

Simone Balloccu | [ESIT] | A.A.: 2017-2018

[Agrumino Flash]

[MEMORY MANAGEMENT PER SMART GARDENING]

Obbiettivo del progetto

Lo scopo del progetto è quello di consentire la memorizzazione di dati e

configurazioni sulla memoria flash del device Agrumino, facendo in modo che i dati

provenienti dai sensori siano memorizzati dopo ogni ora e ritrasmessi ogni 4 ore. Si

devono inoltre consentire il salvataggio sulla flash di eventuali configurazioni

ricevute.

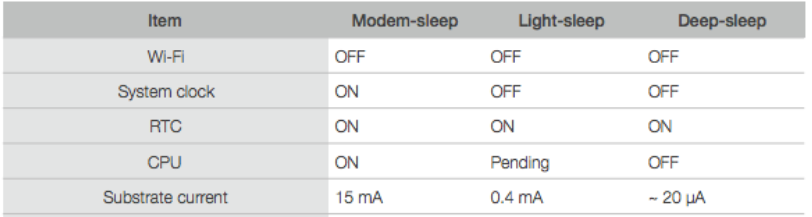
I requisti di dettaglio sulla funzionalità implementato e/o sulla architettura

HW/SW verranno concordati con i docenti su proposta degli studenti.

Contesto, problematiche e scelte progettuali (HW/SW)

La board di riferimento su cui è stato costruito Agrumino è l’ESP8266 (con l’aggiunta di sensori), vista durante il corso di ESIT. Nel contesto dello sviluppo del progetto AgruminoFlash sono state affrontate le seguenti problematiche con le conseguenti scelte progettuali:

* Utilizzo della memoria FLASH: inizialmente è stata effettuata una ricerca per verificare l’esistenza di progetti simili. Si è scoperto che Arduino dispone di   
  una libreria dedita all’utilizzo della memoria EEPROM, e che l’ESP8266 emula  
  questo tipo di memoria proprio tramite FLASH. Di conseguenza è stato possibile  
  utilizzare la libreria senza necessità di porting.
* Strutturazione dei dati: la FLASH dell’ESP8266 è configurata come una sequenza   
  lineare di 4096 registri da 1B completamente general purpose. Non sono quindi   
  fornite metodologie per gestire i diversi tipi di dati. Si è deciso di implementare all’interno della libreria Agrumino un set di metodi atti a lettura e scrittura di quattro tipi di dati fondamentali: int, float, char, bool. Non si è quindi deciso di implementare nessun identificatore per il dato, in quanto avrebbe portato ad un maggiore utilizzo di memoria, né di implementare strutture atte alla memorizzazione di dati “personalizzati” (per esempio strutture) in quanto i dati della sensoristica ricadono sempre nei quattro tipi citati precedentemente, e così si è potuto appurare anche in base ad una ricerca per i vari sensori disponibili per lo smart gardening. Inoltre questa garantisce una gestione semplificata di lettura, scrittura, e gestione della memoria, ed evita all’utente di dover fare controlli a basso livello: la conseguenza è che la fase di coding si focalizza sull’utilizzo della FLASH e non sulla gestione della stessa.
* Tipi di scrittura: si è inoltre deciso di permettere la scrittura sequenziale o a specifici indirizzi indicati dall’utente. Questo permette di utilizzare la libreria in modo più libero e di riorganizzare blocchi omogenei di dati di un certo tipo in blocchi eterogenei; inoltre fornisce all’utente un modo per sfruttare più liberamente la memoria.
* Limiti dei valori dei registri: essendo i registri della memoria da 1B ciascuno, essi sono in grado di contenere solo valori compresi tra 0 e 255. Dopo l’analisi del problema si è concluso che questo limite non è un problema per tre dei quattro tipi di dato: infatti gli interi della sensoristica sono utilizzati solo per rappresentare percentuali, ed essendo il valore 100 disponibile il problema non si è posto; per char il problema non esiste in partenza, in quanto lo standard ASCII non copre 255 caratteri nemmeno in versione estesa; infine per i booleani bastano due valori.  
  Il problema si è posto invece per i tipi di dati float, per cui 1B non è sufficiente: si è deciso quindi di rappresentare il dato tramite una struttura d’unione che contiene il valore in 4B, permettendo di recuperare sempre l’intero valore. Questa scelta ha portato anche a diverse politiche di gestione dei dati, che vengono spiegate nel punto successivo.

* RESET dovuto al DeepSleep: una delle caratteristiche che rende la ESP8266 particolarmente adatta all’Internet Of Things è la sua capacità di esercitare un notevole risparmio energetico grazie alla disponibilità di diverse modalità di “sleep”. La più efficiente è la Deep Sleep in cui la board viene di fatto disattivata, portando il suo consumo ad un valore di circa 20 microAmpere, come mostrato nella seguente tabella riassuntiva:  
    
    
   Tabella riassuntiva dei consumi della ESP8266 nelle varie modalità di sleep

Per risvegliare la board è però necessario l’equivalente del reset della board, operazione che porta a far ripartire lo sketch dall’inizio, perdendo quindi eventuali variabili globali. Questo problema è stato risolto mediante l’utilizzo dei registri dedicati DIRTY e START\_ADDRESS, il cui utilizzo viene spiegato nel punto successivo.

* Gestione dei registri e registri riservati: una volta inseriti i metodi necessari a scrittura e lettura dei dati si sono resi necessari altri metodi per la gestione dello spazio e delle varie informazioni di memoria. Per questo sono stati inseriti i metodi per ripulire i registri inutilizzati (che discriminano il tipo di dato in base alla loro memory size, ossia 1B o 4B), e per la verifica della disponibilità di un particolare registro. Sono inoltre stati riservati alcuni B per la creazione di alcuni registri “riservati”, alcuni dei quali non accessibili all’utente, che conservano informazioni sulla memoria. I registri sono i seguenti:  
   - DIRTY: questo registro contiene un booleano, che comunica all’utente se la  
   memoria dell’Agrumino è “sporca”, ossia se contiene dati non ancora  
   pushati sul server.  
    
   - LASTFREEADD: “last free address”, indica il numero dell’ultimo registro non  
   utilizzato.  
    
   - FREE\_MEMORY: indica la memoria libera in byte  
    
   - START\_ADDRESS: questo registro si è reso necessario a causa del reset che  
   viene eseguito al risveglio dal Deep Sleep. Esso indica in  
   quale registro inizi il blocco di dati che per l’utente è di  
   interesse leggere, e pertanto va gestito e aggiornato   
   manualmente.  
   - HOURS: questo registro contiene il numero di ore passate dall’ultimo push  
   dei dati eseguito. Permette quindi di utilizzare una qualsiasi variabile  
   definita dentro lo sketch della board per eseguire il confronto e capire  
   se bisogna pushare i dati, anche a fronte di un RST.  
    
   - USERSPACE: contiene un valore fisso che indica il primo registro utilizzabile  
   dall’utente. Viene utilizzato per evitare la scrittura in registri   
   riservati.  
    
    
  - MAX\_MEMORY: contiene il valore fisso di 4096MB, ossia la memoria massima del  
   chip, ed è stato introdotto semplicemente per leggibilità del  
   codice.  
    
  Va notato che la validità dei dati contenuti negli indirizzi LASTFREEADD e FREE\_MEMORY viene compromessa se l’utente effettua scritture non sequenziali: infatti questi registri sono pensati per essere utilizzati solo scrivendo blocchi contigui di dati, e il loro utilizzo nel caso questa condizione mancasse sarebbe impossibile

Verifica funzionale del progetto

Per valutare il progetto sono stati prodotti due sketch distinti:

1. AgruminoFlash.ino: questo sketch è stato concepito per dare una breve demo dei  
   metodi disponibili, e la sua esecuzione porta all’inizializzazione della memoria, seguita dalla scrittura/lettura di diversi blocchi omogenei ed eterogenei di dati. In aggiunta è presente la sovrascrittura (scrittura non sequenziale) di blocchi già scritti per dimostrare l’utilizzo di questa funzionalità. L’intero sketch è corredato da un “logger” che mostra come le statistiche di memoria varino durante le operazioni.
2. AgruminoFlashWithSensors.ino: questo secondo sketch è più orientato all’utilizzo di Agrumino e di fatto esegue la routine di task richiesta dal progetto. Prima di tutto viene controllato se la memoria è “dirty”, dopodiché:
   1. Se la memoria è DIRTY e sono passate abbastanza ore dall’ultima lettura dei dati in flash (nell’esempio 4 ore): allora si prosegue all’estrazione di tutti i dati registrati: ciò è fatto utilizzato START\_ADDRESS come registro di shift, semplicemente andando avanti di tanti B quanti sono i dati letti.
   2. Se la memoria è DIRTY e non sono passate abbastanza ore: viene reperito l’ultimo indirizzo disponibile e si scrive un nuovo blocco di dati
   3. Se la memoria non è DIRTY: si veda b.

Al termine di una delle operazioni elencate sopra la board viene messa in Deep Sleep per il tempo specificato (nell’esempio 1 ora). Si noti che il flag DIRTY permette di evitare la re-inizializzazione della memoria al risveglio (che porterebbe alla cancellazione delle informazioni conservate in flash): semplicemente questa viene ri-abilitata per permettere l’accesso.

Istruzioni per riuso e riproduzione:

Il riuso del codice è immediato, in quanto è stata modificata la libreria di Agrumino, senza modifiche da parte del codice già presente: questo significa che si può sostituire la libreria con quella modificata e scrivere gli sketch che utilizzano la flash, e al contempo mantenere funzionanti quelli precedenti. Per quanto riguarda la riproduzione sia gli sketch di DEMO che il codice su GIT è completamente commentato, per cui il codice risulta immediatamente chiaro comprensibile.

Link al repository Github:  
Al seguente link sono disponibili la libreria modificata (inclusa la libreria EEPROM) e gli sketch di esempio sviluppati:

*https://github.com/uccollab/AgruminoFlash*