
- **Procesamiento de datos:** escalado de valores de los ejes según sensibilidad y conversión de datos numéricos a cadenas de texto.
- **¿Comunicación serial activa?:** comprobación de si EN_comms está a 1.

- **Se muestran los datos por UART:** envío de las cadenas formateadas por la interfaz serial y alternancia del LED indicador.
- **No se envía nada por UART:** omisión del envío serial y apagado del LED de actividad.
- **Actualización de pantalla:** limpieza y redibujo de la LCD con los valores de los ejes, el estado de UART y el nivel de batería.

3.1.2. Firmware

Este firmware implementa un sistema de adquisición y visualización de datos sísmicos utilizando un microcontrolador STM32F429ZIT6. Se comunica con un sensor giroscópico L3GD20 mediante SPI para obtener datos de movimiento en tres ejes, utiliza el ADC interno para monitorear el nivel de batería, permite la habilitación o deshabilitación de la comunicación serial mediante un botón de usuario, y presenta la información adquirida en una pantalla LCD utilizando una interfaz gráfica.

El diseño modular y el uso de bibliotecas específicas permiten una fácil adaptación y expansión del sistema para incluir funcionalidades adicionales, como el almacenamiento de datos, la comunicación inalámbrica o la integración con plataformas de IoT.

3.1.3. Inclusión de Bibliotecas

```
#include <libopencm3/stm32/...> // Bibliotecas para controlar perifericos del STM32
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>
#include "clock.h"
#include "console.h"
#include "sdram.h"
#include "lcd-spi.h"
#include "gfx.h"
#include <math.h>
```

Se incluyen:

- Bibliotecas estándar (como `stdio`, `stdint.h`)
- Bibliotecas de `libopencm3` para controlar SPI, ADC, USART, GPIO, RCC.
- Bibliotecas personalizadas para reloj, consola, pantalla LCD y gráficos.

3.1.4. Macros de configuración del giroscopio

```
#define GYR_RNW (1 << 7)
#define GYR_MNS (1 << 6)
#define GYR_WHO_AM_I 0x0F
...
#define L3GD20_SENSITIVITY_500DPS 0.0175f
```

Estas constantes definen direcciones de registros del giroscopio **L3GD20**, así como valores clave para su configuración y escalado (sensibilidad para 500 dps).

3.1.5. Flags globales

```
bool EN_comms = false;  
bool warning_BAT = false;
```

- **EN_comms**: Habilita o deshabilita la comunicación serial.
- **warning_BAT**: Indica si la batería está baja.

3.1.6. Prototipos de funciones

```
static void seismograph_setup(void);  
static void spi_setup(void);  
...
```

Se definen las funciones utilizadas para inicializar y operar el sistema.

3.1.7. Función `spi_setup()`

Inicializa el periférico **SPI5** para comunicarse con el giroscopio.

```
rcc_periph_clock_enable(RCC_SPI5); // Habilita reloj SPI5  
gpio_mode_setup(...); // Configura pines SPI (SCK, MISO, MOSI)  
spi_set_master_mode(SPI5); // Modo maestro  
spi_enable(SPI5); // Habilita SPI
```

3.1.8. Función `button_setup()`

Configura el botón en **GPIOA0** como entrada sin pull-up/down.

3.1.9. Función `led_setup()`

Configura los LEDs en **GPIOG13** (comms) y **GPIOG14** (batería) como salida.

3.1.10. Función `adc_setup()` y `read_adc_naiive()`

Configura el **ADC1** y define una función para leer un canal específico.

- **GPIOA2** y **GPIOA6** como entradas analógicas.
- Lectura bloqueante con `adc_eoc()`.

3.1.11. Función seismograph_setup()

Inicializa todos los periféricos (clock, SPI, botones, LED, ADC), y configura el giroscopio L3GD20:

```
// Enviar configuracion a GYR_CTRL_REG1 y GYR_CTRL_REG4
spi_send(SPI5, GYR_CTRL_REG1);
...
```

- Activa el giroscopio y habilita ejes **X**, **Y** y **Z**.
- Define sensibilidad a ± 500 dps.

3.1.12. Función main()

```
seismograph_setup(); // Inicializa hardware
sdram_init(); // SDRAM externa
lcd_spi_init(); // LCD por SPI
gfx_init(...); // Biblioteca gráfica
```

Pantalla de bienvenida:

Se inicializa la pantalla LCD de 240x320 píxeles y se muestra información del sistema.

```
gfx_puts("seismograph");
...
```

Después de 8 segundos, entra al lazo principal que ejecuta **seismograph()** continuamente.

3.1.13. Función seismograph()

Realiza una iteración de lectura y visualización de datos del giroscopio y batería.

a. Lectura del botón

```
if (gpio_get(GPIOA, GPIO0)) { ... }
```

Altera el estado de **EN_comms** al presionar el botón.

b. Lectura de registros del giroscopio por SPI

Lee:

- WHO_AM_I para verificar ID del dispositivo.
- STATUS_REG y OUT_TEMP (temperatura interna).
- Datos de los ejes **X**, **Y** y **Z** (en 2 bytes cada uno):

```
spi_send(SPI5, GYR_OUT_X_L | GYR_RNW);
X = spi_read(SPI5); // LSB
...
X |= spi_read(SPI5) << 8; // MSB
```

c. Escalado de datos

```
X = X * L3GD20_SENSITIVITY_500DPS;
Y = Y * L3GD20_SENSITIVITY_500DPS;
Z = Z * L3GD20_SENSITIVITY_500DPS;
```

Convierte de valores crudos del giroscopio a grados por segundo (dps).

d. Lectura de batería

```
V = read_adc_naiive(2) * 100 / 38;
```

Convierte la lectura del ADC en voltaje aproximado.

e. Detección de batería baja

```
if (V < 78) {
    warning_BAT = !warning_BAT;
```

Activa la bandera `warning_BAT` si la batería cae por debajo del umbral.

3.2. Análisis electrónico

3.2.1. Lectura del nivel de la batería

Uno de los requerimientos del laboratorio consiste en medir el nivel de una batería cuadrada de 9V. Sin embargo el microcontrolador solo admite un voltaje máximo de 3V en sus pines de entrada, aunque algunos de ellos pueden tolerar hasta 5V. Por esta razón se diseña un circuito divisor de voltaje. En la siguiente Figura se muestra el esquemático:

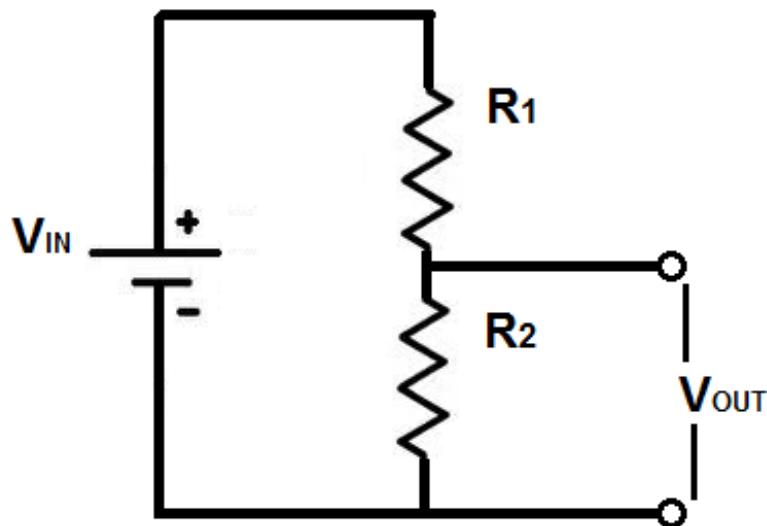


Figura 29: Esquemático de un divisor de tensión ideal

De acuerdo con la ecuación del divisor de tensión:

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in}$$

Para el caso en que $V_{out} = 5V$ y $V_{in} = 9V$, se tiene lo siguiente:

Seteando $R_2 = 4,7k\Omega$

$$\frac{5}{9} \cdot \frac{1}{4,7k} = \frac{1}{R_1 + 4,7k} \Rightarrow R_1 = \frac{9}{5} \cdot 4,7k - 4,7k = 3760\Omega$$

Por lo que en resistencias comerciales:

$$R_1 = 3,9k\Omega \quad (1)$$

$$R_2 = 4,7k\Omega \quad (2)$$

Para comprobar los resultados se realiza una simulación en el software **SimulIDE** en donde se obtienen los siguientes resultados:

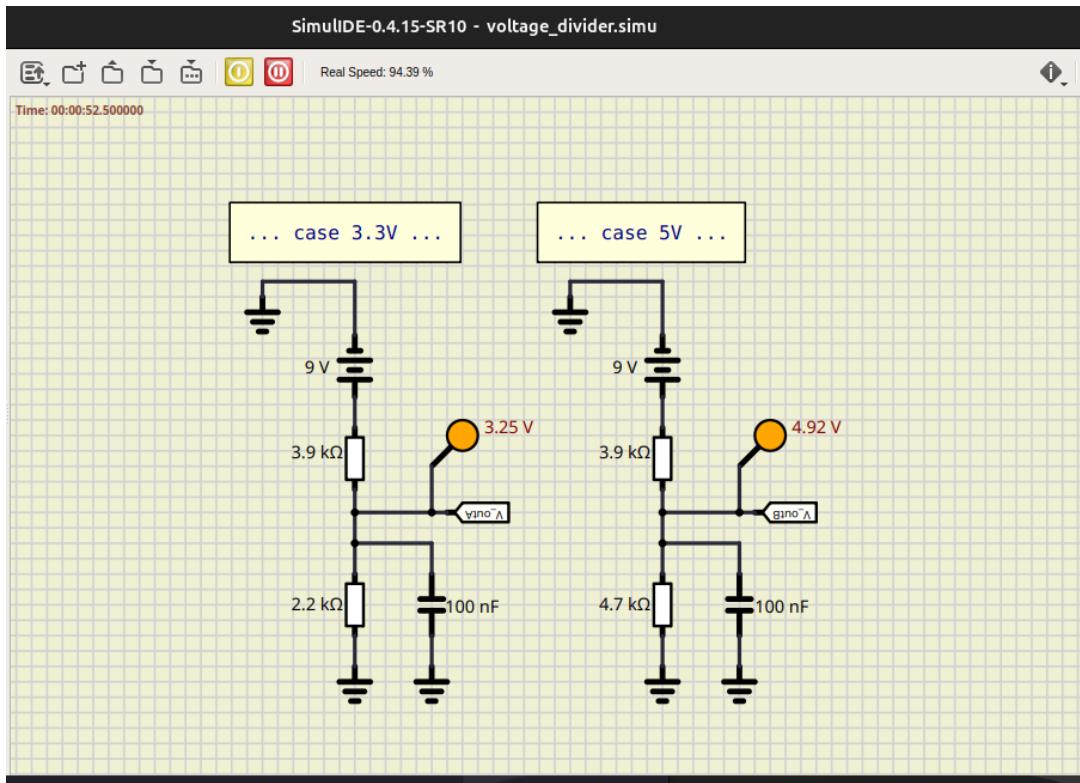


Figura 30: Esquemático de divisor de tensión simulado para obtener 5V a la salida
Elaboración propia

De acuerdo a la Figura anterior se analizan ambos casos pero se trabaja en el caso de 5V.

De igual manera se muestra el resultado de la parte práctica en donde se corrobora los resultados con la ayuda de un multímetro:



Figura 31: Medición de la tensión de salida con la ayuda de un multímetro
Elaboración propia

3.3. Análisis de resultados

A continuación se muestran una serie de capturas de pantalla en donde se observa una funcionalidad específica que comprueba el correcto funcionamiento del laboratorio:

3.3.1. Circuito con divisor de tensión

Como parte del análisis de resultados, se comprueba que el divisor de tensión entrega una salida de aproximadamente 5V, lo cual confirma que su implementación fue exitosa. Esto permite cumplir con el requerimiento de adaptar el voltaje de entrada para los pines del kit, que son utilizados como entradas analógicas y deben recibir una señal en el orden de 5V.



Figura 32: Circuito con divisor de tensión
Elaboración propia

3.3.2. Circuito con la batería ON

En la siguiente Figura se muestra que efectivamente se hace una lectura correcta del nivel de la batería, y como se observa en el valor mostrado es de alrededor de 97 % lo que valida tanto el circuito

divisor de tensión elaborado así como la parte de software que se encarga de medir la lectura del nivel de la batería.



Figura 33: Circuito con la batería ON

Elaboración propia

3.3.3. Circuito con la batería OFF

De igual manera, en la siguiente Figura se muestra una captura en donde se observa el circuito cuando la batería es desconectada, se aprecia que el nivel de batería es de aproximadamente 7% pero esto se debe mas que todo al ruido que entra al circuito lo que marca que haya una lectura diferente del 0% esperado, pero aun así es un valor aceptado para fines demostrativos; también observar que el led correspondiente encargado de avisar si la batería esta baja, se encuentra parpadeando, por lo que en cuanto a este apartado se satisface su funcionamiento.



Figura 34: Circuito con la batería OFF
Elaboración propia

3.3.4. Circuito con la comunicación serial ON

En consiguiente, también de acuerdo a la totalidad de los requisitos, a continuación se muestra una captura que hace referencia a cuando la comunicación serial esta activa, y de igual forma que como se menciono en el nivel de la batería, en este caso el led encargado de verificar que la comunicación serial esta activa se encuentra en estado activo.



Figura 35: Circuito con la comunicación serial ON
Elaboración propia

3.3.5. Circuito con la comunicación serial OFF

Ahora bien, como parte de los requisitos cuando el botón es presionado la comunicación serial se debe desactivar, y como se observa a continuación en la pantalla del microcontrolador la comunicación cambia su estado a OFF y el led correspondiente también cambia su estado a OFF, lo cual verifica que su implementación es correcta.



Figura 36: Circuito con la comunicación serial OFF
Elaboración propia

3.3.6. IoT con batería ON

Finalmente uno de los requisitos del laboratorio es que la información que capta el sismógrafo sea desplegada en una plataforma IoT, por lo tanto a continuación se muestran las capturas de pantalla que corroboran el funcionamiento correcto del laboratorio tanto en la parte práctica-física como en la implementación del firmware, por lo tanto se tienen a continuación los datos capturados por el sismógrafo de las aceleraciones en los ejes (X, Z, Y) y el nivel de la batería.

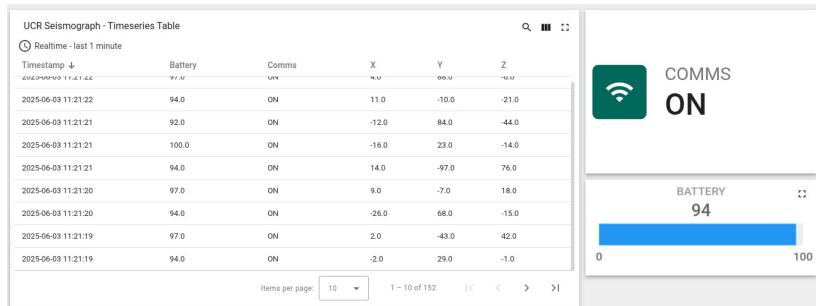


Figura 37: Aceleraciones X, Y, Z y nivel de batería (ON)
Elaboración propia



Figura 38: Sismógrafo y nivel de batería (ON)
Elaboración propia

3.3.7. IoT con batería OFF

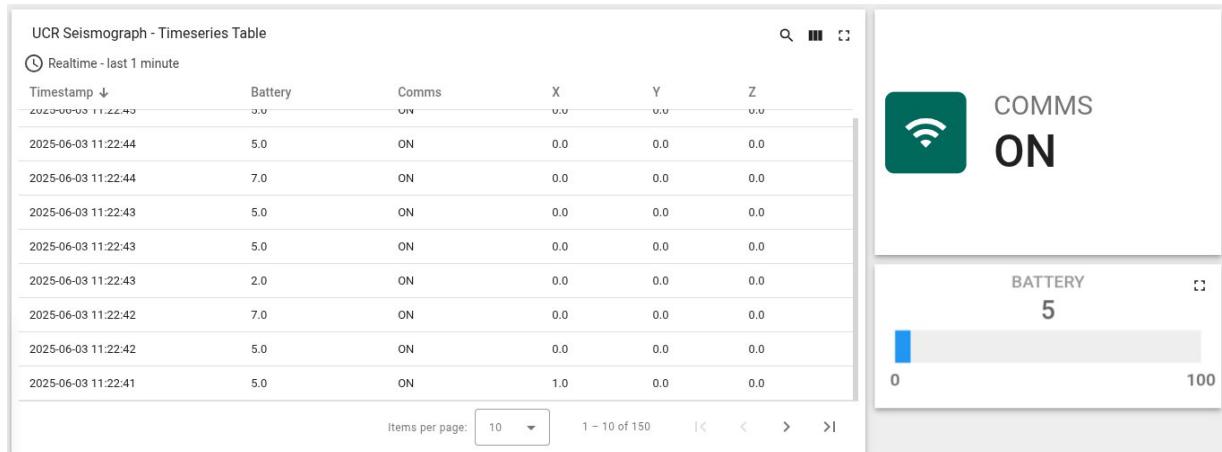


Figura 39: Aceleraciones X, Y, Z y nivel de batería (OFF)
Elaboración propia



Figura 40: Sismógrafo y nivel de batería (OFF)
Elaboración propia

3.3.8. Datos en la terminal

A continuación se muestra una captura de pantalla en donde se observan los datos capturados desplegados en la terminal:

```
[2025-06-03 11:23:50] X=10, Y=27, Z=-1, V=94, Comms=ON
[2025-06-03 11:23:50] X=23, Y=-96, Z=5, V=94, Comms=ON
[2025-06-03 11:23:51] X=-9, Y=55, Z=-5, V=94, Comms=ON
[2025-06-03 11:23:51] X=4, Y=-85, Z=12, V=92, Comms=ON
[2025-06-03 11:23:51] X=-8, Y=34, Z=-5, V=94, Comms=ON
[2025-06-03 11:23:52] X=5, Y=-31, Z=0, V=94, Comms=ON
[2025-06-03 11:23:52] X=5, Y=-13, Z=1, V=97, Comms=ON
[2025-06-03 11:23:53] X=9, Y=-17, Z=-3, V=92, Comms=ON
[2025-06-03 11:23:53] X=-7, Y=8, Z=-2, V=94, Comms=ON
[2025-06-03 11:23:53] X=4, Y=0, Z=0, V=94, Comms=ON
[2025-06-03 11:23:54] X=0, Y=23, Z=-4, V=97, Comms=ON
[2025-06-03 11:23:54] X=5, Y=-19, Z=0, V=94, Comms=ON
[2025-06-03 11:23:55] X=-19, Y=33, Z=-1, V=92, Comms=ON
[2025-06-03 11:23:55] X=7, Y=-24, Z=8, V=94, Comms=ON
[2025-06-03 11:23:55] X=-9, Y=81, Z=-52, V=94, Comms=ON
[2025-06-03 11:23:56] X=26, Y=26, Z=11, V=94, Comms=ON
[2025-06-03 11:23:56] X=34, Y=-144, Z=48, V=94, Comms=ON
[2025-06-03 11:23:57] X=-26, Y=75, Z=-6, V=94, Comms=ON
[2025-06-03 11:23:57] X=11, Y=-41, Z=7, V=97, Comms=ON
[2025-06-03 11:23:57] X=-18, Y=53, Z=-5, V=94, Comms=ON
[2025-06-03 11:23:58] X=13, Y=-19, Z=13, V=92, Comms=ON
[2025-06-03 11:23:58] X=0, Y=-7, Z=2, V=94, Comms=ON
[2025-06-03 11:23:59] X=-19, Y=80, Z=-40, V=94, Comms=ON
[2025-06-03 11:23:59] X=25, Y=-156, Z=16, V=94, Comms=ON
[2025-06-03 11:24:00] X=-8, Y=23, Z=3, V=94, Comms=ON
[2025-06-03 11:24:00] X=-16, Y=81, Z=-21, V=97, Comms=ON
[2025-06-03 11:24:00] X=26, Y=-57, Z=14, V=94, Comms=ON
[2025-06-03 11:24:01] X=-20, Y=81, Z=-18, V=94, Comms=ON
[2025-06-03 11:24:01] X=16, Y=-12, Z=-13, V=92, Comms=ON
[2025-06-03 11:24:02] X=21, Y=-56, Z=9, V=94, Comms=ON
```

Figura 41: Datos en la terminal
Elaboración propia

4. Conclusiones y Recomendaciones

- El STM32F429 resultó ser una tarjeta de desarrollo muy útil debido a su poder de cálculo y cantidad de periféricos para esta aplicación.
- El uso de los protocolos UART y SPI permitió profundizar sobre el funcionamiento de estos.
- La interfaz UART sirvió como canal de depuración y para comunicar a la computadora y que esta comunique por MQTT a thingsboard.
- El bus SPI resultó ser rápido para obtener los datos del giroscopio.
- MQTT fue un protocolo que encajó de manera correcta con el propósito de este laboratorio.
- Gracias a la biblioteca LibOpenCM3 fue fácil configurar e interactuar con la pantalla.
- Se recomienda el uso de divisores de tensión para poder mapear los valores de tensión a tensiones más inofensivas.
- Es importante conocer las capacidades máximas de la tarjeta de desarrollo/microcontrolador para poder sacarle el máximo provecho. Además leer los diagramas de pines.

Bibliografía

1. Material de clase
2. Aguirre, N., Aranda, N. & Balich, N. (2019). “Seguridad en el envío de mensajes mediante protocolo MQTT en IoT”. INNOVA UNTREF. Revista Argentina de Ciencia y Tecnología, 1(4). <https://revistas.untref.edu.ar/index.php/innova/article/view/1000>
3. Arm Mbed. “ST-Discovery-F429ZI Platform”. Disponible en: <https://os.mbed.com/platforms/ST-Discovery-F429ZI/>. Consultado el 6 de junio de 2025.
4. STMicroelectronics. “STM32F429I-DISC1 Discovery Kit with STM32F429ZI MCU”. Disponible en: <https://mou.sr/4kN5evU>. Consultado el 6 de junio de 2025.

5. Apéndices

5.1. Repositorio Git

Repositorio GitHub: https://github.com/Rossetas/MCU_Lab4

5.2. Archivo .sh

```
#!/bin/sh
socat PTY,link=/tmp/ttyS0,raw,echo=0 PTY,link=/tmp/ttyACM0,raw,echo=0
```

5.3. Archivo get_Data.py

```
import serial
import csv
from datetime import datetime

# Configuración del puerto serial
PORT = '/dev/ttyACM0'
BAUDRATE = 115200
CSV_FILENAME = 'logs.csv'

def main():
    try:
        # Abrir puerto serial
        with serial.Serial(PORT, BAUDRATE, timeout=1) as ser, open(CSV_FILENAME, mode='w', newline='') as csv_file:
            writer = csv.writer(csv_file)
            # Escribir encabezados
            writer.writerow(['Timestamp', 'X', 'Y', 'Z', 'Battery', 'Comms'])

            print("Esperando datos del STM32... (Ctrl+C para detener)")

            while True:
                line = ser.readline().decode('utf-8').strip()

                if not line:
                    continue # ignorar líneas vacías

                try:
                    parts = line.split('\t')
                    if len(parts) != 5:
                        print(f'Línea no válida: {line}')
                        continue

                    x, y, z, battery, comms = parts
                    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                    writer.writerow([timestamp, x, y, z, battery, comms])
                    print(f"[{timestamp}] X={x}, Y={y}, Z={z}, V={battery}, Comms={comms}")

                except Exception as e:
                    print(f'Error procesando línea: {line} -> {e}')

    except KeyboardInterrupt:
        print("\nRegistro detenido por el usuario.")
    except serial.SerialException as e:
        print(f'Error al abrir el puerto serial: {e}')

if __name__ == '__main__':
    main()
```

5.4. Archivo IoT.py

```
import serial
import time
import json
import paho.mqtt.client as mqtt

# Configuración del puerto serial
SERIAL_PORT = '/dev/ttyACM0'
BAUD_RATE = 9600

# Configuración de ThingsBoard
THINGSBOARD_HOST = 'iot.eie.ucr.ac.cr'
ACCESS_TOKEN = 'hpogmawb2dy9gd61sjfq' # Token de dispositivo en ThingsBoard
MQTT_PORT = 1883

# Inicializar conexión MQTT
client = mqtt.Client()
client.username_pw_set(ACCESS_TOKEN)
client.connect(THINGSBOARD_HOST, MQTT_PORT, 60)
client.loop_start()

# Función para parsear los datos del STM32
def parse_data(line):
    try:
        parts = line.strip().split('\t')
        if len(parts) != 5:
            return None
        return {
            "X_axis": float(parts[0]),
            "Y_axis": float(parts[1]),
            "Z_axis": float(parts[2]),
            "battery_level": float(parts[3]),
            "communication": 1 if parts[4].strip() == "ON" else 0
        }
    except:
        return None

# Leer del puerto serial y enviar a ThingsBoard
with serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1) as ser:
    time.sleep(2) # Tiempo para que el puerto se estabilice
    print("Leyendo datos del puerto serial y enviando a ThingsBoard...")
    while True:
        if ser.in_waiting > 0:
```

```
line = ser.readline().decode('utf-8').strip()
print("Datos recibidos:", line)
data = parse_data(line)
if data:
    telemetry = json.dumps(data)
    client.publish('v1/devices/me/telemetry', telemetry)
    print("Enviado a ThingsBoard:", telemetry)
time.sleep(0.5)
```

5.5. Archivo thingsboard.py

```
import serial
import csv
import time
from datetime import datetime
import paho.mqtt.client as mqtt

# Configuración del puerto serial
PORT = '/dev/ttyACM0'
BAUDRATE = 115200
CSV_FILENAME = 'logs.csv'

# Configuración de MQTT
MQTT_BROKER = "iot.eie.ucr.ac.cr"
ACCESS_TOKEN = "hpogmawb2dy9gd61sjfq"
MQTT_PORT = 1883
TOPIC = "v1/devices/me/telemetry"

def main():
    # Inicializar cliente MQTT
    client = mqtt.Client()
    client.username_pw_set(ACCESS_TOKEN)
    client.connect(MQTT_BROKER, MQTT_PORT, 60)
    client.loop_start()

    try:
        with serial.Serial(PORT, BAUDRATE, timeout=1) as ser, open(CSV_FILENAME, mode='w', encoding='utf-8') as csv_file:
            writer = csv.writer(csv_file)
            writer.writerow(['Timestamp', 'X', 'Y', 'Z', 'Battery', 'Comms'])

            print("Esperando datos del STM32...")

            while True:
                line = ser.readline().decode('utf-8').strip()

                if not line:
                    continue

                try:
                    parts = line.split('\t')
                    if len(parts) != 5:
                        print(f'Línea no válida: {line}')
                        continue

                    x, y, z, battery, comms = parts
                    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                    client.publish(TOPIC, f'{x},{y},{z},{battery},{comms}', 0)
                    writer.writerow([timestamp, x, y, z, battery, comms])
                except Exception as e:
                    print(f'Error al procesar la línea: {line}, {e}')

    except KeyboardInterrupt:
        client.loop_stop()
        client.disconnect()
        print("Desconectado del broker MQTT")
```

```
writer.writerow([timestamp, x, y, z, battery, comms])
print(f"[{timestamp}] X={x}, Y={y}, Z={z}, V={battery}, Comms={comms}")

# Enviar datos a ThingsBoard
payload = {
    "X": float(x),
    "Y": float(y),
    "Z": float(z),
    "Battery": float(battery),
    "Comms": comms
}
client.publish(TOPIC, str(payload))

except Exception as e:
    print(f"Error procesando línea: {line} -> {e}")
except KeyboardInterrupt:
    print("\nRegistro detenido por el usuario.")
except serial.SerialException as e:
    print(f"Error al abrir el puerto serial: {e}")
finally:
    client.loop_stop()
    client.disconnect()

if __name__ == '__main__':
    main()
```

5.5.1. Archivo .csv

	A	B	C	D	E	F	G
22	2025-06-06 22:48:56	-177	-8	4	2	ON	
23	2025-06-06 22:48:56	31	-88	-257	2	ON	
24	2025-06-06 22:48:57	12	92	269	2	ON	
25	2025-06-06 22:48:57	-58	-153	-224	1	ON	
26	2025-06-06 22:48:57	17	-5	83	2	ON	
27	2025-06-06 22:48:58	17	-131	-227	1	ON	
28	2025-06-06 22:48:58	-73	123	129	1	ON	
29	2025-06-06 22:48:59	35	-37	-37	1	ON	
30	2025-06-06 22:48:59	22	-29	15	2	ON	
31	2025-06-06 22:48:59	36	38	-52	1	ON	
32	2025-06-06 22:49:00	27	-9	31	1	ON	
33	2025-06-06 22:49:00	26	87	16	1	ON	
34	2025-06-06 22:49:01	11	-71	-54	1	ON	
35	2025-06-06 22:49:01	64	12	1	1	ON	
36	2025-06-06 22:49:01	103	6	133	1	ON	
37	2025-06-06 22:49:02	-77	-89	-83	1	ON	
38	2025-06-06 22:49:02	146	33	179	1	ON	
39	2025-06-06 22:49:03	-168	-141	-199	1	ON	
40	2025-06-06 22:49:03	136	-25	125	2	ON	
41	2025-06-06 22:49:03	-72	-3	-121	1	ON	
42	2025-06-06 22:49:04	123	-42	-55	1	ON	
43	2025-06-06 22:49:04	-188	124	-110	1	ON	
44	2025-06-06 22:49:05	226	-3	-71	1	ON	
45	2025-06-06 22:49:05	-158	19	-81	2	ON	
46	2025-06-06 22:49:05	92	-150	95	1	ON	
47	2025-06-06 22:49:06	28	62	-62	1	ON	
48	2025-06-06 22:49:06	-66	-297	186	1	ON	
49	2025-06-06 22:49:07	206	74	-152	2	ON	
50	2025-06-06 22:49:07	-51	-236	89	1	ON	
51	2025-06-06 22:49:07	3	155	-194	2	ON	
52	2025-06-06 22:49:08	68	50	-48	2	ON	
53	2025-06-06 22:49:08	-57	-257	93	1	ON	
54	2025-06-06 22:49:09	159	519	-328	1	ON	
55	2025-06-06 22:49:09	-130	-337	130	0	ON	
56	2025-06-06 22:49:09	-100	17	59	1	ON	
57	2025-06-06 22:49:10	97	390	-227	1	ON	
58	2025-06-06 22:49:10	-105	-248	125	1	ON	
59	2025-06-06 22:49:11	41	160	-167	0	ON	
60	2025-06-06 22:49:11	75	-127	142	1	ON	
..	2025-06-06 22:49:11	205	120	201	1	ON	

Figura 42: Archivo .csv con la información
Elaboración propia