Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0624 – Laboratorio de Microcontroladores
II ciclo 2024

Laboratorio 04:
STM32: GPIO, ADC, comunicaciones, Iot

Rocha Morales Mario - B96561
Ruiz Sanchez Junior - B97026

06 de Octubre del 2024

Índice

1. Introducción

Este reporte describe el desarrollo del laboratorio 4 del curso de Laboratorio de Microcontroladores, el cual, consiste en implementar un sismógrafo, este debe entregar valores X, Y, Z correspondientes a los ejes de un giroscópio el cual sería el encargado de sensar las vibraciones, algunas de sus características es que cuenta con un switch que controla cuando el microcontrolador envía información por el puerto serial, este además hace parpadear una luz LED cuando se encuentra transmitiendo o recibiendo datos, otra de sus características es que realiza lectura de la tensión de la batería para poder avisar cuando esta se encuentre casi descargada, además, esta información la usa para encender un LED que alerta cuando la batería se ecuentra baja.

Para desarrollar este laboratorio se usó la placa STM32F429 Discovery kit la cual cuenta además con una pantalla, por lo que, los datos también se mostrarían en dicha pantalla.

Por último, se creó un script de Python el cual se encarga de recibir los datos recolectados mediante conexión por puerto serial y enviarlos por IoT a la plataforma ThingsBoard mostrando tanto los datos del giroscópio como el estado de la batería.

2. Nota Teórica

2.1. Características Generales de la STM32F429 Discovery Kit

La placa **STM32F429 Discovery kit** está basada en el microcontrolador **STM32F429ZI**, el cual cuenta con las siguientes especificaciones:

- **Microcontrolador**: STM32F429ZI, de arquitectura ARM Cortex-M4 de 32 bits.
- **Frecuencia de Operación**: Hasta 180 MHz, optimizada para aplicaciones de alta velocidad.
- **Memoria Flash**: 2 MB, adecuada para aplicaciones complejas que requieren mucho almacenamiento.
- **Memoria RAM**: 256 KB de RAM interna, permitiendo la ejecución de tareas demandantes en memoria.
- **Pantalla**: Incluye una pantalla TFT LCD de 2.4 pulgadas con interfaz de controlador ILI9341.
- **Interfaces de Comunicación**: Soporte de comunicaciones por USART, SPI, I2C, USB OTG (Host/Device), y CAN.
- **Acelerómetro y Giroscopio**: Sensores integrados para mediciones de aceleración y rotación.
- **Puertos de Expansión**: Headers de pines para extender las capacidades de la placa y conectar dispositivos externos.
- **Alimentación**: Funciona con alimentación a través de USB o fuentes externas (5V o 3.3V).
- **Depuración**: Compatible con el depurador ST-LINK/V2 integrado, útil para desarrollo y pruebas.

2.2. Periféricos

- **GPIO (General Purpose Input/Output)**: Pines de entrada/salida de propósito general, configurables para leer datos digitales o controlar periféricos externos.
- **ADC (Convertidor Análogo-Digital)**: Permite convertir señales analógicas a valores digitales, útil para la lectura de sensores analógicos.
- **DAC (Convertidor Digital-Análogo)**: Convierte señales digitales a análogas, ideal para generar señales de audio o control de voltaje.
- **USART (Universal Synchronous/Asynchronous Receiver/Transmitter)**: Comunicación serie, comúnmente usada para enviar y recibir datos entre el microcontrolador y otros dispositivos.
- **SPI (Serial Peripheral Interface)**: Interfaz de comunicación de alta velocidad para comunicación con sensores, pantallas, y módulos de memoria.
- **I2C (Inter-Integrated Circuit)**: Protocolo de comunicación serie que permite conectar varios dispositivos con una cantidad mínima de pines.
- **USB OTG (On-The-Go)**: Capacidad de actuar tanto como dispositivo USB como host, permitiendo conectar periféricos USB como memorias o teclados.
- **CAN (Controller Area Network)**: Comunicación para aplicaciones industriales y automotrices, útil en entornos que requieren tolerancia a fallos.
- **RTC (Real Time Clock)**: Reloj en tiempo real para realizar mediciones de tiempo y manejar funciones de temporización.
- **PWM (Pulse Width Modulation)**: Generación de señales PWM, usada para control de motores, regulación de brillo en LEDs, y otras aplicaciones que requieran señales moduladas.
- **Timers**: Temporizadores y contadores avanzados, útiles para medir intervalos de tiempo o controlar eventos periódicos.

2.3. Usos Comunes

La **STM32F429 Discovery kit** se emplea en una amplia variedad de aplicaciones, incluyendo:

- **Desarrollo de Interfaces de Usuario**: La pantalla LCD integrada permite diseñar interfaces visuales y realizar aplicaciones de visualización de datos.
- **Sistemas Embebidos de Control y Monitoreo**: Gracias a sus diversos periféricos y capacidad de procesamiento, es ideal para aplicaciones industriales y de automatización.
- **Proyectos de IoT (Internet of Things)**: Los múltiples interfaces de comunicación (USB, CAN, USART) y el soporte de conectividad permiten usarla en aplicaciones IoT para recolección y transmisión de datos.
- **Desarrollo de Prototipos para Automoción y Robótica**: Su capacidad de procesamiento, combinada con comunicación CAN y PWM, la hace adecuada para el control de motores y monitoreo de sensores en sistemas robóticos.

- **Aprendizaje y Enseñanza en Sistemas de Microcontroladores**: Con su soporte para periféricos complejos y facilidad de uso, es ampliamente utilizada en laboratorios académicos y prácticas de ingeniería.

3. Desarrollo y Análisis

3.1. Desarrollo del código del microcontrolador

Para este microcontrolador se implementaron funciones por separado. Se configuró el reloj, la comunicación usart, la salida de printf, delay, la configuración de spi y el giroscopio. Además una función para obtener who_am_i y una función para leer los ejes. En el codigo main, se hizo uso del las funciónes para configurar, de la función para obtener el regitro del giros copio, y en el while1 se hizo uso de la obtención de los datos y el delay.

Librerias usadas:

```
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/spi.h>
#include <libopencm3/stm32/usart.h>
#include <libopencm3/cm3/systick.h>
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
```

Configuración del giroscopio:

```
static void gyro_setup(void) {
    gpio_clear(GYRO_CS_PORT, GYRO_CS_PIN);

    // Configurar CTRL_REG1: Frecuencia de salida de 95 Hz, modo normal, ejes XYZ acti
    spi_xfer(SPI_BUS, 0x20); // CTRL_REG1
    spi_xfer(SPI_BUS, 0x0F); // Configuración: 95 Hz, modo normal, ejes XYZ activos

//    // Configurar CTRL_REG2: Sin filtro de paso alto
//    spi_xfer(SPI_BUS, 0x21); // CTRL_REG2
//    spi_xfer(SPI_BUS, 0x00); // Sin filtro
//
//    // Configurar CTRL_REG4: Rango de ±250 dps
//    spi_xfer(SPI_BUS, 0x23); // CTRL_REG4
//    spi_xfer(SPI_BUS, 0x00); // Configuración: ±250 dps
//
    gpio_set(GYRO_CS_PORT, GYRO_CS_PIN);
}
```

Lectura del giroscopio:

```
static void gyro_read_axes(int16_t *x, int16_t *y, int16_t *z) {
    uint8_t data[6];

    gpio_clear(GYRO_CS_PORT, GYRO_CS_PIN);
    spi_xfer(SPI_BUS, 0x28 | 0x80); // Lectura del registro OUT_X_L
    for (int i = 0; i < 6; i++) {
        data[i] = spi_xfer(SPI_BUS, 0x00);
    }
    gpio_set(GYRO_CS_PORT, GYRO_CS_PIN);

    *x = (int16_t)((data[1] << 8) | data[0]);
    *y = (int16_t)((data[3] << 8) | data[2]);
    *z = (int16_t)((data[5] << 8) | data[4]);
}
```

3.2. Desarrollo de la conexión IoT

Para desarrollar el código en python se usaron las librerías paho.mqtt.client, time, serial y json, la primera de ellas; paho.mqtt.client es la encargada de gestionar el protocolo MQTT usado para la comunicación con ThingsBoard, la librería time sirve para manejar el tiempo y pausas en la ejecuación el código, por lo que se usa para manejar la frecuencia a la que se actualizan los datos, en cuanto a la librería serial esta es usada para poder interpretar los datos recibidos por puerto serial.

En cuanto a la plataforma ThingsBoard, se creó un nuevo dispositivo llamado holaSismo el cual recibe la información para mostrarla posteriormente mediante un widget.

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

# Configuracion de la conexion serial
puerto = '/tmp/ttyS1'  # Cambia '/tmp/ttyS1' por tu puerto especifico
ser = serial.Serial(puerto, 9600, timeout=1)
time.sleep(2)

# Configuracion de ThingsBoard
THINGSBOARD_HOST = "iot.eie.ucr.ac.cr"
ACCESS_TOKEN = "dcnf5vs5u543r1a21fju"

# Configuracion de MQTT
client = mqtt.Client()
client.username_pw_set(ACCESS_TOKEN)

# Funcion de conexion MQTT
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Conexion a ThingsBoard exitosa")
    else:
        print("Error al conectar a ThingsBoard, codigo:", rc)

# Conectar el cliente
client.on_connect = on_connect
client.connect(THINGSBOARD_HOST, 1883, 60)
client.loop_start()

```

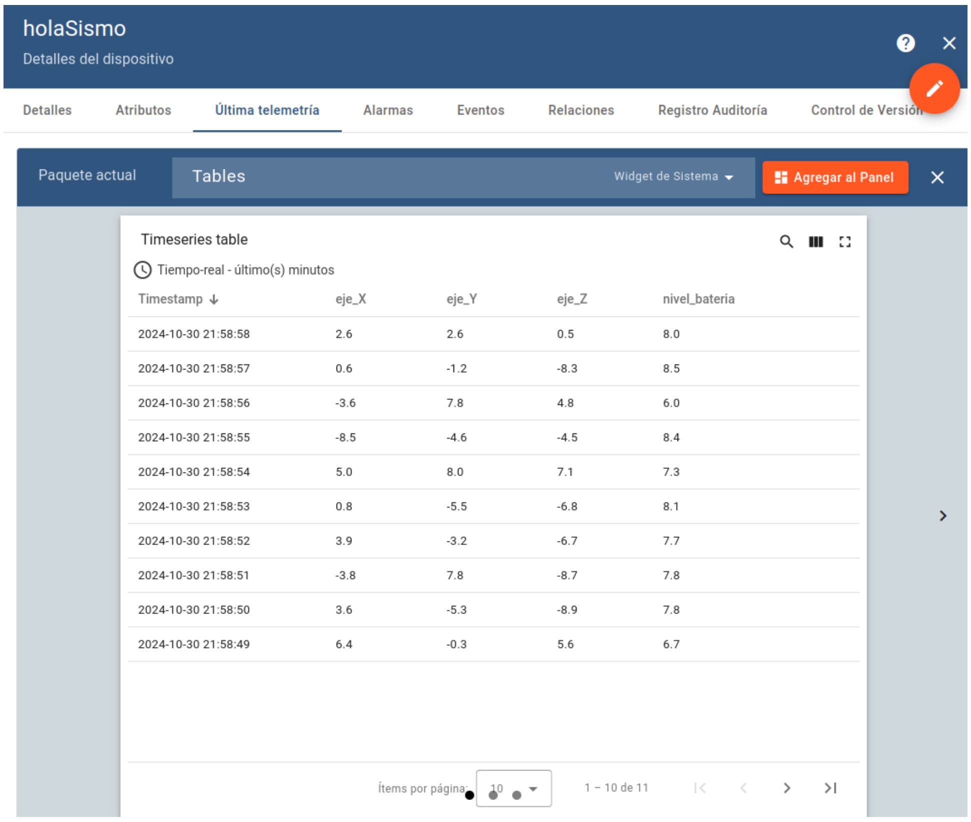
```python
def enviar_telemetria(x, y, z, voltage):
    payload = json.dumps({
        "eje_X": x,
        "eje_Y": y,
        "eje_Z": z,
        "nivel_bateria": voltage
    })
    client.publish("v1/devices/me/telemetry", payload)
    print("Datos enviados:", payload)

while True:
    if ser.in_waiting > 0:
        # Inicializar variables para almacenar datos
        x = y = z = battery = None

        # Leer datos del puerto serial
        for _ in range(5):  # Lee 5 lineas (4 valores y 1 separador)
            linea = ser.readline().decode('utf-8').strip()
            try:
                # Extraer valores segun etiqueta
                if linea.startswith("X:"):
                    x = float(linea.split(":")[1].strip())
                elif linea.startswith("Y:"):
                    y = float(linea.split(":")[1].strip())
                elif linea.startswith("Z:"):
                    z = float(linea.split(":")[1].strip())
                elif linea.startswith("Battery:"):
                    battery = float(linea.split(":")[1].strip())
            except ValueError:
                print("Error en el formato de datos:", linea)

        # Verificar que todos los datos se han leido correctamente
        if x is not None and y is not None and z is not None and
           battery is not None:
            enviar_telemetria(x, y, z, battery)
        else:
            print("Error: datos incompletos recibidos")

    time.sleep(1)
```


holaSismo
Detalles del dispositivo

Detalles | Atributos | Última telemetría | Alarmas | Eventos | Relaciones | Registro Auditoría | Control de Versión

Paquete actual | Tables | Widget de Sistema | Agregar al Panel

Timeseries table
Tiempo-real - último(s) minutos

Timestamp	eje_X	eje_Y	eje_Z	nivel_bateria
2024-10-30 21:58:58	2.6	2.6	0.5	8.0
2024-10-30 21:58:57	0.6	-1.2	-8.3	8.5
2024-10-30 21:58:56	-3.6	7.8	4.8	6.0
2024-10-30 21:58:55	-8.5	-4.6	-4.5	8.4
2024-10-30 21:58:54	5.0	8.0	7.1	7.3
2024-10-30 21:58:53	0.8	-5.5	-6.8	8.1
2024-10-30 21:58:52	3.9	-3.2	-6.7	7.7
2024-10-30 21:58:51	-3.8	7.8	-8.7	7.8
2024-10-30 21:58:50	3.6	-5.3	-8.9	7.8
2024-10-30 21:58:49	6.4	-0.3	5.6	6.7

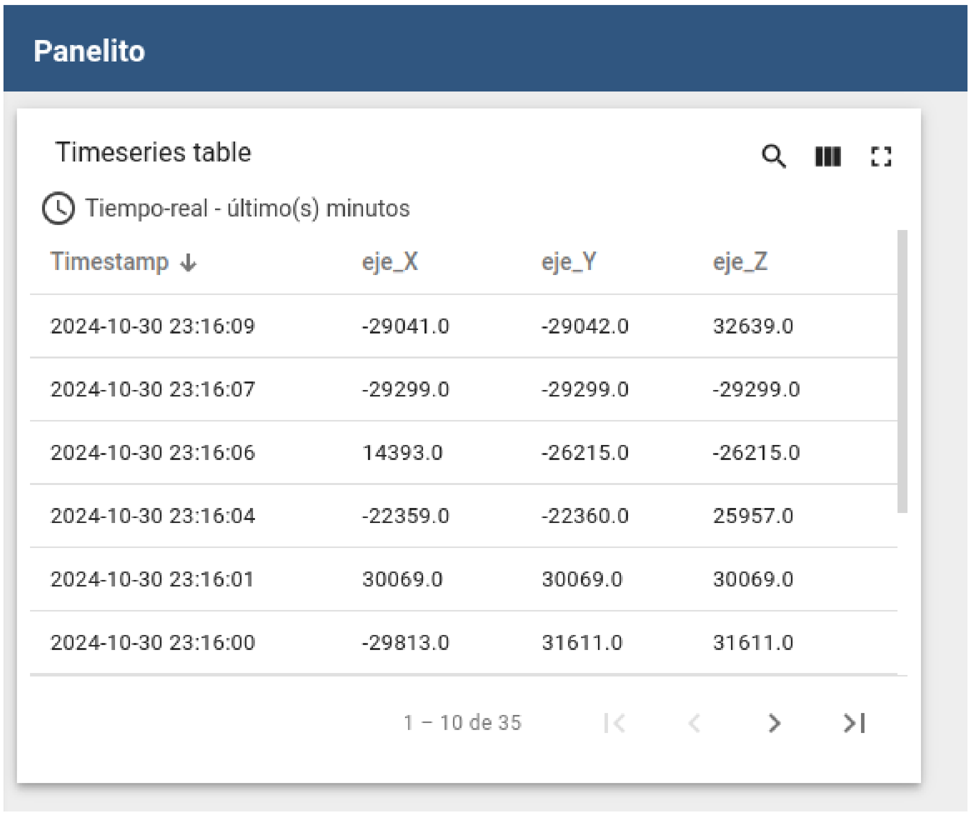
Ítems por página 10 | 1 – 10 de 11

Figura 1: Prueba del dispositivo agregado a ThingsBoard con valores aleatorios.

3.3. Análisis de Resultados

Tras realizar la conexión entre el microcontrolador y el código de python la transmisión de información por MQTT fue exitosa, sin embargo, por dificultades en la configuración del microcontrolador los datos no son correctos, viendonos en la obligación de remover el dato de estado de batería, sin embargo, se puede ver una transmisión exitosa de los datos del giroscópio.

Siendo este el eje del proyecto en cuestión, dado por qué si se realiza la conexión entre el microcontrolador y la plataforma de thingsboard.


Panelito

Timeseries table

Tiempo-real - último(s) minutos

Timestamp ↓	eje_X	eje_Y	eje_Z
2024-10-30 23:16:09	-29041.0	-29042.0	32639.0
2024-10-30 23:16:07	-29299.0	-29299.0	-29299.0
2024-10-30 23:16:06	14393.0	-26215.0	-26215.0
2024-10-30 23:16:04	-22359.0	-22360.0	25957.0
2024-10-30 23:16:01	30069.0	30069.0	30069.0
2024-10-30 23:16:00	-29813.0	31611.0	31611.0

1 – 10 de 35

Figura 2: Resultados obtenidos tras mostrar en panel la información del giroscopio.

4. Apéndices.

UM1662
User manual

Getting started with the STM32F429 Discovery kit

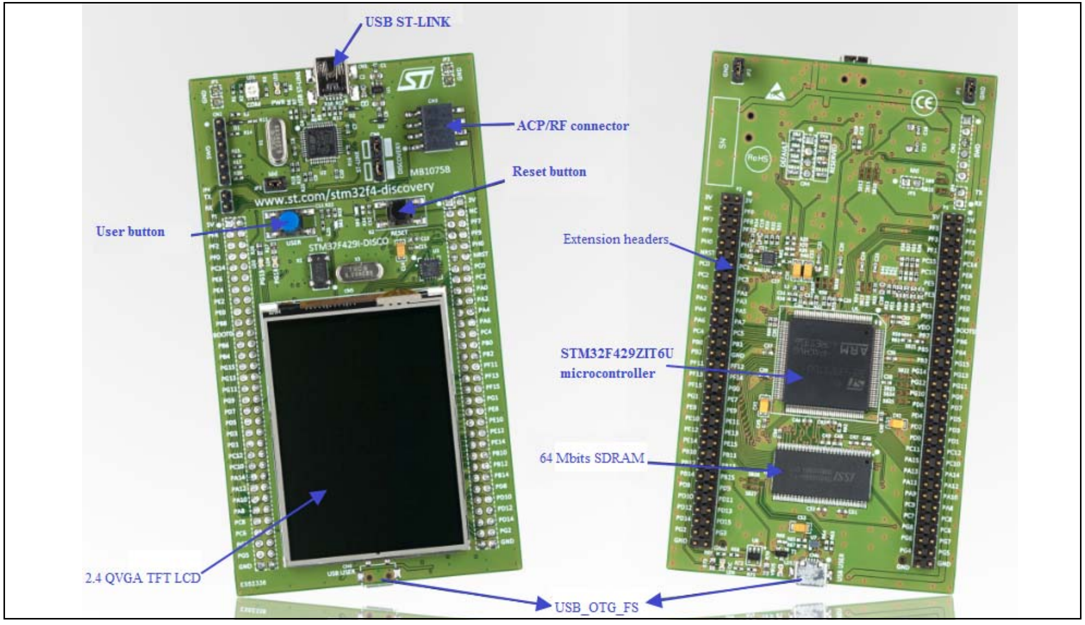
Introduction

This document describes the software, firmware and hardware environments and development recommendations required to build an application around the STM32F429 Discovery kit (32F429IDISCOVERY) with demonstration firmware (STSW-STM32138).

The STM32F429 Discovery kit is a low-cost and easy-to-use development kit to quickly evaluate and start applications with an STM32F4 32-bit ARM® Cortex™-M4 CPU with FPU high-performance microcontroller. Before installing and using the product, please accept the Evaluation Product License Agreement from *www.st.com/stm32f4-discovery*.

For more information on the STM32F429 Discovery kit visit *www.st.com/stm32f4-discovery*. To order the STM32F429 Discovery kit, use the STM32F429I-DISCO order code.

Figure 1. STM32F429 Discovery board: STM32F429I-DISCO



References

- STM32F429xx Datasheet
- STM32F40xxx, STM32F41xxx, STM32F42xxx, STM32F43xxx advanced ARM-based 32-bit MCUs reference manual (RM0090)
- Discovery kit for STM32F429/439 lines (UM1670)
- Getting started with STM32F429 Discovery software development tools
- Forum user question/ discussion.

Contents

List of figures

1 Hardware configuration and layout

1.1 Features

The STM32F429 Discovery offers the following features:

- STM32F429ZIT6 microcontroller featuring 2 MB of Flash memory, 256 KB of RAM in an LQFP144 package
- On-board ST-LINK/V2 with selection mode switch to use the kit as a standalone ST-LINK/V2 (with SWD connector for programming and debugging)
- Board power supply: through the USB bus or from an external 3 V or 5 V supply voltage
- L3GD20, ST MEMS motion sensor, 3-axis digital output gyroscope
- TFT LCD (Thin-film-transistor liquid-crystal display) 2.4", 262K colors RGB, 240 x 320 dots
- SDRAM 64 Mbits (1 Mbit x 16-bit x 4-bank) including an AUTO REFRESH MODE, and a power-saving
- Six LEDs:
  - LD1 (red/green) for USB communication
  - LD2 (red) for 3.3 V power-on
  - Two user LEDs:
    LD3 (green), LD4 (red)
  - Two USB OTG LEDs:
    LD5 (green) VBUS and LD6 (red) OC (over-current)
- Two pushbuttons (user and reset)
- USB OTG with micro-AB connector
- Extension header for LQFP144 I/Os for a quick connection to the prototyping board and an easy probing

1.2 Microcontroller

The STM32F429ZIT6U device is 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 STM32F429ZIT6U device incorporates high-speed embedded memories (2 Mbytes of Flash memory, 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.

1.3 System requirement

- Windows PC (XP, Vista, 7)
- USB type A to Mini-B USB cable
- ST-LINK/V2
- Supported IDE are EWARM (IAR Embedded Workbench®), MDK-ARM™ and Atollic TrueSTUDIO®

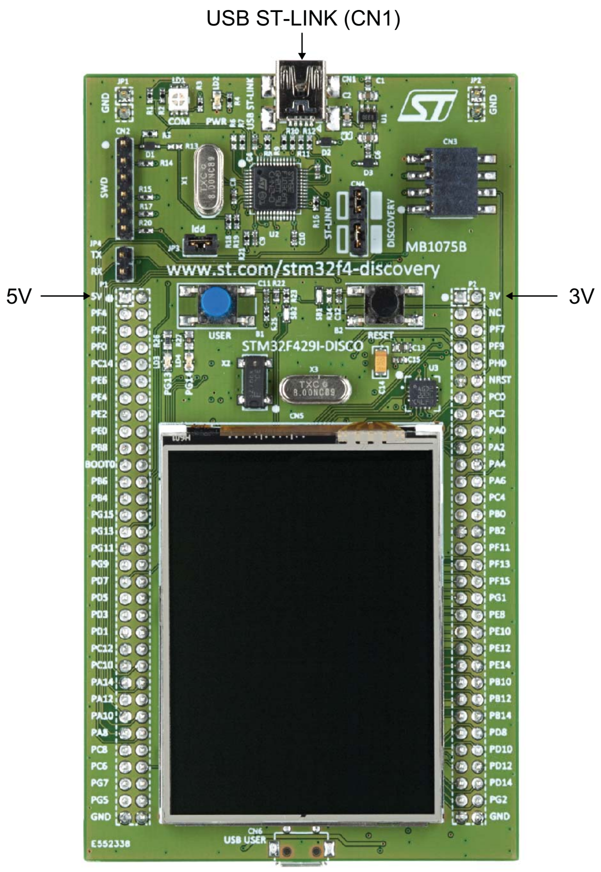
Note: Required information to download and install desired IDE and ST-LINK/V2 are detailed in ***Getting started with STM32F429 Discovery software development tools*** *document.*

1.4 Powering up the board

The STM32F429I-DISCO board can be powered up from three sources.

- USB ST-LINK: To power the board from the USB connector CN1, use the 'USB type A to Mini-B' cable and connect it between the host and the board USB connector CN1.
- External sources: DC power supply can be inserted in the GND and 3 V (or 5 V) pin.

Figure 2. STM32F429I-DISCO power sources



1.5 Reset the board

There are three ways to reset the board:

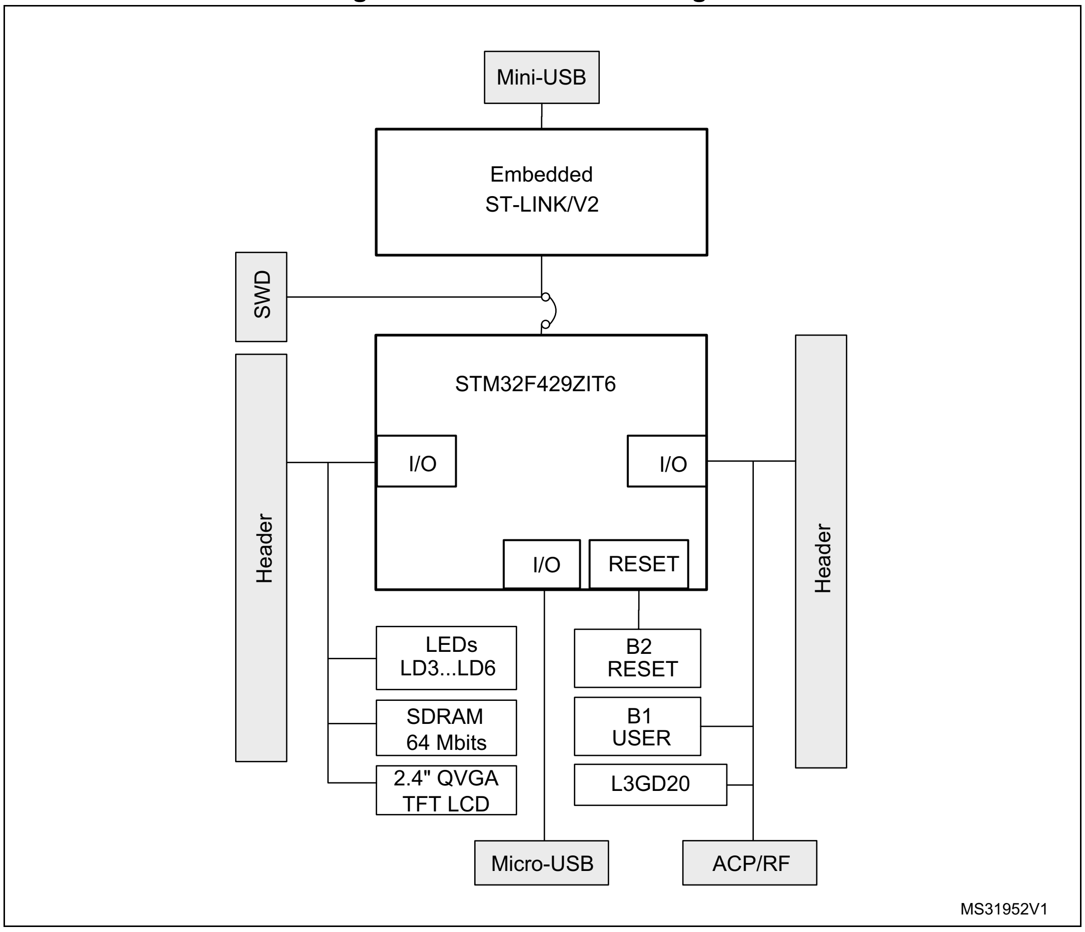
- Push the reset button mounted on the STM32F429I-DISCO.
- Remove and reinsert the USB cable.
- The MCU can also be reset by debuggers.

1.6 Hardware block diagram

The STM32F429I-DISCO is designed around the STM32F429ZIT6U microcontroller in a 144-pin LQFP package. *Figure 3* illustrates the connections between the STM32F429ZIT6U and its peripherals (STLINK/V2, pushbutton, LED, USB and connectors).

Please refer to schematic under *www.st.com/stm32f4-discovery* for more details.

Figure 3. Hardware block diagram



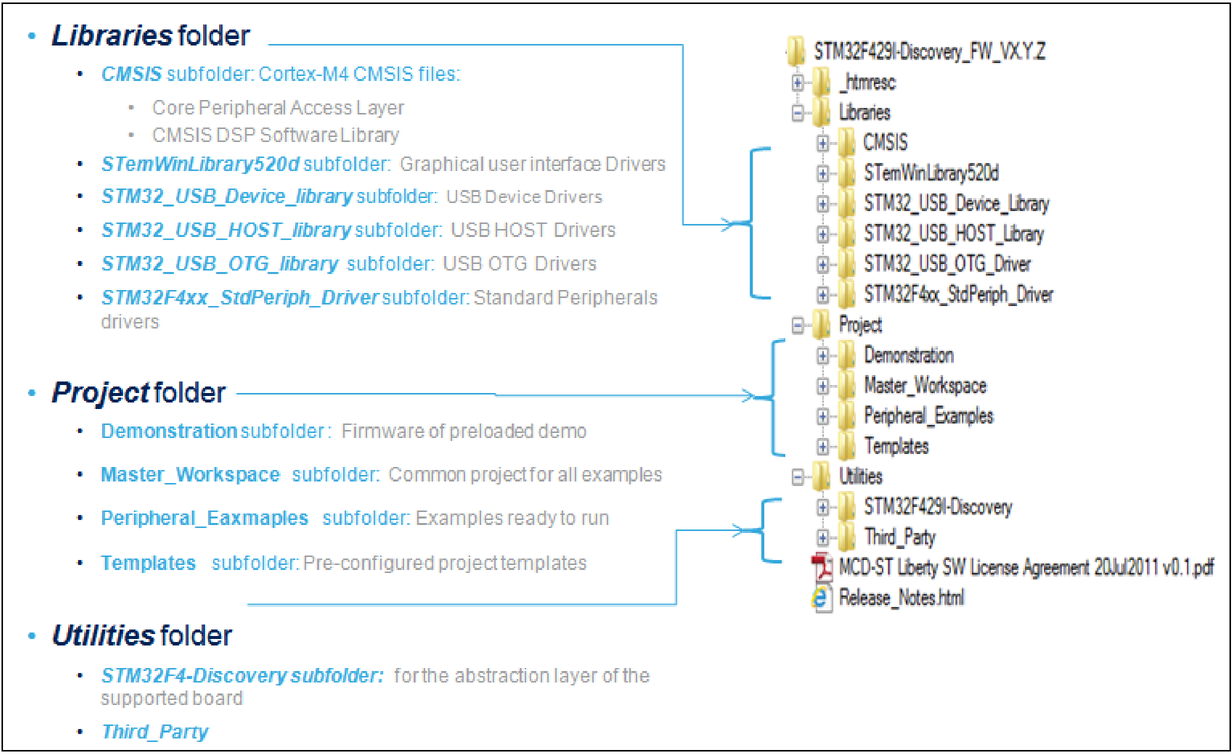
2 Firmware package

To get started with the STM32F429 Discovery kit, a firmware package that contains a set of IP examples and demonstrations of some features exists under *www.st.com/stm32f4-discovery*.

2.1 Package description

The STM32F429 Discovery firmware applications, demonstration and IPs examples are provided in one single package and supplied in one single zip file. The extraction of the zip file generates one folder, *STM32F429I-Discovery_FW_VX.Y.Z*, which contains the following subfolders:

Figure 4. Package contents



- ***Libraries*** **folder**
  - ***CMSIS*** subfolder: Cortex-M4 CMSIS files:
    - Core Peripheral Access Layer
    - CMSIS DSP Software Library
  - ***STemWinLibrary520d*** subfolder: Graphical user interface Drivers
  - ***STM32_USB_Device_library*** subfolder: USB Device Drivers
  - ***STM32_USB_HOST_library*** subfolder: USB HOST Drivers
  - ***STM32_USB_OTG_library*** subfolder: USB OTG Drivers
  - ***STM32F4xx_StdPeriph_Driver*** subfolder: Standard Peripherals drivers
- ***Project*** **folder**
  - **Demonstration** subfolder: Firmware of preloaded demo
  - **Master_Workspace** subfolder: Common project for all examples
  - **Peripheral_Eaxmaples** subfolder: Examples ready to run
  - **Templates** subfolder: Pre-configured project templates
- ***Utilities*** **folder**
  - ***STM32F4-Discovery subfolder:*** for the abstraction layer of the supported board
  - ***Third_Party***

User can run examples provided within this package. A set of examples for each peripheral are ready to be run.

2.2 Programming firmware application

To start programming, user must:

- Install preferred Integrated Development Environment (IDE)
- Install the ST-LINK V2 driver from ST web site

2.2.1 Programming application

To program application (demonstration or example), follow the sequence below:

1. Go under application folder
2. Chose the desired IDE project
3. Double click on the project file (ex. *STM32F429I-Discovery_Demo.eww* for EWARM)
4. Rebuild all files: Project->Rebuild all
5. Load project image: Project->Debug
6. Run program: Debug->Go

Please refer to ***Getting started with STM32F429 Discovery software development tools*** for more details.

2.2.2 Run pre-loaded demo

To run and develop any firmware applications on your STM32F429 Discovery board, the minimum requirements are as follows:

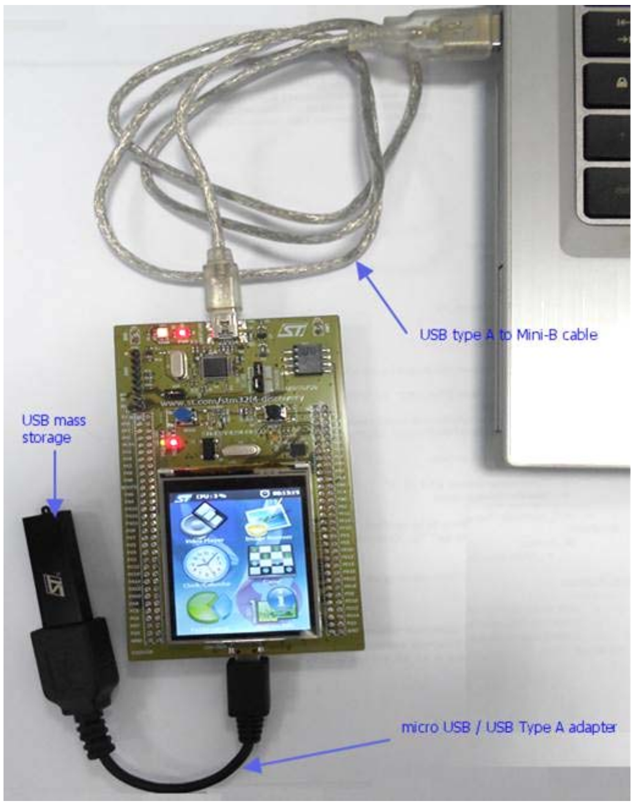
- Windows PC (XP, Vista, 7)
- 'USB type A to Mini-B' cable, used to power the board (through USB connector CN1) from host PC and connect to the embedded ST-LINK/V2 for debugging and Programming.

Additional hardware accessories will be needed to run some applications:

- 'USB type A to Micro-B' cable, used to connect the board (through USB connector CN5) as USB Device to host PC.

Establish the connection with the STM32F429 Discovery board as follows:

Figure 5. Hardware environnement



The demonstration software, based on the STemWin GUI library, is already preloaded in the board's Flash memory. It uses the LCD TFT mounted on the board to show the Menu based-on-icon view widget (Image Browser, Game, Performance, Clock/Calendar, Video and System Info module). The status bar indicate the CPU Usage, date, USB disk flash connection state, alarm and time.

Follow the sequence below to configure the STM32F429 Discovery board and launch the DISCOVER application:

1. Ensure that the jumpers JP3 and CN4 are set to "on" (Discovery mode).
2. Connect the STM32F429 Discovery board to a PC using a USB cable type A/mini-B through the USB ST-LINK connector CN1, to power the board. The LEDs LD2 (PWR) and LD1 (COM).
3. The following applications are available on the screen:
   – Clock/Calendar and Game
   – Video Player and Image Browser (play videos and view images from the USB mass storage connected to CN6)
   – Performance monitor (watch the CPU load and run a graphical benchmark)
   – System Info
4. The demo software, as well as other software examples that allow you to discover the STM32 F4 series features, are available on *www.st.com/stm32f4-discovery*.
5. Develop your own applications starting from the examples.

3 Revision history

Table 1. Document revision history

Date	Revision	Changes
09-Sep-2013	1	Initial release.