Agrumino Remote Firmware Programming

1. Obiettivo del progetto

Lo scopo del progetto è consentire il caricamento wireless di sketch sul device Agrumino, in modo che sia possibile riprogrammarlo da remoto attraverso la connessione Wi-Fi, in particolare facendo uso di metodi OTA (Over The Air).

Nelle specifiche concordate è stato suggerito inoltre di effettuare l'attivazione delle funzionalità OTA attraverso la pressione dell'user button nel device agrumino per una certa quantità di tempo.

Ci siamo preposti di mettere a disposizione le seguenti funzioni per la gestione delle funzioni OTA, le quali verranno spiegate nel dettaglio successivamente:

- httpUpdate – Download diretto da un server http del nuovo sketch.
- ideUpdate – Gestisce la connessione fra l'ide e agrumino mediante socket.
- webServer – Consente di hostare un web server per caricare lo sketch.
- wifiConnect – Connette la board ad un access point wifi.
- isConnected – Restituisce lo stato della connessione.
- OTAModeStart – Si occupa delle operazioni che deve eseguire la board prima dell'update.

Il progetto quindi coordinerà l'uso di diverse librerie che predispongono funzionalità OTA adattandole per l'uso sulla board di Agrumino, e rendendo le funzionalità disponibili attraverso l'uso di una libreria apposita.

Abbiamo deciso di implementare diversi tipi di modalità, complicando quindi il processo di ricerca e le varie problematiche che sorgono dai vari tipi di modalità OTA, per aumentare la complessità del progetto.

2. Contesto e problematiche

Organizzazione del progetto

È stato deciso di creare una libreria wrapper che semplificherà e adatterà le funzioni già implementate per l'ESP8266. Il wrapper verrà messo a disposizione in una repository github e le librerie dipendenti verranno incluse nel progetto, queste librerie sono inoltre già incluse nella board agrumino per default.

Per la programmazione della libreria è stato usato il metodo pair programming (programmazione in coppia), mentre per la ricerca di risorse, documentazione e librerie, è stato deciso di lavorare separatamente per velocizzare il processo di ricerca confrontando poi i risultati e scegliendo le librerie con più supporto e più recenti.

Le ricerche iniziali hanno portato a siti con documentazione carente, ma verso la fine dell'attività progettuale è stata trovata una documentazione esaustiva nel seguente documento:
https://github.com/esp8266/Arduino/blob/master/doc/ota_updates/readme.rst

Al quale abbiamo fatto riferimento per documentare la nostra libreria, e per confermare il comportamento delle librerie che abbiamo deciso di utilizzare, le quali coincidono in gran parte con quelle discusse nel documento.

La maggior parte delle problematiche che abbiamo incontrato erano dovute ai firewall configurati nelle macchine utilizzate per effettuare le prove.

Gestione dello sketch

Gli aggiornamenti OTA consentono di caricare firmware su dispositivi IoT usando la connessione Wi-Fi, uno dei maggiori vantaggi dell'OTA è la possibilità di avere un dispositivo centrale che si occupa di distribuire aggiornamenti a tutti i dispositivi interessati.

È necessario notare una particolarità riguardante il contenuto degli aggiornamenti remoti, per garantire la possibilità di futuri aggiornamenti, i nuovi sketch devono richiedere a loro volta la presenza di un modo per riattivare successivamente le routines OTA.

L'immagine del nuovo sketch verrà memorizzata nello spazio libero del dispositivo, più lontano possibile dallo spazio del vecchio sketch, questo porta quindi all'imposizione di limiti nella dimensione degli sketch da caricare e alla gestione degli errori correlati alla mancanza dello spazio. Ricordiamo che può essere usata la funzione **ESP.getFreeSketchSpace()** per ottenere lo quantità di spazio libero rimanente.

Riavvio

In questo progetto l'hardware utilizzato è la board di Agrumino, i dispositivi OTA devono avere la possibilità di connettersi ad una rete Wi-Fi, essendo basato sul chip open-source ESP8266, Agrumino risulta essere dotato di queste funzionalità. Per ogni aggiornamento avvenuto con

successo sarà necessario il riavvio della dispositivo per completare il caricamento del nuovo sketch, questo dovrebbe avvenire in maniere automatica.

Una grossa limitazione che ha questo progetto è dovuta al fatto che non è momentaneamente presente un metodo per effettuare un reset del chip via software, questo obbliga l'utilizzatore a premere manualmente il tasto di reset dopo l'upload dello sketch. Le librerie utilizzate dovrebbero fare automaticamente un reset ma esso fallisce con il nostro hardware, si tratta di un noto problema di queste librerie con ESP8266, si pensa sia causato dall'upload attraverso porta seriale e che non sia risolvibile direttamente.

Si pensa che il problema sia causato dalla modifica di un registro interno che avviene dopo un upload seriale, e che un reset manuale dopo l'upload seriale e prima dell'attivazione delle routines OTA possa dare la possibilità di far avvenire con successo successivi reset software.

Un'altra possibile soluzione potrebbe richiedere una modifica della board collegando il tasto di reset ad un controllo digitale dell'ESP8266 in modo da mandare il segnale di reset settando attivo il pin usato per il reset.

Sicurezza

Sorgono inoltre problematiche di security e safety, devono essere previste delle metodologie per mettere in uno stato sicuro l'equipaggiamento gestito dal dispositivo durante l'aggiornamento, e cercare in qualche modo di proteggere il dispositivo da tentativi di aggiornamento da parte di terzi.

3. Scelte progettuali svolte

Per implementare le funzionalità OTA e coordinare le varie librerie che forniscono funzionalità OTA, il nostro gruppo ha deciso di scrivere una libreria apposita separata da quella standard di agrumino.

La libreria conterrà anche dei metodi di supporto per connettersi ad una rete Wi-Fi e controllare lo stato della connessione, per facilitare la scrittura degli sketch.

Abbiamo deciso di includere diversi metodi per effettuare l'aggiornamento remoto:

- HTTP Server
- ArduinoIDE
- Web Browser

HTTP Server
La modalità HTTP server consiste in un download diretto dello sketch compilato, l'immagine viene scaricata da un IP o indirizzo accessibile dalla rete Wi-Fi utilizzata.

Il server che fornisce il download ha la possibilità di effettuare ulteriori considerazioni attraverso la richiesta che riceve, analizzando gli header, abbiamo sfruttato questa possibilità data dalla libreria utilizzata, aggiungendo la possibilità di inserire la stringa della versione dello sketch corrente tra i parametri della richiesta come parametro opzionale (verrà usato l'header HTTP_X_ESP8266_VERSION), per far questo abbiamo fatto l'overload dello stesso metodo con un parametro aggiuntivo che permette di specificare la versione.

Il server può quindi utilizzare queste informazioni per fornire l'update solo quando necessario e fornire risposte differenti (ad esempio se la versione comunicata dal dispositivo corrisponde a quella attuale, il server può rispondere che l'update non è necessario, o può fornire update solo a dispositivi con un determinato MAC).

Arduino IDE
La modalità Arduino IDE gestisce gli update utilizzando Arduino IDE o in qualsiasi caso con dei comandi python. In Arduino IDE il dispositivo, se la modalità è stata avviata con successo, dovrebbe apparire con il nominativo Agrumino-[ChipID] nella sezione di menù Tools -> Port -> Network ports, potrebbe essere necessario un riavvio dell'IDE.

Tramite overload, abbiamo fornito una versione del metodo che permette di specificare una password prima di procedere all'upload, un prompt per l'inserimento della password verrà visualizzato nel dispositivo che sta tentando di effettuare l'upload. È necessario specificare che password precedentemente inserite potrebbero essere rivelate dall'IDE se questo non è stato precedentemente chiuso.

Web browser

In questa modalità il dispositivo effettua l'host di un sito web (customizzabile dagli sviluppatori), attraverso questo sito viene gestito l'upload di immagini binarie contenti il nuovo sketch da caricare. Quando la modalità è attivata, viene utilizzato uno dei parametri forniti (la stringa host), per generare un url che porta al sito, in alternativa l'url può essere rimpiazzato dall'ip del dispositivo (l'url viene scritto sul monitor seriale insieme alla versione con l'ip).

Ulteriori problematiche

Alcune di queste modalità (Arduino IDE, Web browser) richiedono l'entrata in un ciclo dove il dispositivo gestisce la ricezione degli aggiornamenti, in questa occasione le normali funzionalità del dispositivo vengono interrotte, abbiamo quindi provveduto un modo per tornare allo stato precedente (con la pressione dell'user button in questa modalità).

Funzionalità di security possono essere implementate lato server nel caso della modalità HTTP Server oppure l'uso di una password nella modalità Arduino IDE. Mentre per quanto riguarda la safety, abbiamo previsto la chiamata del metodo **OTAModeStart()** all'attivazione di qualsiasi modalità OTA, in questa funzione l'equipaggiamento collegato al dispositivo deve essere messo in uno stato sicuro prima dell'esecuzione dell'aggiornamento (essendo fuori dallo scopo del nostro progetto abbiamo solo fatto in modo che l'irrigazione si fermi, come esempio di possibile problematica da considerare).

Abbiamo inoltre previsto il lampeggiamento del led tre volte consecutive per segnalare l'attivazione della modalità scelta.

4. Verifica funzionale

È stato fornito uno sketch d'esempio che ha lo scopo di illustrare il funzionamento dei metodi forniti dalla libreria e può essere utilizzato da scheletro per successivi progetti richiedenti funzionalità OTA.

Lo sketch gestisce vari casi OTA, la modalità viene scelta prima della compilazione dal valore **OTA_TYPE**, il cui valore corrisponde alle seguenti modalità:

1 - HTTP Server

2 - Web Browser

3 - IDE update

4 - IDE update with password

La modalità selzionata viene attivata come da specifica premendo l'user button per una certa quantità di tempo, definita dal valore **pressInterval** (3 secondi per default).

Quando il pulsante risulta premuto per la quantità specificata di tempo, il dispositivo cercherà di connettersi, per un certo numero di volte (definito da **WI_FI_FAILED_TRIES_MAX**), alla rete Wi-Fi configurata, la connessione viene stabilita solo quando è necessario, in modo da conservare batteria.

Una volta verificata la presenza della connessione il dispositivo entrerà nella modalità specificata (dando alternativamente un errore), dichiarandone il nome nel monitor seriale e facendo lampeggiare tre volte il led.

I valori richiesti dallo sketch per connettersi alla rete e gestire la modalità OTA specificata devono essere riempiti prima della compilazione.

Per provare il corretto funzionamento di ogni modalità abbiamo riempito i campi necessari per il funzionamento e abbiamo caricato sul dispositivo lo sketch, abbiamo poi utilizzato le funzionalità OTA per effettuare l'upload successivo dello stesso sketch con un diverso valore **OTA_TYPE**, usando quindi sempre una diversa modalità e verificandone il corretto funzionamento, una volta esauriti i possibili valori per **OTA_TYPE** tutte le modalità risultano funzionanti, specifichiamo che per completare l'aggiornamento OTA è sempre necessario il reset della board.

Per testare la modalità HTTP server è stato necessario utilizzare un semplice server per l'host dell'immagine, è stata recuperata l'immagine di uno sketch compilato (necessaria anche per la modalità Web Server) dalla cartella temporanea utilizzata da Arduino IDE nel processo di compilazione.

Abbiamo inoltre testato il comportamento in casi in cui la connessione è assente o altri casi in cui l'aggiornamento non può andare a buon fine (ad esempio password sbagliata), verificando che i corretti messaggi vengano visualizzati sul monitor seriale.

Tutti i test eseguiti sono andati a buon fine, una volta risolte le problematiche esterne relative ai permessi del firewall.

5. Istruzioni per il riuso autonomo del codice

Il codice è stato commentato e documentato il più possibile ed è stata generata una documentazione HTML utilizzando Doxygen.

I metodi OTA forniti dalla libreria sono static, quindi possono essere chiamati in maniera equivalente usando il nome della classe o un'istanza della classe, a seconda delle preferenze dello sviluppatore:

```
AgruminoOTA agruminoOTA;
agruminoOTA.isConnected(); //Using class instance
AgruminoOTA::isConnected(); //Using class name
```

Per il riuso del codice è necessario importare la board Agrumino su ArduinoIDE (è necessario scaricare esp8266 usando il board manager per la corretta compilazione, usando Agrumino Lemon) e la libreria AgruminoOTA (rispettivamente nelle cartelle hardware e libraries), è necessario inoltre Python 2.7 per effettuare upload OTA con Arduino IDE ed è necessaria particolare attenzione con i settaggi dei firewalls, perché potrebbero impedire l'upload in vari stadi del processo.

Librerie utilizzate

Agrumino.h	Funzionalità della board di Agrumino
ESP8266WiFi.h	Gestisce la connessione alle reti WiFi
ESP8266httpUpdate.h	Aggiornamenti HTTP diretti
ESP8266mDNS.h	Fornisce supporto per risolvere query DNS (usato dal web server)
WiFiUdp.h	Per inviare e ricevere messaggi UDP
ArduinoOTA.h	Aggiornamenti tramite Arduino IDE
ESP8266WebServer.h	Gestisce l'host di un semplice web server
ESP8266HTTPUpdateServer.h	Accetta richieste HTTP post nell'url "/update" url (usato dal web server)

6. Documentazione

◆ httpUpdate() [1/2]

```
void AgruminoOTA::httpUpdate
(
   Agrumino agrumino,
   const char * ota_server,
   int ota_port,
   const char * ota_path,
   const char * ota_version_string
)
```

Description

Direct download of binary file from HTTP server and upload into agrumino board.

Parameters

[in]	**agrumino**	Agrumino instance
[in]	**ota_server**	Server url or ip
[in]	**ota_port**	Server port
[in]	**ota_path**	Path to compiled sketch
[in]	**ota_version_string**	Optional sketch version string, included in request to server as HTTP_X_ESP8266_VERSION header

Requires the board to be connected the a wi-fi network beforehand

◆ httpUpdate() [2/2]

```
void AgruminoOTA::httpUpdate
(
   Agrumino agrumino,
   const char * ota_server,
   int ota_port,
   const char * ota_path
)
```

Description

Direct download of binary file from HTTP server and upload into agrumino board.

Parameters

[in]	**agrumino**	Agrumino instance
[in]	**ota_server**	Server url or ip
[in]	**ota_port**	Server port
[in]	**ota_path**	Path to compiled sketch

Requires the board to be connected the a wi-fi network beforehand

◆ ideUpdate() [1/2]

```
void AgruminoOTA::ideUpdate
(
  Agrumino agrumino
)
```

Description

Handles upload of sketch using Arduino IDE or console.

Parameters

[in]	**agrumino**	Agrumino instance

Requires the board to be connected the a wi-fi network beforehand

◆ ideUpdate() [2/2]

```
void AgruminoOTA::ideUpdate
(
  Agrumino agrumino,
  const char *  password
)
```

Description

Handles upload of sketch using Arduino IDE or console using a password.

Parameters

[in]	**agrumino**	Agrumino instance
[in]	**password**	IDE password

Requires the board to be connected the a wi-fi network beforehand.

◆ isConnected()

boolean AgruminoOTA::isConnected
(
)

Description

Checks if the board is connected to the network.

Returns

Returns result of check

◆ OTAModeStart()

void AgruminoOTA::OTAModeStart
(
  Agrumino agrumino
)

Description

Method called at the start of OTA mode.

Parameters

[in]	**agrumino**	Agrumino instance

Handles safety of connected equipment

◆ webServer()

```
void AgruminoOTA::webServer
(
   Agrumino agrumino,
   const char * host,
   int ota_port,
)
```

Description

The board will host a web server where the upload of a binary file is possible.

Parameters

[in]	**agrumino**	Agrumino instance
[in]	**host**	Decides the url
[in]	**ota_port**	

Requires the board to be connected the a wi-fi network beforehand.

◆ wifiConnect() [1/2]

```
boolean AgruminoOTA::wifiConnect
(
  const char * ssid,
  const char * password
)
```

Description

Agrumino wifi connect.

Parameters

[in]	**ssid**	Name of wifi network
[in]	**password**	Password of wifi network

Returns

Returns true if connected, false if not connected.

◆ wifiConnect() [2/2]

```
boolean AgruminoOTA::wifiConnect
(
  const char * ssid,
  const char * password,
  int max_tries
)
```

Description

Agrumino wifi connect with a specified number of attempts.

Parameters

[in]	**ssid**	Name of wifi network
[in]	**password**	Password of wifi network
[in]	**max_tries**	Maximum number of connection tries

Returns

Returns true if connected, false if not connected