Controllo Accessi laboratorio

* Introduzione, cosa vuol dire monitorare gli accessi

Controllare e conteggiare gli accessi di una determinata area/locale è molto importante per diverse ragioni.

Principalmente quando si mette a disposizione un luogo ad altra gente è fondamentale poter salvaguardare gli oggetti al suo interno, proteggendoli quindi dall’usura dovuta al cattivo utilizzo.

Questo si può attuare sicuramente limitando il numero di persone che possono entrare contemporaneamente all’interno del luogo prescritto.

Successivamente un controllo degli accessi consente anche di tenere traccia dell’affluenza di visitatori per orari e quindi di generare statistiche di ogni tipo: sia relative agli orari più o meno affollati sia al grado di piacimento di quanto esposto.

* Obbiettivi: precisione, scopo, funzionalità

L’obiettivo di questo progetto è quello di proporre una soluzione semplice, ma funzionale e precisa per il monitoraggio degli accessi ad un’area ad uso pubblico avente fino a 10 possibilità di accesso.

Il sistema mira a garantire un alto livello di precisione (minimo livello di errore anche su casi ambigui) seguendo però sempre una filosofia di basso consumo energetico. Questo per consentire un lunga durata anche utilizzando un’alimentazione a batteria o tramite accumulatore.

Tutti i dati raccolti sono riconsultabili tramite un interfaccia utente intuitiva ed alla portata di tutti.

Il sistema in questione è stato studiato appositamente per poter funzionare correttamente in ogni condizione termo-climatica, con ogni livello di luminosità (luce/buio) e con qualunque persona.

* Ruoli e funzionalità\*

Questa è da considerarsi la prima fase vera e propria della parte di progettazione.

Il nostro obiettivo era quello di avere un’unità master ed N unità slave.

Il ruolo dell’unità master è quello di accumulatore, unificatore e correttore dei dati ricevuti degli altri dispositivi nel tempo. Si potrebbe quasi definirla un’unità di controllo principale in grado di fornire quanto richiesto all’utente, presentandolo in modo a lui comprensibile ed intuitivo.

Le unità slave, invece, svolgono un ruolo di acquisizione ed approssimazione dei dati.

Infatti ad esse sono direttamente collegati i sensori di monitoraggio ingressi/uscite.

* Hardware scelto e motivazioni\*

La scelta dell’hardware da utilizzare è stata fatta basandosi su un concetto principale.

Ciò che conserva i dati deve essere un dispositivo con una memoria verosimilmente illimitata, affidabile ed abbastanza potente da poterne consentire la consultazione in qualunque momento.

A tutto questo era inoltre necessario tenere in considerazione la questione del basso consumo energetico.

Per il ruolo di master, infatti, è stata scelta una scheda STM Nucleo F401RE con uno shield WiFi IDW04A1.

Ci siamo basati su una scheda di sviluppo STM in quanto i prodotti che questa marca rilascia sono spesso sviluppati per essere utilizzati sul professionale. Questo ci ha quindi dato una garanzia sia per quanto riguarda l’Hardware stesso che per il supporto in relazione al firmware.

La scheda Nucleo F401RE è una demo board che monta un F401RET6U come processore le cui caratteristiche sono le seguenti:

Core Size: 32bit

CPU Speed: 168MHz

STM32 F 401 RET 6 / Microcontroller 168MHz, 32bit, CORTEX-M4

Flash: 512KB

RAM: 96KB

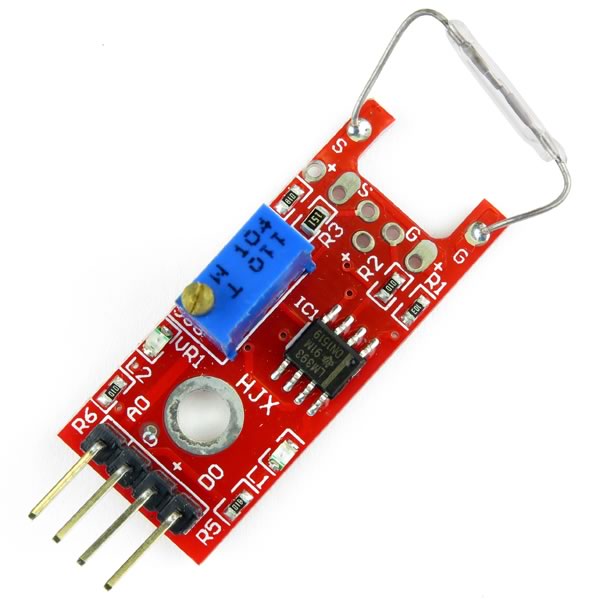
Lo shield IDW04A1 contiene un modulo WiFi SPWF04SA con un firmware di fabbrica aggiornabile tramite seriale o “On Air” (passaggio dell’url).

SPWF04SA è direttamente collegato ad uno slot SD-CARD da noi utilizzato come storage dei dati.

$ Slave modulino esp8266, low tutto, costo ultra ridotto.

Sensori esterni.

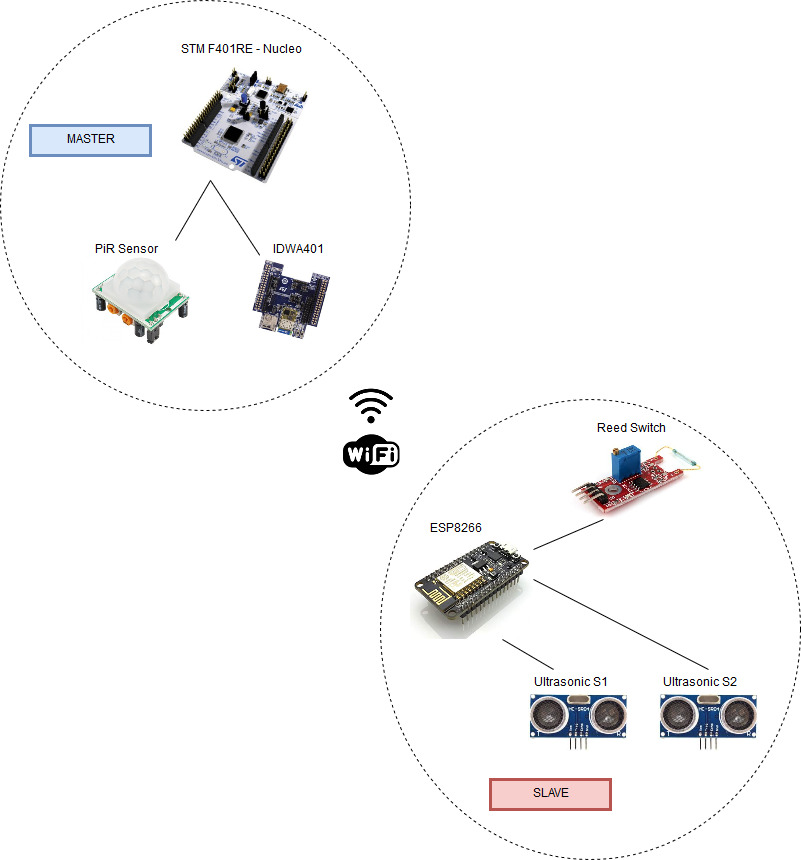
Come sensori esterni sono stati utilizzati moduli ad ultrasuoni HC-SR04, un reed switch magnetico ed un sensore PIR classico.

Il costo di questi sensori è pressoché nullo, ma nonostante ciò la loro precisione è risultata sufficiente per raggiungere gli scopi prefissati.

* Architettura della comunicazione/protocollo

La struttura del sistema risulta quindi essere la seguente.



Come è stato già descritto in precedenza, il sistema è composto da due ruoli principali 1 Master ed N Slave.

L’unita Master comprende la scheda STM Nucleo, lo shield IDW04A1 ed il sensore PIR.

Un’unità Slave invece è composta da una scheda con sopra un ESP8266 (Node MCU nel nostro caso).

La comunicazione tra i diversi slave e l’unità master avviene attraverso una connessione TCP su WiFi 802.11.

Entrambe le unità si aspettano una rete provvista di DHCP e viene utilizzato il multicast DNS per il discovery tra le board. Infatti l’unità master attraverso pacchetti di multicast DNS fornisce alle unità slave tutte le proprie informazioni di connessione.

Ogni comando di un unità slave ha la seguente struttura:

0xFE 1 byte

0xFF 1 byte

0x02 1 byte (\* Identificatore Notifica\*)

Node\_ID 1 byte (\* 0 per il Nodo 1 e 1 per il Nodo 2 …\*)

Code 1 byte (\* Indica quale proprietà è stata modificata \*) [0x01 = Ingresso, 0x02 Uscita]

Valore 1 byte (\* Nuovo valore \*)

In questo pacchetto viene indicato che tipo di comando si sta inviando (notifica == 0x02), l’identificativo del nodo mittente, il codice della proprietà modificata (ingresso o uscita) ed infine la quantità aggiunta o sottratta al valore.

Il fatto che la comunicazione avvenga tramite TCP garantisce la presenza dell’unità Master sull’indirizzo e la porta specificati ed inoltra che il comando sia stato ricevuto con successo (numero di ritrasmissioni di default ed ACK di conferma).

Nel caso in cui per qualche motivo una modifica al valore di una proprietà non venga trasmesso con successo, la trasmissione della modifica successiva includerà anche quella precedente.

Un’altra comunicazione molto importante è quella tra la scheda Nucleo o lo shield WiFi. Il protocollo con il quale questi due componenti comunicano è un set di comandi chiamato “AT”.

Questa tipologia di comandi viene spesso utilizzato via seriale, ma nel nostro caso vi era la possibilità di portarlo anche su SPI (funzionalità che non è però stata utilizzata).

I comandi AT hanno la seguente sintassi

AT+ [Comando] = [Propietà],[Valore]

* Ambienti di sviluppo/librerie\*

Nucleo

Per sviluppare il firmware della nucleo vi era due strade possibili: L’utilizzo di MBed o di strumenti un po’ più a basso livello come uVision Keil oppure Atollic. La scelta è stata però costretta dal fatto che per MBed non vi era nulla da cui partire per lo sviluppo della comunicazione con lo shield IDW.

Le uniche librerie e/o esempi per questo shield erano progetti prodotti per Keil ed Atollic.

E’ stato dunque scelto di utilizzare Keil come IDE. Questo un prodotto Arm che consente di sviluppare, compilare e caricare il codice su microcontrollori in linguaggio C.

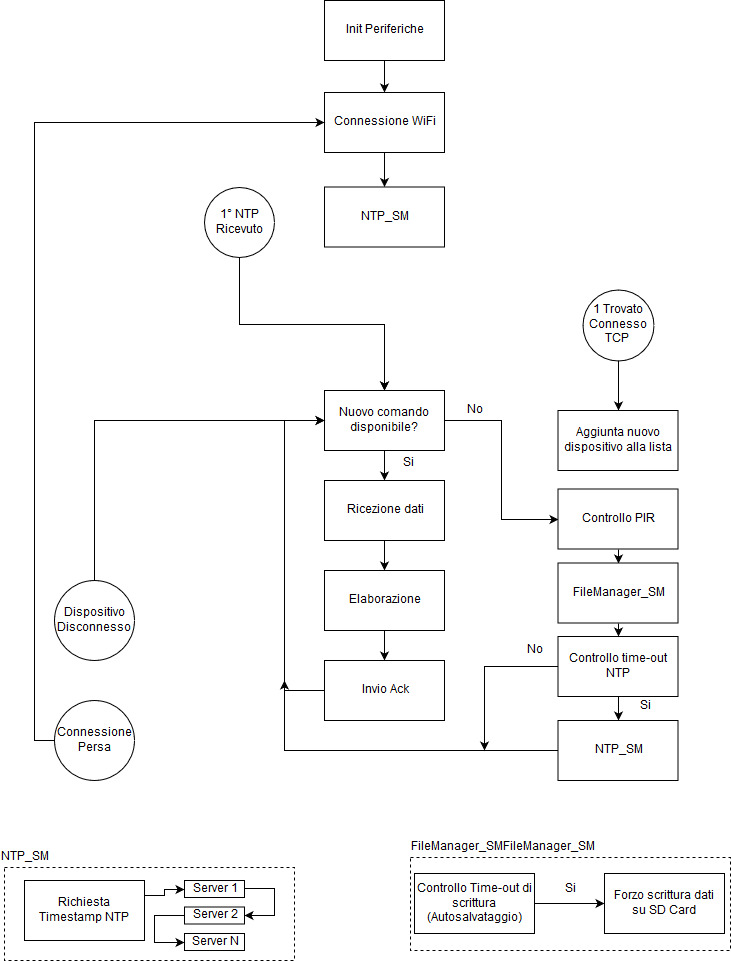
Le librerie contenevano già un parte di protocollo AT di comunicazione con la board IDW, la quale è stata da noi espansa ed integrata con funzionalità delle quali avevamo bisogno.

La gestione dello shield IDW è un po’ simile a quello di un router ovvero va configurato impostando parametri tramite i comandi AT e lui ti invia comandi in caso si verifichino eventi (Esempio connessione stabilita oppure ricevuto pacchetto).

Per lo sviluppo del firmware del microcontrollore non è stato utilizzato nessun RTOS, infatti l’applicativo è single thread e gestisce le priorità attraverso interrupt e timer.

* Descrizione del funzionamento dei singoli dispositivi (macchina a stati)\*

Macchina stati unità Master



La macchina a stati scheda master ha come primo step l’inizializzazione di tutte le periferiche ovvero led, pir, porta seriale, modulo WiFi ed i vari timer.

Successivamente tenta di connettersi alla reta impostata di default. Questo processo va avanti ad oltranza fino a quando o si connette oppure viene riavviata la scheda.

Una volta ottenuto un indirizzo IP viene configurato anche il servizio di multicast DNS inserendo le informazioni di cui le unità slave hanno bisogno per connettersi ed inviare i dati al master (IP e porta).

A seguire viene inviato un pacchetto NTP per la configurazione dell’orologio interno. Anche questo è uno step bloccante in quanto senza avere l’orologio sincronizzato non è possibile elaborare i dati e salvarli in modo corretto.

Le richieste NTP hanno 3 server sui quali fare tentativi, quindi se per qualche motivo uno di essi non è disponibile il firmware proverà con i restanti 2.

Le richieste NTP vengono effettuate ogni 2 minuti circa, questo perché il timer di auto aggiornamento dell’orologio interno ha uno sfasamento di circa 12 ms ogni secondo calcolato.

Una volta che almeno una risposta NTP è stata ricevuta viene sincronizzato l’orologio interno e viene aperto il socket tcp sul quale dovranno poi connettersi le unità slave.

D’ora in poi tutti i dispositivi che si collegheranno al socket prima citato potranno inviare informazioni con la sintassi indicata nel paragrafo precedente.

La macchina a stati prosegue ora iterando sul controllo di ricezione di nuovi comandi.

Nel caso non vi siano comandi da processare, il sistema controlla lo stato del PIR, verifica se è il momento di effettuare un auto-salvataggio dei dati in memoria e se il timer NTP indica che è ora di effettuare nuovamente una richiesta.

Il PIR serve per azzerare gli errori quando il luogo da monitorare è vuoto. Infatti se il PIR non rileva movimenti entro un tempo determinato a priori, il contatore di persone nel locale viene azzerato.

L’auto salvataggio garantisce una minor perdita di dati in caso di caduta di tensione sull’alimentazione.

La scrittura su SD Card prevede l’utilizzo di un buffer non più grande di 1 KB quindi tutti i salvataggi vengono eseguiti ogni volta che il buffer temporaneo raggiunge quella dimensione oppure quando scatta l’auto salvataggio.

I dati vengono salvati in formato JSON su file di testo denominati [Timestamp in seconds].txt ed aventi la seguente struttura.

[

{

“time”:000000000,

“value”:00

},

.

.

.

{

“time”:000000000,

“value”:00

}

]

Tutti i file vengono poi indicizzati in un ulteriore file chiamato db\_index.txt che contiene, separati da una “,” tutti i file salvati nella memoria. Questo, come vedremo successivamente, serve per evitare l’utilizzo di script lato server come php o python.

Nel caso, invece, in cui venga ricevuto un comando, questo viene controllato ed elaborato aggiornando il relativo contatore degli ingressi, infine salvato in memoria per poi essere successivamente scritto, quando sarà il momento, su SD Card.

Agli slave è la sciata la possibilità di connettersi e disconnettersi dal tcp in qualunque momento.

Se invece si verifica una disconnessione dall’access point WiFi, la board si preoccuperà di effettuare nuovamente la connessione quando questa tornerà ad essere disponibile.

* Strumenti di debug\*

Un primo struemento di debug è appunto l’ST-LINKG debugger integrato nella board, il quale consente di utilizzare i break point nel codice sorgente ed avere una visione completa della memoria del microcontrollore in tempo reale.

Questo è probabilmente lo strumento di debug più comodo ed affidabile in assoluto perché consente di bloccare il processo in qualunque momento e verificare lo stato delle variabili del programma.

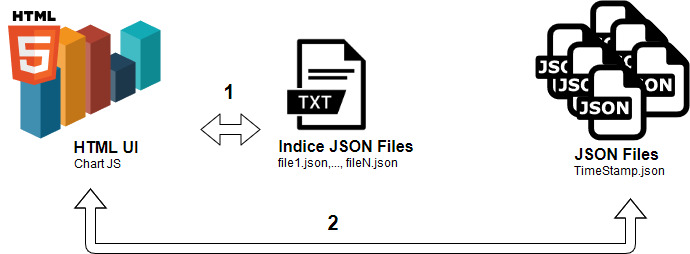
In supporto molto utile usufruire delle funzionalità di un oscilloscopio attraverso il quale abbiamo misurato con precisione la discrepanza tra il nostro timer locale ed il reale scorrere del tempo.

* Interfaccia utente e tracciamento dei dati

L’interfaccia è stata sviluppata in formato html in modo da poterla inserire all’interno dell’SD Card dello shield IDW. Questo infatti svolge un ulteriore funzione, ovvero quella di web server. Nonostante il supporto integrato di micro-python si è scelto di non utilizzare nessuno script lato server per non appesantire il microprocessore dello shield.

Al fine di consentire di eseguire query sui dati si è scelto di implementare un mini database basato su file JSON.

La struttura dell’applicativo è la seguente.



Inizialmente all’interno della UI viene richiesto di selezionare un intervallo di tempo che ha come limite superiore il giorno e l’ora corrente. Successivamente viene analizzato il file degli indici ed effettuata una prima query di selezione sul nome dei file.

Dato un intervallo chiuso [A,B] vengono tenuti in considerazione tutti i file il cui nome, ovvero timestamp, è incluso nell’intervallo più il primo file disponibile antecedente all’intervallo prescritto.

Tutti i file JSON hanno come dimensione 1 KB, il che vuol dire circa 40 coppie timestamp-numero di persone. Questa scelta è stata fatta per alleggerire le query e filtrare quindi molti dati semplicemente dal nome del file stesso.

La seconda selezione dei dati consiste nell’apertura di tutti i file prima selezionati ed a seguire l’eliminazione di tutti i dati al di fuori dell’intervallo.

L’intervallo viene poi diviso in N (variabile a seconda delle preferenze) sottointervalli e vengono calcolate le medie pesate per ognuno di essi. Esattamente a metà di ogni sottointervallo di tempo viene disegnato il dato medio relativo alle persone presenti all’interno della stanza.

Per il plot dei dati è stata utilizzata la libreria Chart JS ed è scegliendo la visualizzazione ad istogramma verticale. Le letture dei file vengono effettuate utilizzando JQuery.

* Test effettuati\*
* Consumi energetici

MASTER

STM Nucleo F401RE + Shield IDW04A1 = 110 mA (5V) ≈ 550 mW

PIR Sensor = 9.3 mA (5V) ≈ 46.5 mW

Totale = 596.5 mW

SLAVE

Esp8266 = 90 mA (5V) ≈ 450 mW

2 \* HC-SR04 (ultrasonic) = 2 \*15 mA (5V) ≈ 2 \* 75 mW = 150 mW

Reed Switch = pochi uA ≈ 0 mW

Totale = 600 mW

Come si può notare, i valori calcolati di consumo energetico risultano perfettamente in linea con la politica a basso consumo adottata.

* Problemi riscontrati\*
* Risultati ottenuti\*

\* = Gimmy scrivi la tua parte