Oltre Arduino: dal prototipo al prodotto con microcontroller ESP32

UniVR - Dipartimento di Informatica

Fabio Irimie

1° Semestre 2024/2025

Indice

1 Introduzione

1.1 Tipi di microcontrollori

1.1.1 STM

Hanno più risorse hardware e ampio supporto dalla community. Fornisce come strumento **STM32CubeMX** che permette di configurare il microcontrollore in modo grafico, come ad esempio i GPIO (General Purpose Input Output) ecc... per dire alla CPU come interagire con ogni pin.

1.1.2 Espressif

L'ESP32 è il microcontrollore utilizzato all'interno di questo corso. Usato per il basso costo e facilità di utilizzo e l'ambiente di sviluppo (**ESP-IDF SDK**) è fatto molto bene. La documentazione è completa con tanto di esempi. Questo microcontrollore è basato su **FreeRTOS**, un sistema operativo real time.

1.2 Sistema operativo FreeRTOS

È un sistema operativo open source altamente configurabile che fornisce primitive per la creazione di thread, mutex, semafori, timer e memoria dinamica. Questo più che un sistema operativo come Linux, è un kernel che fornisce le primitive per la gestione dei thread e delle risorse, come ad esempio lo scheduler ecc...

Un sistema operativo real time ha la caratteristica di avere un tempo di risposta garantito, cioè se un evento deve essere gestito in un tempo massimo, il sistema operativo deve garantire che l'evento venga gestito entro quel tempo.

1.2.1 Schemmi di allocazione

FreeRTOS mette a disposizione varie strategie di allocazione:

- Solo allocazione
- Allocazione e deallocazione semplice
- Allocazione e deallocazione in coalescenza
- Coalescenza e heap frammentato
- Allocazione e deallocazione completa con mutua esclusione

1.2.2 Estensioni

FreeRTOS dà la possibilità di estendere il sistema operativo mediante librerie con varie funzionalità come:

- Stack TCP/IP
- File system
- Command line interface
- Interfaccia per i driver
- Ecc...

2 Configurazione dell'ambiente di sviluppo

La seguente configurazione è stata fatta su un sistema operativo Linux con il microcontrollore ESP32.

2.1 Installazione del toolchain

Per scaricare la toolchain su Visual Studio Code installare l'estensione **ESP-IDF** e seguire le istruzioni presenti nell'estensione.

2.1.1 Linux + Qualsiasi editor

Il manuale di installazione si trova a: *getting-started*.

1. Installare i prerequisiti

   - Fedora

   ```
   sudo dnf install git wget flex bison gperf python3 cmake
       ninja-build ccache dfu-util libusbx
   ```

   - Ubuntu e Debian

   ```
   sudo apt-get install git wget flex bison gperf python3
       python3-pip python3-venv cmake ninja-build ccache
       libffi-dev libssl-dev dfu-util libusb-1.0-0
   ```

   - CentOS 7 & 8

   ```
   sudo yum -y update && sudo yum install git wget flex bison
        gperf python3 cmake ninja-build ccache dfu-util
       libusbx
   ```

   - Arch

   ```
   sudo pacman -S --needed gcc git make flex bison gperf
       python cmake ninja ccache dfu-util libusb
   ```

2. Scaricare le librerie fornite da Espressif in *ESP-IDF repository*.

   ```
   mkdir -p ~/esp
   cd ~/esp
   git clone -b v5.4 --recursive https://github.com/espressif/esp
       -idf.git
   ```

   ESP-IDF verrà scaricato in `~/esp/esp-idf`.

3. Installazione dei tool

   Oltre alle librerie, è necessario installare gli strumenti necessari usati da ESP-IDF, come ad esempio il compilatore, il debugger, i pacchetti python ecc...

   ```
   cd ~/esp/esp-idf
   ./install.sh esp32
   ```

   Questo script installerà tutti i pacchetti necessari per lo sviluppo con ESP32. Se si vuole installare per più microcontrollori, basta passare il nome del microcontrollore come argomento:

```
cd ~/esp/esp-idf
./install.sh esp32,esp32s2
```

Oppure per scaricare gli strumenti per tutti i target supportati:

```
cd ~/esp/esp-idf
./install.sh all
```

4. Configurare le variabili d'ambiente

   ESP-IDF fornisce uno script che imposta le variabili d'ambiente necessarie per compilare il codice. Questo script è da eseguire ogni volta che si vuole sviluppare con ESP-IDF:

   ```
   . $HOME/esp/esp-idf/export.sh
   ```

   Per facilitare l'uso di questo script, si può aggiungere al file `.bashrc` o `.zshrc` il seguente alias (riavviare il terminale, o fare il source del file per rendere effettive le modifiche):

   ```
   alias get_idf='. $HOME/esp/esp-idf/export.sh'
   ```

   In questo modo, ogni volta che si apre un terminale, basta eseguire il comando `get_idf` per impostare le variabili d'ambiente.

2.2 Creazione ed esecuzione di un progetto

Prima di creare un progetto bisogna entrare nell'ambiente di sviluppo con il comando `get_idf`. Questo comando rende disponibile il comando `idf.py`.

- Creare un progetto

  Per creare un progetto eseguire il seguente comando:

  ```
  idf.py create-project <nome-progetto>
  ```

- Connetti il microcontrollore

  Connettere il microcontrollore e verificare in quale porta seriale è disponibile. Su linux le porte seriali si trovano in `/dev/tty*`. Per verificare la lista delle porte seriali disponibili, eseguire il seguente comando:

  ```
  ls /dev/tty*
  ```

  Se collegato tramite usb, il microcontrollore dovrebbe essere disponibile in `/dev/ttyUSB0`. Questa porta seriale potrebbe essere in possesso di **root**, quindi per poterla usare bisogna dare i permessi all'utente:

  ```
  sudo usermod -a -G dialout $USER
  sudo chmod a+rw /dev/ttyUSB0
  ```

- Configurare il progetto

  Per configurare il progetto, entrare nella cartella del progetto e lanciare il seguente comando per impostare il microcontrollore di target:

```
idf.py set-target esp32
```

Successivamente, per configurare il progetto, eseguire il seguente comando:

```
idf.py menuconfig
```

Questo comando aprirà un menu con tutte le configurazioni del progetto.

- Compilare il progetto

  Per compilare il progetto, eseguire il seguente comando:

  ```
  idf.py build
  ```

- Flash del microcontrollore

  Per caricare il programma sul microcontrollore (flash), eseguire il seguente comando:

  ```
  idf.py -p PORT flash
  ```

  Dove PORT è la porta seriale in cui è collegato il microcontrollore.

- Monitorare il microcontrollore

  Per monitorare il microcontrollore, eseguire il seguente comando:

  ```
  idf.py -p PORT monitor
  ```

- Consigli
  - Il flash compila in automatico, quindi per compilare, flashare e fare il monitoraggio si può eseguire il seguente comando:

    ```
    idf.py -p PORT flash monitor
    ```

  - Per eliminare il flash, eseguire il seguente comando:

    ```
    idf.py -p PORT erase-flash
    ```