Stazione Meteo

Taikomo

（タイコモ）

Arduino Due
GY-39

ESP-8266
SD Card Reader

a cura di
Giacomo Guiduzzi
Taisia Marconi

Indice

1a. Introduzione - Overview

Il nostro progetto consiste in una stazione meteo ad utilizzo domestico.

È possibile accedere ai dati raccolti dalla stazione tramite una interfaccia web.

I dati raccolti sono:

- Temperatura;
- Umidità;
- Pressione;
- Altitudine;
- Luminosità ambientale.

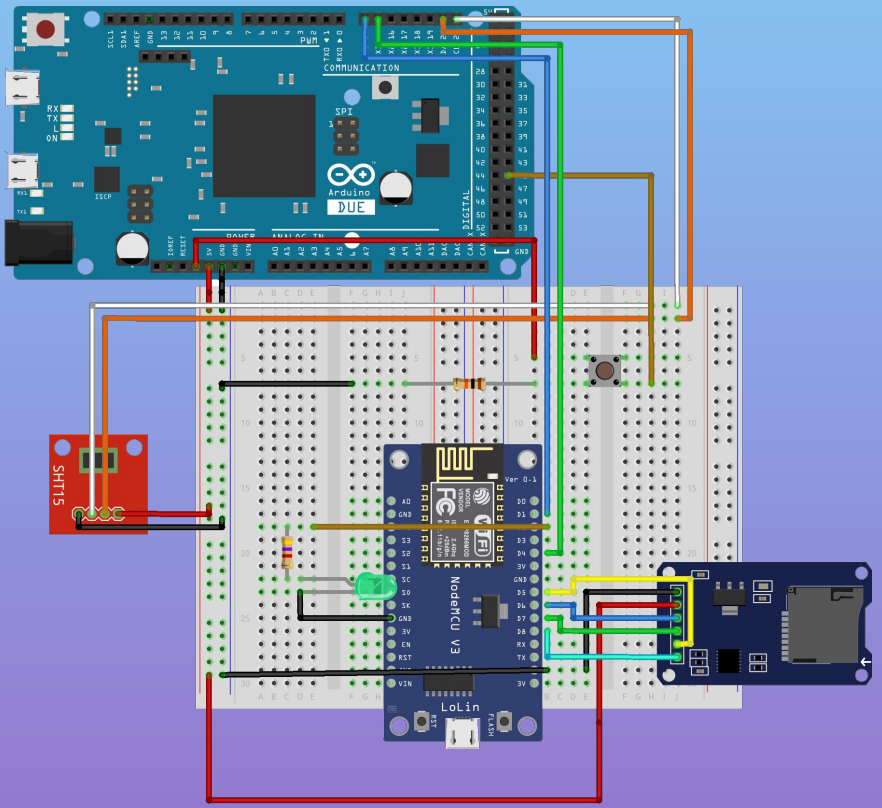
1a. Introduzione - Overview

È possibile modificare la frequenza di aggiornamento tramite l'interfaccia o tramite la pressione di un bottone fisico sulla breadboard.
È presente anche un LED verde che mostra la effettiva comunicazione tra le schede.

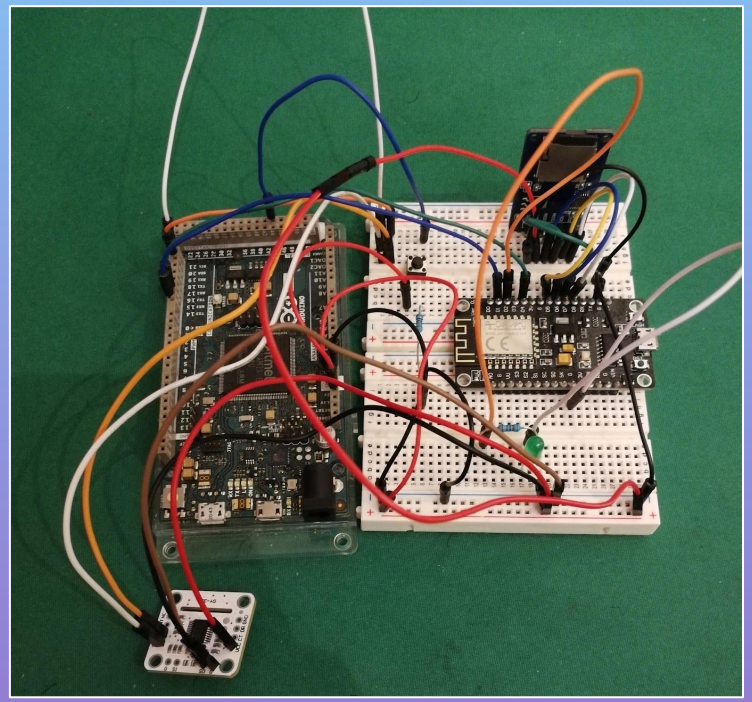
Caratteristiche delle schede:

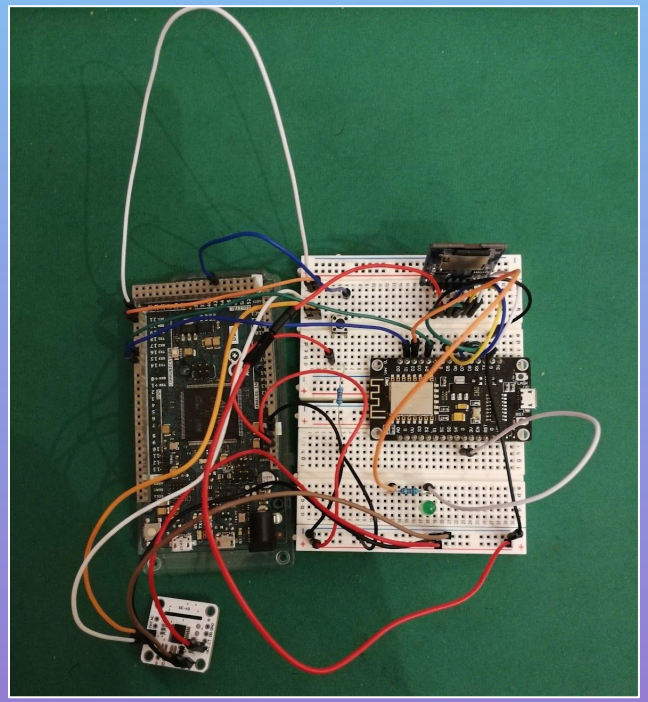
Scheda	CPU	FLASH	RAM
Arduino Due	Atmel SAM3X8E ARM Cortex-M3@84MHz	512 KB	96 KB
Lolin ESP-8266 NodeMCU 1.0 (ESP-12E)	ESP-8266@80 MHz	4 MB	64 KB

1a. Introduzione - Overview (come dovrebbe essere)



1a. Introduzione - Overview (com’è in realtà)





1b. Introduzione - Struttura del progetto

Il progetto è suddiviso su un totale di **4** file:

1. **FreeRTOS_ArduinoDue.ino** - implementazione del SO FreeRTOS;
2. **ESP8266_Webserver.ino** - webserver su ESP-8266;
3. **index.html** - pagina principale dell'interfaccia web;
4. **functions.js** - file di script Javascript per le librerie grafiche e le richieste asincrone AJAX.

Entrambe le schede sono state programmate e testate attraverso **ArduinoIDE**.

2a. Funzionalità built-in

- setup() & loop();
- LED Control;
- Hardware Serial (anche per debugging);
- ISR Handling.

2b. Funzionalità - Librerie

- ESP8266:
  - **ESP8266WiFi.h** - gestisce la connessione al WiFi;
  - **WiFiClient.h** - gestisce il client e la sua connessione;
  - **ESP8266WebServer.h** - gestisce il webserver e le risposte;
  - **SD.h** - gestisce la comunicazione con la scheda SD;
  - **ESPSoftwareSerial.h** - gestisce la comunicazione con l'Arduino Due;
- Arduino Due:
  - **FreeRTOS.h** - libreria che implementa il SO FreeRTOS per Arduino Due;
  - **task.h** - libreria di FreeRTOS che implementa i task;
  - **Wire.h** - libreria di Arduino che permette di comunicare fisicamente con il sensore;
  - **Adafruit_BME280.h** - libreria per recuperare temperatura, umidità, pressione e altitudine;
  - **MAX44009.h** - libreria per recuperare la luminosità.

2c. Funzionalità - Logica operativa (Arduino Due)

FreeRTOS, sull'Arduino Due, inizializza 7 task:

- **TaskGetTemp** - recupera la temperatura dal sensore;
- **TaskGetHum** - recupera l'umidità;
- **TaskGetPress** - recupera la pressione;
- **TaskGetAlt** - recupera l'altitudine;
- **TaskGetBright** - recupera la luminosità;
- **TaskSendData** - comunica alla scheda ESP-8266 i dati raccolti quando pronti;
- **TaskGetDelay** - controlla potenziali messaggi dall'ESP-8266.

I primi 5 tasks hanno priorità più bassa, *TaskSendData* ha priorità media e *TaskGetDelay* priorità massima (e periodo più breve di tutti).

TaskSendData non ha periodo in quanto è svegliato tramite un meccanismo di notifica (*xTaskNotifyGive()* e *xTaskNotifyTake()*);

2c. Funzionalità - Logica operativa (Arduino Due)

I task che comunicano con il sensore scrivono all'interno di un array il proprio completamento.
Quando questo array è "pieno" significa che tutti i task hanno raccolto il proprio dato; solo a questo punto viene svegliato *TaskSendData* che tramite UART manda la nuova struttura dati al webserver.

TaskGetDelay può ricevere un nuovo delay da parte del webserver, oppure può ricevere una risposta di conferma del nuovo delay impostato quando questo viene modificato dal bottone fisico.

Il delay, che può essere influenzato dal bottone o dal client web, determina il periodo dei task che interrogano il sensore. *TaskGetDelay* ha un periodo fisso.

2c. Funzionalità - Logica operativa (ESP-8266)

Per quanto riguarda l'*ESP-8266*, viene preparato il webserver con tutte le pagine correlate e nella funzione *loop()* si gestiscono le richieste del client e la comunicazione con l'Arduino Due.

L'*ESP-8266* può comunicare con l'Arduino in 3 modi:

- Nuova struttura dati;
- Nuovo delay dal bottone;
- Conferma del nuovo delay impostato dal client web.

Le pagine web di risposta al client vengono direttamente mandate in streaming tramite un handle al file sulla SD.

3a. Risultati - Interfaccia web

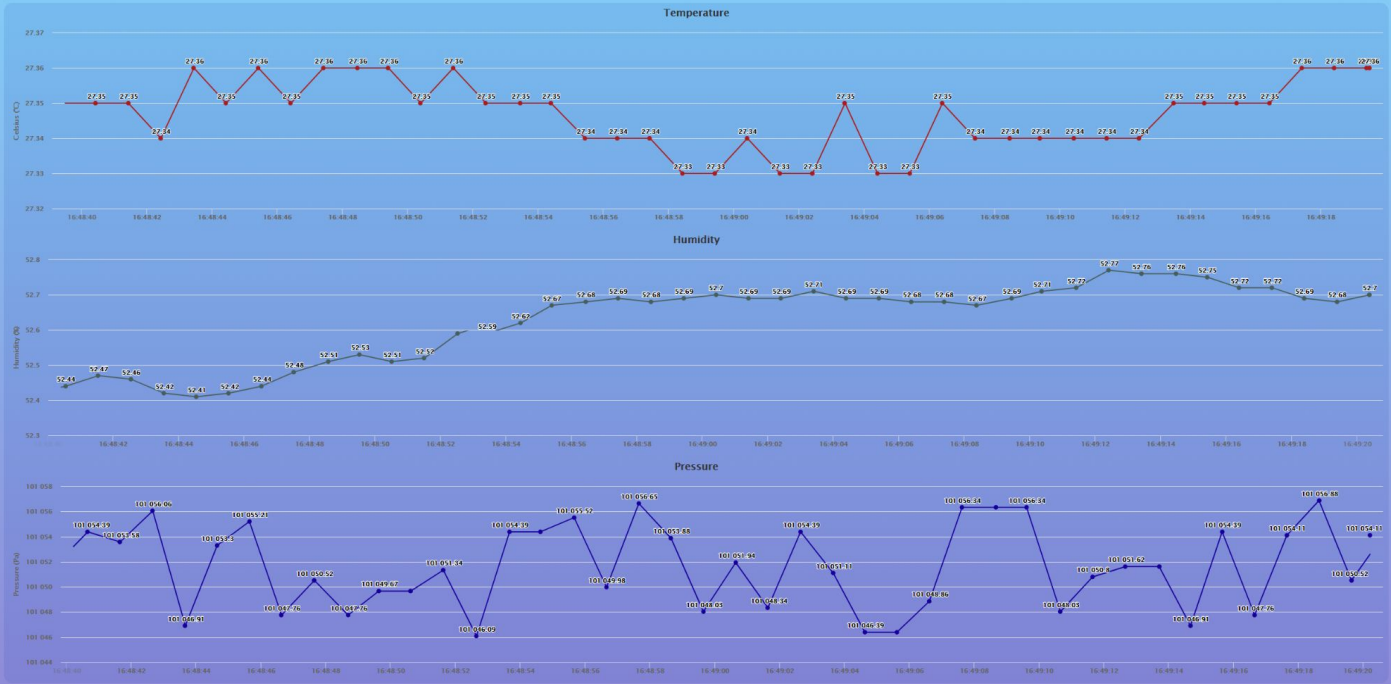
L'interfaccia presenta 3 grafici che illustrano l'andamento dei dati ***Temperatura***, ***Umidità*** e ***Pressione*** nel tempo.

Sotto questi grafici sono presenti i valori di ***Altitudine*** e ***Luminosità***.
In fondo alla web page sono presenti 2 sezioni, dedicate rispettivamente alla *modifica del delay* e alla *modifica* delle *unità di misura* della Temperatura e della Pressione.

Cliccando sui rispettivi bottoni è possibile inviare una richiesta AJAX al webserver, che risponderà positivamente o negativamente nel caso in cui ci siano stati errori di comunicazione.

Weather Station

Arduino Due + GY-39 + ESP8266 + SD card adapter



Altitude: 22.58m

Brightness: 0.00 lux

Sensors refresh rate

- (•) Medium
- ( ) Slow
- ( ) Very slow
- ( ) Take a break

Set refresh rate

Check unit of measure

Temperature:

- (•) Celsius (°C)
- ( ) Fahrenheit (°F)

Pressure:

- (•) Pascal (Pa)
- ( ) Bar (bar)

Set units of measure

3b. Segnali di funzionamento

Pagina che si aggiorna nel tempo →

Bottoni che appaiono per conferma modifica delay e unità di misura →

← LED verde durante la trasmissione dati

4a. Conclusioni

La stazione meteo risulta user-friendly grazie all'interfaccia web semplice e visivamente gradevole.

L'interazione è molto intuitiva ed è evidente quali siano le features messe a disposizione per l'utente.

Dopo alcuni test possiamo dire anche che il sistema è in grado di continuare ad operare nel tempo senza interruzioni del servizio di alcun tipo.

4a. Conclusioni - Requisiti fondamentali

- Uso di 1 task per canale input/output: ✔
  - *TaskGetData* e *TaskGetDelay* permettono l'interazione con il web server tramite UART;
- Utilizzo di 3 o più canali di input: ✔
  - Bottone fisico sulla breadboard;
  - Bottone per cambiare delay;
  - Bottone per le unità di misura sulla pagina web;
- Utilizzo di 2 o più canali di output: ✔
  - Pagina web;
  - LED verde che mostra l'effettiva trasmissione di dati tra le schede.

4a. Conclusioni - Requisiti fondamentali

- Utilizzo di 2 o più sensori o 1 sensore ed una scheda di rete: ✔️
  - Sensore GY-39 che comprende i sensori di temperatura, umidità, pressione, altitudine e luminosità;
  - Scheda ESP-8266 con funzionalità di rete wireless.
- Utilizzo di 1 o più task di comunicazione many-to-one: ✔️
  - I task che raccolgono i dati dal sensore (many) si occupano di notificare TaskSendData (one) che invierà i dati al web server.

4a. Conclusioni - Requisiti addizionali

- **Easy**:
  - Git versioning: ✔
    - https://github.com/ServantGrunt/rtes (repo privata)
  - Periodic execution: ✔
    - Tutti i 7 tasks dell'Arduino Due (8 compreso il Default Task) considerano il tempo di esecuzione nel calcolare il tempo rimanente al prossimo periodo tramite la funzione *xTaskGetTickCount()*;
  - Tuning: ✔
    - Task's constraint optimisation: Il periodo dei task è regolabile e un task, che non ha senso abbia un periodo, viene svegliato solo se soddisfatte certe condizioni;
- **Challenging:**
  - MISRA C compliance: ✔
    - abbiamo appurato tramite un software chiamato **Cppcheck** che il nostro software è predicibile in tutte le sue parti e rispetta lo standard.

Grazie per l’attenzione!