Master Degree in Embedded Computing Systems
A.Y. 2016- 2017

Report of RT-Arduino's Project
Real-Time Systems

RT-Arduino. Develop a concurrent application on the Arduino platform using a real-time interface provided by the RETIS Lab for periodic task management.

Student

Giovanni Falzone

Student ID: 464756

Full Professor

Giorgio C. Buttazzo

Delivery date: 04/03/17

Falzone Giovanni, email: falzone.giovanni2@gmail.com

Index

Falzone Giovanni, email: falzone.giovanni2@gmail.com

1 Descrizione

1.1 Progetto

Il progetto riguarda un sistema di gestione automatizzato di una stanza.

Il dispositivo, tramite il suo set di sensori tiene traccia dello stato del sistema, costituito da Temperatura, umidità, luminosità, presenza di Gas e presenza di movimento nella stanza.

Tramite quattro relè controlla l'accensione e lo spegnimento del sistema d'illuminazione, del riscaldamento, dell'allarme, e del sistema di chiusura.

Offre all'utente due diverse interfacce, una tramite Pannello di controllo TouchScreen ed un'altra tramite Interfaccia Web.

Il sistema, tramite i dati prelevati dai sensori, elabora informazioni relative ai casi di allarme quali allarme intrusione e presenza Gas, ed agisce di conseguenza attivando il corrispondente relè associato ad eventi di allarme, un opportuno messaggio verrà visualizzato tramite il Pannello di controllo.

L'utente tramite le due interfacce controlla lo stato del sistema e ne setta vari parametri, quali **temperatura desiderata**, **luminosità minima**, **accensione/spegnimento dell'illuminazione e del riscaldamento, attivazione/disattivazione gestione automatica di riscaldamento ed illuminazione**.

Il Pannello di Controllo introduce la possibilità di **chiudere od aprire la stanza** e di visualizzare tramite un'opportuna pagina, il log di sistema per visionare eventuali messaggi del sistema.

La chiusura della stanza implica, oltre ad azionare la chiusura elettrica, anche lo spegnimento dell'impianto di illuminazione e del sistema di riscaldamento, per una maggiore sicurezza non verranno accettate richieste provenienti dall'interfaccia Ethernet.

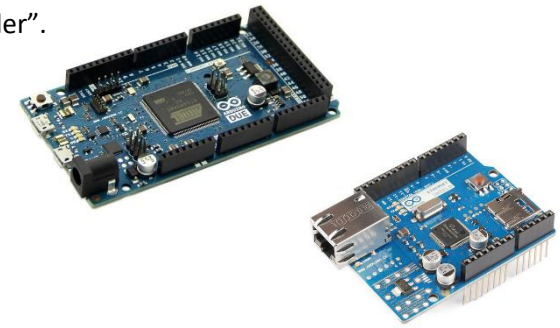
Il sistema provvede al salvataggio automatico dei parametri e dello stato del sistema, e provvede a ripristinarlo ogni qualvolta si avvia il sistema.

1.3 Hardware Utilizzato

Arduino Due

"The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller".

- CPU Clock at 84Mhz.
- 96 KBytes of SRAM.
- 512 KBytes of Flash memory for code.



Ethernet Shield

"The Arduino Ethernet Shield V1 allows an Arduino board to connect to the internet. It is based on the Wiznet W5100 ethernet chip (datasheet). The Wiznet W5100 provides a network (IP) stack capable of both TCP and UDP."

Display Nextion 2.8"

"Nextion offers a new and easy way to interface, help to make GUI easier".

Il display viene programmato separatamente, comunichiamo con il display tramite comunicazione seriale UART, gli oggetti rappresentati vengono identificati per id e nome.

Il display viene pilotato inviando il comando corrispondente all'azione da svolgere, due esempi di seguito.

porta il valore dell'oggetto con nome name a 5: name.val=5

visualizza "hello world" sulla textline tl: tl.txt="hello world"

In quest'ultimo caso le dimensioni della stringa devono essere opportunamente gestite.

DHT22

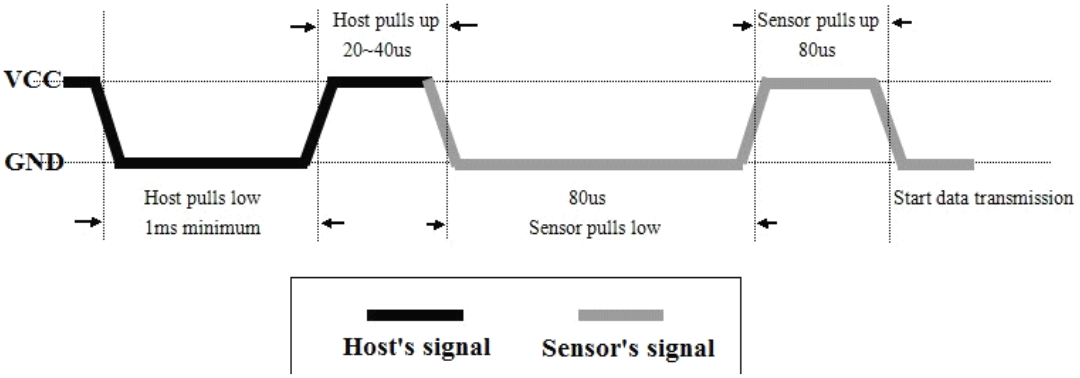
"The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). Its fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old."

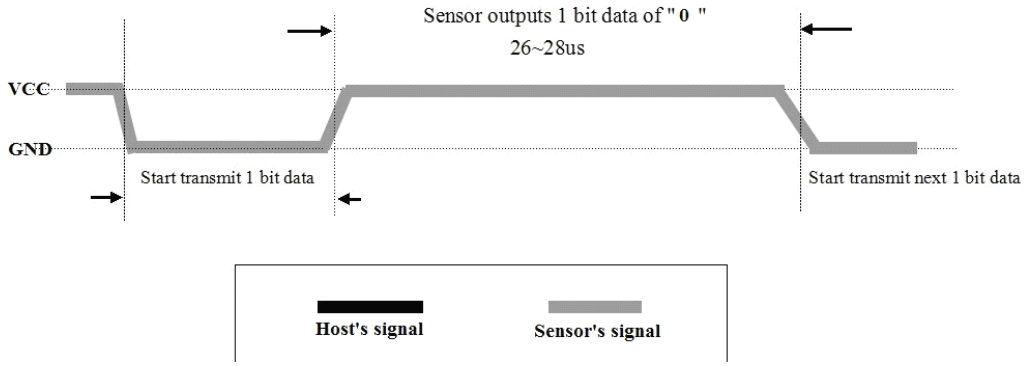
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-100% humidity readings with 2-5% accuracy
- Good for -40 to 80°C temperature readings ±0.5°C accuracy

Visto il tempo minimo di lettura del sensore, il sistema ha un Task per le operazioni che richiedono una frequenza superiore a 0.5 Hz.

Inoltre questo sensore utilizza una comunicazione seriale proprietaria su un singolo pin digitale, la quale ha particolari vincoli temporali di handshake e trasmissione dei bit, di seguito le immagini

descrittive ricavate dai datasheet





Mentre per la trasmissione di un “1” il segnale deve rimanere a livello logico alto per 70us.

Consideriamo il caso peggiore ovvero che vengano trasmessi tutti “1”, il frame da trasmettere è costituito da 16bit per temperatura, 16bit per umidità e 8bit di checksum.

Handshake = 10ms + 40us + 80us + 80us = 10.2ms

PayLoad + checksum + end of trasmission = (16 + 16 + 8) * (70us + 50us) + 80us = 4.8ms

Totale tempo di trasferimento per WorstCase = 14.8ms

La parte di libreria che si occupa della comunicazione con il sensore è stata protetta bloccando lo Scheduler, tuttavia altri tipi di interferenze si possono presentare durante la comunicazione, interferenze dovute ad interrupt sollevati da SPI e UART, in questi casi la parte di libreria torna un errore e scartiamo il dato.

Nel caso in cui la distanza temporale tra due letture sia inferiore ai 2s, viene utilizzato il valore letto precedentemente.

TSL2591

“The TSL2591 luminosity sensor is an advanced digital light sensor, ideal for use in a wide range of light situations.”

- Lux Range: 188 uLux sensitivity, up to 88,000 Lux input measurements.
- Temperature range: -30 to 80 *C
- Interface: I2C

MQ-4

"This is a simple-to-use compressed natural gas (CNG) sensor, suitable for sensing natural gas (composed of mostly Methane [CH4]) concentrations in the air. The MQ-4 can detect natural gas concentrations anywhere from 200 to 10000ppm."

Il valore viene letto tramite un Analog Pin, la Breakout Board dispone inoltre di un pin digitale, il quale viene portato a livello logico basso se il valore letto dal sensore supera una soglia selezionata tramite un potenziometro.

PIR (Passive Infra Red)

"PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out."

Dispone di un pin digitale che viene portato a livello logico alto quando viene rilevato un movimento, il segnale permane a livello alto per un tempo minimo che viene selezionato tramite un potenziometro.

Per calcolare il numero di intrusioni consideriamo la periodicità del Task che effettua le letture e il tempo di durata del segnale, tramite un altro parametro definiremo il numero minimo di intrusioni per generare un evento che il sistema andrà a gestire come allarme intrusione.

Relè Breakout Board

Dispone di 4 pin digitali attivi bassi, uno per pilotare ogni relè opportunamente optoisolato, i relè possono essere alimentati esternamente.

Per evitare condizioni in cui il sistema effettui uno switch-on switch-off dei relè, quando andiamo a valutare le condizioni di soglia relative alle funzionalità automatiche di riscaldamento ed illuminazione, prendiamo in considerazione anche di un errore per la temperatura ed uno per la luminosità, in modo da evitare che piccole oscillazioni si ripercuotano sugli attuatori.

Switch-**off** Riscaldamento:

$$Temp \geq Desired_{temp} + error_{temp}$$

Switch-**on** Riscaldamento:

$$Temp \leq Desired_{temp} - error_{temp}$$

Switch-**off** Illuminazione:

$$Luminosity \geq Desired_{lux} + error_{lux}$$

Switch-**on** Illuminazione:

$$Luminosity \leq Desired_{lux} - error_{lux}$$

2 Organizzazione del sistema

Il sistema è organizzato in tre parti concettualmente separate con compiti separati, le quali interagiscono con il sistema invocandone i metodi, queste sono:

- Una parte relativa al sistema
- Una per l'Interfaccia Control Panel
- Una per l'Interfaccia Web

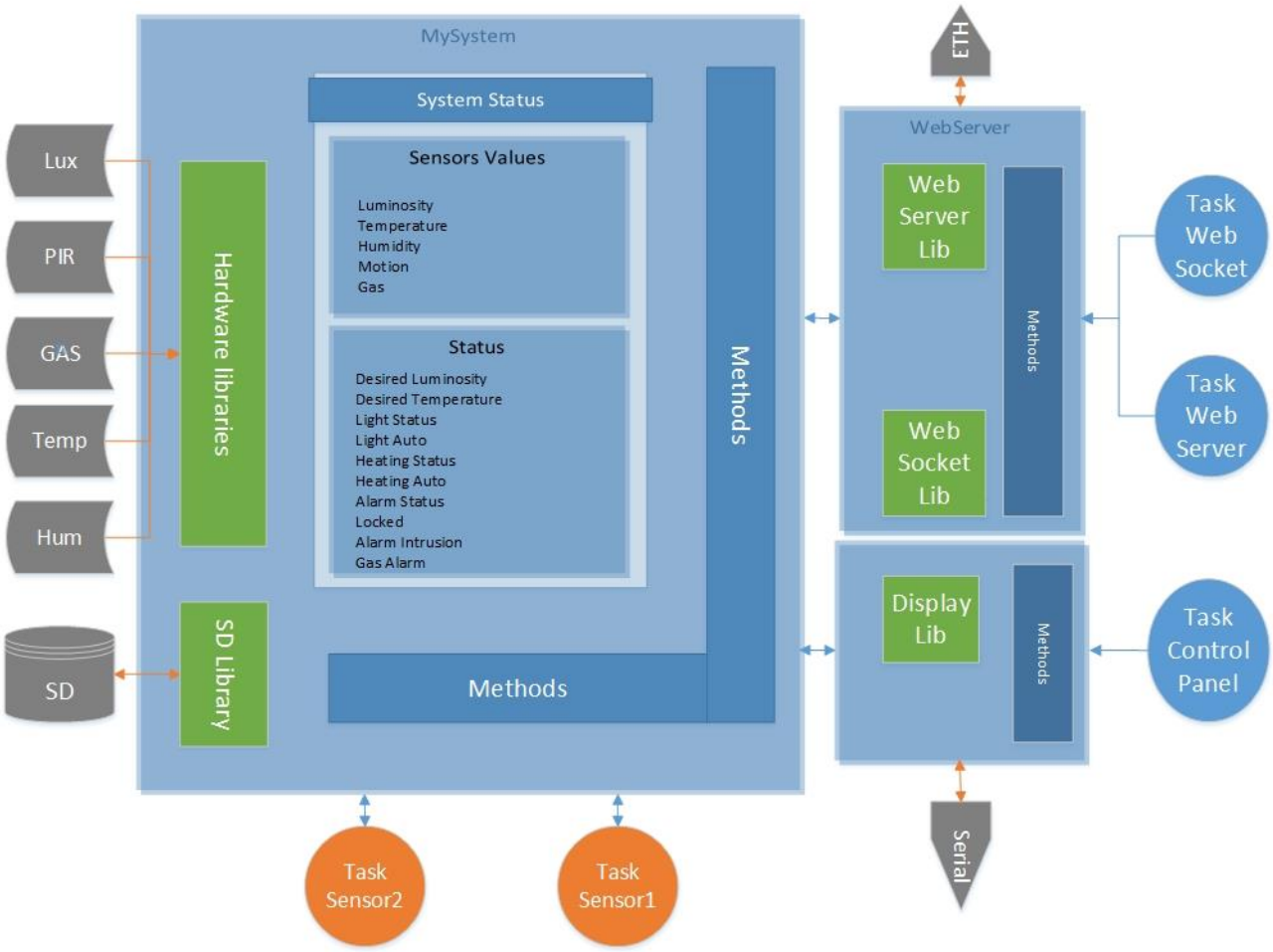


FIGURE 1 PROJECT DESIGN

2.1 Strutture dati condivise

Come dalla precedente figura, lo stato del sistema è costituito da un insieme di informazioni relative ai **dati acquisiti tramite i sensori**, ed una parte relativa alle informazioni riguardanti lo **stato** di accensione/ spegnimento **e modalità delle varie funzioni** svolte dal sistema.

Al fine di interagire con il sistema nel modo corretto, sono stati implementati vari **metodi che permettono di modificarne lo stato**, è necessario proteggere **alcuni di questi metodi come sezioni critiche** per non portare il sistema in uno stato inconsistente, essendo ognuno di questi metodi invocabile da più entità in modo concorrente.

Per garantire l'atomicità di questi metodi, viene bloccato lo Scheduler tramite le funzioni offerte da ARTe, ArteLock() ed ArteUnlock().

Prendiamo come esempio la situazione in cui l'attivazione del sistema di illuminazione non avvenga in modo atomico, e una richiesta per mandare il sistema in lock, da parte di un Task a priorità maggiore, si presenta durante questo metodo, in questo caso lo Scheduler manda in esecuzione il task a priorità maggiore, questo mette in stato di lock il sistema e in seguito il secondo task riprende l'esecuzione attivando il sistema d'illuminazione, cosa non prevista dalle specifiche. (Entrare in lock implica lo spegnimento del riscaldamento e dell'illuminazione)

2.2 Divisione dei compiti

Vediamo adesso quali sono i compiti che devono essere svolti per quanto riguarda il sistema principale, ovvero sensori, attuatori e funzionalità del sistema:

Come visto nel paragrafo relativo l'hardware, alcune di queste componenti sono "veloci" ed altre "lente", per questo motivo i compiti relativi alla gestione del sistema vengono suddivisi in due Task uno per ogni categoria.

Abbiamo quindi un Task System1, con frequenza di attivazione maggiore, il quale si occupa di:

- Aggiornare il valore di luminosità
- Aggiornare il valore di gas
- Gestione automatica del Riscaldamento
- Gestione automatica dell'Illuminazione
- Controllo presenza Gas

Il Task System2 invece si occupa di:

- Aggiornare il valore di temperatura e umidità (DHT22 lento)
- Aggiornare il valore relativo al movimento (PIR durata del segnale lunga)
- Controllo presenza intrusione
- Salvataggio sistema

Il Task System1 si occupa della gestione automatica del riscaldamento, poiché questa funzionalità dipende dalla scelta dell'utente, evento che può pervenire con una frequenza maggiore rispetto il Task System2.

2.3 Sicurezza del sistema

Per quanto riguarda la sicurezza del sistema, vi è la necessità che i due Task System abbiano una priorità maggiore rispetto gli altri, ed in particolare rispetto al Task WebServer. Supponiamo che un “bad guy” inizi ad inviare http_request opportunamente modificate, se queste http_request hanno una dimensione notevole, il Task relativo al WebServer impiegherebbe un tempo proporzionale alla dimensione.

Per garantire una certa velocità di risposta del sistema, l’interfaccia Web è stata realizzata tramite la coppia WebServer e WebSocket, quest’ultima viene gestita da un task a priorità inferiore rispetto il Task System1, ma superiore rispetto il Task System2, cosa che permetterebbe ad un “bad guy” di manomettere il sistema per quanto riguarda i compiti del Task System2, per garantire una sicurezza del sistema ancora superiore dovremmo limitare i compiti del Task System2 a quei compiti che non gestiscono parametri relativi la sicurezza.

(per distribuire il carico di lavoro, ho comunque lasciato al Task System2 la gestione del sensore di movimento, il controllo di intrusioni ed il salvataggio dei dati di sistema, se il WebSocket non permette l’esecuzione del Task System2 non verrebbero rilevate le intrusioni, né salvato lo stato di System Locked).

3 Interfacce Utente

Per ognuna delle interfacce è stata realizzata un’apposita libreria che interagisce con il Sistema invocandone i metodi ogni qual volta si presenta un evento tramite l’interfaccia.

Gli unici metodi che offrono le due librerie sono relativi alla fase di inizializzazione, l’aggiornamento delle informazioni ed il controllo della presenza di eventi.

Le due librerie, invocano i metodi del sistema per ottenere le informazioni relative allo stato del sistema o per modificarne lo stato.

Ogni libreria mantiene informazioni relative allo stato della propria interfaccia al fine di minimizzare il flusso di dati da inviare nella fase di Update, questo tipo di ottimizzazione non verrà considerata per quanto riguarda il Worst Case.

Il Task che gestisce un’interfaccia, ad ogni job, si preoccupa di aggiornare le informazioni rappresentate, di verificare la presenza di eventi provenienti dall’interfaccia e, per ognuno di questi, provvede ad eseguire il corrispondente metodo di sistema se previsto.

3.1 Control Panel Interface

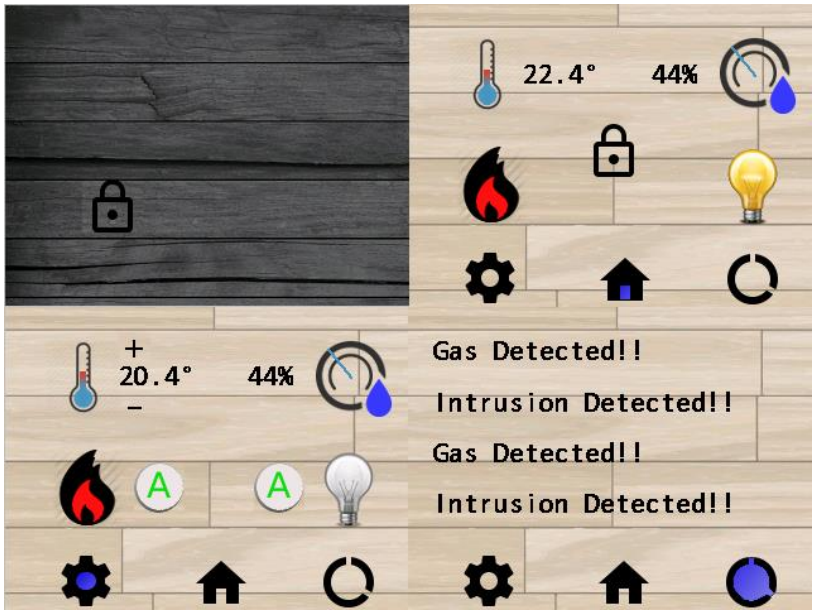


FIGURE 2 CONTROL PANEL'S PAGES

Sono state realizzate quattro diverse pagine:

- Lock Page
- Home Page
- Settings Page
- Log Page

Nella **Lock Page** è possibile attivare il sistema, se questo si trova in stato di lock.

Nella **Home Page**, oltre che portare il sistema in stato di Lock, vengono rappresentati i valori di temperatura e umidità tramite testo e apposita icona, la quale in percentuale rispetto il range definito ne rappresenta l'opportuno stato.

Tramite opportune icone, è possibile visionare o modificare lo stato di accensione/spegnimento di Riscaldamento ed Illuminazione.

Qualora fosse attiva la modalità di funzionamento automatico per Riscaldamento o Illuminazione, la volontà di accendere/spegnere la relativa funzione, viene interpretata come incremento/decremento della soglia di funzionamento automatico.

Nella **Settings Page** viene rappresentata la temperatura desiderata ed è possibile incrementarla o decrementarla tramite opportune icone, le icone di riscaldamento ed illuminazione hanno le stesse funzionalità della Home Page, vi sono due icone aggiuntive per attivare/disattivare lo stato di funzionamento automatico della corrispondente funzione.

La **Log Page** permette di visionare eventuali messaggi del sistema, quali rilevamento intrusione e rilevamento gas, il sistema conserva un massimo di 64 + 4 possibili messaggi non letti.

3.2 Web Interface

Al fine di ottenere un'interattività migliore, l'interfaccia Web è stata realizzata tramite una componente relativa al Web Server, ed un'altra tramite Web Socket che, dato il ridotto flusso di dati da inviare, permette di aumentare la frequenza del Task relativo a quest'ultima.

Risulta dunque necessario avere un Task relativo all'interfaccia WebServer che si occupa di ricevere le richieste http, analizzarle ed inviare la rispettiva HTTP_response e pagina Web, ed un Task relativo al WebSocket, che si occupa di inviare soltanto le informazioni da rappresentare tramite la pagina Web e di ricevere le richieste da parte dell'utente.

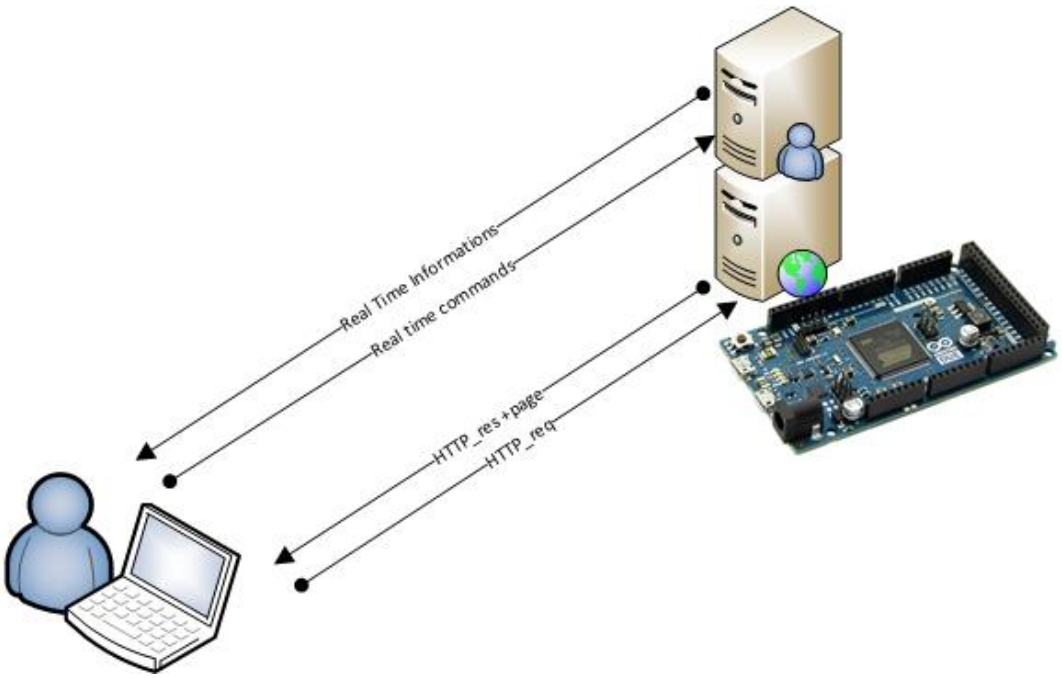


FIGURE 3 WEBSERVER AND WEBSOCKET

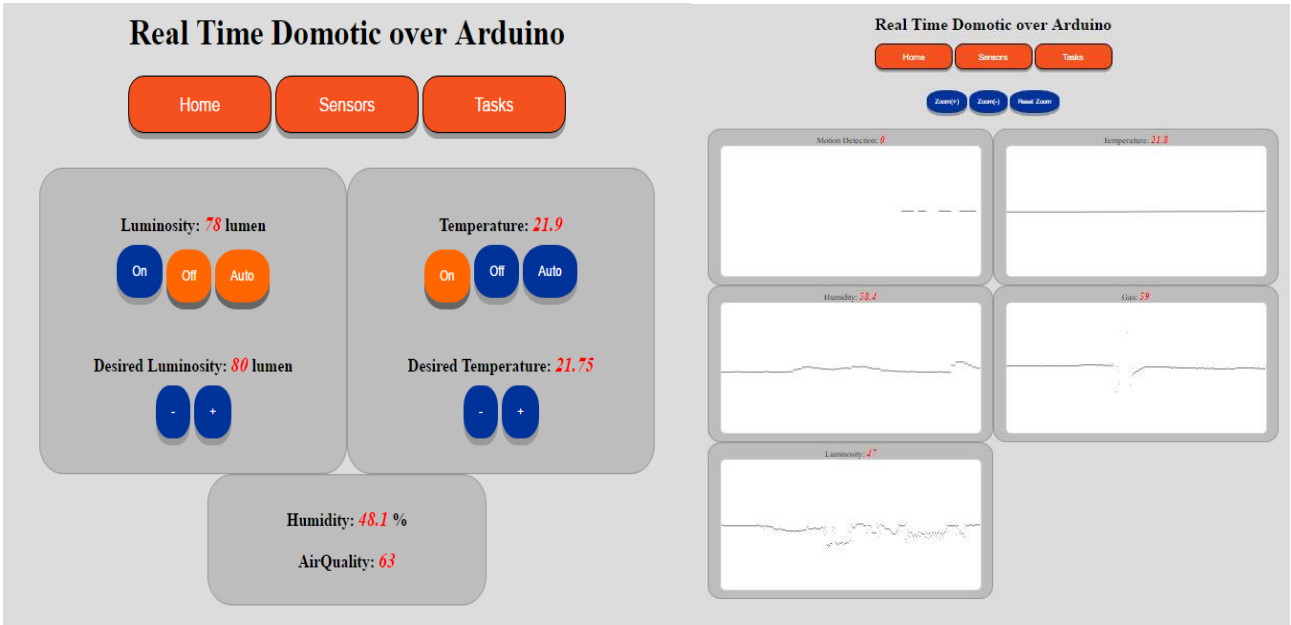


FIGURE 4 WEB INTERFACE'S PAGES

4 Tasks

Al fine di analizzare le caratteristiche di esecuzione dei Task durante la fase di Debug, ho realizzato una parte di libreria a supporto dei Task in modo da valutarne le caratteristiche tramite comunicazione Seriale.

È stata integrata un ulteriore struttura dati, la quale contiene informazioni riguardo i Task e sono:

- Period
- StartExecution ultimo job
- Ultima Activation
- Prossima Deadline
- Numero di Deadline Miss
- Max Execution time
- Execution Time
- Sum Execution Time
- Numero Job effettuati
- Average Execution Time
- First Activation

4.1 Stima del Worst-Case Execution Time

Per ottenere una stima dei Tempi di Esecuzione riguardo i singoli task, ho estrapolato dal sistema la parte relativa ad ogni Task e ne ho analizzato le caratteristiche per almeno 1000 iterazioni, inoltre il sistema è stato modificato in modo tale che ogni task esegua tutti i compiti che gli sono stati assegnati indipendentemente dallo stato del sistema.

Per quanto riguarda le interfacce è stata considerata la situazione in cui le informazioni rappresentate siano le massime possibili, ed il flusso di dati scambiato anch'esso il massimo possibile.

Assumendo come Worst Case Execution Time di ogni task il corrispondente Valore stimato di Max Execution Time, otteniamo il seguente task Set schedulabile secondo RTA.

Tasks	Period	Max Execution Time	AverageExecution Time	Uwc	Uavg
Task system 1	850	120	119	0,141176	0,14
Control Panel	850	290	20	0,341176	0,023529
WebSocket	1000	112	43	0,112	0,043
Task System 2	5000	358	317	0,0716	0,0634
WebServer	6000	584	20	0,097333	0,003333
			U	**0,763286**	**0,273263**

Tuttavia per ottenere la garanzia di Schedulability per il worst case abbiamo perso in prestazioni.

Consideriamo il sistema come Soft Real-Time, e mantenendo l'ordine di priorità come segue nel prossimo paragrafo, è preferibile ridurre i periodi dei Task relativi alle interfacce e rischiare qualche Deadline Miss, piuttosto che avere un sistema poco interattivo.

Consideriamo il caso particolare della coppia WebServer - WebSocket con il vincolo di una connessione, per instaurare una nuova connessione con il WebSocket Server il Client deve prima aver ricevuto la pagina.

Se consideriamo come worst-case la fase di instaurazione della connessione possiamo elaborare la seguente relazione:

“Ad ogni WebPage inviata consegue un solo WCET del WebSocket.”

Al fine di calcolare un nuovo tempo di esecuzione ponderato, andiamo a considerare il numero di job del WebSocket nel periodo del WebServer e di questi solo uno sarà un WCET:

$$Occur = \left\lfloor \frac{Period_{WebServer}}{Period_{WebSocket}} \right\rfloor$$

$$ExecutionTime_{WebSocket} = WCET_{WebSocket} * \left(\frac{1}{Occur}\right) + AvgExecutionTime * \left(1 - \frac{1}{Occur}\right)$$

Per quanto riguarda il Control Panel, la frequenza con la quale si presentano i casi di WCET dipende dall’utente, e supponiamo che questi eventi si presentino con un’occorrenza di un WCET ogni quattro job

$$Occur_{control\ panel} = 4$$

Tasks	Period	1/Occur	WCET	AvgET	New ET	Uwc	Uavg	Udes
Task system 1	350	1	120	119	120	0,342857	0,34	0,342857
Control Panel	350	0,25	290	20	87,5	0,828571	0,057143	0,25
WebSocket	500	0,125	112	43	51,625	0,224	0,086	0,10325
Task System 2	4000	1	358	317	358	0,0895	0,07925	0,0895
WebServer	4000	1	584	20	584	0,146	0,005	0,146
					U	**1,630929**	**0,567393**	**0,931607**
					HB	**3,752624**	**1,668616**	**2,312201**

Secondo le precedenti ipotesi di funzionamento otteniamo un differente Task Set Schedulabile, per quanto riguarda il caso medio ed il nuovo caso desiderato.

Nel caso in cui si presenta il Worst Case del Control Panel, il set non è Schedulabile per nessuna delle configurazioni viste.

Riassumendo, dato il sistema come Soft Real-Time, ho preferito aumentare la frequenza dei Task per ottenere migliori prestazioni, il sistema è soggetto a “transient overloads” causati dal Task Control Panel.

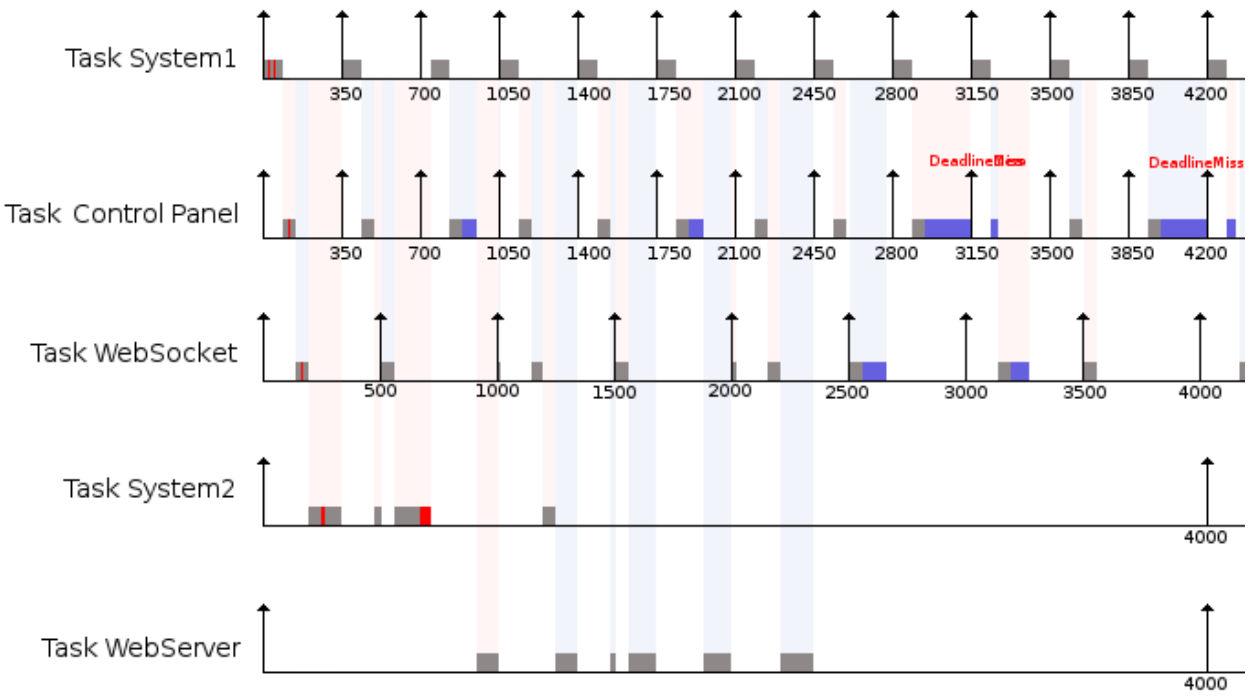


FIGURE 5 POSSIBLE EXECUTION

In **grigio** i job con tempo di esecuzione calcolato tramite le precedenti considerazioni (caso desiderato).

In **rosso** le sezioni critiche, nel primo job dei Task ho riportato alcune delle sezioni critiche, ma andrebbero considerate anche le sezioni critiche relative alle librerie SPI e Wire, la sezione critica più consistente si trova nel Task System2 ed è relativa al salvataggio dello stato sistema su memoria SD, tempo stimato 60ms.

In **blu** il possibile incremento del tempo di esecuzione previsto dei Task relativi alle interfacce.

4.3 Priorità

Le priorità vengono assegnate dal sistema in modo proporzionale rispetto la frequenza del Task e, a parità di frequenza rispetto l'ordine dei loop, il sistema utilizza Rate Monotonic come Scheduler.

Viste le considerazioni riguardo l'interattività e la sicurezza del sistema, le priorità dei Task devono seguire questo schema:

Task System 1	1
Task Control Panel	2
Task Web Socket	3
Task System 2	4
Task Web Server	5

4.4 Blocking Time

La presenza di sezioni critiche blocca lo Scheduler, di conseguenza un Task a più alta priorità potrebbe essere ritardato di un tempo pari alla durata della sezione critica, la sezione critica più consistente è data dal salvataggio dello stato del sistema, il quale viene effettuato su memoria SD ed ha una durata stimata di 60ms.

4.5 Deadline Miss

È possibile, manomettendo la comunicazione tra WebSocket Client e WebSocket Server, che il Task relativo al WebSocket abbia tempi di esecuzione maggiori rispetto quelli previsti, cosa che porterebbe WebSocket, System2 e WebServer a possibili eventi di deadline miss essendo questi a priorità inferiore, se questa situazione permane nel tempo si potrebbero verificare eventi di "Permanent Overload" che porterebbero alla non esecuzione dei job relativi a questi task.

5 Possibili ulteriori ottimizzazioni

5.1 Web

Il sistema tiene traccia dell'attuale pagina che viene rappresentata, questa informazione viene aggiornata automaticamente ad ogni richiesta di nuova pagina, è possibile aggiornare tale informazione inviando un apposito comando tramite WebSocket, questa ulteriore possibilità permette di utilizzare pagine dinamiche piuttosto che sole pagine statiche, eliminando così la necessaria connessione e disconnessione dal WebSocket per ogni pagina consecutiva richiesta, oltre che l'invio di una sola pagina web.

5.1 Control Panel

Il Display comunica con Arduino DUE tramite Seriale UART, il baud rate è stato settato a 57600 bps, è possibile, con opportune modifiche alle librerie del dispositivo e riprogrammando il display, portare il baundrate a 115200 bps riducendo così il WCET del Task relativo al Control Panel.

5.2 SDfat library

Questa libreria non è ottimizzata per l'utilizzo concorrente, nonostante utilizzi la libreria SPI adattata per ARTe, si presenta la situazione in cui due dispositivi rimangono selezionati contemporaneamente, per ovviare a questo problema è stato necessario proteggere con sezioni critiche le operazioni su SD.

Ottimizzando questa libreria, la sezione critica più consistente viene ridotta alla più lunga tra le sezioni critiche delle librerie a supporto dell'hardware.

6 Grafici

6.1 Task Control Panel

I picchi sopra i 200 ms sono dovuti ad una fase di “stress test” del dispositivo, mentre quelli al di sotto sono dovuti ad un utilizzo “standard”.

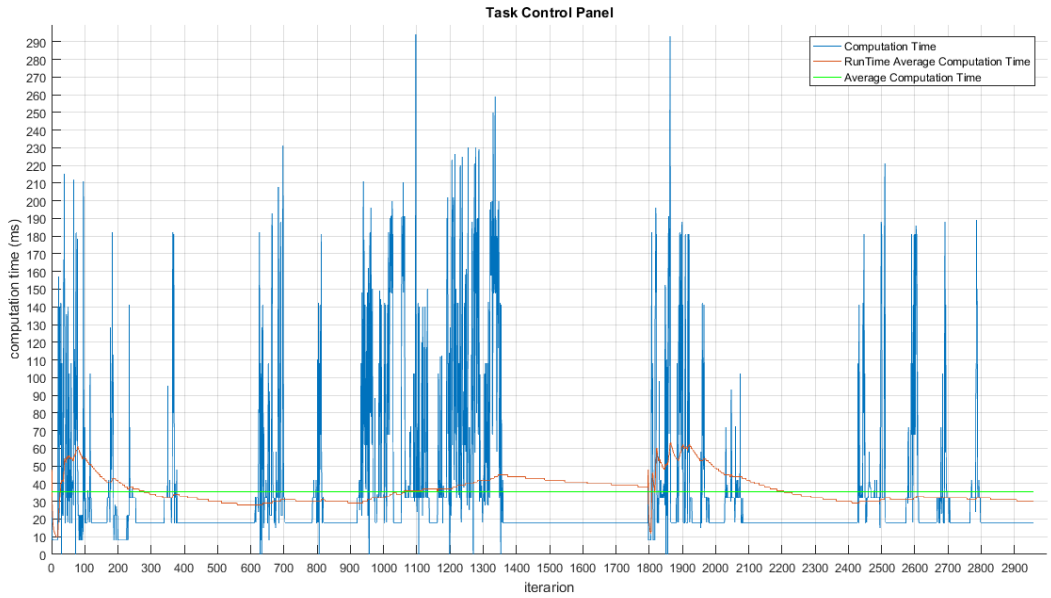


FIGURE 6 COMPUTATION TIME OF TASK CONTROL PANEL

I casi di Worst-Case Computation Time sono isolati.

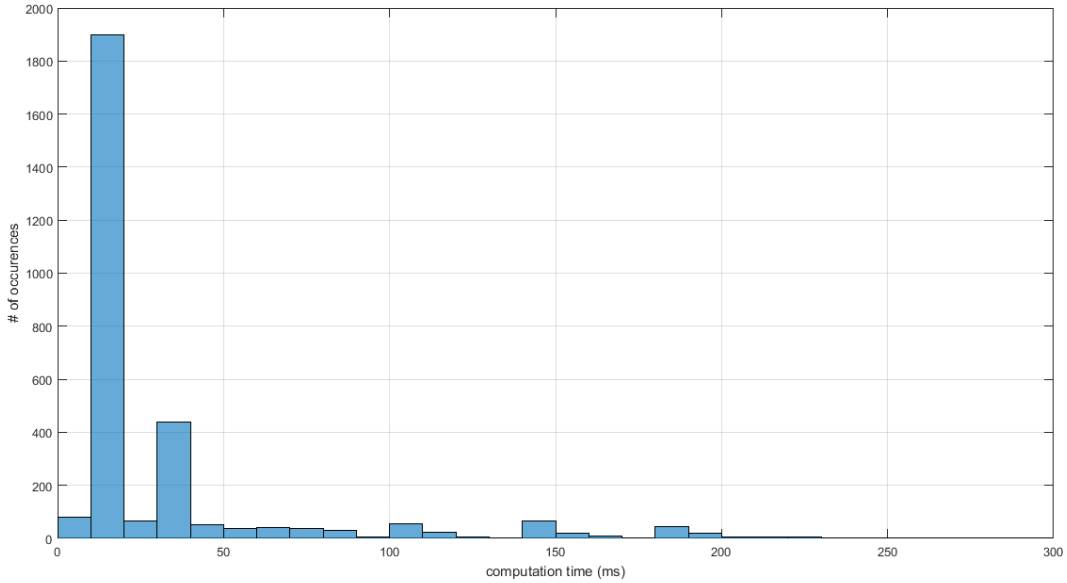


FIGURE 7 CONTROL PANEL EXECUTION TIME DISTRIBUTION

6.2 Task Web Server

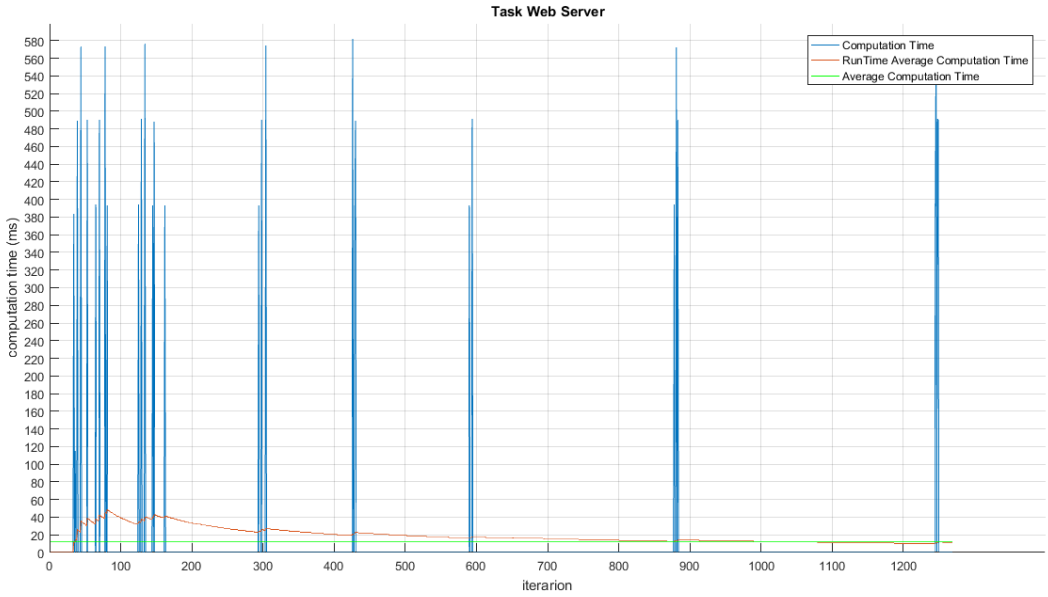


FIGURE 8 COMPUTATION TIME OF TASK WEB SERVER

Si notano tre diversi picchi, uno per ogni pagina web trasferita, la più consistente è la pagina dei Task.

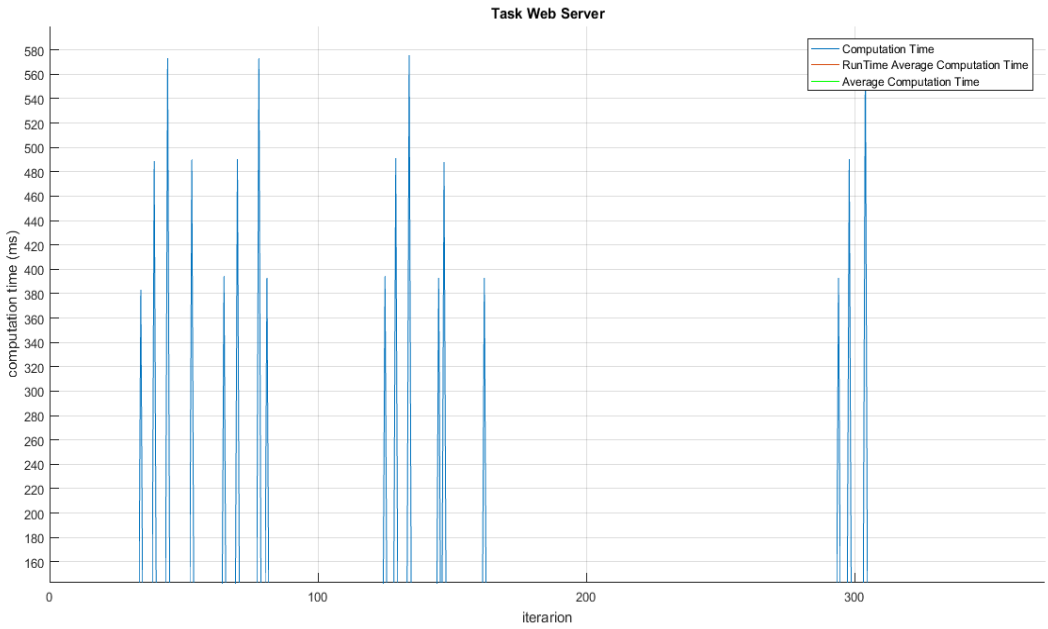


FIGURE 9 COMPUTATION TIME OF TASK WEB SERVER

6.3 Task Web Socket

I tre grossi picchi, con Computation Time maggiore di 1000ms, sono dovuti alla chiusura errata del WebSocket, con conseguente Deadline Miss da parte del Task Web Socket.

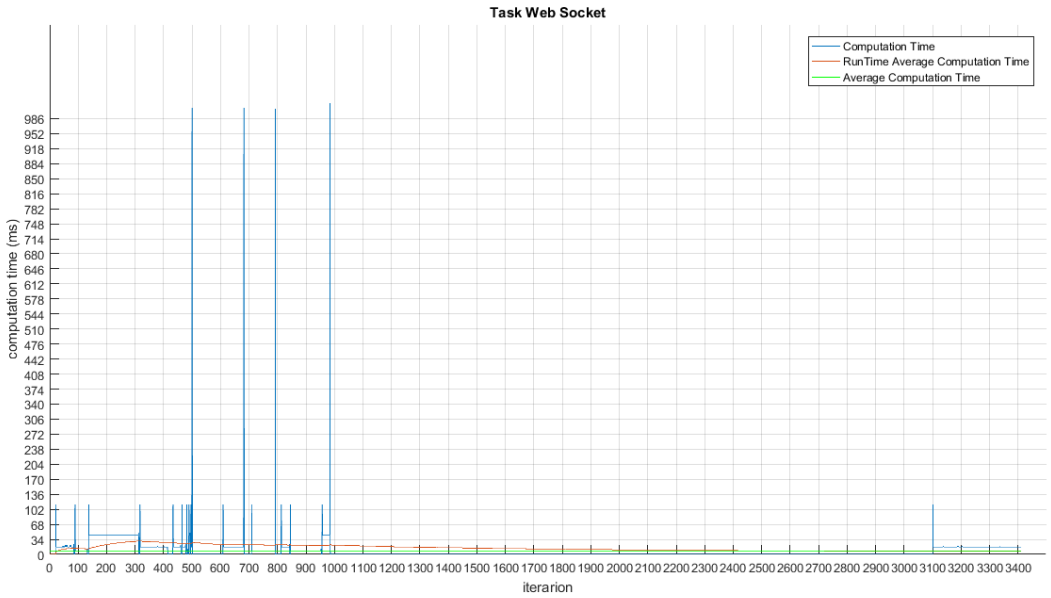


FIGURE 10 COMPUTATION TIME OF TASK WEB SOCKET

Durante la fase di funzionamento regolare si notano alcuni picchi a ~120ms, questi sono relativi alla fase di instaurazione della connessione con il Web Socket e come dalle ipotesi fatte ne abbiamo uno per ogni pagina richiesta.

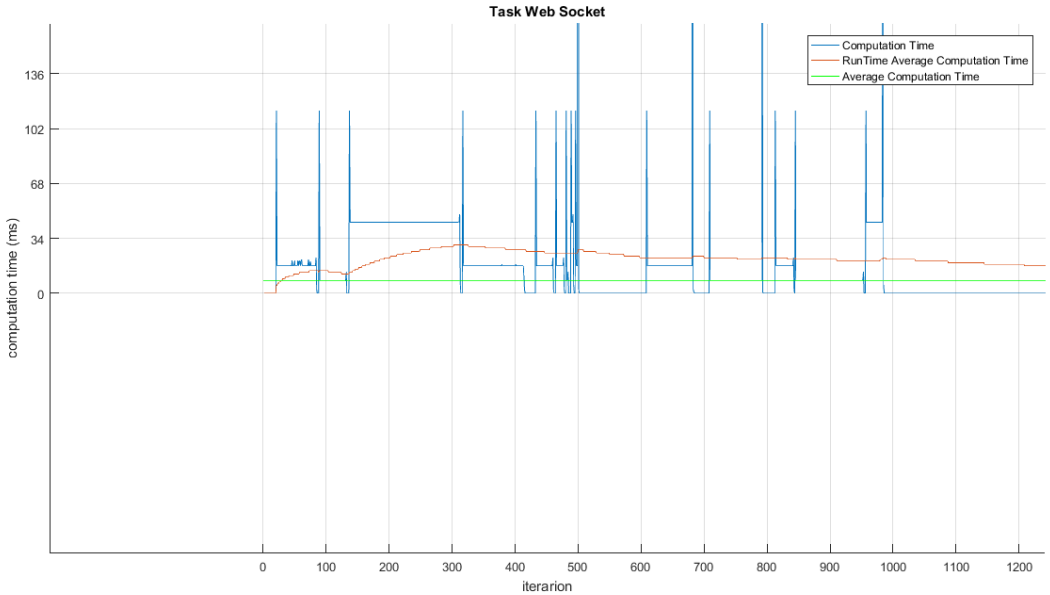


FIGURE 11 COMPUTATION TIME OF TASK WEB SOCKET