

Interfaccia USB per hard disk PATA/IDE su scheda STM32F4DISCOVERY

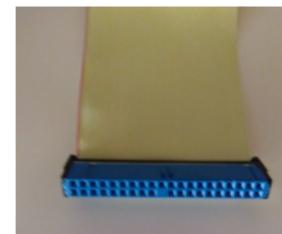
Gianluca Nitti

Esame di Progettazione di Sistemi Operativi

8 ottobre 2021

Introduzione

- ATA (Advanced Technology Attachment): protocollo di comunicazione per periferiche di archiviazione di massa
- PATA (parallel ATA): interfaccia fisica per ATA formata da un bus parallelo di 16 bit e segnali di controllo
- USB (Universal Serial Bus): interfaccia per vari tipi di dispositivi, con *classi* standard che definiscono un protocollo di comunicazione unificato per dispositivi con le stesse funzionalità



Connettore PATA/IDE



Connettore USB

Obiettivo del progetto

Realizzare con un microcontrollore un convertitore da PATA/IDE ad USB per utilizzare hard disk interni di vecchi PC come dischi esterni USB



Un adattatore commerciale di cui si intende replicare la funzionalità

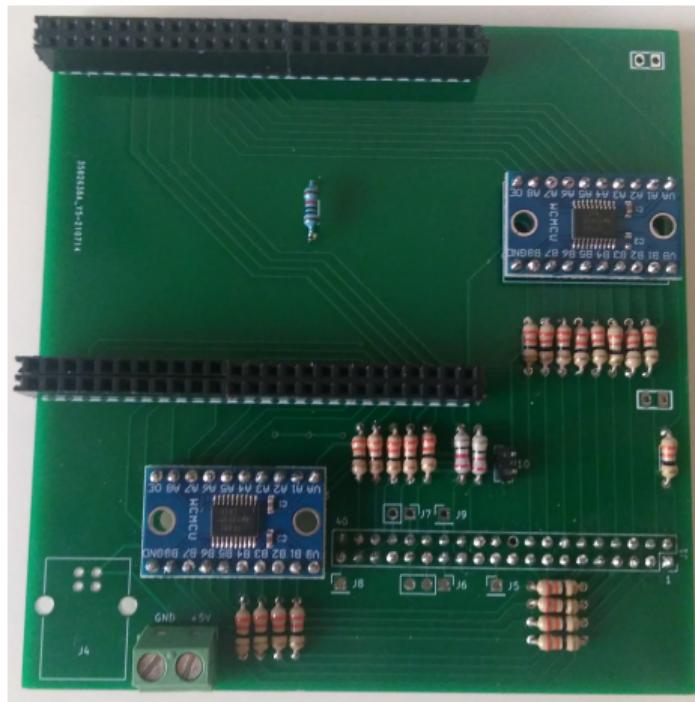
Hardware: scheda microcontrollore

Scheda di sviluppo STM32F4DISCOVERY

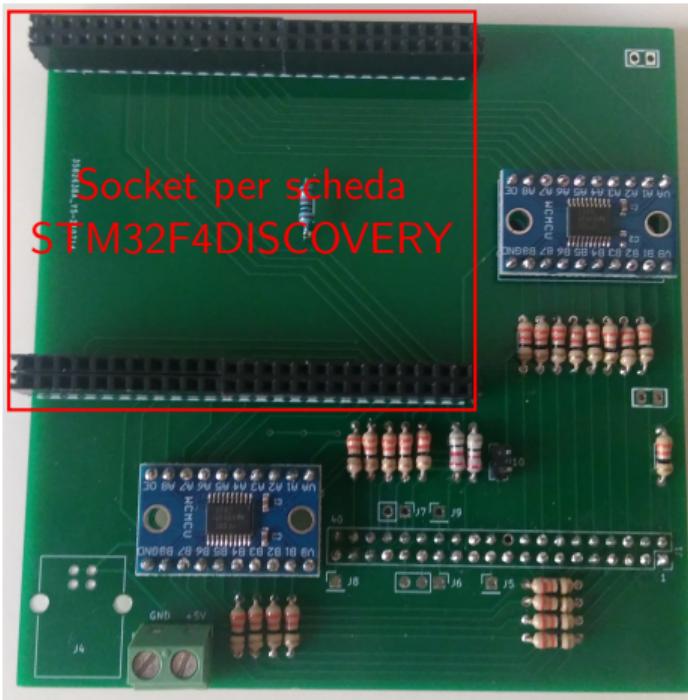
- Basata su MCU STM32F407VGT6, con CPU 32-bit ARM Cortex-M4 fino a 168MHz e 192 KB di RAM
- L'MCU contiene un controller USB *Full-Speed* (1.1), la cui interfaccia è esposta con un connettore micro-B sulla scheda; limite teorico di velocità dell'USB 1.1: 1.5MB/s
- Numerosi GPIO che possono essere utilizzati per implementare l'interfaccia PATA/IDE



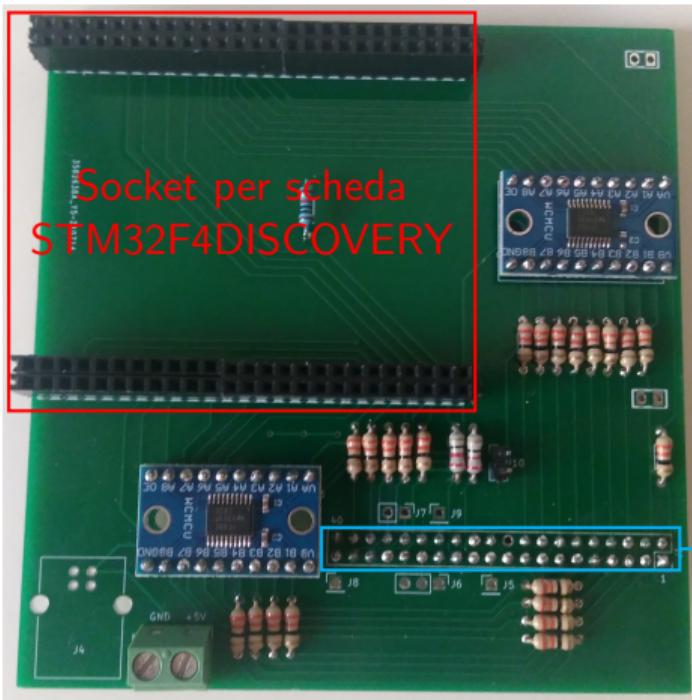
Hardware: scheda custom per interfaccia IDE/PATA



Hardware: scheda custom per interfaccia IDE/PATA

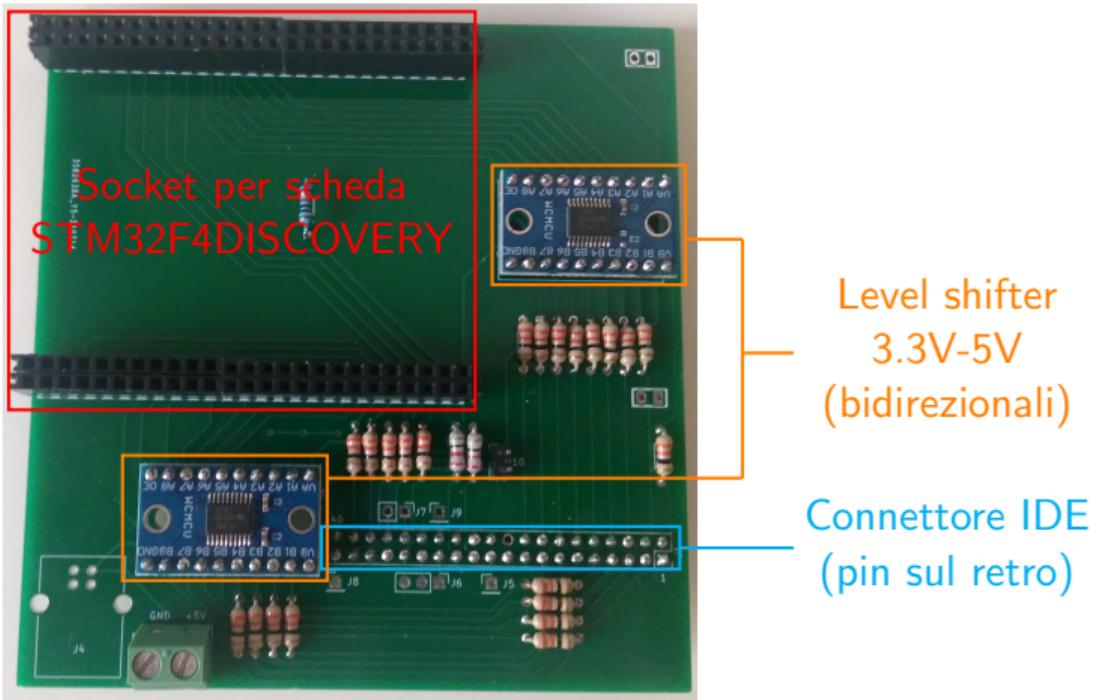


Hardware: scheda custom per interfaccia IDE/PATA



Connettore IDE
(pin sul retro)

Hardware: scheda custom per interfaccia IDE/PATA



Hardware: assemblaggio completo



PATA/IDE: principali segnali

- DD0-DD15: bus dati bidirezionale da 16 bit
- CS0-CS1, DA0-DA2: bus indirizzi: seleziona il registro della periferica da leggere/scrivere
- DIOR, DIOW: comandi di lettura e scrittura; entrambi normalmente alti, vengono abbassati per leggere o scrivere il registro selezionato
- RESET: normalmente alto, può essere abbassato per riportare la periferica nello stato di default post-accensione

PATA/IDE: lettura di un registro

- Configurazione dei pin GPIO del bus dati in modalità input
- Selezione del registro tramite scrittura dei pin DA0-DA2 e CS0-CS1
- Scrittura di uno 0 logico sul pin DIOR, per richiedere al device di porre sul bus il contenuto del registro
- Lettura dei pin del bus dati, da cui si ottiene il contenuto del registro
- Scrittura di un 1 logico sul pin DIOR, per richiedere al device di rilasciare il controllo del bus

PATA/IDE: scrittura di un registro

- Configurazione dei pin GPIO del bus dati in modalità output
- Selezione del registro tramite scrittura dei pin DA0-DA2 e CS0-CS1
- Scrittura del valore desiderato sui pin del bus dati
- Scrittura di uno 0 logico sul pin DIOW, per richiedere al device di acquisire il contenuto del bus e di scriverlo nel registro
- Scrittura di un 1 logico sul pin DIOW, per richiedere al device di terminare il monitoraggio/lettura del bus

PATA/IDE: i principali registri

<i>Registro/i</i>	<i>Lettura</i>	<i>Scrittura</i>
STATUS/COMMAND	Ciascun bit indica la presenza o meno di una condizione; es. READY, BUSY, DRQ	“Opcode” del comando da impartire al device
DATA	Dati letti dal disco ¹	Dati da scrivere sul disco
Indirizzamento (4 registri)	N/A	LBA/CHS iniziali per comando di R/W
SECTOR COUNT	N/A	Numero di settori interessati da comando di R/W

¹Non idempotente: ogni lettura restituisce la word successiva

- Attesa bit di READY (lettura, se necessario ripetuta, di **STATUS/COMMAND**)
- Scrittura dei parametri del comando di lettura nei relativi registri
 - LBA iniziale
 - Numero di settori (*nsect*)
- Scrittura del comando di lettura (0x20) in **STATUS/COMMAND**
- Per *nsect* volte:
 - Attesa bit DRQ
 - Sequenza di 256 letture di **DATA**: ognuna restituisce 2 bytes di contenuto del settore (1 settore = 512 bytes)

PATA/IDE: scrittura su disco

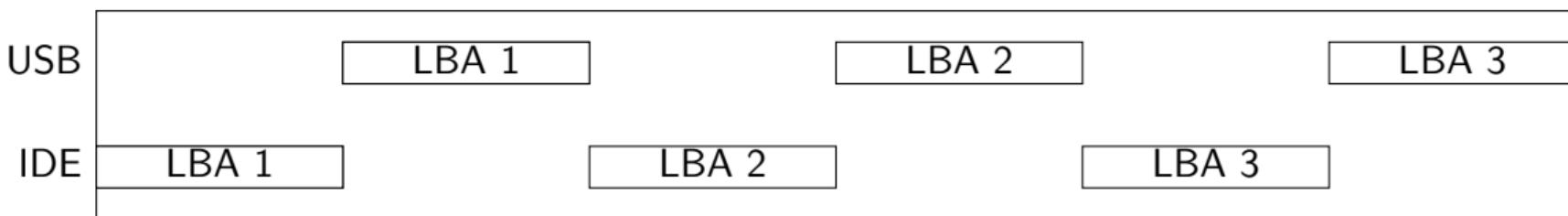
- Attesa bit di READY (lettura, se necessario ripetuta, di **STATUS/COMMAND**)
- Scrittura dei parametri del comando di scrittura nei relativi registri
 - LBA iniziale
 - Numero di settori (*nsect*)
- Scrittura del comando di scrittura (0x30) in **STATUS/COMMAND**
- Per *nsect* volte:
 - Attesa bit DRQ
 - Sequenza di 256 scritture di **DATA**, ciascuna contenente 2 bytes di dati da scrivere nel settore

Funzioni (callback) da implementare per realizzare un dispositivo USB MSC:

```
int8_t STORAGE_GetMaxLun(void);
int8_t STORAGE_Init(uint8_t lun);
int8_t STORAGE_GetCapacity(uint8_t lun, uint32_t *n_blks, uint16_t *blk_size);
int8_t STORAGE_IsReady(uint8_t lun);
int8_t STORAGE_IsWriteProtected(uint8_t lun);
int8_t STORAGE_Read(uint8_t lun, uint8_t *buf, uint32_t lba, uint16_t n_blks);
int8_t STORAGE_Write(uint8_t lun, uint8_t *buf, uint32_t lba, uint16_t n_blks);
```

Libreria USB STMicroelectronics: limitazioni

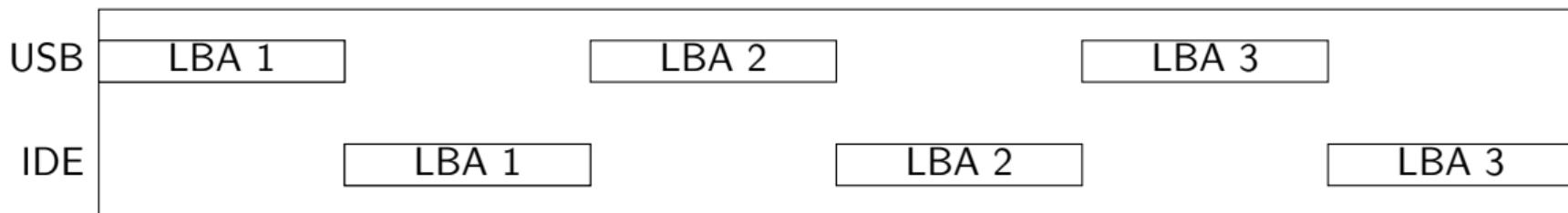
Lettura disco: STORAGE_Read deve obbligatoriamente caricare in memoria tutti i settori richiesti, ed il trasferimento verso l'host USB ha inizio solo quando questa ritorna



Massima velocità di trasferimento raggiunta: 350kB/s

Libreria USB STMicroelectronics: limitazioni

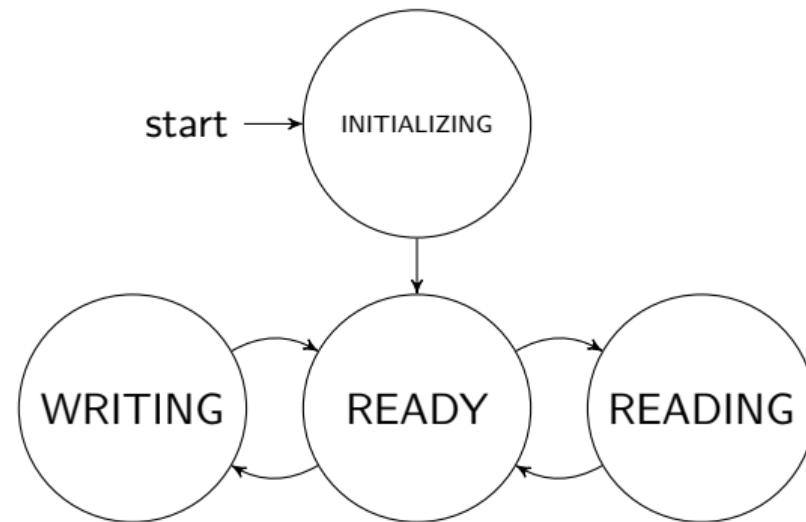
Scrittura disco: analogamente, `STORAGE_Write` viene chiamata solo dopo che l'intero “blocco” di settori di cui l'host USB ha richiesto la scrittura è stato trasferito, e deve obbligatoriamente completare l'intera operazione



- Implementazione open source dello stack USB (host e device) per vari microcontrollori, tra cui quello in uso in questo progetto
- Integrazione con FreeRTOS: può essere eseguita da un task che chiama in loop `tud_task()`, da cui la libreria si occupa di chiamare i callback quando necessario
- Funzioni di callback da implementare per un device USB MSC:

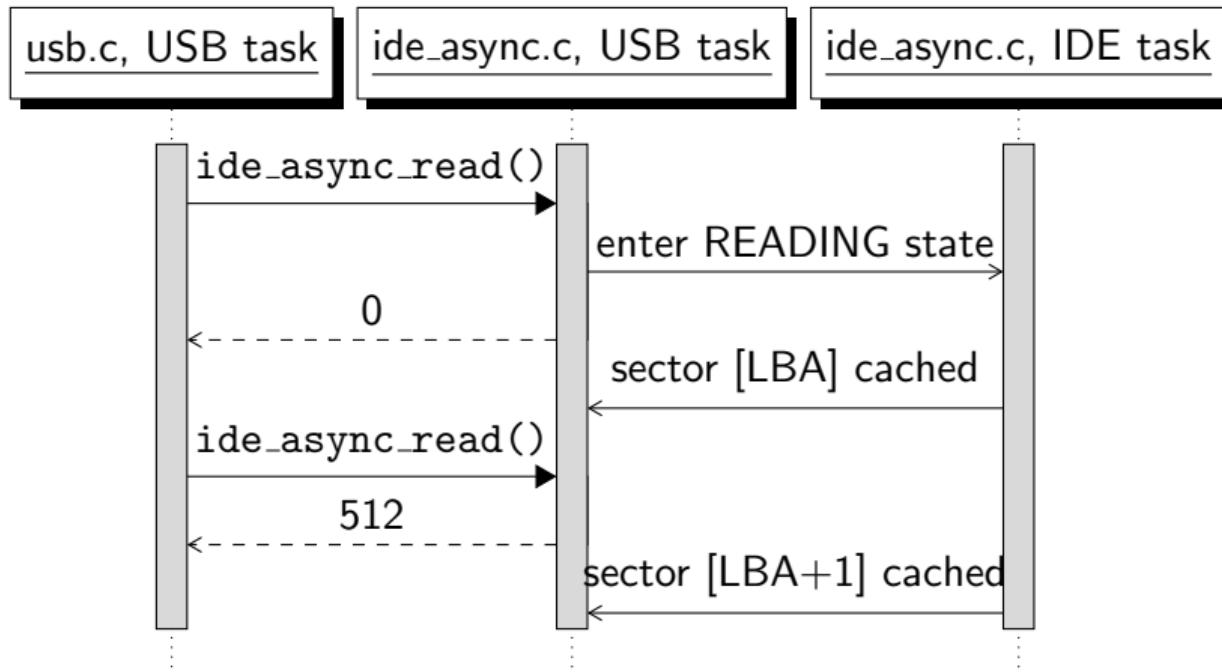
```
uint8_t tud_msc_get_maxlun_cb(void);
bool tud_msc_test_unit_ready_cb(uint8_t lun);
void tud_msc_capacity_cb(uint8_t lun, uint32_t* blk_count, uint16_t* blk_sz);
// ritornano il numero di bytes trasferiti:
int32_t tud_msc_read10_cb(uint8_t lun, uint32_t lba, uint32_t offset,
                           void* buffer, uint32_t bufsize);
int32_t tud_msc_write10_cb(uint8_t lun, uint32_t lba, uint32_t offset,
                           uint8_t* buffer, uint32_t bufsize);
```

Stato task di gestione interfaccia PATA/IDE

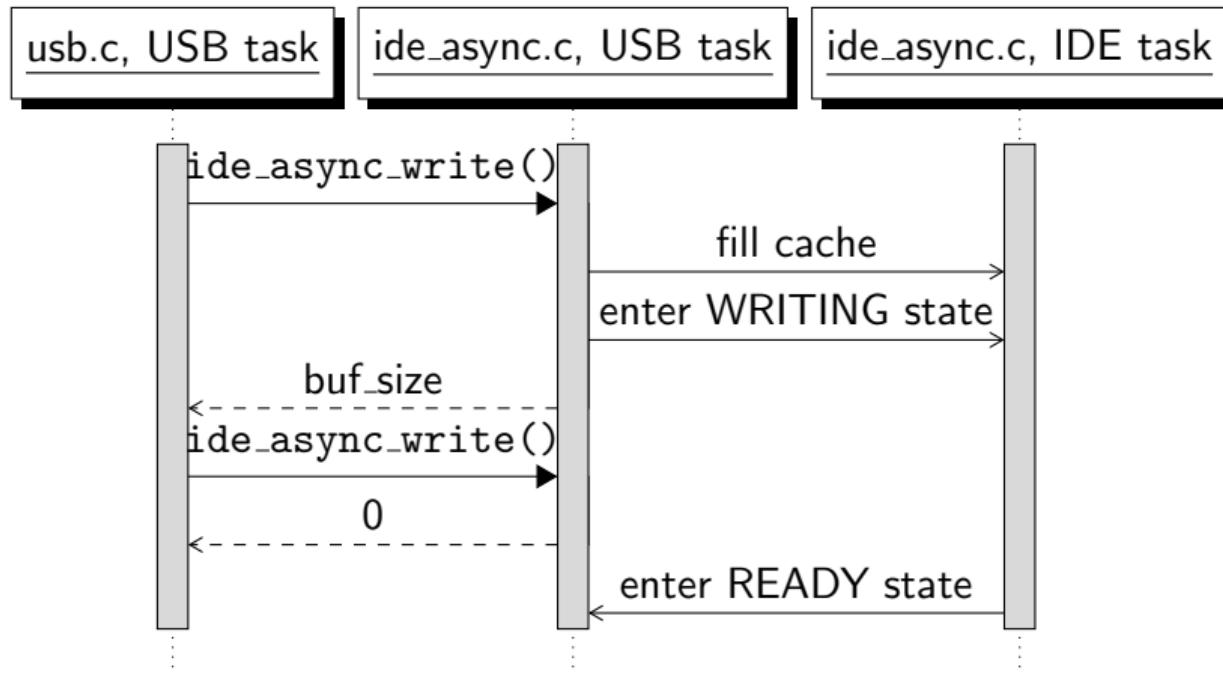


```
enum {INITIALIZING, READY, READING, WRITING} state;  
uint32_t requested_lba;  
uint8_t cache[32 * 512];  
uint32_t cache_len;
```

Lettura ottimizzata



Scrittura ottimizzata



Utilizzo risorse ottimizzato

Lettura

USB	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5
IDE	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5

Scrittura

USB	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6
IDE	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 5

Massima velocità di trasferimento raggiunta: 1.1MB/s

Riferimenti

- American National Standard for Information Technology. ATA/ATAPI Parallel Transport (2013)
- American National Standard for Information Technology. ATA Command Set (2007)
- USB Implementers Forum. Universal Serial Bus Mass Storage Class Bulk-Only Transport (1999)
- USB Implementers Forum. Universal Serial Bus Mass Storage Class Specification Overview (2010)
- Thach Ha. An open source cross-platform USB stack for embedded system (<https://github.com/hathach/tinyusb>)
- Manuali STMicroelectronics su MCU STM32F407, scheda di sviluppo STM32F4DISCOVERY e librerie STM32Cube