Progetto PROS:
relazione di progetto

Alessia Libertucci, Michele Zenoni

27 settembre 2022

1 Introduzione

L'obiettivo del progetto è quello di dare un'indicazione sulle principali caratteristiche che determinano la potabilità dell'acqua in modo semplice e intuitivo; questo viene fatto attraverso la realizzazione di un sistema per la misurazione della qualità dell'acqua utilizzando tre diversi tipi di sensori: sensore del pH, sensore per il residuo fisso (anche noto come misura della "durezza" dell'acqua) e sensore della torbidità sfruttando il microcontrollore STM32L053C8T6.
Le misurazioni vengono avviate dall'utente premendo un apposito bottone e il risultato di esse verrà visualizzato sull'ePaper Display (EPD) della board, insieme a una rappresentazione grafica che ne rende più semplice la comprensione.
Inoltre, al fine di determinare correttamente la bontà dell'acqua, viene utilizzato un sistema di supporto alle decisioni la cui logica è stata realizzata tendendo conto della scala del pH[1] e di quella del residuo fisso[2] mostrate in figura 1. Infatti, da esse si può dedurre che il pH è ottimo nella zona verde indicata, così come il residuo fisso risulta buono se è minore di 100 ppm e ottimo se si trova al di sotto di 50 ppm.

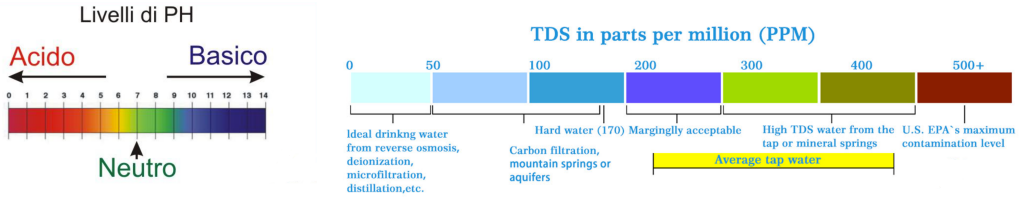


Figura 1: Scale di riferimento per determinare la qualità dell'acqua

Per quanto riguarda la torbidità, invece, essa risulta ottima se ha un valore al di sotto di 1 NTU (*Nephelometric Turbidity Unit*) (concetti approfonditi nelle sezioni seguenti).
La relazione del progetto è cosi strutturata: nella sezione 2 viene descritta l'architettura del software utilizzata, mentre in sezione 3 vi è la presentazione dei componenti fondamentali divisi in tre sottosezioni. Segue poi la sezione 4 contenente le informazioni riguardo i tempi di esecuzione e la potenza dissipata.
I documenti/manuali di riferimento e il codice sorgente utilizzati per lo sviluppo di questo progetto sono disponibili nel *repository* GitHub[3] e citati nei riferimenti bibliografici, le immagini che costituiscono le figure di questo elaborato sono contenute nella cartella 'Images'.

[1] https://www.intech-cr.com/Scala-del-PH-2.htm
[2] http://www.cqrobot.wiki/index.php/TDS_(Total_Dissolved_Solids)_Meter_Sensor_SKU:_CQRSENTDS01#Specifications
[3] https://github.com/Levyathanus/WaterQualitySystem

2 Architettura Software

In figura 2 viene illustrato il comportamento del sistema nei suoi componenti architetturali principali; si noti che il diagramma di sequenza utilizzato non riporta tutte le chiamate di funzione, ma solo quelle utili a capire il funzionamento generale del sistema. In particolare possiamo distinguere tre fasi:

1. inizializzazione: nel diagramma questa prima fase si colloca prima che l'utente interagisca con il sistema tramite il bottone e comprende: l'inizializzazione delle periferiche (in particolare l'EPD al quale, grazie alla libreria, viene reso "trasparente" l'accesso tramite il protocollo *Serial Peripheral Interface* (SPI)) e la transizione in STOP mode, la modalità ad alto risparmio energetico approfondita nella sezione 4;

2. gestione dell'interrupt: questa fase, invece, inizia a seguito dell'interazione con l'utente. Infatti, essa fa riferimento alla gestione dell'interrupt mediante la funzione di *callback* che permette di riportare il sistema in modalità "RUN" e di eseguire il *toggle* del led verde presente sulla board;

3. acquisizione, elaborazione e visualizzazione: questa parte riguarda il blocco più corposo dell'intero sistema, dove si ha l'esecuzione della funzione `Show_Measure` che, anche attraverso le chiamate ad altre funzioni, permette di acquisire i dati, elaborarli e visualizzarli sullo schermo. Maggiori dettagli sono spiegati in sezione 3.

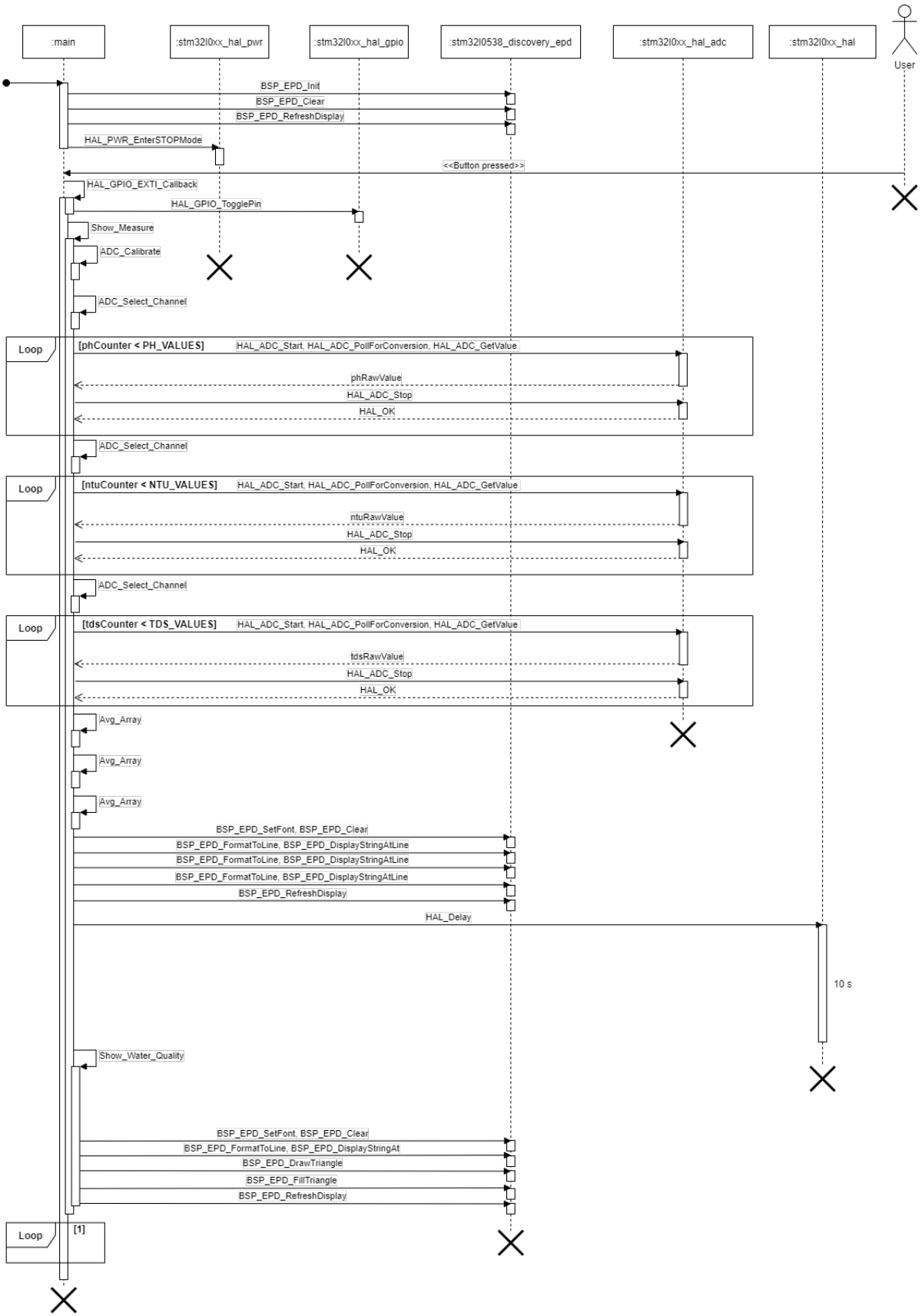


Figura 2: Diagramma di sequenza del codice sorgente

3 Componenti fondamentali

3.1 Acquisizione

Dopo aver segnalato il passaggio da STOP a RUN mode tramite l'accensione del led verde on-board, il sistema si prepara ad acquisire i dati dai tre sensori. In primo luogo vengono disattivati gli interrupt esterni per evitare di saturare la CPU in caso l'utente prema molte volte il bottone per l'inizio della misurazione, dopodiché si procede ad una calibrazione via software dell'*Analog-to-Digital Converter* (ADC) attraverso la funzione dedicata `ADC_Calibrate`, che si assicura che il dispositivo sia pronto per la conversione e inizializzato correttamente. A questo punto si può procedere alla conversione dei segnali analogici in arrivo dai sensori; grazie alla funzione `ADC_Select_Channel` comunichiamo al sistema da quale canale vogliamo campionare e procediamo all'acquisizione in modalità polling; i valori sono ora pronti per essere elaborati.
Nel prossimo paragrafo si descrive nel dettaglio la conversione da "valore grezzo" (*raw value*) a voltaggio e, successivamente, a valore della grandezza fisico/chimica misurata con annessa calibrazione software dei sensori.

3.2 Elaborazione

Per cercare di limitare le imprecisioni dovute a rumore nel segnale o ad errori di conversione, i dati acquisiti dai sensori, dopo essere stati trasformati nei valori misurati della rispettiva grandezza fisico/chimica, vengono salvati in appositi buffer con dimensione definita dalle constanti: `PH_VALUES`, `NTU_VALUES` e `TDS_VALUE` fissate sperimentalmente a 128, rispettivamente per i valori di pH, torbidità e residuo fisso. Una volta acquisiti e convertiti i valori da tutti i sensori si procederà ad effettuare una media aritmetica sui valori contenuti nei buffer (grazie alla funzione `Avg_Array`) per avere una misura più accurata.
Vediamo ora come i dati acquisiti vengono elaborati per raggiungere la misura finale a seconda del sensore utilizzato (l'ordine di presentazione rispecchia l'ordine di acquisizione nel sistema reale):

- **sensore di pH**: una volta acquisito il *raw value* si effettua il calcolo del voltaggio associato tenendo conto, però, che già qui si è deciso di effettuare un passo di calibrazione introducendo il coefficiente moltiplicativo `PH_VOLTAGE_CALIBRATION`. La formula standard, infatti, sarebbe:

  $$Voltage = \frac{rawValue \cdot V_a}{2^{res} - 1}$$

  dove V_a rappresenta la tensione di alimentazione del sensore in Volt (nel nostro caso 5 V per tutti i sensori) e res è la risoluzione dell'ADC in bit (nel nostro caso 12 bit per tutti i sensori).
  Una volta ottenuto il voltaggio si può ricavare il valore della grandezza fisico/chimica misurata attraverso la funzione di conversione caratteristica di ogni sensore. Nel caso del sensore di pH si ha una dipendenza lineare tra $Voltage$ e il valore di pH $pHValue$. Questa relazione è stata determinata sperimentalmente avendo a disposizione delle soluzioni campione con pH noto (7.0 e 4.0) e risulta:

  $$pHValue = -4.4522 \cdot Voltage + 18.73$$

  dove il coefficiente angolare della retta è codificato nella costante `PH_SLOPE`.

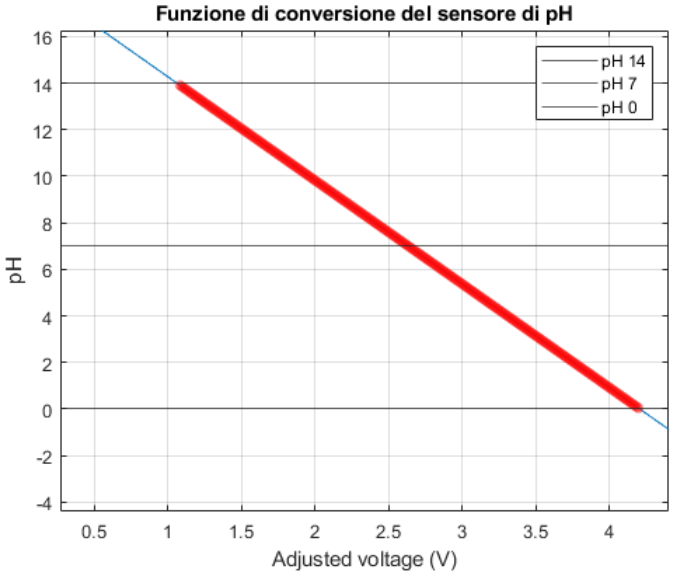


Figura 3: Funzione di conversione del sensore di pH, in rosso il range di validità

- **sensore di torbidità**: analogamente al sensore di pH, partendo dal "valore grezzo" (*rawValue*) si ottiene *Voltage* introducendo, anche in questo caso, un fattore moltiplicativo (costante `TURBIDITY_VOLTAGE_CALIBRATION`), utilizzato per mappare i valori di voltaggio con quelli richiesti dal manuale del sensore per poi applicare invariata la formula di conversione. In questo caso possiamo scomporre la funzione di conversione come una funzione a tratti del tipo:

$$f(x) = \begin{cases} -1120.4x^2 + 5742.3x - 4353.8 & \text{per} \quad x \geq 2.63 \\ 0 & \text{per} \quad x < 2.63 \end{cases}$$

dove x è il voltaggio mappato e $f(x)$ è il valore di torbidità in NTU[4]. NTU sta per *Nephelometric Turbidity Unit* (unità nefelometrica di torbidità) e indica che lo strumento misura la luce diffusa da un campione ad un angolo di 90 gradi rispetto alla luce incidente.

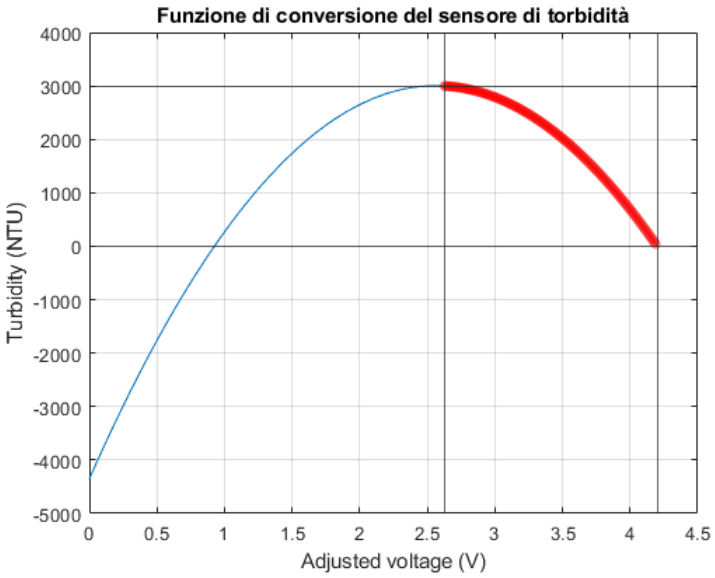


Figura 4: Funzione di conversione del sensore di torbidità, in rosso il range di validità

[4] https://how2electronics.com/diy-turbidity-meter-using-turbidity-sensor-arduino/

- **sensore di residuo fisso (TDS)**: come per gli altri due sensori, anche qui si parte dal "valore grezzo" (*rawValue*) per ottenere il $Voltage$ utilizzando la formula standard. Successivamente viene fatto un passo di calibrazione del voltaggio (in questo caso per compensare la temperatura codificata nella variabile `temperature`) nel modo seguente:

$$compensationVoltage = \frac{Voltage}{1 + 0.02 \cdot (temperature - 25)}$$

Il divisore fa riferimento alla formula di compensazione della temperatura in cui *temperature* rimanda alla temperatura della soluzione, mentre 25°C è la temperatura di riferimento. Il valore 0.02 è il coefficiente di temperatura che viene associato all'acqua dolce o di rubinetto.
Questa formula di compensazione serve in quanto il TDS è direttamente correlato alla conduttività la quale aumenta all'aumentare della temperatura, poiché la prima dipende sia dalla concentrazione di ioni, sia dalla mobilità ionica la quale aumenta di circa il 2% per ogni 1°C.

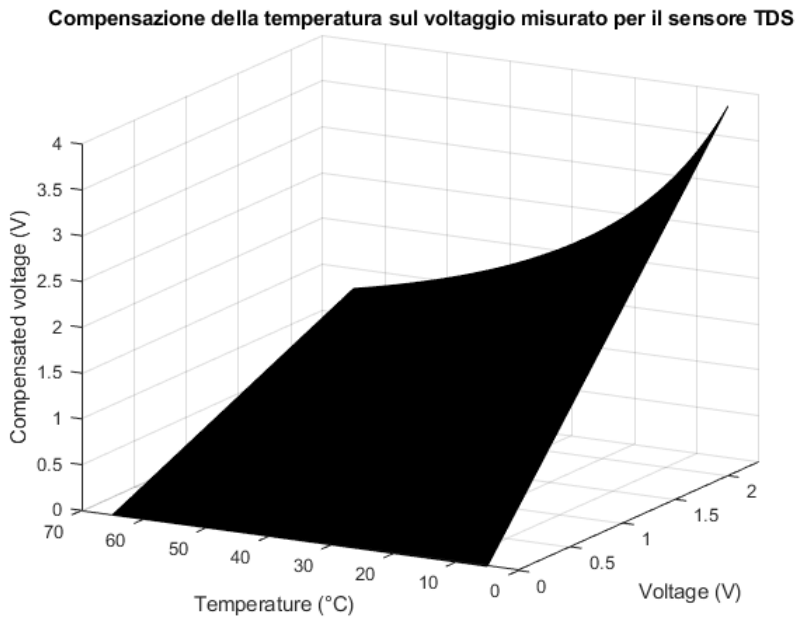


Figura 5: Compensazione della temperatura sul voltaggio misurato per il sensore TDS

A seguire, il valore del $compensationVoltage$ (x) è convertito nel valore del TDS come segue:

$$TDS = k \cdot 0.5 \cdot (133.42x^3 - 255.86x^2 + 857.39x)$$

Questa relazione è stata determinata sperimentalmente e fa riferimento alla documentazione[5] dello specifico sensore. Inoltre, il valore k rappresenta il coefficiente moltiplicativo per la calibrazione del sensore ed è salvato nella costante `TDS_CALIBRATION_COEFFICIENT` equivalente a 0.75.

[5] http://www.cqrobot.wiki/index.php/TDS_(Total_Dissolved_Solids)_Meter_Sensor_SKU:_CQRSENTDS01#Specifications

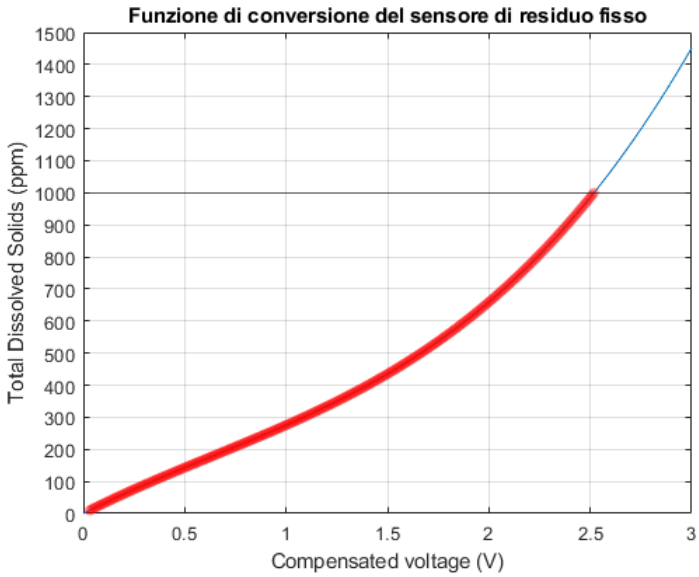


Figura 6: Funzione di conversione del sensore di residuo fisso ($k = 0.75$), in rosso il range di validità

3.2.1 Sistema di decisione della qualità dell'acqua

Dopo aver raccolto e mediato i valori dai corrispondenti sensori, è stata utilizzata la logica mostrata in figura 8 per dare una stima della qualità dell'acqua.
La variabile `qualityPoints` viene utilizzata per memorizzare il punteggio, il quale viene suddiviso come mostrato in figura 7. Al fine di considerare tutti i parametri, si è pensato di assegnare un punteggio massimo a ciascuno di loro nel modo seguente:

- in base al valore della variabile `ntu` (valore di torbidità), è possibile raggiungere un punteggio massimo uguale a 5.0;
- in base al valore della variabile `pH` (valore di pH), è possibile raggiungere un punteggio massimo uguale a 1.5;
- in base al valore della variabile `tds` (valore del residuo fisso), è possibile raggiungere un punteggio massimo uguale a 3.5.

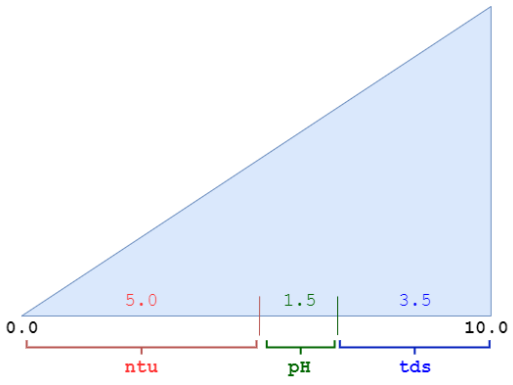


Figura 7: Scala della qualità dell'acqua

Seguendo il diagramma di flusso in figura 8, si può vedere quale valore assume la variabile `qualityPoints` quando i valori dei tre parametri non permettono di ottenere il punteggio massimo specificato precedentemente. In particolare, la stima viene fatta analizzando le tre grandezze (variabili): `pH`, `ntu`, `tds`.

La prima condizione considerata riguarda `pH` e `tds` in quanto risultano essere i valori più importanti per determinare la bontà dell'acqua; infatti, se `pH` è al di fuori del range $[6.5, 8.5]$ o se `tds` è maggiore di 400, allora `qualityPoints = 0` in quanto l'acqua non è potabile.
In caso contrario, si procede prendendo in esame il parametro della torbidità, il quale sarà buono solo se risulta essere minore o uguale a 10 e, in questo caso, la variabile `qualityPoints` aumenterà di 5 punti. In realtà, come riportato nella sezione 1, il valore della torbidità è ottimo quando è strettamente minore di 1, ma dato che il sensore restituisce solamente valori interi e poiché si è stabilito sperimentalmente un offset di accettabilità dovuto alla compensazione del rumore e all'accuratezza del sensore, la condizione da verificare è `ntu <= 10` (e non `ntu == 0`).
Successivamente, si prosegue andando a studiare più dettagliatamente il valore del pH nel seguente modo:

- se $7.0 \leq$ `pH` ≤ 7.5 allora `qualityPoints` aumenterà del valore massimo assegnato al `pH` che è stato definito precedentemente (1.5 punti);

- se `pH` > 7.5 allora, in base al suo determinato valore, ci sarà un decremento di 0.1 punti per ogni punto decimale di `pH` partendo da 1.5. Quindi, ad esempio, se `pH == 7.6` allora `qualityPoints += 1.4` ($1.4 = 1.5 - 0.1 = 7.6 - 7.5$).
  Se si raggiunge questo ramo, `qualityPoints` potrà aumentare massimo di 1.4 punti e minimo di 0.5 perché il `pH` potrà avere valori nell'intervallo $[7.6, 8.5]$;

- se `pH` < 7.0 allora in base al suo determinato valore, ci sarà un decremento di 0.2 punti per ogni punto decimale di `pH` partendo da 1.5. Quindi, ad esempio, se `pH == 6.8` allora `qualityPoints += 1.1` ($1.1 = 1.5 - 0.4 = 1.5 - (7.0 - 6.8) \cdot 2$).
  In questo caso, `qualityPoints` potrà aumentare massimo di 1.3 punti e minimo di 0.5 perché il `pH` potrà avere valori nell'intervallo $[6.5, 6.9]$.

A seguire, si andrà ad analizzare il valore dell'ultimo parametro rimasto: il residuo fisso. Come già anticipato, il punteggio massimo che si potrà raggiungere considerando il `tds` è uguale a 3.5 e ciò avverrà solo se $0 \leq$ `tds` ≤ 100. Negli altri casi si avrà che:

- se $100 <$ `tds` ≤ 170 allora si procede dividendo questo intervallo in 4 gruppi (sottointervalli di uguale ampiezza) e, in base al gruppo in cui si troverà lo specifico valore del residuo fisso, ci sarà un decremento di 0.1 punti partendo da 3.5. Quindi, ad esempio, se `tds == 140` allora farà parte del terzo blocco e di conseguenza `qualityPoints += 3.2`.
  Siccome l'intero intervallo viene suddiviso in 4 sottointervalli, `qualityPoints` potrà aumentare massimo di 3.4 punti e minimo di 3.1;

- se $170 <$ `tds` ≤ 200 allora si procede come nel caso precedente ma dividendo l'intervallo in 3 gruppi.
  Quindi, se `tds == 185` allora `qualityPoints += 2.9` perché il valore del residuo fisso farà parte del secondo raggruppamento.
  Siccome l'intero intervallo viene suddiviso in 3 sottointervalli, `qualityPoints` potrà aumentare massimo di 3.0 punti e minimo di 2.8;

- i due casi rimanenti sono simili tra loro e fanno riferimento alle situazioni in cui $200 <$ `tds` ≤ 300 o $300 <$ `tds` ≤ 400. Infatti, in entrambi i rami, l'intervallo viene diviso in 14 sottogruppi e, in base al gruppo in cui ci si troverà, `qualityPoints` potrà aumentare massimo di 2.7 punti e minimo di 1.4 nel primo caso oppure massimo di 1.3 punti e minimo di 0 nel secondo caso.

3.3 Visualizzazione

In questa fase viene utilizzato lo schermo EPD della board per visualizzare i valori elaborati e il risultato dato dal sistema di decisione della qualità dell'acqua descritto in sezione 3.2.1. Al fine di utilizzare lo schermo, è stata importata nel codice la libreria `stm32l0538_discovery_epd.h` per facilitarne l'interazione.
In figura 9 viene illustrato come appare l'output del programma. La stampa dei valori viene fatta attraverso la funzione `Show_Measure` la quale, dopo aver raccolto le misurazioni dai vari sensori e dopo aver fatto la media, fa uso di alcune funzioni della libreria sopracitata per poter settare lo schermo in

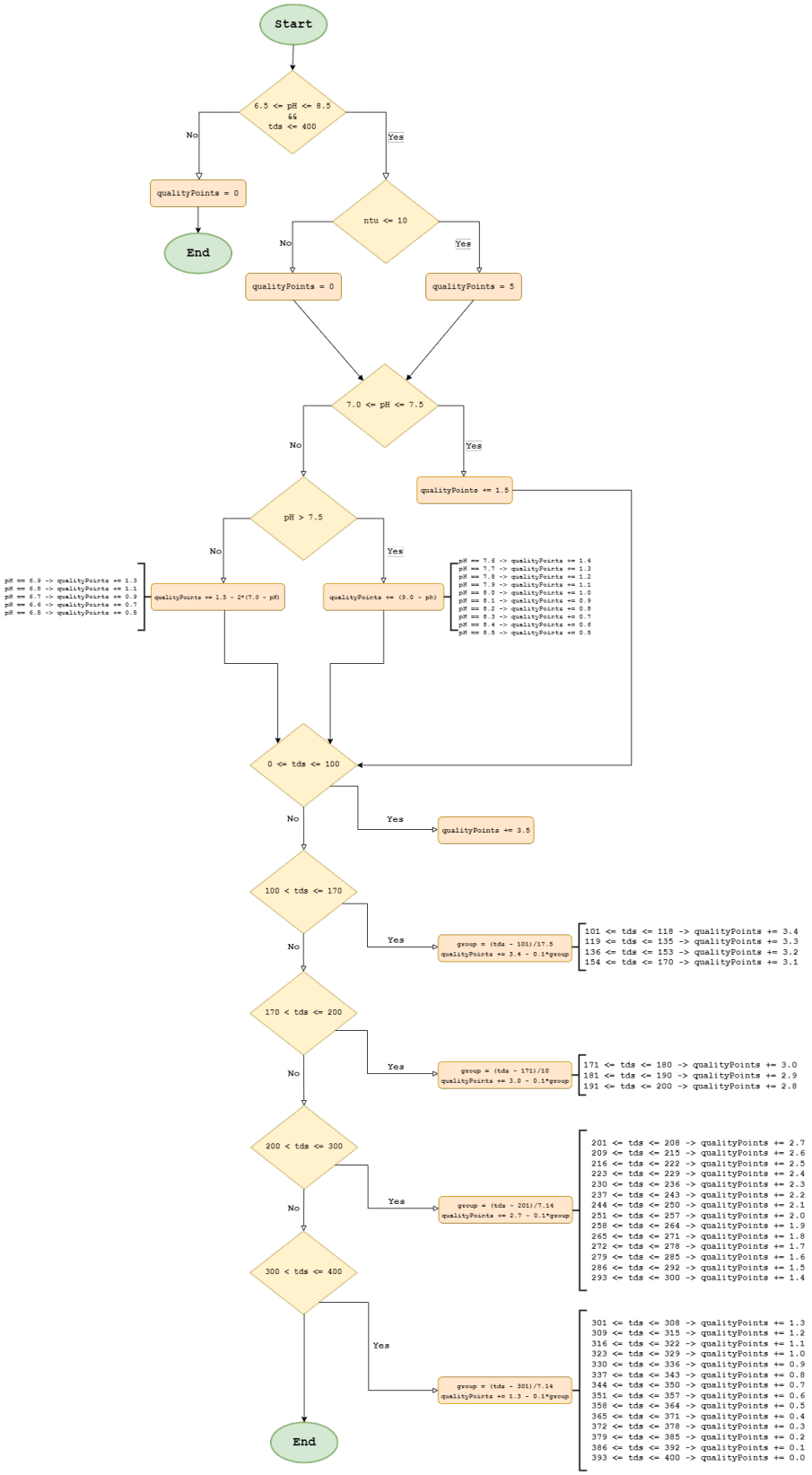


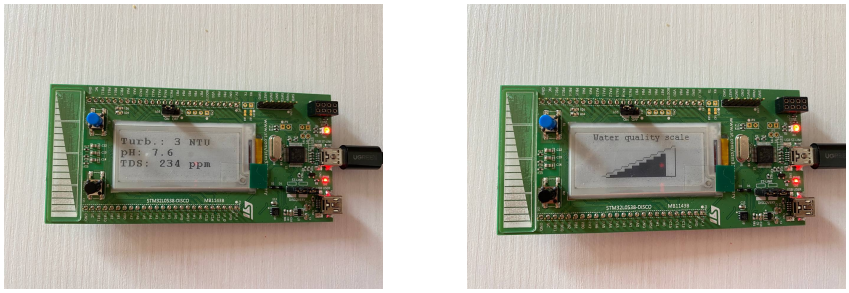
Figura 8: Flow chart per determinare il punteggio di qualità

modo da avere la formattazione mostrata in figura.
Oltre a ciò, come si può vedere, l'esito riguardante la bontà dell'acqua viene descritto intuitivamente attraverso un triangolo rettangolo il cui cateto maggiore rappresenta la scala di valori che va da 0 a 10, dove parte del triangolo viene colorato in base al valore della variabile `qualityPoints` (spiegata in sezione 3.2.1).
Per fare ciò, la libreria `stm32l0538_discovery_epd.h` è stata integrata con alcune funzioni che hanno permesso di disegnare il triangolo e colorarlo: `BSP_EPD_DrawTriangle` e `BSP_EPD_FillTriangle`.

- `BSP_EPD_DrawTriangle`: questa funzione ha 6 argomenti che rappresentano le coordinate dei tre punti utilizzati per disegnare il triangolo. All'interno vengono richiamate le funzioni `BSP_EPD_DrawHLine`, `BSP_EPD_DrawVLine` e `BSP_EPD_DrawLine` che fanno già parte della libreria. `BSP_EPD_DrawHLine` e `BSP_EPD_DrawVLine` permettono rispettivamente di disegnare una linea orizzontale e verticale e prendono in input le coordinate del punto da cui si vuole partire per disegnare la linea e la lunghezza di quest'ultima. `BSP_EPD_DrawLine` permette di disegnare una linea generica tra due punti prendendo in input le coordinate del punto di inizio e di fine.

- `BSP_EPD_FillTriangle`: questa funzione è utilizzata per colorare il triangolo all'interno, in base al valore di `qualityPoints`. Ha come argomenti le coordinate dei vertici dell'ipotenusa del triangolo e il punteggio della qualità dell'acqua. L'idea è quella di disegnare tante linee verticali partendo da ogni punto del cateto maggiore e arrivando fino all'ipotenusa.
  Inoltre, all'inizio di questa funzione è presente un'istruzione che ha il compito di convertire il punteggio in base 100, in quanto il cateto maggiore è stato costruito con lunghezza 100 ma i punti relativi alla qualità dell'acqua che vengono passati alla funzione fanno riferimento a una scala da 0 a 10. Di conseguenza, la conversione serve per poter riportare il risultato sul triangolo e colorarlo correttamente.

Le funzioni ausiliarie appena descritte vengono utilizzate nel componente `Show_Water_Quality` che troviamo in `main.c`. Infatti, `Show_Water_Quality` implementa tutta la logica del sistema di decisione (sezione 3.2.1) ed infine imposta lo schermo in modo da poter visualizzare la qualità dell'acqua correttamente. Inoltre, le due visualizzazioni in figura 9 distano di un tempo `EPD_INFO_TIME` equivalente a 10 s.



(a) (b)

Figura 9: Output del programma

4 Analisi dei tempi di esecuzione e della potenza dissipata

4.1 Scelta delle *power mode*

Per ottimizzare i consumi energetici tenendo conto del contesto in cui il sistema si troverebbe ad operare, sono state fatte delle scelte che sfruttano la principale caratteristica dei microcontrollori STM32 della famiglia L0 con microprocessore ARM Cortex-M0+: la loro capacità di operare ad un livello personalizzabile via software di consumo energetico. Prima di analizzare i consumi mostrando un ipotetico scenario d'uso, è conveniente mostrare l'utilizzo massimo delle risorse hardware del microcontrollore (memorie FLASH e RAM):

Utilizzo delle risorse hardware				
Risorsa	Dimensione	Libera	Utilizzata	Utilizzo
RAM	8 KB	3.74 KB	4.26 KB	53.22%
FLASH	64 KB	6.75 KB	57.25 KB	89.46%

Allo stato attuale il sistema, appena è alimentato, entra in STOP mode dopo aver configurato ed inizializzato le periferiche; questa famiglia di microcontrollori, infatti, dispone di cinque modalità di risparmio energetico: LOW-POWER RUN, SLEEP, LOW-POWER SLEEP, STOP e STANDBY, elencate secondo un ordine decrescente di consumo. La decisione di utilizzare la STOP mode è dovuta al fatto che è la modalità a più basso consumo energetico che mantiene inalterato il contenuto di RAM e FLASH permettendo, quindi, di evitare di ri-inizializzare le periferiche quando si esegue il passaggio in RUN mode. Quest'ultima è la modalità di utilizzo "standard" che, grazie alla tecnologia *Dynamic Voltage Scaling* (DVS) implementata sui microcontrollori della serie STM32L, permette di selezionare via software il livello più adatto di consumo scegliendo da tre scale di voltaggio; per questo sistema la scala 2 (*medium range scale*) è stata valutata la più pertinente in quanto permette di avere una frequenza di clock di 16 MHz riducendo il consumo interno (V_{core}) di circa il 15%.

4.2 Analisi di un ipotetico scenario di utilizzo con stima del tempo di esecuzione

Al fine di condurre un'analisi della potenza dissipata in uno scenario verosimile, si ipotizzi di utilizzare un sistema automatico di generazione degli interrupt al posto del bottone che deve essere premuto manualmente. In particolare, lo scenario proposto ipotizza che il sistema (microcontrollore e sensori) sia costantemente alimentato da una batteria alcalina con voltaggio nominale di 9 V, potenza di 625 mAh e coefficiente di scaricamento pari a 0.3%/mese, senza bisogno di un interruttore manuale di accensione e spegnimento. Un interrupt viene generato ogni 1 h (3600 s) e il tempo di esecuzione è stato misurato essere composto da: 2.5 s per ottenere le misure visualizzate su schermo, 10 s di attesa e circa 1.5 s per la visualizzazione della scala di qualità, per un totale di circa 14 s.
La sequenza campione, analizzata grazie al PCC (*Power Consumption Calculator*) integrato nel STM32CubeIDE, consta di tre passi che rappresentano le fasi principali di esecuzione:

1. STOP mode: come descritto in precedenza in questa modalità il consumo è minimo ed è stimato a 410 nA per 3600 s;

2. Wake up from STOP mode: questo passo (della durata stimata di 4.9 μs) ha un consumo di corrente pari a 1 mA dovuto alla riattivazione del clock (HSI) a 16 MHz;

3. RUN mode: durante l'effettivo utilizzo della potenza di calcolo del microprocessore il consumo totale si alza fino a 51.36 mA suddivisi in: 2.36 mA legati al microcontrollore (di cui 257 μA per le linee di: ADC, GPIOA, GPIOB e SPI), fino a 40 mA per il sensore di torbidità[6], $\sim$ 3 mA per il sensore di pH[7] e fino a 6 mA (3 - 6 mA) per il sensore di residuo fisso[8].

Questi risultati sono riassunti automaticamente dal PCC come in figura 10.
In questo scenario la durata della batteria sopracitata è stimata a 4 mesi, 7 giorni e 6 ore.

[6] https://wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189
[7] https://files.atlas-scientific.com/Gravity-pH-datasheet.pdf
[8] https://wiki.dfrobot.com/Gravity__Analog_TDS_Sensor___Meter_For_Arduino_SKU__SEN0244

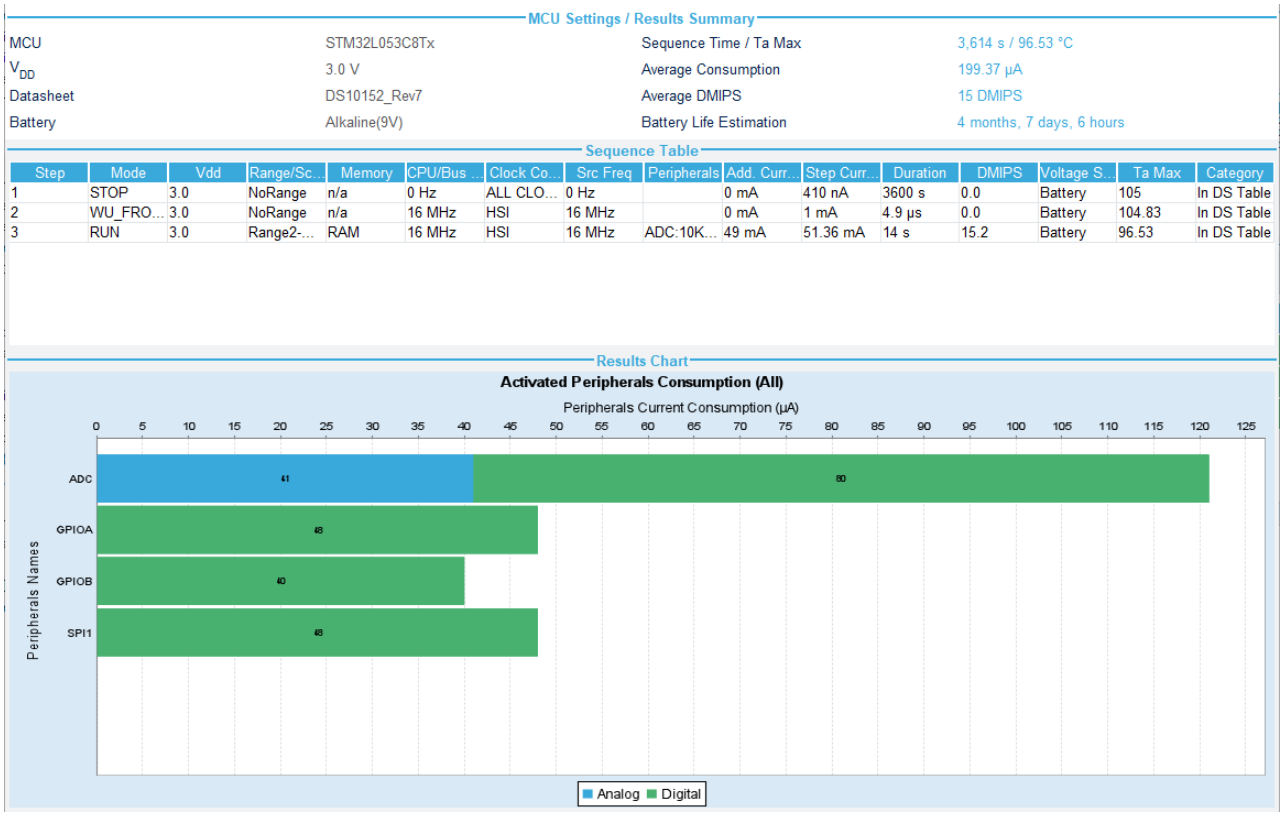


Figura 10: Schermata riassuntiva del consumo energetico

Riferimenti bibliografici

[Noviello, 2017] Noviello, C. (2017). *Mastering STM32*. Leanpub.

[STMicroelectronics, 2014] STMicroelectronics (2014). *AN4500 Application note*. STMicroelectronics. How to display size-optimized pictures on a 4-grey level E-Paper from STM32 embedded memory.

[STMicroelectronics, 2019] STMicroelectronics (2019). *STM32L053C6 STM32L053C8 STM32L053R6 STM32L053R8*. STMicroelectronics. Ultra-low-power 32-bit MCU Arm-based Cortex-M0+, up to 64KB Flash, 8KB SRAM, 2KB EEPROM, LCD, USB, ADC, DAC.

[STMicroelectronics, 2021] STMicroelectronics (2021). *STM32L05xxx/L06xxx device errata*. STMicroelectronics.

[STMicroelectronics, 2022a] STMicroelectronics (2022a). *RM0367 Reference manual*. STMicroelectronics. Ultra-low-power STM32L0x3 advanced Arm-based 32-bit MCUs.

[STMicroelectronics, 2022b] STMicroelectronics (2022b). *UM1775 User manual*. STMicroelectronics. Discovery kit with STM32L053C8 MCU.