ESIOT– A.A. 2017/2018

Progetto
Agrumino Configuration by AT Commands

Relazione Conclusiva

Obiettivi del progetto

L'obiettivo principale del progetto é stato la realizzazione di un interprete per il dispositivo Agrumino, che consenta la configurazione dello stesso attraverso comandi AT (ad esempio ricevere e interpretare comandi per restituire informazioni come dati dei sensori presenti o per effettuare settaggi). Durante le fasi conclusive del progetto si è resa evidente la necessità di realizzare anche un'applicazione, lato client, che, aprendo un canale di comunicazione con lo stesso, permettesse di inviare al dispositivo una serie di comandi AT, contenuti all'interno di un file, consentendo una modalità semplificata per l'invio dei dati ed un notevole risparmio di tempo in caso di operazioni ripetute.

Contesto e problematiche da affrontare

Durante la realizzazione del progetto sono state evidenziate alcune problematiche relative all'utilizzo di un dispositivo quale è Agrumino e delle ridotte risorse disponibili, soprattutto per quanto attiene al consumo di energia, alla persistenza dei dati di configurazione ed alla necessità della board di dover passare la maggior parte del tempo in modalità deep sleep.

Principalmente per quanto riguarda la realizzazione del software non ci sono stati grossi problemi in quanto l'interprete non richiede grandi capacità di calcolo e l'unica preoccupazione è stata quella di trovare il modo di non tenere occupata la scheda nel polling per un tempo indefinito quando non necessario. L'obiettivo è stato raggiunto mediante la temporizzazione della funzione di polling, che allo scadere dell'intervallo impostato, o se viene meno il collegamento della board con la porta USB, restituisce il controllo al ciclo principale.

Un ulteriore problema, che abbiamo dovuto affrontare, è stato quello di riuscire a capire perché la board, in alcune situazioni non chiare, presentava un problema legato al trasferimento del firmware, che abbiamo risolto tramite un software disponibile per Esp8266 che, tramite una piccola interfaccia

terminale permetteva di capire se la board fosse stata rilevata e in quale fase fosse (es. board rilevata, ack ricevuto, inizio scrittura e fine scrittura).

Il software utilizzato e presente nel seguente link di github.
https://github.com/nodemcu/nodemcu-flasher

Scelte progettuali

Hardware
L'hardware utilizzato, la board Agrumino Lemon, versione non ancora definitiva, presenta attualmente qualche problema di alimentazione e, soprattutto, un problema ricorrente in fase di trasferimento del firmware.
Sicuramente sulla versione definitiva tali problemi saranno risolti, ma, fino ad allora, per far ripartire la scheda, dopo un periodo di inutilizzo o in caso di scaricamento della batteria ricaricabile al litio, è stato necessario utilizzare quattro batterie stilo nell'apposito "4xAA Battery Holder Box", in dotazione nel kit node-mcu utilizzato a lezione.
In relazione al frequente errore durante il trasferimento del firmware, dovuto, come ci è stato detto, all'errato valore di una resistenza sulla board, è stato necessario l'utilizzo del software per windows "ESP8266Flasher.exe", che si è dimostrato in grado di rilevare correttamente la scheda a seguito della pressione del tasto di reset sulla stessa, immediatamente dopo aver avviato la procedura di trasferimento. Naturalmente il firmware da trasferire è stato ottenuto mediante la procedura di esportazione di Arduino IDE.
Lato pc, data la non ingente quantità di risorse hardware richieste, sono stati usati dei normalissimi notebook ASUS dotati di sistema operativo Linux (distro Ubuntu 16.04).

Software
Come detto al paragrafo precedente, per la realizzazione dell'interprete dei comandi AT, si è operato utilizzando Arduino IDE versione 1.8.5, mentre per la realizzazione del programma di comunicazione, scritto in C, si è utilizzato un semplice editor di testo (gedit).
Si è operato quasi esclusivamente in ambiente Linux, tranne quando si è dovuto procedere al recupero della board con il programma "ESP8266Flasher.exe" di cui purtroppo non si è trovata una versione Linux.
Dal punto di vista dell'implementazione, il progetto è stato portato a compimento con la realizzazione di una libreria per Arduino IDE in cui sono contenuti, oltre ai due file AgruminoAt.h ed AgruminoAt.cpp con cui è stata implementata la classe AgruminoAt, due esempi di sketch basati su di essa.
È stata preferita, come forma, la libreria per la facilità con cui possono essere integrate in qualsiasi sketch.
Per facilitare sviluppi futuri, si è operato in modo da poter aggiungere e/o modificare i comandi AT e le funzioni da essi richiamate in maniera completamente indipendente dall'interprete.
Per ottenere tale risultato:

- sono state definite:
    - ROW_LENGHT 255 (lunghezza massima della linea di comandi AT)
    - HELP_LENGHT 128 (lunghezza massima help di un comando)
    - COMMAND_LENGHT 15 (lunghezza massima di un singolo comando AT)

    - COMMANDS_NUM 26 (numero dei comandi attualmente implementati)

- le funzioni, richiamate dai comandi, hanno tutte la stessa intestazione ed i puntatori ad esse sono memorizzati in un array:
  `void (*command_function [COMMANDS_NUM]) (char* , AgruminoAt*)`

- le stringhe contenenti i comandi sono memorizzate anche esse in un array nello stesso ordine delle funzioni:
  `[char commands[COMMANDS_NUM][COMMAND_LENGHT]`

- le stringhe contenenti l'help sono memorizzate allo stesso modo:
  `char help[COMMANDS_NUM][HELP_LENGHT]`

- è stato definito il tipo struttura `config` che contiene tutti i dati (tempo di deep sleep, wifi ssid, wifi password et.) che devono essere memorizzati nella flash e richiamati ad ogni ciclo (dopo il deep sleep)

I comandi implementati allo stato attuale sono solo un piccolo insieme scelto, innanzitutto, per offrire la possibilità di leggere i valori dei sensori già presenti sulla board, settarne i parametri di base per il corretto funzionamento e fornire il supporto agli sketch di esempio compresi nella libreria.

Inoltre si è scelto di inserire una copia dei file di definizione della classe AgruminoAt nelle cartelle degli esempi, in modo che fossero aperti dall'Ide insieme agli sketch dando la possibilità di modificarli, se necessario.

La procedura per l'inserimento di un nuovo comando AT può essere riassunta come segue:

- su AgruminoAt.h:
    - modifica `+1` del valore definito per `COMMANDS_NUM` che diventa `27`
    - aggiunta della dichiarazione della nuova funzione in coda alle esistenti nella sezione definita come `// Command methods -` che sarà:
      ```
      static void AT_funz26(char* param, AgruminoAt* agruminoAt);
      ```

- su AgruminoAt.cpp:
    - aggiunta delle stringhe relative al comando ed all'help in coda alla sezione definita come `//Preparazione tabelle comandi ed help`
      ```
      strcpy(&commands[26][0], "AT+NEWCMD");
      strcpy(&help[26][0], "New AT command");
      command_function[26] = AT_funz26;
      ```

    - aggiunta del codice, relativo alla funzione da eseguire in risposta al nuovo comando AT, in coda alla sezione definita come `// Command methods //`
      ```
      /**
         AT+NEWCMD
      */
      void AgruminoAt::AT_funz26(char* param, AgruminoAt* agruminoAt) {
        //insert new code here
      }
      ```

      Si noti che alla funzione vengono passati:

- la stringa param contenente i parametri necessari all'esecuzione del comando AT e della cui interpretazione deve farsi carico la funzione stessa;
- l'attuale istanza di AgruminoAt, tramite cui è possibile accedere ai valori della structure config e all'istanza attuale di Agrumino (valori di tutti i sensori e parametri).

Risultati ottenuti e Considerazione finali

L'interprete dei comandi implementato, disponibile sotto forma di libreria (AgruminoAt.h) consente di avere con la board Agrumino un'interfaccia a riga di comando, all'interno della quale è possibile eseguire dei comandi per visualizzare lo stato dei sensori, per attivare o disattivare delle funzionalità e per impostare i parametri per la connessione WIFI o quelli di un server MQTT, ma soprattutto, è possibile salvare questi parametri sulla flash e recuperarli all'inizio di ogni ciclo, dopo che la procedura di deep sleep, necessaria per il risparmio energetico, ha riportato tutti i valori a quelli di default, come impostati nella sezione setup dello sketch. Tale possibilità evita soprattutto di dover intervenire sullo sketch manualmente e trasferire nuovamente il firmware dal pc alla board ogni volta che avviene una modifica dei parametri.
Inoltre, cosa non da poco, l'interprete non incide sui consumi energetici poiché può essere attivato solo quando la board è collegata alla USB e quindi alimentata.
Altre funzionalità ottenute, a nostro avviso molto importanti, sono la semplicità con cui si possono integrare nuovi comandi e la possibilità di inviare alla board serie di comandi memorizzati su file.

Riuso del codice

Per l'integrazione dell'interprete AT in qualsiasi applicazione basata sulla libreria Agrumino è sufficiente l'inserimento nella sessa delle righe in rosso nel seguente esempio, che è basato sullo sketch di default della board Agrumino Lemon:

`#include <Agrumino.h>`
`#include "AgruminoAt.h"`
`Agrumino agrumino;`
`AgruminoAt interpreter;`
`config* st;`
`void setup() {`
`  Serial.begin(115200);`
`  agrumino.setup();`
`  agrumino.turnBoardOn();`
`  interpreter.setup_interpreter(agrumino);`
`  st = interpreter.get_config();`
`}`
	Include la libreria
	
	Dichiara l'oggetto di classe AgruminoAt
	Dichiara il puntatore alla struttura che contiene la configurazione
	
	esegue il setup dell'interprete
	recupera la configurazione salvata nella flash

`void loop() {`
`  if (!agrumino.isBoardOn()) {`
`  agrumino.turnBoardOn();`
`  }`
 	
`  boolean isButtonPressed = agrumino.isButtonPressed();`
`  /*`
`  boolean isAttachedToUSB = agrumino.isAttachedToUSB();`
`  boolean isBatteryCharging = agrumino.isBatteryCharging();`
`  */`	

`  if (agrumino.isAttachedToUSB()) {`
 	
`  Serial.println("Press a key within 2 sec. to go to AT interpreter");`
`  delay(2000);`
`  if (Serial.available()) {`
`  interpreter.atInterpreter();`
`  }`
 	
`  }`
	Se Agrumino è collegata alla porta USB apre la finestra di due secondi in cui la stessa attende un input dalla seriale: se vengono rilevati caratteri in input il controllo passa all'interprete, che lo restituirà in seguito all'immissione del condo AT+Q o alla disconnessione dalla porta USB.

`  if (isButtonPressed) {`
`  agrumino.turnWateringOn();`
`  delay(2000);`
`  agrumino.turnWateringOff();`
`  }`
 	
`  blinkLed();`
 	
`  agrumino.turnBoardOff(); // Board off before delay/sleep to save battery :)`	

`  if (st->sleep_mode == 'd') {`
 	
`delaySec(interpreter.get_config()->sleep_time_sec);`
	Se nella configurazione è impostato il valore 'd' per lo sleep mode, il sistema

`// The ESP8266 stays powered, executes the loop repeatedly`
`  } else {`
 	
`agrumino.deepSleepSec(interpreter.get_config()->sleep_time_sec); // ESP8266 enter in deepSleep and after the selected time starts back from setup() and then loop()`
`  }`
`}`
 	
`//////////////////////`
`// Utility methods //`
`//////////////////////`
 	
`void blinkLed() {`
`  agrumino.turnLedOn();`
`  delay(200);`
`  agrumino.turnLedOff();`
`}`
 	
`void delaySec(int sec) {`
`  delay (sec * 1000);`
`}`
	rimane acceso ed esegue esclusivamente un delay, altrimenti va in deep sleep per ottimizzare al massimo l'utilizzo delle risorse energetiche.

La nuova applicazione dovrà essere scritta in modo che utilizzi i valori contenuti della struttura "config" per connettersi a servizi (wifi, MQTT) o per decidere il proprio comportamento (deep sleep, wifi on/off).

Come illustrato nella documentazione, per l'automatizzazione di operazioni ripetitive quali la configurazione dei parametri per l'effettuazione di test su più board, è stato predisposto il client **Agrumino_AT_serial_cmd**, realizzato in C per linux e basato sulla libreria "termios.h", il cui codice, per poter svolgere la semplice funzione di invio di un file di comandi alla board, non dovrebbe aver bisogno di modifiche tranne nel caso in cui nello sviluppo di una nuova applicazione non si vadano a modificare la stringa di apertura della finestra di attesa input da parte dell'interprete (`"Press a key within 2 sec. to go to AT interpreter"`) o l'output del logo di Agrumino all'avvio della scheda.