Università degli Studi di Trieste - Dipartimento di Ingegneria e Architettura

Relazione progetto Reti Logiche
Weather Station

Andrea Gonzato

Maggio 2019

Indice

1 Introduzione

2 Scopo del progetto
- 2.1 Descrizione per l'utilizzo
- 2.2 Specifiche
- 2.3 Risultato ottenuto

3 Realizzazione
- 3.1 Metodo usato
- 3.2 Struttura della programmazione ad oggetti implementata
- 3.3 Componenti utilizzati
- 3.4 Schema
- 3.5 Strumenti usati
- 3.6 Scelte progettuali e difficoltà

4 Sviluppi futuri

5 Rilascio materiale

6 Conclusioni

7 Sitografia

8 Dichiarazione finale

1 Introduzione

Questa è la relazione del progetto di Arduino svolto per il corso di reti logiche tenuto dal Professore Marsi dell'Università di Trieste.

2 Scopo del progetto

Lo scopo del progetto "Weather Station", da come suggerisce il nome è quello di realizzare una stazione meteorologica amatoriale. In aggiunta all'opportunità di poter visualizzare la temperatura e l'umidità attuale, sono state aggiunte funzionalità secondarie come la possibilità di poter leggere l'ora, data attuale e di poter utilizzare la stazione anche come fosse una sveglia. Ovviamente è previsto che data ora e sveglia possono essere regolate e la sveglia inoltre sia anche disattivabile.

2.1 Descrizione per l'utilizzo

Per controllare la stazione meteorologica ci sono tre pulsanti: *SET*, +, -. Il funzionamento ricorda molto l'utilizzo dei classici orologi da polso digitali. Dunque per poter scorrere lungo tutte le funzionalità si utilizzano i pulsanti + e -, mentre per impostare data, ora e sveglia si necessita di tener premuto per un secondo il pulsante *SET*, dopodichè nella regolazione si usano gli analoghi pulsanti per aumentare e diminuire, mentre per passare a impostare il campo successivo oppure per attivare/disattivare la sveglia si necessita di premere una singola volta il pulsante *SET*.

2.2 Specifiche

Come principali specifiche da rispettare, era necessario che l'orologio abbia una precisione che permetta di regolarlo mensilmente. Quindi ragionevolmente si richiedeva un errore inferiore di un secondo dopo 24 ore dalla sincronizzazione con un orologio affidabile.

Le altre specifiche sui dati meteo richiedono un errore inferiore a 1 grado Celsius per la temperatura e 2 punti percentuali per l'umidità.

2.3 Risultato ottenuto

In figura 1 si può vedere la realizzazione del progetto completato.

Tutte le specifiche nella realizzazione finale del progetto sono state rispettate.

Ne suo complesso il sistema si può ritenere funzionante, stabile e facile nell'utilizzo.

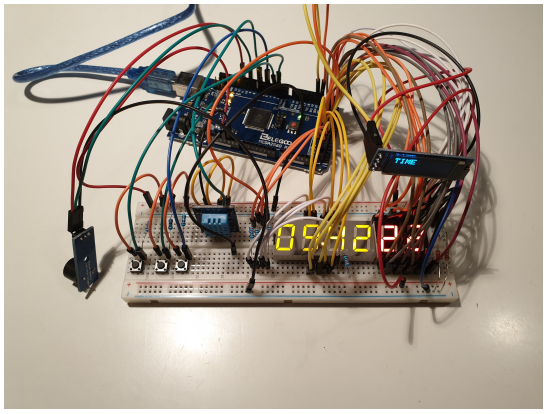

Figura 1: una immagine della realizzazione del progetto

3 Realizzazione

Tutto il progetto è stato realizzato tra Marzo 2019 e Maggio 2019.

3.1 Metodo usato

Il primo periodo è stato utilizzato per recuperare il materiale necessario. Una volta reperiti i componenti sono stati eseguiti una serie di test per comprendere l'utilizzo e le capacità dei singoli componenti.

Nella seconda fase, momento in cui è iniziata la vera realizzazione del progetto, la metodologia adottata era ispirata al paradigma AGILE: Si aggiungevano funzionalità una alla volta passando ciclicamente da una fase di progettazione a sviluppo per poi passare ai test e verificare la correttezza delle implementazioni.

La figura 2 illustra bene il paradigma adottato.

Nell'ultima fase del progetto, una volta implementata gran parte delle funzionalità, c'era la necessità di ristrutture il codice, creando una nuove versione strutturata ad oggetti, per poter avere un codice più leggibile ed uniforme.

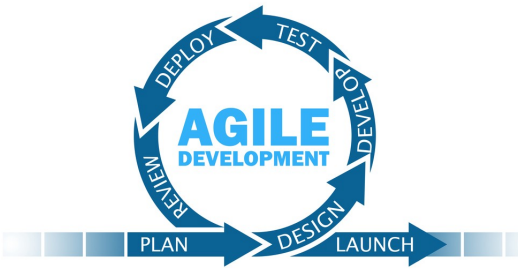

Figura 2: paradigma AGILE

3.2 Struttura della programmazione ad oggetti implementata

Di seguito viene fornita una breve descrizione di come è stato strutturato ad oggetti il progetto.

Tutti gli oggetti vengono creati nel file *Code/code.ino* e per ogni classe viene implementato un solo oggetto apparte per la classe *Button* dove ne vengono creati tre. I metodi "principali" di ogni classe si chiamano *update()* e in tali metodi i vari oggetti comunicano tra di loro i cambiamenti registrati durante il ciclo di eseguzione della funzione *loop()*

Le classi implementate sono:

- MyTime: una classe che contiene tutte le variabili temporali e scandisce nel tempo l'evoluzione del sistema, questa classe serve per esempio per l'attivazione dell'allarme della sveglia.
- Mode: una classe che gestisce e aggiorna il sistema informandolo su cosa bisogna visualizzare sui display a 7 segmenti: nella stazione meteo ci sono 5 dati fondamentali da far vedere all'utente che sono: tempo, data, temperatura, umidità, e l'orario a cui è impostata la sveglia. Ma osserva che per questioni progettuali sono uno alla volta di questi dati può essere visualizzato sui display.
- Button: una classe che gestisce tutte le attività che iniziano alla pressione di un pulsante, per esempio aggiorna il sistema comunicando che durante un settaggio si è premuto un pulsante per far aumentare le ore dell'orologio. Questa classe è in continua comunicazione con MyTime e Mode.
- controllSingleDigitDisplay: una classe che si occupa di visualizzare i dati richiesti sui 2 display a 7 segmenti 1 digit.

3.3 Componenti utilizzati

Dopo aver definito le specifiche, durante la prima fase è stato necessario reperire tutto il materiale necessario.
Questa è una lista completa del materiale utilizzato

- Arduino Mega 2560
- breadboard (63x10)
- sensore digitale DHT11 per rilevare temperatura e umidità
- schermo OLED (risoluzione: 128*32)
- display 7 segmenti (4 digits, catodo comune)
- 2 display 7 segmenti (1 digit, catodo comune)
- cicalino (buzzer passivo)
- 3 bottoni
- 7 resistenze da 220 Ohm
- 3 resistenze da 1k Ohm
- 50/55 cavi elettrici M/M
- 8 cavi elettrici M/F

Si osservi che per la realizzazione di questo progetto non è sufficiente un Arduino Uno perché, da come si vede in Figura 3, sono necessarie 34 connessioni digitali.

3.4 Schema

In fono alla relazione si può osservare la figura 3 che indica lo schema utilizzato per il progetto.

Nota che nello schema non sono presenti le connessioni, del cicalino e dello schermo OLED. Ma queste sono particolarmente semplici e intuitive. Se si vuole usare il codice del progetto, i pin del cicalino I/O e VCC vanno connessi rispettivamente a i pin Arduino 9 e 11.

3.5 Strumenti usati

I principali strumenti che sono stati utilizzati sono: l'ambiente Arduino e Fritzing come software per disegnare gli schemi elettronici.
Nel codice del progetto sono presenti come librerie esterne:

- *SevSeg*
- *DHT*

- *Adafruit Sensor*
- *Adafruit GFX*
- *Adafruit SSD1306*

SevSeg è stata utile per "scrivere" sul display a 7 segmenti 4 digits; mentre *DHT* e *Adafruit Sensor* sono servite per leggere i dati dal sensore che rilevava la temperatura e l'umidità; infine le ultime due librerie hanno permesso di utilizzare lo schermo OLED.

3.6 Scelte progettuali e difficoltà

Una scelta progettuale effettuata, consisteva nel scegliere come far scandire il tempo al sistema. La prima opzione consisteva nell'acquisire un modulo RTC appositamente designato per queste attività; ma la scelta progettuale invece attuata è stata di far scandire il tempo direttamente ad Arduino, creando una variabile *long unsigned* che incrementa al passare di ogni millisecondo. Scegliendo questa realizzazione si è riusciti a risparmiare un modulo ma ad un costo: infatti la variabile teoricamente dopo 4,294,967,295 millisecondi, cioè circa in 49 giorni arriva in overflow e questo potrebbe portare in crash il sistema (questo avvenimento non è ancora stato testato). Per questo motivo sopracitato si raccomanda di resettare la stazione prima dello scadere del quarantottesimo giorno.

La difficoltà più grande incontrata è stata sicuramente, quella di gestire e sincronizzare i display a 7 segmenti. Nel progetto originale, l'idea era quella di realizzare il sistema di visualizzazione dati utilizzando due display a 7 segmenti con 4 digits, quindi poter disporre di un totale di 8 digits. Il problema di questa realizzazione era che la libreria *SevSeg* utilizzata non supportava l'opportunità di comandare più display contemporaneamente. Per tale motivo la realizzazione finale del progetto è avvenuta con l'implementazione di un display 4 digits controllato dalla libreria, affiancato da due display sigle digit controllati senza libreria ma con comandi standard *digitalWrite()* di Arduino. Questo compromesso ha reso l'aggiornamento dei display non uniforme e non ottimizzato, infatti ad ogni eseguzione dell'*loop()* ciclicamente un solo digit del display composto da quattro si aggiorna, mentre entrambi i display single digit si aggiornano. In futuro, si dovrebbe migliorare questa non uniformità nell'illuminazione dei display. Ma attualmente non si ritiene sia fondamentale.

Come protocollo di comunicazione tra il display OLED e Arduino si è deciso di utilizzare il protocollo I^2C considerando il fatto che l'Arduino Mega 2560 possiede due pin addetti per tale scopo.

L'utilizzo della libreria *SevSeg* ha portato come compromesso che all'interno del *loop()* non possa essere inserito alcun ritardo di qualunque tipo. In caso questi avvenissero, il display a 7 segmenti 4 digits inizia a sfarfallare. A causa di questa problematica il sensore di umidità e temperatura rileva i dati non in maniera continuata, ma esclusivamente ogni 20 secondi. Questo perché altrimenti si verificherebbero dei ritardi nell'*loop()*. Infatti se si osserva bene il display, ogni 20 secondi c'è un piccolo

sfarfallamento, ma non è particolarmente fastidioso.

4 Sviluppi futuri

I prossimi sviluppi per questo progetto riguarderanno aspetti secondari. Di seguito vengono elencate alcune possibili futuri sviluppi per il progetto:

- introduzione di nuove melodie più piacevoli per la sveglia.
- introduzione di una nuova funzionalità che permetta l'uso della stazione anche come cronometro.
- introduzione di una nuova funzionalità che permetta l'uso della stazione anche come timer.
- implementazione di un sensore che rilevi la pressione (barometro) e di una funzionalità che permetta di vedere il suo stato attuale.
- introduzione di effetti sonori, durante le attività di settaggio degli strumenti.
- implementazione di un sistema di memorizzazione che scriva i dati meteorologici (temperatura, umidità e pressione) in una scheda sd. Permettendo di creare degli storici/report sui dati rilevati.

5 Rilascio materiale

Tutto il materiale di questo progetto è rilasciato il licenza open source ed è reperibile al seguente sito: *https://github.com/AndreaGonzato/WeatherStation*

In questa pagina è possibile trovare anche la vecchia versione del codice non strutturato ad oggetti. Tale versione, chiamata *"Code No OOP"*, non supporta le ultime funzionalità come ad esempio la sveglia; ma può rilevarsi utile per chi intende contribuire a questo progetto ma non è molto famigliare con la programmazione ad oggetti.

6 Conclusioni

Sicuramente il progetto non è ne perfetto ne completo, ma nel suo complesso si può definire un prototipo funzionante.

7 Sitografia

Le librerie utilizzate sono reperibili ai seguenti siti:

- https://github.com/DeanIsMe/SevSeg

- https://github.com/adafruit/Adafruit-GFX-Library
- https://github.com/adafruit/Adafruit_SSD1306
- https://github.com/adafruit/DHT-sensor-library
- https://github.com/adafruit/Adafruit_Sensor

Altri siti dove sono state lette guide e tutorial sono:

8 Dichiarazione finale

Tutto il lavoro è stato svolto da me in completa autonomia.

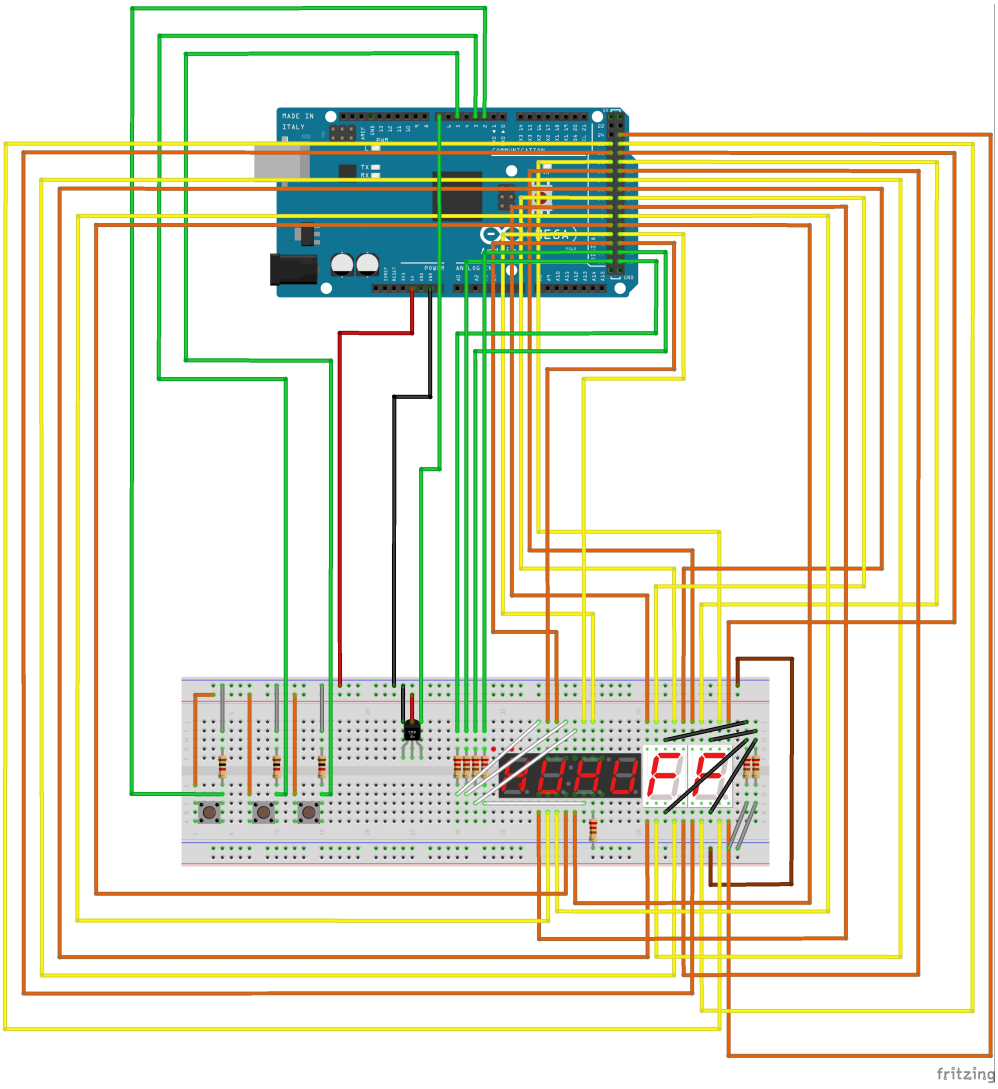

Figura 3: Schema del circuito