UNIVERSIDAD DE COSTA RICA

ESCUELA DE INGENIERÍA ELÉCTRICA

IE0624
Laboratorio de Microcontroladores

Laboratorio IV

STM32: GPIO, ADC, timer, comunicaciones e Iot

Estudiantes:
Kendall Saborio Picado - B87103
Alexander Rojas Brenes - B86869

Profesor:
Ing. Marco Villalta Fallas

Fecha de entrega: 29/05/2024

Índice

1. INTRODUCCIÓN

Este trabajo aborda el desarrollo de un sistema de monitoreo de pendientes, diseñado para detectar peligros potenciales en una etapa temprana y proporcionar alertas y predicciones tempranas. El objetivo principal de este sistema es permitir a las autoridades pertinentes tomar medidas adecuadas antes de que ocurra un incidente. El sistema se construyó utilizando una placa STM32F429 Discovery kit y la biblioteca libopencm3. Para lograr el monitoreo de pendientes, se leen los ejes del giroscopio (X, Y, Z) y la temperatura. Se implementó un mecanismo de alerta mediante un LED parpadeante que se activa cuando se detecta una deformación angular mayor a 5 grados en cualquier eje. Además, se envía una notificación de advertencia al dashboard de ThingsBoard, una plataforma de IoT de código abierto basada en Java. El sistema también monitorea el nivel de la batería, cuyo rango operativo es de [0,9]V. Si el nivel de la batería se acerca al límite mínimo de operación del microcontrolador (7V), se enciende un LED de alarma parpadeante y se envía una notificación de batería baja al dashboard de ThingsBoard. Para ello, se utiliza una batería de 9V y un circuito que condiciona el nivel de voltaje dentro del rango de operación del microcontrolador. En la pantalla LCD del sistema se despliegan el nivel de batería, la temperatura, los valores de los ejes X, Y, Z y el estado de la comunicación serial/USB. Se utiliza un botón de la placa para habilitar o deshabilitar las comunicaciones por USART/USB. Además, se desarrolló un script en Python que permite leer y escribir al puerto serial/USB, enviando la información del giroscopio, temperatura y nivel de batería al dashboard de ThingsBoard. Para consultar el trabajo realizado, puede consultarse el siguiente repositorio de trabajo, en la rama llamada “Lab4”: https://github.com/SaboEIE/IE-0624_IS_2024_B87103.git. El código principal en C `lab4.c`, los archivos necesarios extraídos de la librería, el archivo en python `iot.py` y el Makefile puede consultarse en la ruta `Laboratorio_04/src`.

2. NOTA TEÓRICA

Antes de iniciar con el uso y manipulación del lenguaje de programación C para la creación de los algoritmos objetivo del presente laboratorio, es necesario tener a mano una serie de conceptos importantes, los cuales se explican a continuación:

2.1. Microcontrolador STM32F429

Tal como se detalla en la hoja del fabricante [1], el STM32F429 es un microcontrolador de 32 bits desarrollado por STMicroelectronics, perteneciente a la familia STM32, que se basa en el núcleo ARM Cortex-M4 que opera a una frecuencia de hasta 180 MHz.. Este microcontrolador es conocido por su alto rendimiento, amplia gama de periféricos y altas capacidades que lo hacen ideal para aplicaciones complejas en una variedad de campos, como la electrónica de consumo, la automatización industrial y las telecomunicaciones.

Incluye instrucciones específicas para el procesamiento digital de señales (DSP), que son particularmente útiles para aplicaciones que requieren cálculos matemáticos intensivos, una memoria flash con capacidad de 2 MB y una SRAM de 256Kb. Asimismo, cuenta con hasta 168 GPIO programables, 17 timers (6 timers generales de 16 bits y 4 timers generales de 32 bits). Posee un controlador TFT-LCD que permite la conexión directa a pantallas TFT, facilitando el desarrollo de interfaces gráficas. Soporte para interfaces de cámaras digitales, útil en aplicaciones de visión artificial. Incluye varios interfaces como USB OTG (On-The-Go), Ethernet, CAN (Controller Area Network), y varios puertos UART, SPI, I^2C, entre otros. Este microcontrolador dispone de 3 ADCs independientes de 12 bits, puede manejar hasta 24 canales, permitiendo la adquisición de señales de múltiples fuentes analógicas.

En la Figura 1 puede observarse la distribución de pines del microcontrolador y sus respectivas funciones:

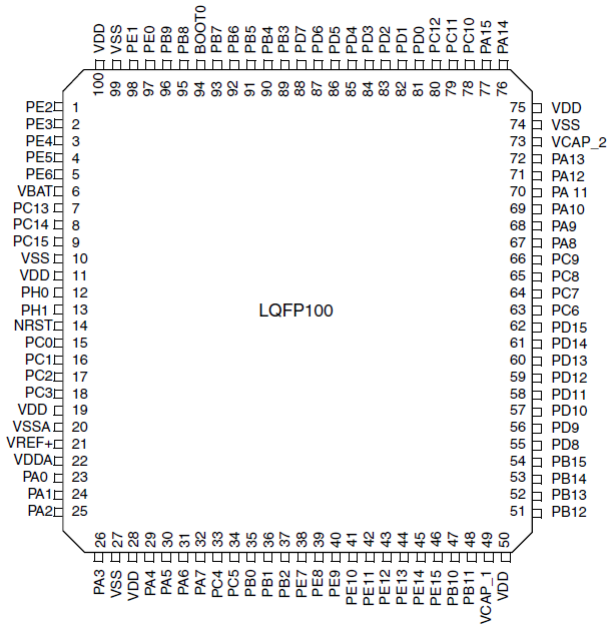


Figura 1: Diagrama de pines y sus respectivas funciones. (Fuente: Imagen tomada de [1])

Para más información general del dispositivo, observar el Anexo 5.1. Por otro lado, en la Figura 2 puede consultarse el diagrama de bloques del microcontrolador en el cual puede consultarse a alto nivel la conexión interna de este (no incluye el CPU):

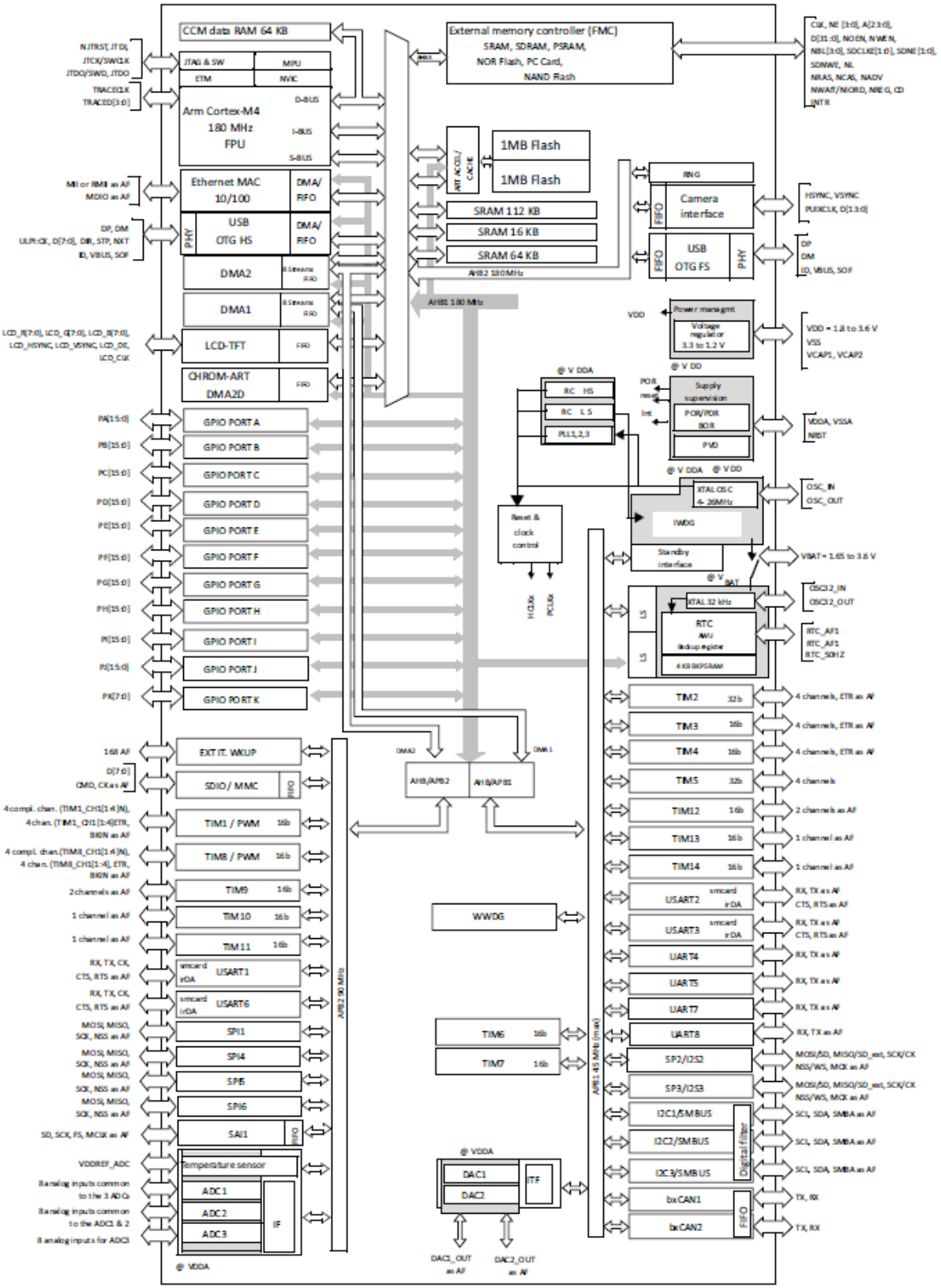


Figura 2: Diagrama de bloques del STM32F429. (Fuente: Imagen tomada de [1])

2.1.1. Registros de importancia

Según la hoja del fabricante [1], el mapa de memoria del STM32F427xx y STM32F429xx está dividido en varias regiones, cada una asignada a diferentes tipos de periféricos y funcionalidades. A continuación, se muestran los rangos de direcciones de registros de importancia:

Bus	Boundary address	Peripheral
	0x4008 0000- 0x4FFF FFFF	Reserved
AHB1	0x4004 0000 - 0x4007 FFFF	USB OTG HS
	0x4002 BC00- 0x4003 FFFF	Reserved
	0x4002 B000 - 0x4002 BBFF	DMA2D
	0x4002 9400 - 0x4002 AFFF	Reserved
	0x4002 9000 - 0x4002 93FF	ETHERNET MAC
	0x4002 8C00 - 0x4002 8FFF	
	0x4002 8800 - 0x4002 8BFF	
	0x4002 8400 - 0x4002 87FF	
	0x4002 8000 - 0x4002 83FF	
	0x4002 6800 - 0x4002 7FFF	Reserved
	0x4002 6400 - 0x4002 67FF	DMA2
	0x4002 6000 - 0x4002 63FF	DMA1
	0X4002 5000 - 0X4002 5FFF	Reserved
	0x4002 4000 - 0x4002 4FFF	BKPSRAM
	0x4002 3C00 - 0x4002 3FFF	Flash interface register
	0x4002 3800 - 0x4002 3BFF	RCC
	0X4002 3400 - 0X4002 37FF	Reserved
	0x4002 3000 - 0x4002 33FF	CRC
	0x4002 2C00 - 0x4002 2FFF	Reserved
	0x4002 2800 - 0x4002 2BFF	GPIOK
	0x4002 2400 - 0x4002 27FF	GPIOJ
	0x4002 2000 - 0x4002 23FF	GPIOI
	0x4002 1C00 - 0x4002 1FFF	GPIOH
	0x4002 1800 - 0x4002 1BFF	GPIOG
	0x4002 1400 - 0x4002 17FF	GPIOF
	0x4002 1000 - 0x4002 13FF	GPIOE
	0X4002 0C00 - 0x4002 0FFF	GPIOD
	0x4002 0800 - 0x4002 0BFF	GPIOC
	0x4002 0400 - 0x4002 07FF	GPIOB
	0x4002 0000 - 0x4002 03FF	GPIOA

Figura 3: Rangos direcciones de registros. (Fuente: Imagen tomada de [1])

Bus	Boundary address	Peripheral
	0x4001 6C00- 0x4001 FFFF	Reserved
APB2	0x4001 6800 - 0x4001 6BFF	LCD-TFT
	0x4001 5C00 - 0x4001 67FF	Reserved
	0x4001 5800 - 0x4001 5BFF	SAI1
	0x4001 5400 - 0x4001 57FF	SPI6
	0x4001 5000 - 0x4001 53FF	SPI5
	0x4001 5400 - 0x4001 57FF	SPI6
	0x4001 5000 - 0x4001 53FF	SPI5
	0x4001 4C00 - 0x4001 4FFF	Reserved
	0x4001 4800 - 0x4001 4BFF	TIM11
	0x4001 4400 - 0x4001 47FF	TIM10
	0x4001 4000 - 0x4001 43FF	TIM9
	0x4001 3C00 - 0x4001 3FFF	EXTI
	0x4001 3800 - 0x4001 3BFF	SYSCFG
	0x4001 3400 - 0x4001 37FF	SPI4
	0x4001 3000 - 0x4001 33FF	SPI1
	0x4001 2C00 - 0x4001 2FFF	SDIO
	0x4001 2400 - 0x4001 2BFF	Reserved
	0x4001 2000 - 0x4001 23FF	ADC1 - ADC2 - ADC3
	0x4001 1800 - 0x4001 1FFF	Reserved
	0x4001 1400 - 0x4001 17FF	USART6
	0x4001 1000 - 0x4001 13FF	USART1
	0x4001 0800 - 0x4001 0FFF	Reserved
	0x4001 0400 - 0x4001 07FF	TIM8
	0x4001 0000 - 0x4001 03FF	TIM1

Figura 4: Rangos direcciones de registros. (Fuente: Imagen tomada de [1])

Bus	Boundary address	Peripheral
	0x4000 8000- 0x4000 FFFF	Reserved
APB1	0x4000 7C00 - 0x4000 7FFF	UART8
	0x4000 7800 - 0x4000 7BFF	UART7
	0x4000 7400 - 0x4000 77FF	DAC
	0x4000 7000 - 0x4000 73FF	PWR
	0x4000 6C00 - 0x4000 6FFF	Reserved
	0x4000 6800 - 0x4000 6BFF	CAN2
	0x4000 6400 - 0x4000 67FF	CAN1
	0x4000 6000 - 0x4000 63FF	Reserved
	0x4000 5C00 - 0x4000 5FFF	I2C3
	0x4000 5800 - 0x4000 5BFF	I2C2
	0x4000 5400 - 0x4000 57FF	I2C1
	0x4000 5000 - 0x4000 53FF	UART5
	0x4000 4C00 - 0x4000 4FFF	UART4
	0x4000 4800 - 0x4000 4BFF	USART3
	0x4000 4400 - 0x4000 47FF	USART2
	0x4000 4000 - 0x4000 43FF	I2S3ext
	0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3
	0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2
	0x4000 3400 - 0x4000 37FF	I2S2ext
	0x4000 3000 - 0x4000 33FF	IWDG
	0x4000 2C00 - 0x4000 2FFF	WWDG
	0x4000 2800 - 0x4000 2BFF	RTC & BKP Registers
	0x4000 2400 - 0x4000 27FF	Reserved
	0x4000 2000 - 0x4000 23FF	TIM14
	0x4000 1C00 - 0x4000 1FFF	TIM13
	0x4000 1800 - 0x4000 1BFF	TIM12
	0x4000 1400 - 0x4000 17FF	TIM7
	0x4000 1000 - 0x4000 13FF	TIM6
	0x4000 0C00 - 0x4000 0FFF	TIM5
	0x4000 0800 - 0x4000 0BFF	TIM4
	0x4000 0400 - 0x4000 07FF	TIM3
	0x4000 0000 - 0x4000 03FF	TIM2

Figura 5: Rangos direcciones de registros. (Fuente: Imagen tomada de [1])

Es importante mencionar que el caso de este microcontrolador, se puede utilizar ciertas librerías como LibOpenCM3 que proporcionan una capa de abstracción más alta sobre el hardware, lo que significa que facilita manipular los registros del microcontrolador.

2.1.2. Características eléctricas

A continuación se muestra las especificaciones eléctricas del STM32F429:

Symbol	Ratings	Min	Max	Unit
V_{DD}–V_{SS}	External main supply voltage (including V_{DDA}, V_{DD} and VBAT)(1)	– 0.3	4.0	V
V_{IN}	Input voltage on FT pins(2)	V_{SS} – 0.3	V_{DD}+4.0	
	Input voltage on TTa pins	V_{SS} – 0.3	4.0	
	Input voltage on any other pin	V_{SS} – 0.3	4.0	
	Input voltage on BOOT0 pin	V_{SS}	9.0	
$\lvert\Delta V_{DDx}\rvert$	Variations between different V_{DD} power pins	-	50	mV
$\lvert V_{SSX} - V_{SS}\rvert$	Variations between all the different ground pins including V_{REF-}	-	50	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (human body model)	see *Section 6.3.15: Absolute maximum ratings (electrical sensitivity)*		

Figura 6: Características de tensión. (Fuente: Imagen tomada de [1])

Symbol	Ratings	Max.	Unit
ΣI_{VDD}	Total current into sum of all V_{DD_x} power lines (source)(1)	270	mA
ΣI_{VSS}	Total current out of sum of all V_{SS_x} ground lines (sink)(1)	– 270	
I_{VDD}	Maximum current into each V_{DD_x} power line (source)(1)	100	
I_{VSS}	Maximum current out of each V_{SS_x} ground line (sink)(1)	– 100	
I_{IO}	Output current sunk by any I/O and control pin	25	
	Output current sourced by any I/Os and control pin	– 25	
ΣI_{IO}	Total output current sunk by sum of all I/O and control pins (2)	120	
	Total output current sourced by sum of all I/Os and control pins(2)	– 120	
$I_{INJ(PIN)}$ (3)	Injected current on FT pins (4)	– 5/+0	
	Injected current on NRST and BOOT0 pins (4)		
	Injected current on TTa pins(5)	±5	
$\Sigma I_{INJ(PIN)}$ (5)	Total injected current (sum of all I/O and control pins)(6)	±25	

Figura 7: Características de corriente. (Fuente: Imagen tomada de [1])

2.1.3. Sensor L3GD20

El sensor MEMS L3GD20 es un giroscopio digital de 3 ejes fabricado por STMicroelectronics, diseñado para medir la velocidad angular en aplicaciones de movimiento y orientación. Este sensor

puede medir la velocidad angular alrededor de tres ejes ortogonales (X, Y, Z), con un rango de medición seleccionable entre ±250 dps, ±500 dps, y ±2000 dps. Posee una alta resolución de 16 bits para cada eje y una sensibilidad ajustable según el rango de medición seleccionado, lo que permite obtener datos precisos en diferentes aplicaciones. El L3GD20 se comunica con los microcontroladores a través de interfaces SPI (Serial Peripheral Interface) y I2C (Inter-Integrated Circuit), con una tasa de salida de datos (ODR) seleccionable de 95 Hz a 760 Hz. [2]

El L3GD20 también incluye funcionalidades avanzadas, como la integración de filtros paso bajo y paso alto programables, detección de interrupciones programables para eventos específicos (como caída libre y movimiento), y una función de auto-prueba para verificar su correcto funcionamiento. Funciona con un voltaje de operación de 2.4V a 3.6V y en un rango de temperatura de -40°C a +85°C, haciéndolo adecuado para diversas aplicaciones ambientales [2].

El sensor puede integrarse con el microcontrolador STM32F429 utilizando sus interfaces SPI o I2C. Esto implica conectar los pines SPI/I2C del L3GD20 a los pines apropiados del STM32F429 y asegurar una correcta alimentación del sensor. Luego, se configura el STM32F429 para comunicarse con el L3GD20 utilizando las bibliotecas HAL (Hardware Abstraction Layer) o LL (Low Layer) proporcionadas por STMicroelectronics [2]. A continuación algunos de los registros esenciales para su configuración y funcionamiento:

- **CTRL_REG1** (0x20): Controla el encendido del sensor y la configuración básica de la tasa de salida de datos y los filtros.
  - Bits 7-6: Selección de la tasa de salida de datos (ODR).
  - Bits 5-4: Selección del ancho de banda del filtro de paso bajo.
  - Bit 3: Control de encendido del giroscopio.
  - Bits 2-0: Habilitación de los ejes X, Y y Z.
- **CTRL_REG2** (0x21): Configuración del filtro de paso alto.
  - Bits 5-4: Modo de filtro de paso alto.
  - Bits 3-0: Frecuencia de corte del filtro de paso alto.
- **CTRL_REG4** (0x23): Configuración del rango de escala y del auto-prueba.
  - Bits 5-4: Selección del rango de escala (±250 dps, ±500 dps, ±2000 dps).
  - Bit 3: Control del modo de auto-prueba.
  - Otros bits para configuraciones adicionales.
- **OUT_TEMP** (0x26): Registro de salida de la temperatura, proporciona datos de temperatura.
- **OUT_X_L** (0x28) y **OUT_X_H** (0x29) : Registros de datos de salida del eje X (bajo y alto).
- **OUT_Y_L** (0x2A) y **OUT_Y_H** (0x2B) : Registros de datos de salida del eje Y (bajo y alto).
- **OUT_Z_L** (0x2C) y **OUT_Z_H** (0x2D) : Registros de datos de salida del eje Z (bajo y alto).

2.1.4. Pantalla y gráficos

La pantalla TFT-LCD integrada en los kits de desarrollo típicos del STM32F429, suele tener un tamaño de 2.4 a 2.8 pulgadas con una resolución de 240x320 píxeles (QVGA). La pantalla utiliza una interfaz RGB paralela de 16/18/24 bits para la transmisión de datos, lo que permite una rápida actualización de la imagen. Esta pantalla utiliza el controlador gráfico ILI9341 y un controlador táctil XPT2046 para la detección de toques. A nivel de comunicación, esta pantalla ofrece interfaces I2C y SPI, y en el contexto de la placa STM32F429 Discovery Kit, se comunica mediante la interfaz SPI5. Algunos otras características indicadas en la hoja del fabricante [1] son:

- 2 capas de pantalla con FIFO dedicado (64x32 bits)
- Tabla de búsqueda de colores (CLUT) de hasta 256 colores (256x24 bits) por capa
- Hasta 8 formatos de color de entrada seleccionables por capa
- Mezcla flexible entre dos capas usando valor alfa (por píxel o constante)
- Parámetros programables flexibles para cada capa
- Hasta 4 eventos de interrupción programables

2.2. Librerías de importancia

A continuación se detallan las librerías más importantes que fueron utilizadas en el laboratorio:

2.2.1. LibOpenCM3

LibOpenCM3 es una biblioteca diseñada para ser utilizada con microcontroladores ARM Cortex-M de diferentes fabricantes, como STMicroelectronics, NXP, Texas Instruments, y otros. Ofrece una API coherente y fácil de usar para manejar los periféricos comunes de estos microcontroladores, lo que simplifica el desarrollo de aplicaciones embebidas. LibOpenCM3 soporta una amplia gama de microcontroladores de varios fabricantes, incluyendo STM32 (de STMicroelectronics), LPC (de NXP), EFM32 (de Silicon Labs), y otros. Compatible con diversas herramientas de compilación y entornos de desarrollo, como GCC (GNU Compiler Collection), y puede integrarse con IDEs populares como Eclipse, Keil, y otros [3].

2.3. IOT: Internet of things

El Internet de las cosas (IoT, por sus siglas en inglés) se refiere a una red abierta y completa de objetos inteligentes que tienen la capacidad de autoorganizarse, compartir información, datos y recursos, reaccionando y actuando ante situaciones y cambios en el entorno. Es un concepto que ha evolucionado desde la idea de que la primera versión de Internet trataba sobre datos creados por personas, mientras que la siguiente versión se trata de datos creados por cosas [4].

2.4. Diseño del circuito de alimentación sistema de monitoreo de pendiente

Según se indica en el enunciado, el presente laboratorio tiene como objetivo construir un sistema encargado de monitorear los cambios en la pendiente de un terreno. Al existir la necesidad de que este circuito se encuentre en un lugar específico, es necesario que se autónomo respecto a la energía que consume. Por lo anterior, es necesario que el microcontrolador sea conectado a una batería de 9 V y que la tensión restante también sea reportada para evitar que el sistema se quede sin energía. Sin embargo, existe un problema y es que la tensión máxima permitida por el microcontrolador según su hoja del fabricante [1] es de 5 V, por lo que será necesario aplicar una división de tensión utilizando resistores.

Un divisor de voltaje es un simple circuito de resistores en serie que proporciona una fracción específica del voltaje de entrada como voltaje de salida. La relación entre el voltaje de entrada y salida se define por los valores de dos resistores, tal como se explica en [5]. En la Figura 8 puede observarse la disposición necesaria de los resistores:

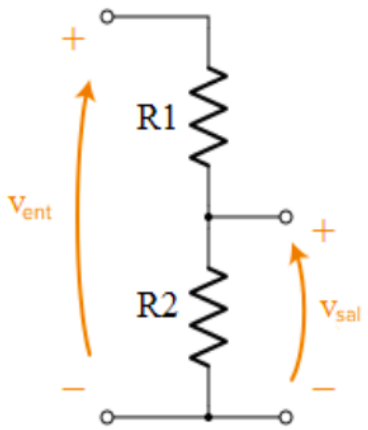


Figura 8: Circuito de división de tensión.(Tomada de [5])

Al pasar el diagrama anterior a expresiones matemáticas, de obtiene lo siguiente:

$$V_{sal} = V_{ent} \cdot \frac{R_2}{R_1 + R_2} \tag{1}$$

Para el presente diseño, se seleccionó $R_2 = 6{,}8\ k\Omega$, además se sabe que la tensión de entrada es de 9 V y la tensión de salida debe ser de 5 V, por lo que sólo resta despejar el valor R_1:

$$V_{sal} = V_{ent} \cdot \frac{R_2}{R_1 + R_2}$$

$$V_{sal} \cdot (R_1 + R_2) = V_{ent} \cdot R_2$$

$$V_{sal} \cdot R_1 + V_{sal} \cdot R_2 = V_{ent} \cdot R_2$$

$$V_{sal} \cdot R_1 = R_2 \cdot (V_{ent} - V_{sal})$$

$$R_1 = R_2 \cdot \frac{(V_{ent} - V_{sal})}{V_{sal}} \tag{2}$$

Sustituyendo los valores conocidos:

$$R_1 = (6{,}8 \times 10^3) \cdot \frac{(9-5)}{5} = 5{,}44 \times 10^3 = 5{,}44\ k\Omega \tag{3}$$

Se selecciona un valor estándar, por lo que $R_1 = 5{,}6\ k\Omega$. Lo anterior permitirá evitar sobrecargas de tensión en el microcontrolador y mantener su funcionamiento en condiciones controladas.

2.5. Lista de componentes y precios

En la Tabla 1 pueden consultarse los componentes utilizados y sus precios, tomando como referencia los precios del sitio web de la tienda de componentes MicroJPM (https://www.microjpm.com/) y la página oficial de ST (https://www.st.com/content/st_com/en.html):

Tabla 1: Lista de componentes.

Componente	Cantidad	Precio
STM34F429I-DISC1	1	$29.30
Resistor de kΩ	1	$0,07
Resistor de kΩ	1	$0,07
Batería de 9 V Ω	1	$5.64
Total	-	$35.08

Según la Tabla 1, para realizar este proyecto son necesario ₡18 073,50, según el valor del dólar al momento de escribir el presente informe.

3. DESARROLLO Y ANÁLISIS DE RESULTADOS

A continuación se muestran los métodos de desarrollo utilizados para resolver cada uno de los problemas planteados, así como la funciones utilizadas, el método de operación a partir de diagramas de flujo y los resultados obtenidos:

3.1. Desarrollo de la solución

A continuación podrá consultarse una descripción de alto nivel de cada una de las funciones implementadas para conseguir los objetivos del presente laboratorio:

3.1.1. Función `gpio_setup()`

Esta función es la encargada de configurar los pines GPIO necesarios. Se configura GPIOA0 como entrada y GPIOG13 y GPIOG14 como salidas. Además habilita los relojes para los puertos GPIOA y GPIOG.

3.1.2. Función `adc_setup()`

Esta función es la encargada de configurar el ADC (Convertidor Analógico-Digital). Se configura el ADC para utilizar el canal GPIO3 como entrada analógica. Se apaga el ADC, se desactiva el modo de escaneo, se establece el tiempo de muestreo en todos los canales y luego se enciende el ADC.

3.1.3. Función `spi_setup()`

Esta función se encarga de configurar el bus de SPI. Inicia habilitando los relojes para el bus SPI5 y los puertos GPIOC y GPIOF. Configura el pin GPIOC1 como salida para el control del chip de esclavo, los pines GPIOF7, GPIOF8 y GPIOF9 para sus funciones alternas correspondientes (probablemente SCK, MISO, MOSI de SPI), los parámetros de SPI5, como modo maestro, velocidad de baudios, polaridad del reloj, fase del reloj, modo full duplex, modo unidireccional, control de esclavo mediante software, orden de transmisión de bits y configuraciones adicionales. Finalmente, inicializa el giroscopio escribiendo en sus registros de control.

3.1.4. Función `usart_setup()`

Esta función configura la USART (Universal Synchronous/Asynchronous Receiver/Transmitter). Específicamente configura la USART1 para la comunicación serie asíncrona. Establece la velocidad de baudios en 115200, el número de bits de datos en 8, el número de bits de parada en 1, sin paridad, sin control de flujo, y en modo de transmisión solamente. Finalmente, habilita la USART1.

3.1.5. Función `init_system()`

Esta función inicializa todos los periféricos y configuraciones necesarios para el sistema. Contiene el llamado a las diversas funciones de setup creadas.

3.1.6. Función `write_reg()`

Esta función envía una secuencia de datos SPI para escribir un valor en un registro específico del dispositivo. Recibe como parámetros el registro al que se va a escribir el valor y el valor que se va a escribir en el registro.

3.1.7. Función `read_reg()`

Esta función envía una secuencia de datos SPI para leer un registro específico del dispositivo. Recibe como parámetro la dirección para leer el registro. y retorna el valor leído del registro.

3.1.8. Función `read_xyz_temp()`

Esta función lee los valores de los ejes X, Y, Z y la temperatura del giroscopio mediante la comunicación SPI con el dispositivo. Retorna una estructura que contiene los valores de los ejes X, Y, Z y la temperatura leídos del giroscopio.

3.1.9. Función `read_adc_naiive()`

Esta función realiza una lectura del valor analógico del ADC mediante una secuencia de operaciones básicas. Recibe como parámetro el número de canal del ADC del que se leerá el valor y retorna el valor digital leído del ADC.

3.1.10. Función `led_control()`

Esta función controla el estado un LED en función del voltaje de la batería. Si el voltaje de la batería es inferior a 7.5V, el LED se encenderá intermitentemente (cambiando su estado). En caso contrario, el LED se apagará. Tambien controla el estado un LED en función de las lecturas de los ejes. Si alguno de los ejes tiene una deformación mayor a 5 grados, el led se encenderá intermitentemente (cambiando su estado). La función recibe como parámetros el voltaje de la batería y un boleano que indica si hay deformación o no.

3.1.11. Función `display_data()`

Esta función muestra los datos proporcionados en la pantalla LCD. La función recibe como parámetros una estructura que contiene los valores de los ejes X, Y, Z y la temperatura, el voltaje de la batería y un indicador booleano que indica si se está en modo de transmisión.

3.1.12. Función `main()`

A continuación se muestra un diagrama de flujo de la función principal utilizada para este laboratorio:

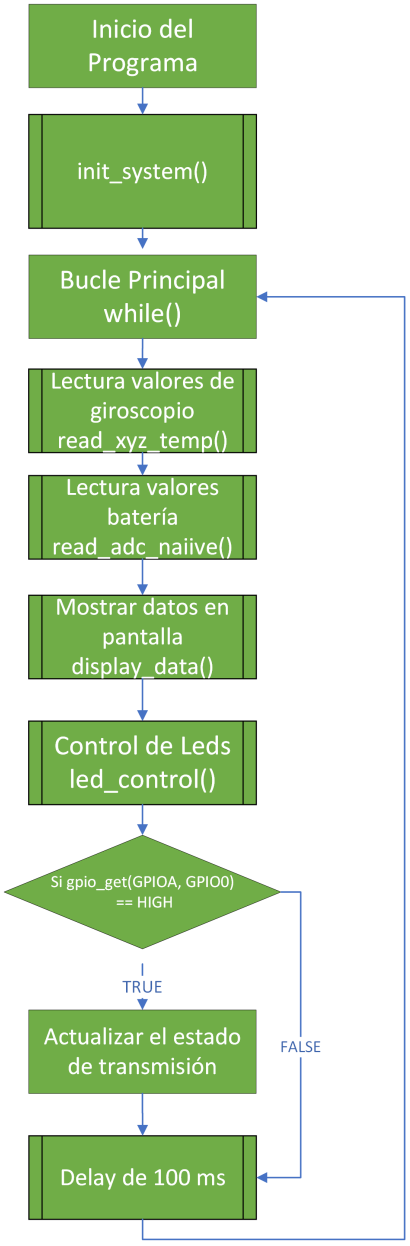


Figura 9: Diagrama de flujo que describe el funcionamiento del sistema.

3.1.13. *Script* de Python `iot.py`

Este *script* en Python establece una comunicación serial el STM32F429 y utiliza MQTT para enviar los datos leídos a un *broker*. Primero, configura la comunicación serial y el cliente MQTT, definiendo las funciones de conexión y los parámetros necesarios. Luego, verifica continuamente la conexión al broker MQTT y, una vez establecida, entra en un bucle donde lee datos del dispositivo, procesa estos datos (incluyendo detección de batería baja y deformación), los convierte a formato JSON, y los publica en el tema MQTT especificado. Este proceso se ejecuta indefinidamente, asegurando una transmisión continua de los datos a la plataforma Thingsboard.

3.1.14. Construcción del circuito utilizado

Para iniciar con la construcción de este circuito, es necesario tomar en cuenta el métodos de protección de tensión descrito en la sección 2.4. Dado lo anterior, se toma como tensión de entrada la tensión entre la parte superior del resistor R_2. Como según el enunciado es necesario hacer una medición de tensión de la batería, el pin 3 del puerto A se conecta a la parte superior del resistor R_2 y el pin GND se conecta a la tierra del circuito. Por otro lado, para notificar una tensión baja (menor a 7 V), se utilizó el pin 14 del puerto G como salida, el cual corresponde a uno de los LED integrados, este parpadea cuando ocurre el caso mencionado. Asimismo, para notificar una deformación en el terreo, se utilizó el pin 13 de puerto G como salida, el cual corresponde a uno de los LED integrados, este también parpadea cuando ocurre el caso mencionado. Por otro lado, se utilizó el pin 0 del puerto A, el cual se configuró como entrega, ya que esta asociado al pulsador azul integrado, este permite seleccionar si la transmisión está o no habilitada. Finalmente, se utilizó la pantalla LCD del microcontrolador para desplegar los datos de tensión disponible en la batería, las velocidades angulares de cada eje, la temperatura y el estado de la transmisión de datos hacia Thingsboard. El circuito resultante tras implementar todas as consideraciones anteriores puede consultarse en la Figura 10:

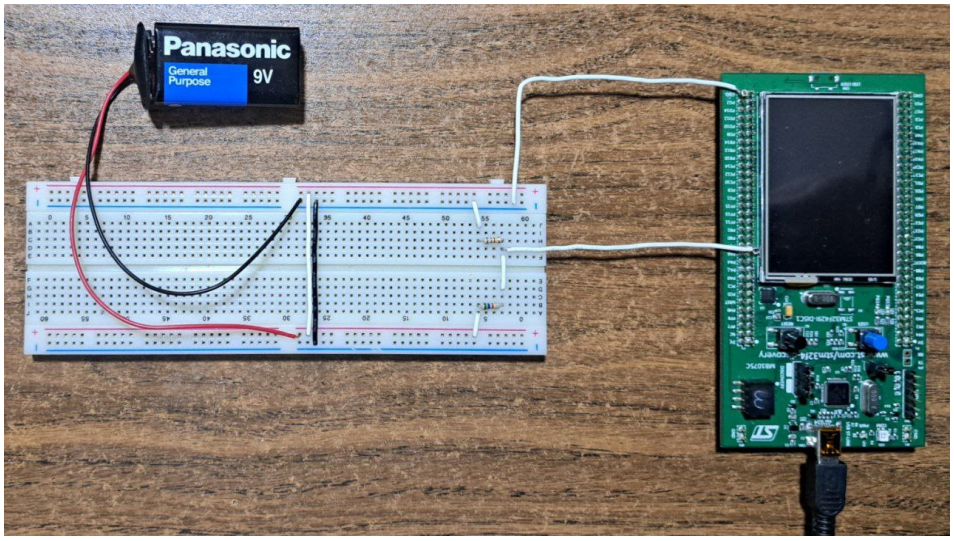


Figura 10: Circuito final tras los cálculos de magnitudes y la implementación de las consideraciones necesarias.

3.2. Análisis de los resultados

A continuación se mostrarán los resultados para cada una de las funcionalidades solicitadas por el enunciado:

3.2.1. Tensión de entrada adecuada

En la Figura 11 puede observarse la medida de tensión en R_2:

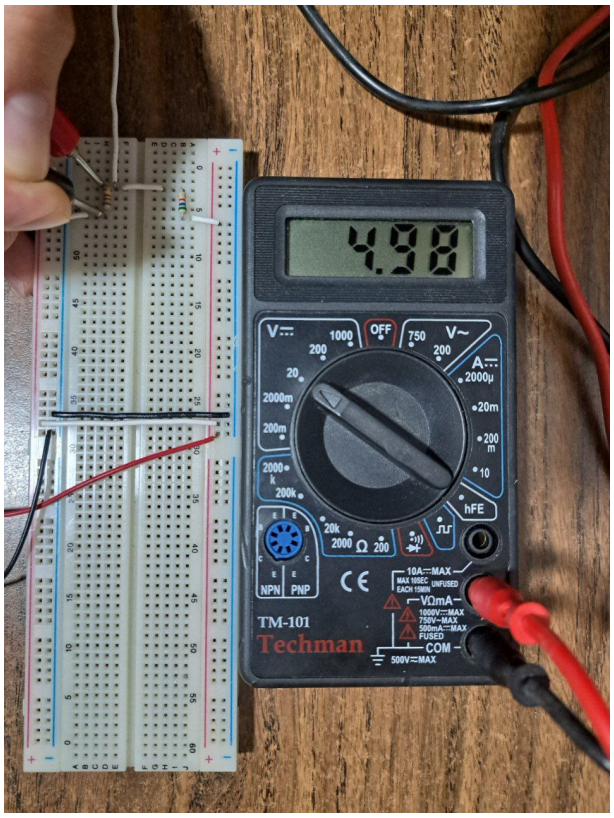


Figura 11: Tensión de entrada adecuada, la cual permitirá evaluar la tensión restante en la batería.

Como puede verse en la Figura 11, la tensión de entrada es un valor cercano a los 5 V, tal como fue calculado en la sección 2.4. Lo anterior permite que al microcontrolador entre una tensión adecuada y que se pueda hacer la evaluación de la tensión restante en la batería.

3.2.2. Despliegue de información en el microcontrolador

En la Figura 12 puede observarse la información mostrada en el microcontrolador una vez que el programa fue cargado en este y la batería fue conectada:

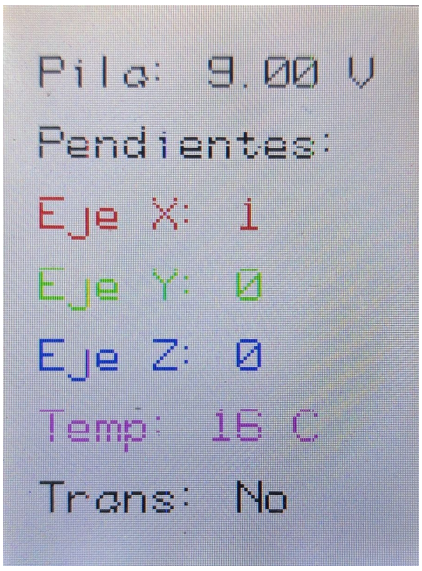


Figura 12: Información desplegada en la pantalla al iniciar el sistema.

Como puede verse en la Figura 12, en la pantalla efectivamente puede verse el valor de tensión de la batería (“Pila”), el valor de cada uno de los ejes (“Pendientes”), el valor de temperatura (“Temp”) y el estado de la transmisión (“Trans”). El valor de la temperatura no alcanza el valor correcto y esto se debe a que el sensor de temperatura del L3GD20 está destinado a compensar la variación del chip y no a medir la temperatura ambiental, además no se especifica un punto de referencia claro para el valor del offset en la hoja del fabricante, tal como se detalla en [6]. A pesar de esto, lo anterior confirma que la programación realizada para extraer datos del acelerómetro fue exitosa y que el circuito para regular la entrada de tensión funciona correctamente.

En la Figura 13, puede observarse el resultado en pantalla cuando el botón de usuario es presionado:

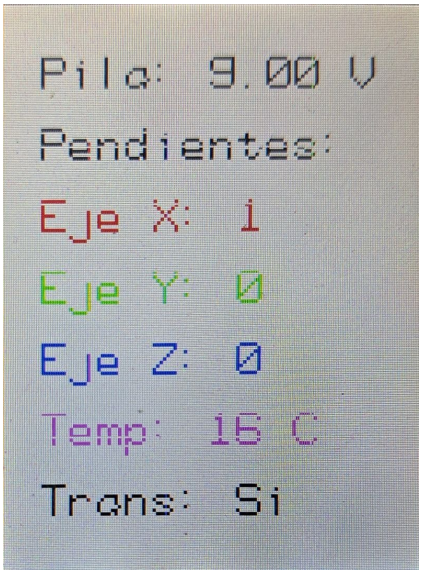


Figura 13: Información desplegada en la pantalla cuando el botón de usuario fue presionado para habilitar la transmisión.

Como puede verse en la Figura 13, la opción “Trans” pasó de estar en “No” a estar en “Si”, confirmando que el botón para habilitar la transmisión funciona de manera correcta. Los resultados obtenidos en la plataforma de IoT puede consultarse en la siguiente sección.

Por último, para demostrar el funcionamiento de los LED que funciona como banderas para notificar que cambios en la pendiente del terreno en el que se encuentra el sisema y que la batería está baja, pueden consultarse en el siguiente video: `https://youtu.be/xIwpAetdkUw`. En este pueden observarse tres estados:

1. Al inicio del video puede observarse como ninguno de los LED parpadean, ya que la batería está conectada y cuenta con un valor adecuado.

2. El LED verde parpadea cuando ocurre un cambio de la pendiente que es superior a 5 grados. Asimismo, se pueden notar los cambios en la velocidad angular de los ejes.

3. El LED rojo parpadea cuando se desconecta la batería, indicando un valor inferior a 7 V. También puede verse como la tensión baja en la pantalla y como cambia debido al ruido de lectura en el pin.

Estos tres estados confirman la notificación adecuada de incidentes utilizando la información ambiental, para cada uno de los casos solicitados en el enunciado. Al igual que con los resultados de transmisión, los casos anteriores podrán verse reflejados en la siguiente sección.

3.2.3. Transmisión de datos a Thingsboard

Para el despliegue de datos, se utilizó la plataforma Thingsboard. Con el sistema conectado, el programa cargado en el microcontrolador, la transmisión habilitada y el identificador del dispositivo configurado en la plataforma, se corre el `script` Python. Al correr el *script*, se evalúa la conexión y una vez que se logra, se imprime “Conexión exitosa.” en la terminal. Ya con la conexión establecida, se obtiene el siguiente resultado en la terminal, el cual puede ser observado en la Figura 14:


```
Conexión exitosa.
{'Eje X': '1', 'Eje Y': '0', 'Eje Z': '0', 'Temperatura': '14', 'Bateria': '9.00', 'Bateria Baja': 'No', 'Deformacion': 'No'}
{"Eje X": "1", "Eje Y": "0", "Eje Z": "0", "Temperatura": "14", "Bateria": "9.00", "Bateria Baja": "No", "Deformacion": "No"}
{'Eje X': '1', 'Eje Y': '0', 'Eje Z': '0', 'Temperatura': '14', 'Bateria': '9.00', 'Bateria Baja': 'No', 'Deformacion': 'No'}
{"Eje X": "1", "Eje Y": "0", "Eje Z": "0", "Temperatura": "14", "Bateria": "9.00", "Bateria Baja": "No", "Deformacion": "No"}
{'Eje X': '1', 'Eje Y': '0', 'Eje Z': '0', 'Temperatura': '13', 'Bateria': '9.00', 'Bateria Baja': 'No', 'Deformacion': 'No'}
{"Eje X": "1", "Eje Y": "0", "Eje Z": "0", "Temperatura": "13", "Bateria": "9.00", "Bateria Baja": "No", "Deformacion": "No"}
{'Eje X': '1', 'Eje Y': '0', 'Eje Z': '0', 'Temperatura': '13', 'Bateria': '9.00', 'Bateria Baja': 'No', 'Deformacion': 'No'}
{"Eje X": "1", "Eje Y": "0", "Eje Z": "0", "Temperatura": "13", "Bateria": "9.00", "Bateria Baja": "No", "Deformacion": "No"}
{'Eje X': '1', 'Eje Y': '0', 'Eje Z': '0', 'Temperatura': '13', 'Bateria': '9.00', 'Bateria Baja': 'No', 'Deformacion': 'No'}
{"Eje X": "1", "Eje Y": "0", "Eje Z": "0", "Temperatura": "13", "Bateria": "9.00", "Bateria Baja": "No", "Deformacion": "No"}
{'Eje X': '1', 'Eje Y': '0', 'Eje Z': '0', 'Temperatura': '13', 'Bateria': '9.00', 'Bateria Baja': 'No', 'Deformacion': 'No'}
{"Eje X": "1", "Eje Y": "0", "Eje Z": "0", "Temperatura": "13", "Bateria": "9.00", "Bateria Baja": "No", "Deformacion": "No"}
{'Eje X': '1', 'Eje Y': '0', 'Eje Z': '0', 'Temperatura': '13', 'Bateria': '9.00', 'Bateria Baja': 'No', 'Deformacion': 'No'}
{"Eje X": "1", "Eje Y": "0", "Eje Z": "0", "Temperatura": "13", "Bateria": "9.00", "Bateria Baja": "No", "Deformacion": "No"}
```


Figura 14: Transmisión de datos Thingsboard utilizando el `script` de Python.

Como puede verse en la Figura 14, los datos relevantes del sistema son transmitidos a la plataforma de IoT. Estos fueron utilizados para la construcción de un *dashboard* que permite visualizar de manera más accesible los datos y controlar las alertas de casos de interés. Dicho *dashboard* puede

consultarse en el siguiente link: Slope Monitoring Dashboard. Asimismo, este puede observarse en la Figura 15:

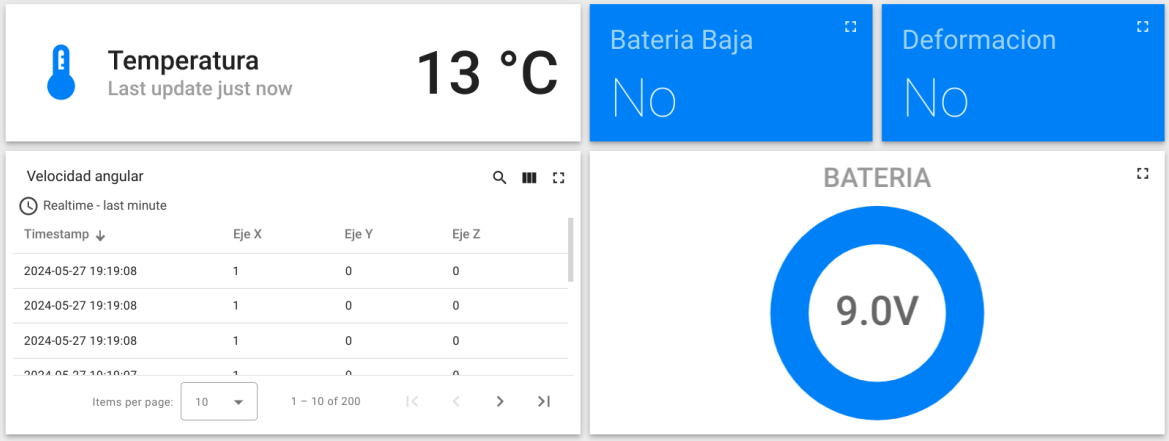


Figura 15: *Dashboard* construido en Thingsboard para monitorear el sistema de monitoreo de cambios en la pendiente de un terreno.

Como puede verse en la Figura 15, en este se muestran los datos de los tres ejes del acelerómetro, la temperatura, la información sobre la carga de la batería y las dos notificaciones que muestran si la batería tiene un carga menor a 7 V y si hubo una deformación en el terreno.

Finalmente, pueden observarse los tres casos de estado descritos en la sección anterior:

1. El *dashboard* tiene datos estáticos, ya que no hay cambios de pendiente y la batería se encuentra en su carga óptima. Lo anterior puede consultarse en la Figura 15.

2. Se notifica una deformación en el terreno de más de 5 grados. Asimismo, se pueden notar los cambios en la velocidad angular de los ejes. Lo anterior puede consultarse en la Figura 16:

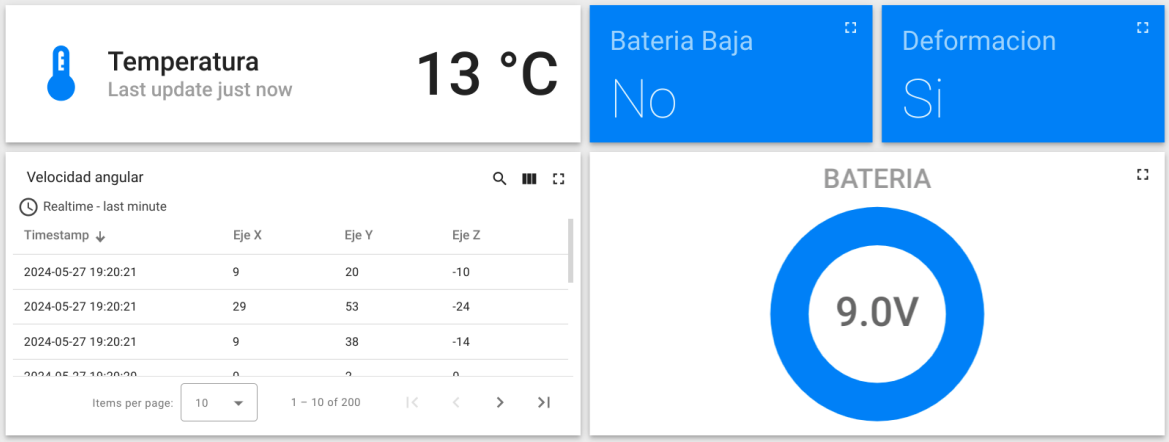


Figura 16: Comportamiento del *dashboard* tras un cambio en la pendiente del terreno en el que se encuentra el sistema.

3. Se notifica que la batería tiene una carga inferior a 7 V. Lo anterior también se puede notar en la métrica encargada de la carga de batería. Lo anterior puede consultarse en la Figura 17:

Figura 17: Comportamiento del *dashboard* tras una disminución de la carga de la batería por debajo del umbral de tensión requerido.

Estos tres estados confirman la notificación adecuada de incidentes utilizando la información extraída del microcontrolador y enviada a la plataforma de IoT, confirmando el cumplimiento de los objetivos del laboratorio, detallados en el enunciado.

4. CONCLUSIONES Y RECOMENDACIONES

A lo largo del desarrollo del informe se pusieron a prueba una serie de conceptos y prácticas importantes para el manejo básico del STM32F429, así como sus GPIO, interfaces de comunicación, dispositivos para mostrar información y formas de programación. A continuación algunas conclusiones y recomendaciones recolectadas a lo largo de trabajo realizado:

- La tensión de entrada al microcontrolador es cercana a los 5 V, garantizando una operación eficiente y permitiendo la correcta evaluación de la batería, confirmando la utilidad del circuito regulador de entrada.
- La pantalla del microcontrolador muestra correctamente la tensión de la batería, los ejes del acelerómetro, la temperatura y el estado de transmisión. La temperatura no es precisa debido a la limitación del sensor, pero el resto de los datos son fiables.
- Los LEDs indicaron correctamente los cambios de pendiente y niveles bajos de batería, proporcionando una respuesta visual inmediata y efectiva.
- La transmisión de datos a Thingsboard fue exitosa, y el *dashboard* interactivo permite un monitoreo claro y en tiempo real de los datos del sistema, cumpliendo con los objetivos del laboratorio.
- Se recomienda integrar un sensor de temperatura ambiental adecuado y añadir una funcionalidad de calibración automática para los sensores del microcontrolador. Esto aumentará la precisión y confiabilidad del sistema en el monitoreo y respuesta a cambios ambientales.

REFERENCIAS BIBLIOGRÁFICAS

[1] STMicroelectronics, *STM32F429xx ARM Cortex-M4 32b MCU+FPU, 225DMIPS, up to 2MB Flash/256+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 20 comm. interfaces, camera & LCD-TFT*. 2018. Disponible en: https://www.st.com/resource/en/datasheet/stm32f427vg.pdf.

[2] STMicroelectronics, *MEMS motion sensor: three-axis digital output gyroscope*. 2013. Disponible en: https://www.st.com/resource/en/datasheet/l3gd20.pdf.

[3] *Libopencm3: Open source ARM Cortex-M microcontroller library*. Disponible en: https://github.com/libopencm3/libopencm3/tree/master.

[4] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, no. 5, pp. 164–173, 2015.

[5] W. McAllister, "Divisor de voltaje." https://es.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-voltage-divider, s.f.

[6] B. Earl, "Adafruit Triple Axis Gyro Breakout." https://learn.adafruit.com/adafruit-triple-axis-gyro-breakout/arduino, 2013.

5. Anexos

5.1. Hoja del fabricante del STM32F429: información general

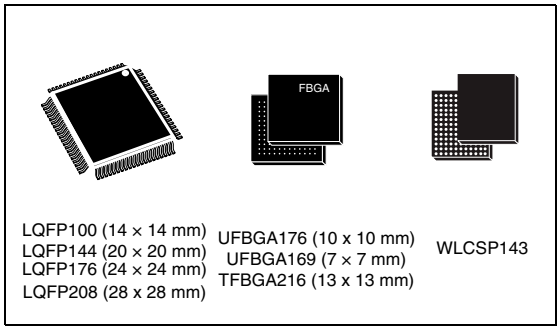
STM32F427xx STM32F429xx

32b Arm® Cortex®-M4 MCU+FPU, 225DMIPS, up to 2MB Flash/256+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 20 com. interfaces, camera & LCD-TFT

Datasheet - production data

Features

- Core: Arm® 32-bit Cortex®-M4 CPU with FPU, Adaptive real-time accelerator (ART Accelerator™) allowing 0-wait state execution from Flash memory, frequency up to 180 MHz, MPU, 225 DMIPS/1.25 DMIPS/MHz (Dhrystone 2.1), and DSP instructions
- Memories
  - Up to 2 MB of Flash memory organized into two banks allowing read-while-write
  - Up to 256+4 KB of SRAM including 64-KB of CCM (core coupled memory) data RAM
  - Flexible external memory controller with up to 32-bit data bus: SRAM, PSRAM, SDRAM/LPSDR SDRAM, Compact Flash/NOR/NAND memories
- LCD parallel interface, 8080/6800 modes
- LCD-TFT controller with fully programmable resolution (total width up to 4096 pixels, total height up to 2048 lines and pixel clock up to 83 MHz)
- Chrom-ART Accelerator™ for enhanced graphic content creation (DMA2D)
- Clock, reset and supply management
  - 1.7 V to 3.6 V application supply and I/Os
  - POR, PDR, PVD and BOR
  - 4-to-26 MHz crystal oscillator
  - Internal 16 MHz factory-trimmed RC (1% accuracy)
  - 32 kHz oscillator for RTC with calibration
  - Internal 32 kHz RC with calibration
- Low power
  - Sleep, Stop and Standby modes
  - V_{BAT} supply for RTC, 20×32 bit backup registers + optional 4 KB backup SRAM
- 3×12-bit, 2.4 MSPS ADC: up to 24 channels and 7.2 MSPS in triple interleaved mode
- 2×12-bit D/A converters
- General-purpose DMA: 16-stream DMA controller with FIFOs and burst support
- Up to 17 timers: up to twelve 16-bit and two 32-bit timers up to 180 MHz, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input



- Debug mode
  - SWD & JTAG interfaces
  - Cortex-M4 Trace Macrocell™
- Up to 168 I/O ports with interrupt capability
  - Up to 164 fast I/Os up to 90 MHz
  - Up to 166 5 V-tolerant I/Os
- Up to 21 communication interfaces
  - Up to 3 × I^2C interfaces (SMBus/PMBus)
  - Up to 4 USARTs/4 UARTs (11.25 Mbit/s, ISO7816 interface, LIN, IrDA, modem control)
  - Up to 6 SPIs (45 Mbits/s), 2 with muxed full-duplex I^2S for audio class accuracy via internal audio PLL or external clock
  - 1 x SAI (serial audio interface)
  - 2 × CAN (2.0B Active) and SDIO interface
- Advanced connectivity
  - USB 2.0 full-speed device/host/OTG controller with on-chip PHY
  - USB 2.0 high-speed/full-speed device/host/OTG controller with dedicated DMA, on-chip full-speed PHY and ULPI
  - 10/100 Ethernet MAC with dedicated DMA: supports IEEE 1588v2 hardware, MII/RMII
- 8- to 14-bit parallel camera interface up to 54 Mbytes/s
- True random number generator
- CRC calculation unit
- RTC: subsecond accuracy, hardware calendar
- 96-bit unique ID

2 Description

The STM32F427xx and STM32F429xx devices are based on the high-performance Arm® Cortex®-M4 32-bit RISC core operating at a frequency of up to 180 MHz. The Cortex-M4 core features a Floating point unit (FPU) single precision which supports all Arm® single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) which enhances application security.

The STM32F427xx and STM32F429xx devices incorporate high-speed embedded memories (Flash memory up to 2 Mbyte, up to 256 Kbytes of SRAM), up to 4 Kbytes of backup SRAM, and an extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses and a 32-bit multi-AHB bus matrix.

All devices offer three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers. They also feature standard and advanced communication interfaces.

- Up to three I^2Cs
- Six SPIs, two I^2Ss full duplex. To achieve audio class accuracy, the I^2S peripherals can be clocked via a dedicated internal audio PLL or via an external clock to allow synchronization.
- Four USARTs plus four UARTs
- An USB OTG full-speed and a USB OTG high-speed with full-speed capability (with the ULPI),
- Two CANs
- One SAI serial audio interface
- An SDIO/MMC interface
- Ethernet and camera interface
- LCD-TFT display controller
- Chrom-ART Accelerator™.

Advanced peripherals include an SDIO, a flexible memory control (FMC) interface, a camera interface for CMOS sensors. Refer to *Table 2: STM32F427xx and STM32F429xx features and peripheral counts* for the list of peripherals available on each part number.

The STM32F427xx and STM32F429xx devices operates in the –40 to +105 °C temperature range from a 1.7 to 3.6 V power supply.

The supply voltage can drop to 1.7 V with the use of an external power supply supervisor (refer to *Section 3.17.2: Internal reset OFF*). A comprehensive set of power-saving mode allows the design of low-power applications.

The STM32F427xx and STM32F429xx devices offer devices in 8 packages ranging from 100 pins to 216 pins. The set of included peripherals changes with the device chosen.

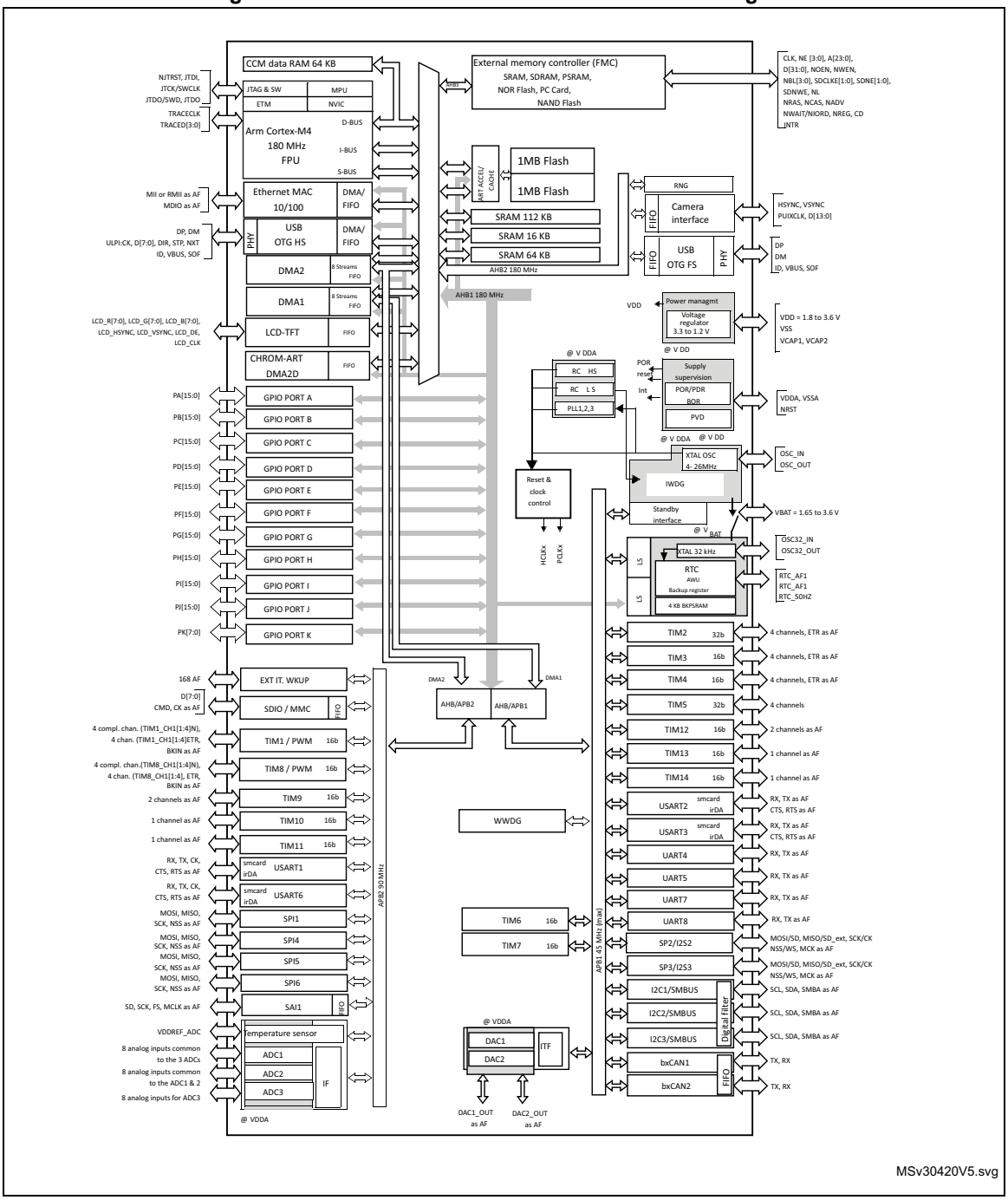
These features make the STM32F427xx and STM32F429xx microcontrollers suitable for a wide range of applications:

- Motor drive and application control
- Medical equipment
- Industrial applications: PLC, inverters, circuit breakers
- Printers, and scanners
- Alarm systems, video intercom, and HVAC
- Home audio appliances

Figure 4 shows the general block diagram of the device family.

arm

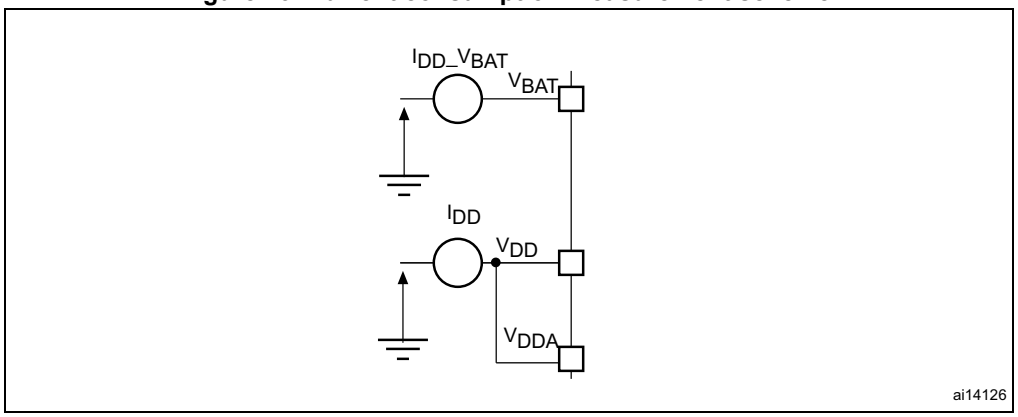
Figure 4. STM32F427xx and STM32F429xx block diagram



1. The timers connected to APB2 are clocked from TIMxCLK up to 180 MHz, while the timers connected to APB1 are clocked from TIMxCLK either up to 90 MHz or 180 MHz depending on TIMPRE bit configuration in the RCC_DCKCFGR register.
2. The LCD-TFT is available only on STM32F429xx devices.

6.1.7 Current consumption measurement

Figure 23. Current consumption measurement scheme



6.2 Absolute maximum ratings

Stresses above the absolute maximum ratings listed in *Table 14: Voltage characteristics*, *Table 15: Current characteristics*, and *Table 16: Thermal characteristics* may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Device mission profile (application conditions) is compliant with JEDEC JESD47 Qualification Standard, extended mission profiles are available on demand.

Table 14. Voltage characteristics

Symbol	Ratings	Min	Max	Unit
V_{DD}–V_{SS}	External main supply voltage (including V_{DDA}, V_{DD} and VBAT)(1)	− 0.3	4.0	V
V_{IN}	Input voltage on FT pins(2)	V_{SS} − 0.3	V_{DD}+4.0	
	Input voltage on TTa pins	V_{SS} − 0.3	4.0	
	Input voltage on any other pin	V_{SS} − 0.3	4.0	
	Input voltage on BOOT0 pin	V_{SS}	9.0	
$\lvert\Delta V_{DDx}\rvert$	Variations between different V_{DD} power pins	-	50	mV
$\lvert V_{SSX} - V_{SS}\rvert$	Variations between all the different ground pins including V_{REF-}	-	50	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (human body model)	see *Section 6.3.15: Absolute maximum ratings (electrical sensitivity)*		

1. All main power (V_{DD}, V_{DDA}) and ground (V_{SS}, V_{SSA}) pins must always be connected to the external power supply, in the permitted range.
2. V_{IN} maximum value must always be respected. Refer to *Table 15* for the values of the maximum allowed injected current.

Table 15. Current characteristics

Symbol	Ratings	Max.	Unit
ΣI_{VDD}	Total current into sum of all V_{DD_x} power lines (source)(1)	270	mA
ΣI_{VSS}	Total current out of sum of all V_{SS_x} ground lines (sink)(1)	− 270	
I_{VDD}	Maximum current into each V_{DD_x} power line (source)(1)	100	
I_{VSS}	Maximum current out of each V_{SS_x} ground line (sink)(1)	− 100	
I_{IO}	Output current sunk by any I/O and control pin	25	
	Output current sourced by any I/Os and control pin	− 25	
ΣI_{IO}	Total output current sunk by sum of all I/O and control pins (2)	120	
	Total output current sourced by sum of all I/Os and control pins(2)	− 120	
$I_{INJ(PIN)}$ (3)	Injected current on FT pins (4)	− 5/+0	
	Injected current on NRST and BOOT0 pins (4)		
	Injected current on TTa pins(5)	±5	
$\Sigma I_{INJ(PIN)}$(5)	Total injected current (sum of all I/O and control pins)(6)	±25	

1. All main power (V_{DD}, V_{DDA}) and ground (V_{SS}, V_{SSA}) pins must always be connected to the external power supply, in the permitted range.
2. This current consumption must be correctly distributed over all I/Os and control pins. The total output current must not be sunk/sourced between two consecutive power supply pins referring to high pin count LQFP packages.
3. Negative injection disturbs the analog performance of the device. See note in *Section 6.3.21: 12-bit ADC characteristics*.
4. Positive injection is not possible on these I/Os and does not occur for input voltages lower than the specified maximum value.
5. A positive injection is induced by $V_{IN}>V_{DDA}$ while a negative injection is induced by $V_{IN}<V_{SS}$. $I_{INJ(PIN)}$ must never be exceeded. Refer to *Table 14* for the values of the maximum allowed input voltage.
6. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values).

Table 16. Thermal characteristics

Symbol	Ratings	Value	Unit
T_{STG}	Storage temperature range	− 65 to +150	°C
T_J	Maximum junction temperature	125	°C