Laboratorio parte Hardware

Laboratorio 1)

1. Il primo esercizio è stato eseguito seguendo le consegne e non ha particolari punti di interesse.

2. Questo esercizio è stato svolto in modo simile al primo con l'aggiunta della lettura da seriale nel loop(); abbiamo inoltre aggiunto un controllo sul carattere di escape "\n" per evitare che anche il tasto di invio venisse letto dalla funzione dando errore di comando non valido.

3. In questo esercizio ci siamo serviti della funzione ISR per contare gli eventi segnati dal sensore PIR che veniva chiamata ad ogni cambiamento di stato; il led è stato inizializzato a valore LOW così che alla prima chiamata della ISR (PIR che manda un segnale alto) si accenda e si spenga quando la funzione viene chiamata di nuovo, ovvero con PIR basso; abbiamo dovuto però modificare la sensitività del sensore PIR perché non registrava quasi nessun evento.

4. In questo script abbiamo svolto le richieste facendo attenzione a fare dei giusti controlli sulla velocità che avrebbe avuto il motore in seguito all'incremento/decremento; abbiamo poi fatto anche un controllo sui comandi non validi e sul carattere di escape "\n".

5. Dopo aver letto da pin analogico il valore della resistenza del sensore abbiamo applicato la formula da cui derivare la temperatura per poi stamparne il valore attraverso la porta seriale.

6. Per la scrittura su display abbiamo pensato di scrivere subito dopo l'inizializzazione e sempre nel setup() le parti che non cambiavano della riga di testo ("temperature: ") e in seguito abbiamo spostato il cursore subito dopo quanto scritto per aggiornare solo il valore della temperatura ad ogni ciclo del loop().

Laboratorio 2)

1. In questo lungo sketch abbiamo svolto tutti i punti compreso quello bonus inizialmente partendo con un approccio "seriale", svolgendo tutti i punti uno per uno, ma in seguito abbiamo rimesso mano a tutto il codice per snellirlo attraverso una visione più generale del progetto (per esempio usando una sola funzione per il riscaldamento e il raffreddamento sia che venisse chiamato con i valori standard, sia con il flag della presenza di persone attivo, sia con i dati inseriti da utente).

Particolari problemi sono sorti con la gestione dei due sensori PIR e di rumore che volevamo inserire in due ISR differenti; ci siamo accorti però che l'unico modo per fare questo era collegare uno dei due segnali alla porta 1 digitale, che tuttavia sarebbe dovuta essere preferibilmente lasciarla libera: questo ci ha portati alla conclusione di dover utilizzare una funzione apposita e una chiamata nel loop per uno dei due sensori.
Un altro problema si è rivelato l'utilizzo di un timer, poiché ne erano necessari diversi e TimerOne dava problemi con il reset: abbiamo risolto scrivendo delle funzioni che contassero attraverso sottrazione di millis() quanto tempo fosse passato dall'ultimo evento e un vettore di dimensione 10 in cui si vanno a segnare il numero di eventi degli ultimi dieci minuti per poter controllare facilmente se ce ne siano stati o meno in quel lasso temporale.
Anche l'inserimento da tastiera dei valori ha avuto qualche difficoltà in quanto l'inserimento di valori non numerici creava dei problemi in tutto il resto del codice (in questo caso abbiamo risolto con un controllo per cui tutti i valori inseriti devono essere diversi da zero, ovvero il valore di default per quando vengono inseriti caratteri non numerici, anche se siamo consci del fatto che questo possa limitare l'azione dell'utente non permettendogli di inizializzare uno degli estremi a zero gradi) e senza scartare il valore '\n' mediante una Serial.read() aggiuntiva questo veniva a creare problemi anche in caso di inserzione corretta dei valori.
Infine per quanto riguarda la sezione bonus abbiamo provato empiricamente quanti eventi venissero contati dal sensore di rumore al battito di mani con e senza il rumore di fondo della ventola del sistema di condizionamento e abbiamo considerato validi i valori entro un certo margine dalla media di quelli così ottenuti.

Laboratorio 3)

1. In questo esercizio abbiamo creato il server HTTP da Arduino mettendo a disposizione un led e il sensore di temperatura, mentre tutte le altre richieste non previste sono state reindirizzate

ad un messaggio di errore 404. La conversione in .json è stata svolta mediante funzione di libreria essendo più semplice e non essendo la memoria relativa al codice eccessivamente piena.

2. In questo caso abbiamo usufruito della libreria Process per poter eseguire una curl sul server avviato con cherrypy in vista di effettuare richieste POST alla pagina /log; il valore di ritorno ed eventuali codici errore della curl vengono stampati mediante porta seriale. Per quanto riguarda i file python da modificare, abbiamo aggiunto in entrambi i casi le sezioni relative al metodo POST e esteso la sezione del metodo GET facendo un controllo sull'*uri.

Per memorizzare i valori dati da Arduino era possibile salvarli in una lista di dizionari come variabile globale che veniva aggiornata ad ogni POST dopo le opportune conversioni e ritornata con la GET, oppure si poteva optare per il salvataggio mediante sessioni: il primo metodo ci sembrava più semplice e leggibile, quindi abbiamo scelto quello.

3. In questo sketch abbiamo creato un subscriber al topic */tiot/21/led* a cui devono arrivare dei messaggi in senML che vengono decodificati da un'apposita funzione e dopo un controllo sui valori validi permette l'accensione o spegnimento del led; il publisher sul topic */tiot/21/ temperature*, invece, pubblica periodicamente i valori di temperatura letti dal sensore. Per operare le conversioni in e da json abbiamo usato funzioni di libreria.