SISTEMI EMBEDDED AL TEMPO DEL COVID-19

La relazione mira a condividere come campo accademico-universitario e aziendale possano collaborare al fine di produrre applicativi di quotidiano utilizzo.

Utilizzo delle conoscenze acquisite durante il corso di Sistemi Embedded all'interno di una realtà aziendale.

Sommario

Introduzione 2
Motivazioni relative alla scelta del tema 2
Presentazione dell'argomento trattato 2
Kaligo 2
Totem Ienius 3
Persone coinvolte nel lavoro e loro ruolo 3
Progettazione 4
Kaligo 4
Programma Arduino 5
Rappresentazione Schema elettrico 13
Applicazione Android 14
Tempi di realizzazione 17
Conclusioni 17
Difficoltà incontrare e riflessioni conclusive 17
Idee per il futuro 18
Indice delle Immagini 19
Indice delle tabelle 19

Introduzione

Il progetto, che il seguente documento pone sotto osservazione, scaturisce da una lunga analisi effettuata all'interno dell'azienda IEN Industrie S.p.A di Fano. Tale realtà, da oltre un ventennio, commercializza a livello internazionale macchine automatiche nel campo della rilegatura di album fotografici. IEN Industrie conta oltre 50 dipendenti all'interno della propria sede e una vasta rete di legami con importanti aziende quali HP e Canon, al fine di guidare e suggerire ai propri clienti il miglior servizio e processo produttivo possibile.

Motivazioni relative alla scelta del tema

Ad inizio 2020 l'Italia, come qualsiasi altra Nazione sul pianeta terra, è stata vittima di una delle più grandi pandemie che l'essere umano abbia mai dovuto affrontare. Se da un lato numerosi ricercatori erano al lavoro per ricavare un efficace vaccino contro il COVID-19, dall'altra c'era un'intera popolazione costretta a restare in casa, in quanto il contatto ravvicinato poteva pregiudicare la salute della persona. A seguito di una lunga quarantena, il bisogno, da parte di tutti, di ritornare alla quotidianità, stava diventando cospicuo e le varie amministrazioni dovevano urgentemente proporre una soluzione a ciò. Di fronte a tale domanda molteplici aziende si sono mobilitate per proporre soluzioni volte al contenimento, alla sicurezza e alla sanificazione di persone e oggetti attraverso appositi dispositivi.

Anche IEN Industrie ha voluto farne parte progettando e commercializzando una gamma di prodotti rivolti alla media e grande distribuzione.

Presentazione dell'argomento trattato

Preso atto della crescente richiesta da parte del mercato e della volontà aziendale di entrare in un nuovo settore in fase storica critica e complessa, IEN ha deciso di dedicare risorse, tempo ed investimenti al fine di identificare dei prodotti che potessero avere rilevanza.

Contestualmente, l'obiettivo era anche il concepimento dei suddetti prodotti in tempi rapidi, al fine di sfruttare al massimo la crescente richiesta di prodotti volti alla sanificazione di ambienti, oggetti e persone.

Kaligo

Il primo prodotto individuato, in maniera collegiale da alcuni responsabili aziendali, risulta una macchina che permette di sanificare capi di abbigliamento e/o oggetti generici attraverso l'utilizzo di prodotti chimici, esempio Labiosan, e relativi processi di attivazione. La macchina doveva possedere una struttura modulare per permettere un facile collocamento anche in locali commerciali che possiedono pochi e piccoli spazi disponibili. Il dispositivo sarebbe stato venduto ad un pubblico assai eterogeneo di persone da esercizi commerciali, enti pubblici e privati fino a centri di salute. Ciò ha introdotto numerose sfide sia a livello di Design sia a livello di operabilità della e con la macchina.
Infine, come numerose concorrenti, la macchina doveva generare dei riscontri luminosi o sonori per segnalare all'operatore l'attuale stato del processo di sanificazione

Totem Ienius

Ienius si pone in maniera del tutto complementare alle finalità di Kaligo. Se quest'ultimo ha la volontà di sanificare oggetti, potenzialmente contaminati da batteri, muffe e virus, Ienius intende evitare che il contagio avvenga.

Il totem multifunzione Ienius introduce un meccanismo di controllo e monitoraggio degli accessi, e conseguenti uscite, a luoghi pubblici o privati.
Nel dettaglio definiamo con:

- Controllo: l'azione volta a determinare se un soggetto rispecchia i requisiti di legge d'idoneità riguardanti temperatura corporea e adeguato utilizzo della mascherina sanitaria in luoghi pubblici.
  Inoltre, per alcuni settori, il totem ha lo scopo di fornire procedure di contingentamento dei locali; ovvero rendere inaccessibile un'area dell'edificio al superamento di un limite massimo di persone simultaneamente presenti al suo interno
- Monitoraggio: rispettando le norme in ambito di privacy disposte dal GDPR 2016/679, il sistema doveva memorizzare per un arco di tempo variabile le informazioni all'interno di un database, per poter essere successivamente reperibili in caso di necessità quali la determinazione di persone che sono state a contatto con un potenziale contagiato di SARS Covid-19.

Completiamo l'introduzione del progetto menzionando l'ultima funzionalità concepita in corso d'opera; la possibilità da parte degli acquirenti del prodotto, di poter creare degli appuntamenti per persone (o gruppi). Gli invitati, tendenzialmente non dipendenti dell'azienda da cui proviene l'appuntamento, ricevono un'e-mail contenente delle informazioni e un QRCODE, il quale una volta posizionato di fronte al totem, restituirà un badge per poter avere accesso a determinati reparti dell'edificio.

Persone coinvolte nel lavoro e loro ruolo

La realizzazione di progetti di questa portata viene gestita attraverso un lungo iter, disseminato di incontri collegiali volti a comprendere l'avanzamento delle attività e propedeutici per comprendere se sussiste la necessità di correggere qualcosa. Quanto appena detto è possibile soprattutto attraverso un cospicuo numero di professionisti che collaborano al conseguimento di un obiettivo comune.

Per ragioni di privacy e personali, nella presente relazione non sono presenti i nomi che hanno contribuito alla realizzazione dei prodotti e che hanno gestito la parte di Design e progettazione del prodotto, la parte elettrica e di montaggio ed infine la parte di collaudo e produzione.

A rigor del merito però, va comunicato che il team che ha gestito la parte di progettazione è composto da 7 persone; 3 per la parte di Design e progettazione, 1 per la parte elettrica, 2 per la parte di collaudo ed industrializzazione, oltre alla parte di sviluppo software effettuato dal redattore del presente documento.

Progettazione

I capitoli contenuti all'interno del presente documento permettono di identificare la fase di progettazione e realizzazione del prototipo del progetto Kaligo, successivamente passato ad un processo di produzione su più ampia scala.

Kaligo

A seguito di una lunga analisi in merito alle funzionalità che avrebbe dovuto fornire la macchina, il team ha optato per l'introduzione di un microcontrollore al quale affidare l'incarico di monitorare e governare il funzionamento del dispositivo e contestualmente la comunicazione verso l'operatore attraverso una connessione Bluetooth.

La macchina avrebbe dovuto gestire due diverse tipologie di prodotti sanificanti, Laboisan e UV, e tre modalità di utilizzo:

- Nebulizzazione (a caldo): in cui, per un tempo prestabilito dall'operatore, la macchina avrebbe dovuto nebulizzare il Labiosan producendo fumo sanificante, il quale avrebbe dovuto saturare l'interno della macchina;
- Fotocatalisi: in cui, per un tempo prestabilito dall'operatore, la macchina avrebbe attivato la lampada UV e la ventola di areazione interna. Attivando la lampada si generano ioni attivi che permettono la sanificazione;
- Modalità notte: utilizzata per sanificare interi ambienti domestici e/o lavorativi, in quanto le porte della macchina sarebbero restate aperte e il fumo generato avrebbe pervaso la stanza.

La nebulizzazione del Labiosan al fine di produrre il fumo, rappresentava la vera grande difficoltà del progetto, poiché l'iter di fabbricazione contemplava numerosi dispositivi che dovevano seguire simultaneamente regole ben precise. Ripercorrendo con ordine il processo, al primo step incontriamo il contenitore di Labiosan, dal quale attingere il prodotto allo stato liquido. Il contenitore, posizionato e nascosto all'interno del lato laterale della macchina, era collegato tramite un tubo alla pompa di aspirazione, posizionata nella parte superiore. La pompa comandata attraverso un relay di potenza aveva lo scopo di prelevare il liquido è trasportarlo all'interno della caldaia, in cui attraverso un elevato sbalzo termico cambia stato e si trasforma in gas, contenente elementi chimici finalizzati all'eliminazione di batteri e virus.

La produzione del gas sanificante produce un immediato abbassamento della temperatura interna della caldaia, la quale dev'essere monitorata al fine di interrompere la produzione al superamento di una temperatura settata come limite inferiore di lavorazione. Oltrepassando quel valore non vi sono garanzia che il cambiamento di stato del prodotto avvenga, con il drammatico ed inevitabile esito di danneggiare gli oggetti contenuti all'interno dell'ambiente sanificante per via dell'introduzione del prodotto allo stato liquido e con un'elevata temperatura.

La variazione della temperatura interna della caldaia a seguito dell'introduzione del liquido è visibile attraverso la Figura 1.

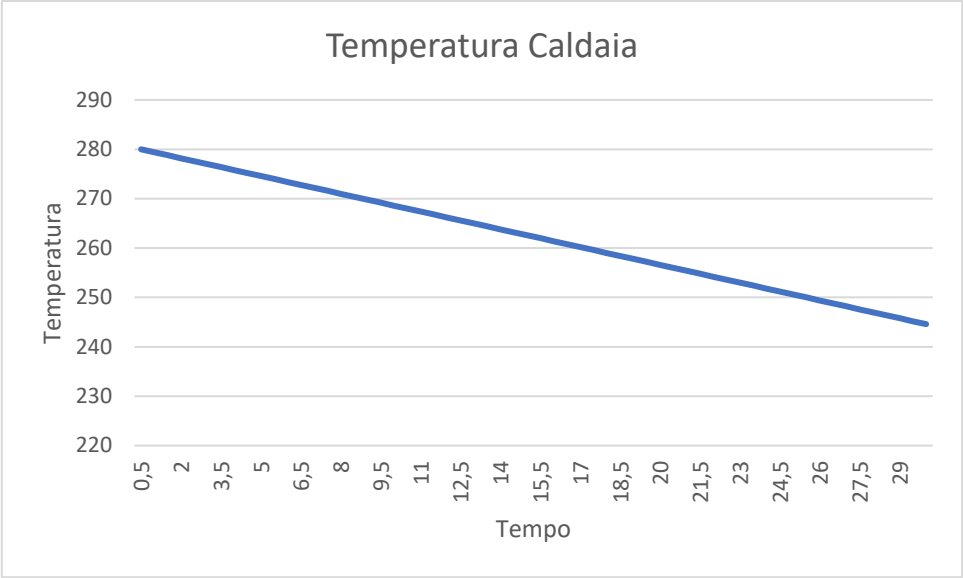

Figura 1 Andamento Temperatura Caldaia

Programma Arduino

Il compito di supervisionare la ciclica della macchina è stato affidato ad Arduino MEGA, un microcontrollore basato su un Atmega 2560.

Un microcontrollore (o MCU) è un single-chip computer, ovvero un microcalcolatore integrato su una singola scheda di dimensioni ridotte e utilizzando tecniche di microelettronica riduce in poco spazio vari componenti quali CPU e memorie. Il chip possiede anche pin di ingresso e uscita, attraverso i quali è possibile interagire e che permettono di realizzare applicazioni special purpose, ovvero dedicate esclusivamente alla funzione prestabilita. In ambito embedded vengono largamente utilizzati in quanto rappresentano un'adeguata soluzione in termini di rapporto costo-prestazione. Infine, occorre precisare che durante la realizzazione del software, attraverso appositi IDE, il programmatore non manipola le istruzioni direttamente contenute nel chip, bensì elabora istruzioni situate logicamente in uno strato superiore.

Orientiamo la narrativa del presente paragrafo verso la parte di progettazione del sistema, introducendo e analizzando le principali parti del progetto, attraverso UML e grafici finalizzati alla comprensione del fenomeno fisico d'interesse.

Introduciamo i primi diagrammi UML attraverso Figura 2 e Figura 3, che rappresentano rispettivamente le tipologie di dispositivi da interfacciare tramite ingressi digitali e ingressi analogici.

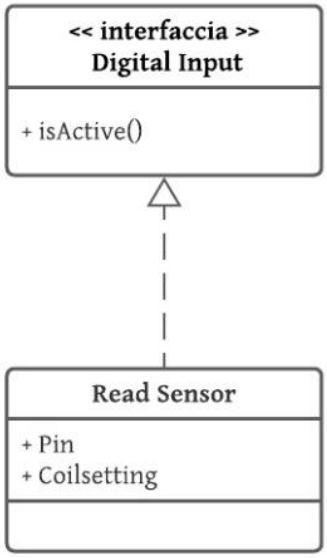

Figura 2 UML classi - ingressi digitali

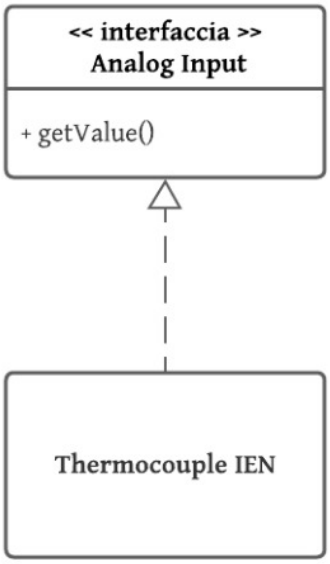

Figura 3 UML classi - ingressi analogici

Completiamo la condivisione con la Figura 4, in cui è possibile osservare le tipologie di classi utilizzate per modellare i dispositivi da interfacciare attraverso uscite digitali del microcontrollore.

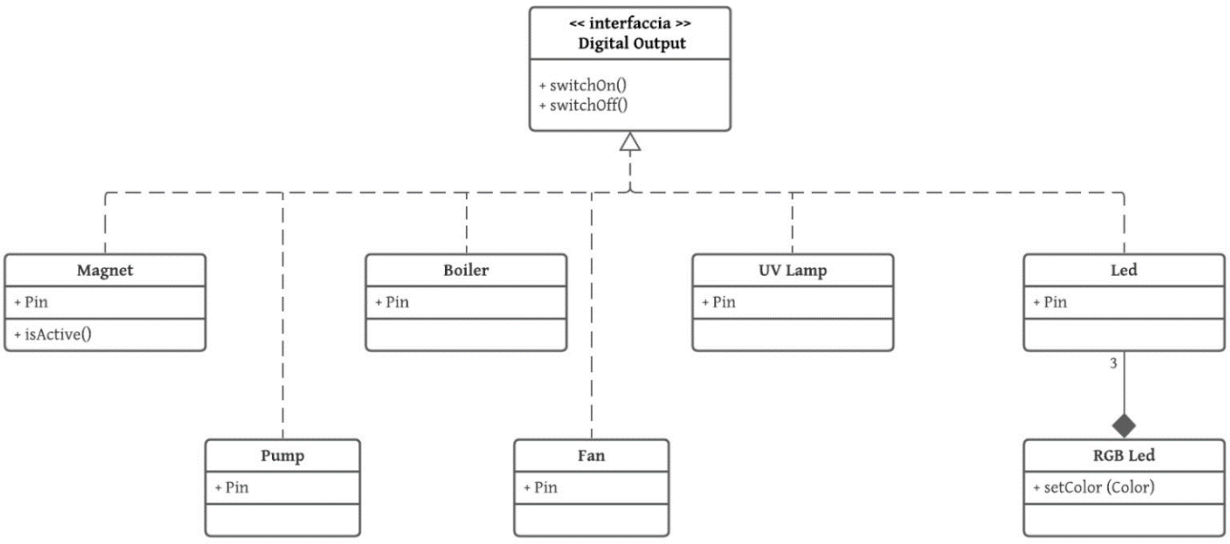

Figura 4 UML classi - uscite digitali

Riprendendo i diagrammi delle classi, posizionati nella zona superiore, è doveroso concedere una discreta riflessione per quanto riguarda la termocoppia.

La termocoppia è un trasduttore di temperatura ampiamente utilizzato, facilmente sostituibile e standardizzato tramite IEC EN 60584. È composto da 2 materiali conduttori, i quali si uniscono in

una delle 2 sezioni principali di una termocoppia: il “giunto caldo”. Quest’ultimo è il punto in cui occorre misurare la temperatura. L’altro capo è definito “giunto freddo”, il quale è connesso allo strumento di misura. La termocoppia sfrutta l’effetto Seebeck, dal nome del fisico estone che studiò il fenomeno, secondo il quale la differenza di potenziale misurata in prossimità del giunto freddo è direttamente proporzionale al gradiente delle temperature dei materiali situati nel giunto caldo. Le termocoppie si dividono in tipologie a secondo delle coppie di materiali che vengono utilizzati per la realizzazione del dispositivo, in quanto combinazioni diverse determinano caratteristiche prestazioni e sensibilità nella lettura differenti. Per il presente progetto, il team ha deciso di utilizzare una termocoppia Tipo K, la quale possiede la seguente coppia di materiali: Chromel e Alumel (entrambe lege a base di nichel).

La termocoppia deve essere affiancata ad un dispositivo che determina la temperatura e la comunica ad Arduino. Tale dispositivo è di seguito visibile e comunica l’informazione tramite SPI, che permette una comunicazione bidirezionale tra master e slave.

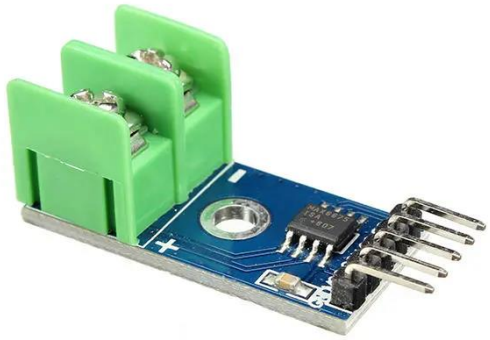

Figura 5 Modulo lettura termocoppia Tipo K

La temperatura rilevata dalla termocoppia è un elemento fondamentale del progetto, in quanto attraverso tale valore, il controllore dovrà determinare l’abilitazione o meno del relay di potenza che pilota l’accensione della caldaia. In questa parte del progetto subentra il fenomeno dell’isteresi. Infatti, per poter operare adeguatamente, la temperatura interna della caldaia necessita di essere all’interno di un range prestabilito.

Il controllore dovrà costantemente leggere la temperatura fornita dalla termocoppia tramite l’apposito oggetto, abilitare l’accensione della caldaia se tale temperatura è scesa al di sotto del limite inferiore prescelto e, analogamente, disabilitare il relay rivolto alla caldaia quando la suddetta temperatura supera il limite superiore. L’operazione appena descritta è facilmente rappresentabile tramite la Figura 6, di seguito riportata.

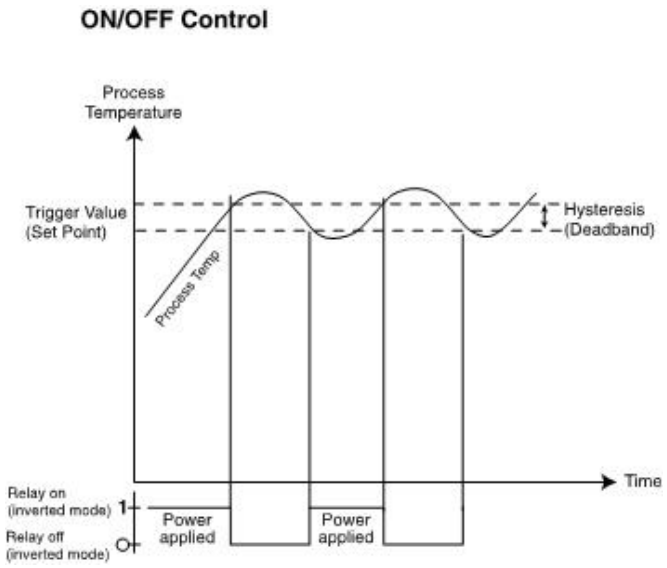

Figura 6 Isteresi temperatura caldaia

Le precedenti frasi sono state un valido punto d'inizio per poter focalizzare l'osservazione verso l'operatività del sistema nel suo insieme. Sono stati illustrati i singoli dispositivi utilizzati e come è avvenuta la modellazione degli stessi.
Procediamo ora analizzando la struttura che attivamente rende possibile il funzionamento del progetto, il quale sarà implementato e caricato sul microcontrollore citato.

Utilizzando un approccio Top-Down inizia analizzando architettura base, la quale contiene uno "Scheduler" fondamentale per l'operatività di "Task" sincroni. La schedulazione del sistema creato avviene secondo il principio di "round-robin", in cui non è presente interleaving e i singoli task eseguono la propria azione quando si invoca un "Tick". I principali vantaggi dell'approccio adottato è che non sono presenti corse critiche che potrebbero disturbare l'operatore di differenti Task.
Ne deriva che la tipologia di multi-task adattata è cooperativa.

Infine, occorre raffinare la modellazione e l'implementazione per i task che dovranno consentire l'operatività dei diversi cicli di lavoro. Essi sono rappresentanti attraverso la classe astratta "AbstracCycleTask" e le corrispettive estensioni contengono macchine a stati sincrone in cui in base allo stato si determina l'azione da compiere.

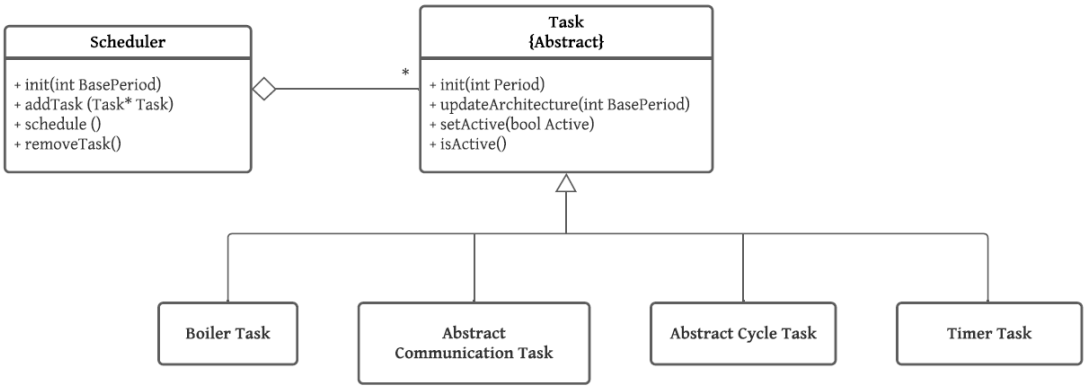

Figura 7 UML Gestione Task

Come è possibile osservare tramite Figura 7, la classe astratta "`Task`" è estesa al fine di gestire i seguenti Task:

- `BoilerTask`: la gestione della temperatura interna della caldaia segue una logica indipendente dal sistema. Rispettando il periodo impostato, il task dovrà monitore l'andamento della temperatura e regolare la caldaia secondo le considerazioni d'isteresi effettuate
- `TimerTask`: i cicli di lavorazione necessitano di timer impostabili secondo esigenze differenti, ma soprattutto con tempistiche che possono variare dai pochi secondi ai diversi minuti. Un esempio calzante riguarda la funzionalità cardine, ovvero il tempo necessario per sanificare gli oggetti contenuti all'interno della macchina. Tali task utilizzano una funzione non bloccante di Arduino per determinare il passaggio del tempo: `millis`(). Quest'ultima restituisce una variabile long corrispondente a millisecondi trascorsi dal momento in cui viene messo in esecuzione il programma. L'utilizzo della funzione `millis` deve essere effettuato con la consapevolezza che tale funzione torna a zero al superamento del 49-esimo giorno di operatività consecutiva del microcontrollore
- `AbstractCommunicationTask`: rappresenta una classe astratta utile per poter interfacciare diversi fonti di comunicazione che potrebbero comunicare con Arduino. All'interno del progetto è utilizzata esclusivamente per definire metodi adeguati allo scambio di informazioni tramite Bluetooth
- `AbstractCycleTask`: come preannunciato, definisce metodi comuni riguardati le macchine a stati che modellano i 3 cicli di lavoro per la sanificazione. Contestualmente, la classe implementa interfacce finalizzate allo scambio d'informazioni tra Task. Di seguito verrà illustrato come i cicli ricevono informazioni provenienti dalla comunicazione bluetooth oppure riguardanti l'insorgere di interrupt di sistema.

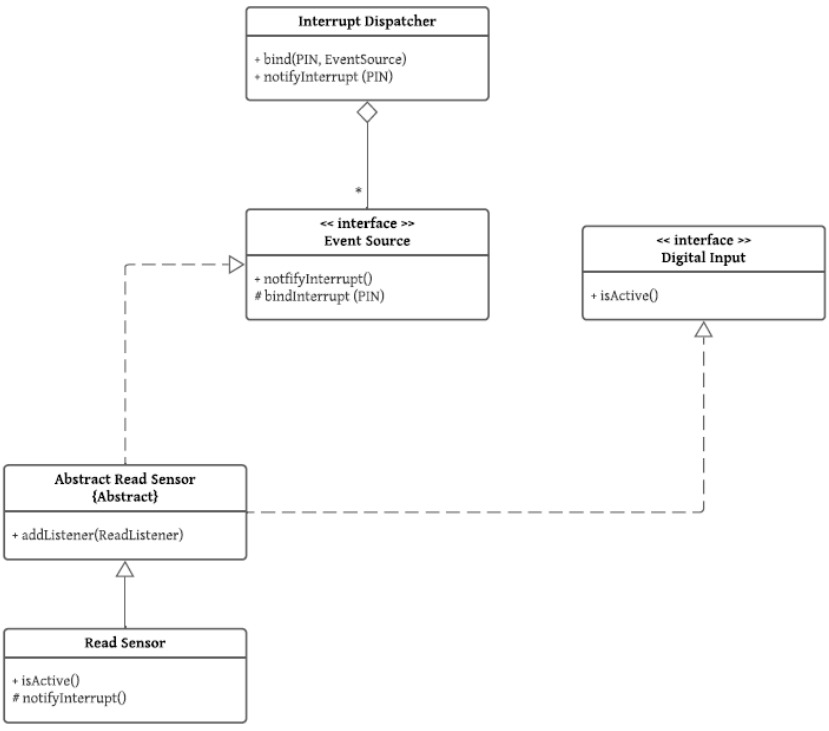

Figura 8 UML gestione Interrupt

Il raffinamento della modellazione prosegue grazie all'introduzione di due ulteriori elementi:

- Gli interrupt: sono eventi, che vengono generati nativamente dal controllore al cambiamento di valore relativo ai pin di I/O. Gli interrupt sono ampiamenti utilizzanti in quanto il programmatore delega al controllore il compito di monitorare costantemente il valore di un pin, evitando quindi di effettuare polling. Quando lo stato cambia si verifica l'interrupt e viene eseguita la funzione precedentemente assegnatagli.
- L'introduzione del pattern di comportamento "Observer". Il pattern definisce una relazione tra un oggetto (o in generale una sorgente) e un insieme di oggetti interessati ad essere notificati quando il primo cambia stato o alternativamente quando al primo succede un evento che deve essere notificato.

A fronte di quanto appena detto, introduciamo la Figura 8 la quale permette di osservare la prima parte di modellazione in cui è stato utilizzato il pattern Observer.

Il diagramma mostra la classe astratta `AbstractReadSensor` che estende le interfacce "`DigitalInput`" e "`EventSource`". Il motivo dell'ereditarietà multipla pocanzi indicata è dovuto alla provenienza dell'evento che vogliamo modellare e monitorare; ovvero il segnale generato dal read collocato sulle porte della macchina. L'apertura di quest'ultime durante la sanificazione e rigorosamente vietata in quanto l'inalazione prolungata dei prodotti, utilizzati durante la sanificazione, potrebbe provocare danni alla salute. Per precauzione e per rispettare misure minime di sicurezza occorre fermare tempestivamente la sanificazione se vengono aperte le porte.

Grazie all'implementazione dell'interfaccia "`EventSource`" gestiamo la creazione dell'interrupt tramite il metodo `bindInterrupt` al quale passiamo il pin che vogliamo sorvegliare. Contestualmente potremo determinare il relativo valore grazie al metodo `isActive` concesso dall'interfaccia "`DigitalInput`".

Per il momento, la discussione si è soffermata sulle classi che determinano la sorgente del pattern Observer, ma non è ancora chiaro cosa succede quando si verifica l'interrupt. Per rispondere occorre continuare a porre l'attenzione nei riguardi della classe astratta `AbstractReadSensor` e soffermarsi sul metodo `addListener`. Tale metodo permette di aggiungere oggetti, che estendono l'interfaccia "`ReadSensorListener`", che vogliono essere notificati al generarsi dell'interrupt.

L'interfaccia "`ReadSensorListener`" viene implementata dalla classe astratta `AbstractCycleTask`. Quest'ultima contestualmente ne implementa delle altre, in quanto lo scopo principale è quello di evitare ridondanze di codice nei task che dovranno rappresentare i cicli di sanificazione. Infatti, la maggior parte dei metodi ereditati dalle interfacce "`ReadSensorListener`" e "`MessageListener`" sono implementati direttamente all'interno della classe astratta, mentre l'esecuzione delle macchine a stati viene delegata interamente alle classi che la estenderanno. Tale approccio riduce i tempi di implementazione e manutenzione in quanto funzionalità simili, tra i diversi cicli, vengono gestite adeguatamente in un solo punto.
La discussione relativa al primo utilizzo del pattern Observer è conclusa e possiamo procedere con la presentazione del secondo punto in cui è stato utilizzato il pattern: quello riguardante la condivisione di informazioni provenienti dalla comunicazione Bluetooth.

La struttura del pattern Observer, in questo secondo utilizzo, resta invariata, per tanto è possibile osservare una sorgente di informazioni rappresentata da `BleutoothTask` e degli oggetti che desiderano essere notificati al verificarsi di determinati eventi. Tali oggetti sono identificabili nei cicli di lavorazione che devono essere svolti. Quanto appena detto, ha validato le considerazioni

precedentemente compiute in merito alla classe astratta `AbstractCycleTask`, in merito al ruolo di ottimizzazione che svolge.

La trattazione e descrizione del progetto si avvia verso la conclusione, non prima di aver definito ed analizzato il compito svolto dai task principali, ovvero quello riguardati la sanificazione, e le relative macchine a stati.

Procediamo differenziando i task in base al prodotto che utilizzato per realizzare la sanificazione; per tanto avremo `NebulizationTask` e `NightModeTask` che utilizzano il liquido Labiosan e `PhotocatalysisTask` che utilizza la fotocatalisi prodotta dalla lampada UV.

La macchina a stati contenuta all'interno di `Photocatalysis` presenta una struttura molto semplice e lineare, la quale inizia da uno stato di INIT in cui si resettano variabili e si disabilitano uscite che potrebbero essere rimaste abilitate da precedenti esecuzioni ed infine avviene il cambio allo stato WAIT. Qui si attende l'abilitazione, tramite l'app Android, all'attivazione del processo sanificante. Quando ciò avviene, si verifica la corretta chiusura delle porte e successivamente si procede con l'attivazione delle lampade per compiere la sanificazione. La temporizzazione viene gestita tramite il timer interno al Task, il quale è stato impostato grazie all'attributo `remainedTime` della classe. Concluso il tempo indicato, si procede con lo spegnimento delle lampade ed infine si torna allo stato di WAIT fino alla comunicazione di un nuovo comando.

Per completare la trattazione di questa modalità operativa, va condivisa la tabella relativa la comunicazione visiva applicata tramite LED RBG.

Tabella 1 Significato colori ciclo fotocatalisi

Colore	Significato
LIGHTBLUE	Macchina accesa e operativa
YELLOW	Impossibile proseguire per porte aperte
BLUE	Sanificazione in corso
RED	Interruzione causata da emergenza
GREEN	Ciclo completato correttamente

La trattazione dei cicli di sanificazione che utilizzano il liquido Labiosan è stata volontariamente lasciata per ultima, in quanto oltre ad essere la più delicata e pericolosa, ha necessitato di maggiore attenzione durante la fase di progettazione per rendere l'architettura la più affidabile e robusta possibile.

L'aspetto critico riguardava il monitoraggio dell'isteresi provocata dall'ingresso del liquido all'interno della caldaia. Entrambi cicli risentono del fenomeno, per tanto la decisione è ricaduta nel gestire in un unico punto la macchina a stati riguardante l'operatività del ciclo ed introdurre metodi astratti richiamabili da essa. Tali metodi, come ad esempio `predicateCheckState`, sono stati sovrascritti all'interno di `NightModeTask`, al fine di poter riutilizzare il codice implementato all'interno di `NebulizationTask` e contemporaneamente soddisfare le diverse esigenze concernenti il ciclo notturno.
La macchina a stati, a seguito dell'attivazione, si trova nello stato INIT in cui vengono disabilitate uscite digitali potenzialmente abilitate in precedenti operazioni e a seguito si effettua la transizione

allo stato WAIT. La macchina, quindi, attende l'autorizzazione proveniente dall'app Android per procedere con la produzione di fumo sanificante generato attraverso il passaggio del liquido Labiosan all'interno della caldaia. L'uscita dallo stato di WAIT introduce la prima differenze riscontrabile tra le due modalità operativa, infatti durante NebulitionTask le porte deve restare rigorosamente serrate. NightModeTask, invece, richiedere un'esigenze opposta in quanto tale modalità permette la sanificazione di ambienti durante periodi di assenza delle persone.
In virtù di ciò che succede durante lo stato di CHECK, la modalità di sanificazione prosegue l'operatività attivando la pompa e trasferendo il liquido all'interno della caldaia. La pompa resta abilitata fin quando la caldaia non raggiunge temperature inadeguate per svolgere la propria funzione e in tale circostanza avviene la transizione allo stato BOILER_WARM_UP, dove si imposta il timer interno del task a 10 secondi. Al termine, si verifica se la temperatura della caldaia è nuovamente adatta alla produzione di fumo ed in caso affermativo si procede con completamento del ciclo. Ultimata la fase di produzione del fumo, occorre lasciare gli oggetti all'interno ed attendere che l'effetto sanificante completi il suo operato. La fase appena indicata si svolge all'interno dello stato di PARKING_WORKING, in cui si lascia il tempo al fumo di decantare.
Se durante le operazioni appena comunicate non sopraggiungono situazioni di emergenza, il ciclo termina e l'operatore è abilitato ad estrarre gli oggetti situati all'interno della macchina.

Lo scenario di emergenze è il secondo punto in cui le due modalità si differenziano, in quanto `NebulitazionTask` effettua la transizione allo stato di emergenza se durante il ciclo l'operatore intenzionalmente apre le porte. Per il `NightModeTask` la condizione è nuovamente opposta, in quanto produrre agenti sanificanti rivolti alla pulizia dell'ambiente è alquanto controproducente se nessuna delle due porte risulta aperta durante il ciclo.

Completiamo la trattazione inserendo Tabella 2, contenente i colori utilizzati durante il ciclo di sanificazione ed il relativo significato.

Tabella 2 Significato colori ciclo Nebulizzazione e ciclo notturno

Colore	Significato
LIGHTBLUE	Macchina accesa e operativa
YELLOW	Impossibile proseguire per porte aperte
WHITE	Produzione fumo sanificante
BLUE	Ciclo entrato nello stato di PARKING, in cui il fumo completa la propria funzione
PURPLE	Attesa che la caldaia torni ad una temperatura adeguata per la produzione di fumo
RED	Interruzione causata da emergenza
GREEN	Ciclo completato correttamente

Quanto illustrato nelle precedenti fasi è parzialmente visibile tramite Figura 9.

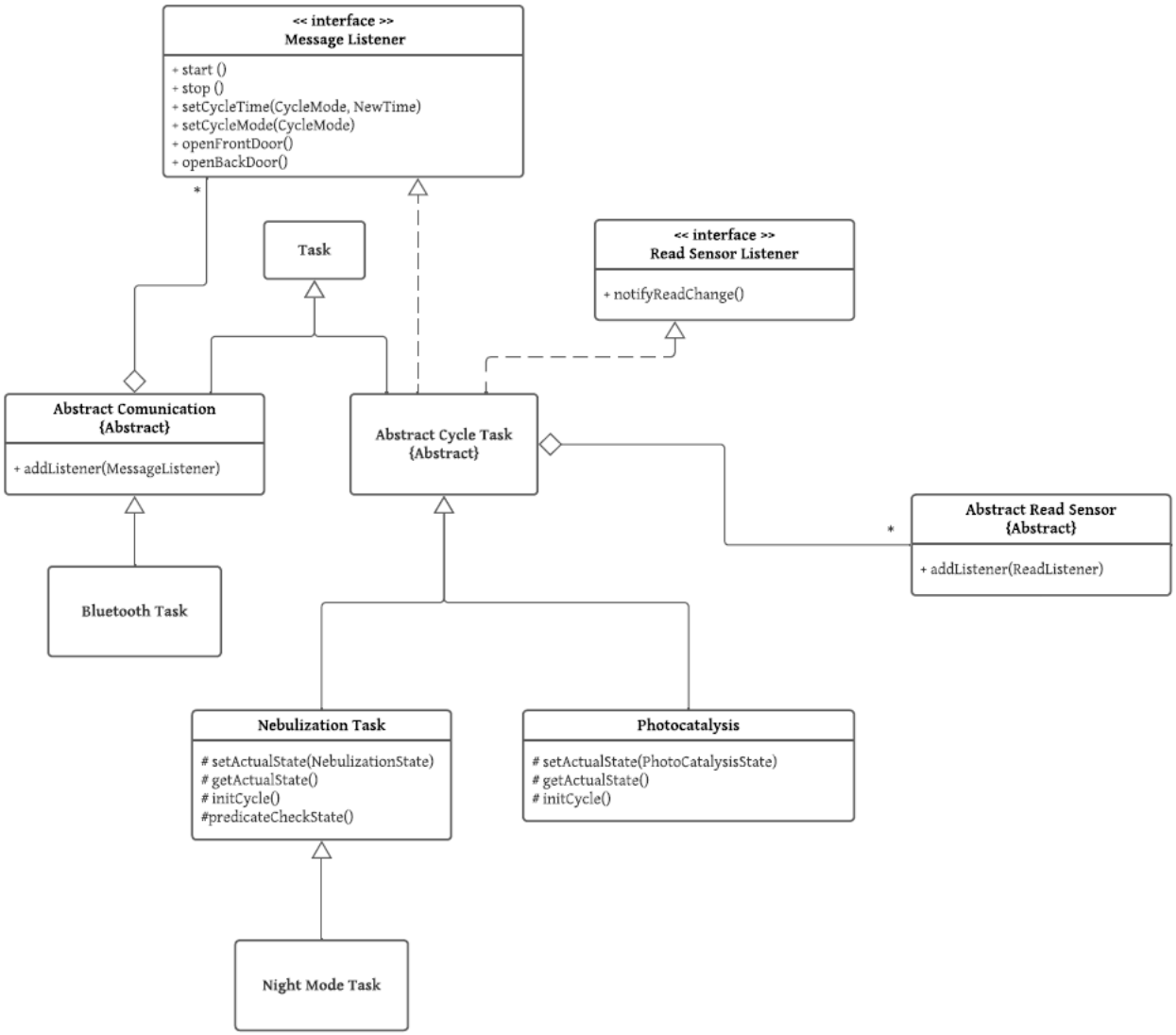

Figura 9 UML Task principali

Rappresentazione Schema elettrico

IEN Industrie dispone di un apposito reparto dedicato alla progettazione e realizzazione di quadri e circuiti elettrici indispensabili per il funzionamento di qualsiasi macchina industriale. A prescindere da questa considerazione per poter realizzare questa tipologia di applicazione è fondamentale la stretta collaborazione tra i vari reparti, quindi anche tra programmatori ed elettricisti.
La discussione non si soffermerà sulle procedure utilizzare per la costruzione del piccolo quadro elettrico, bensì verranno effettuate considerazioni sui segnali ricevuti dalla scheda Arduino.
In particolare, ci soffermeremo brevemente sull'utilizzo di resistenze di pull-down per "disciplinare" il segnale proveniente dai read collocati sulle porte. L'inserimento della resistenza di pull-down è reso necessario in quanto il read apre il circuito quando vengono aperte le porte e conseguentemente l'elettronica del microcontrollore di fronte a tale avvenimento non riesce ad interpretare il corretto valore e attribuirebbe al pin un valore privo di significo.

La resistenza di pull-down invece permette al pin di ricevere in qualsiasi momento un valore. Quando il read chiude il circuito, il pin riceve un livello logico alto, in maniera complementare, quando il read apre il circuito, il pin riceve un livello logico basso in quanto si trova direttamente collegato con il GND.

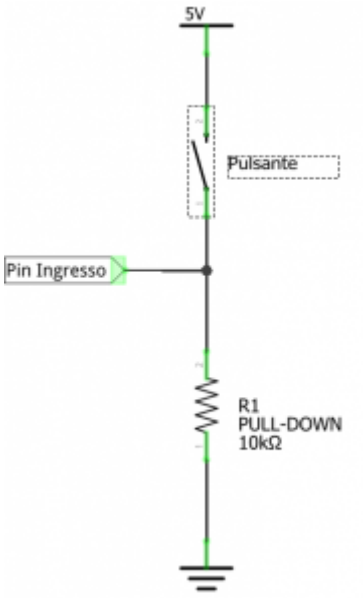

Figura 10 Rappresentazione resistenza di Pull-down

Il progetto possiede anche la relativa rappresentazione realizzata tramite il tool Fritzing. La suddetta rappresentazione è contenuta in allegato alla presente relazione.

Applicazione Android

All'interno dei precedenti capitoli è indicato il funzionamento della macchina e come ciò sia reso possibile tramite i dispositivi che la compongono.
Il percorso si conclude con il seguente capitolo nel quale verranno introdotte le modalità con le quali l'operatore, o l'utente generico, può interfacciarsi con la macchina. Generalmente, in ambito industriale, si utilizzano HMI (Human-Machine Interface) acquistati da produttori specializzati. Per questo progetto si è deciso di introdurre una app per smartphone al fine di rendere l'esperienza d'utilizzo più semplice e piacevole, concepire un design smart e contestualmente abbattere il prezzo di realizzazione eliminando l'HMI, il quale tendenzialmente ha prezzi molto più onerosi.

L'applicazione si compone di tre differenti schermata, che vengono chiamate tecnicamente Fragment. Esiste un'unica Activity principale, la quale conterrà le fragment pocanzi citate, le quali saranno intercambiate secondo la volontà dell'utente, quindi solo una alla volta verrà visualizzata nel display dello smartphone. L'utente ha la possibilità di variare il fragment d'interesse attraverso un apposito menù in cui vengono riportati i titoli delle sezioni corrispondenti ai fragment implementati.

L'implementazione software, realizzata attraverso l'IDE "Android Studio", è avvenuta utilizzando il linguaggio di programmazione Java, nativo per i sistemi Android.
I programmatori del suddetto linguaggio prediligono la creazione delle componenti grafiche per via

programmatica, ovvero con codice sorgente, quindi senza avvalersi di tools o strumenti che generano autonomamente il codice a seguiti di un Drag&Drop di Widgets.

In ambito Android però, tale filosofia, non viene sempre applicata; infatti, l'uso comune è quello di creare GUI tramite un file XML e identificarlo come risorsa dell'applicazione.

Di seguito illustriamo i tre Fragment realizzati per l'applicazione e le relative caratteristiche principali:

- Home: costituita 4 tasti. La pressione del singolo tasto generava un evento, il quale seguirà un semplice iter. In primo luogo, si verifica che il dispositivo abbia precedentemente istaurato una connessione Bluetooth con la macchina, in caso affermativo viene inviato il relativo comando alla macchina.
- Device: la pressione del tasto collocato in alto al Fragment permette di scansionare i dispositivi Bluetooth nelle prossimità dello smartphone. Tali dispositivi vengono inseriti all'interno dell'elenco sottostante. L'utente cliccando sul dispositivo, contenuto nell'elenco, esprime l'intenzione di collegarsi con esso, di conseguenze viene avviata la procedura di connessione. L'esito di tale procedura viene comunicato all'utente attraverso un Toast (componente grafico di Android) sulla zona inferiore del display.
  Se la procedura di accoppiamento ha esito positivo, l'istanza BluetoothDevice viene memorizzata, grazie al Pattern Singleton, e resa disponibile in qualsiasi momento anche agli altri due Fragment.
- Setting: il Fragment è diviso in due zone. La prima permette di selezionare la modalità che si desidera effettuare, mentre la seconda consente all'utente la modifica dei tempi dei cicli macchina. Entrambe le modifiche vengono trasmesse tramite il collegamento Bluetooth precedentemente creato.
  L'implementazione sviluppata ha sempre avuto come prerogativa un'adeguata architettura e per tale motivo le modifiche dei tempi, come la memorizzazione della modalità selezionata, passano rigorosamente tramite un'istanza dell'interfaccia Model.

Figura 11 Screenshot Home Fragment

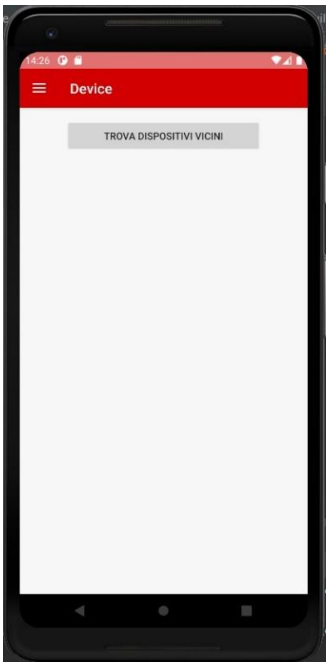

Figura 12 Screenshot Device Fragment

Figura 13 Screenshot Setting Fragment

La comunicazione tramite la connessione Bluetooth è l'aspetto cardine del progetto e indispensabile per l'interazione con la macchina. Di conseguenza, per adempiere a tale compito è stato necessario realizzare una classe dedicata esclusivamente alla funzione pocanzi menzionata. Poiché la comunicazione veniva utilizzata a seguito di un determinato evento, si è deciso di rappresentare ognuno di essi attraverso l'elemento di un enum. Ogni elemento dell'enumeratore BluetoothCommand, è rappresentato dal relativo comando da inoltrare all'interno del canale ed inoltre permette di comporre o completare il comando qualora fosse necessario comunicare un valore scelto dinamicamente. Un pratico esempio è rappresentato dalla designazione di un nuovo tempo a un ciclo macchina.

Prima di concludere analizziamo cosa avviene al termine di ogni lavorazione da parte della macchina. Come qualsiasi buona app che si rispetti, al termine di ogni lavorazione, il dispositivo comunica tale evento all'app, la quale prontamente creerà un NotificationCompat, ovvero la classica notifica che compare nella Home o nella schermata di blocco di qualsiasi smartphone.

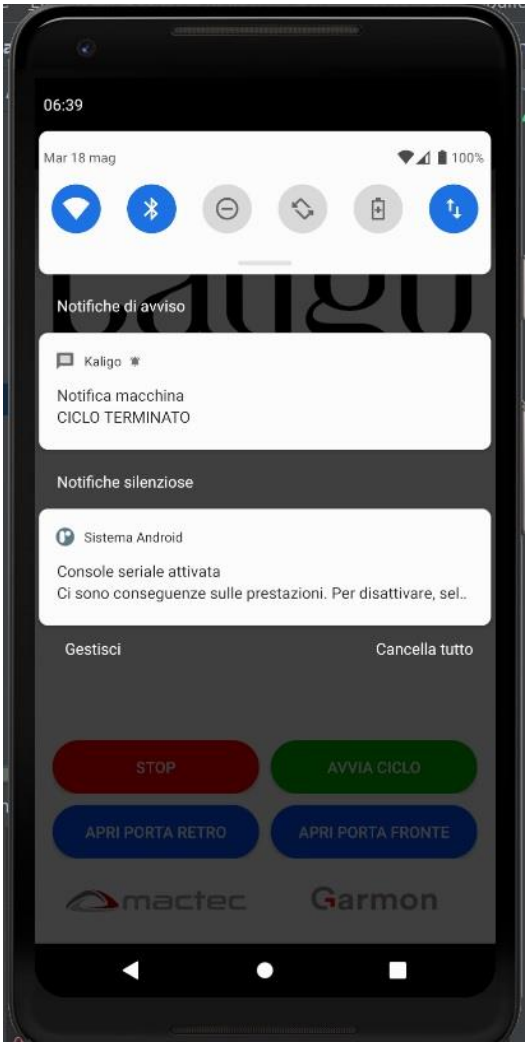

Figura 14 Screenshoot Notifica

Tempi di realizzazione

Come indicato nell'introduzione della trattazione, i tempi di progettazione e realizzazione non potevano essere eccessivi al fine di poter presentare al mercato, quanto prima, i prodotti.

Nel complesso sono state necessarie 3 settimane per decidere e definire i prodotti che l'azienda avrebbe realizzato e contestualmente suddividere il lavoro tra i vari collaboratori a seconda dell'area professionale di competenza.

Condividendo il proseguimento dei lavori per quanto riguarda il reparto software, è possibile affermare che è stato necessario utilizzare:

- Un'intera settimana lavorativa per la progettazione e modellazione dei 2 applicativi rivolti al progetto Kaligo. In questa fase sono stati realizzati documenti di progetto e schemi di modellazione volti alla maggiore comprensione del progetto e all'individuazione di potenziali problematiche.
  Allo stesso tempo è stato necessario interloquire con la parte elettrica per definire quali oggetti utilizzare all'interno del prototipo e che configurazione e pinout applicare
- Circa un mese per la realizzazione dell'applicazione Android e per il microcontrollore Arduino.
  Proseguendo dalle fasi precedenti, la produzione software non ha riscontrato grandi problemi.

Conclusioni

Difficoltà incontrare e riflessioni conclusive

È veramente fortunato chi non incontra difficoltà durante la realizzazione di qualsiasi progetto, dal più banale al più complesso.

L'ironica, per quanto onesta, verità pocanzi riportata non è stata smentita per il progetto posto sotto analisi.
Le difficoltà riscontrate non sono attribuibili al vero è proprio progetto, bensì al contesto nel quale è stato concepito. Infatti, in un momento in cui molti si sono gettati in un settore con la sola finalità di trarre profitto, abbiamo incontrato numerosissime realtà che affermavano di disporre di prodotti adeguanti alla lotta contro il COVID-19, ma in realtà rifilavano tutt'altro. Quello che tendenzialmente si verificava era la promozione e vendita di prodotti sanificanti con efficacia del 99.5% (o del 99.8%) che potrebbe risultare adeguato ad occhi inesperti, ma che in realtà non rappresenta una corretta contromisura nei confronti del COVID. I presidi medici chirurgici, regolamentati da decreti sempre più stringenti, indicano che i prodotti conformi all'eliminazione del virus Sars-CoV-2 devono rispettare una scala logaritmica di base 10, il cui risultato permette di eliminare almeno il 99.99% dei batteri e virus.

Concludiamo enunciando un'altra grossa difficoltà incontrata verso la fine del progetto: la crisi dei semiconduttori.
La crescente richiesta di dispositivi per il lavoro, la didattica, assistenza remota ha portato ad una massiccia riduzione a livello globale delle scorte di chip, processi e schede madri, con il conseguente aumento dei tempi di consegna da parte di molti fornitori.

Idee per il futuro

La situazione continua ad essere ancora incerta anche se i vaccini introdotti stanno portando oggettivi risultati contro la lotta al COVID.

IEN intende proseguire la promozione e vendita dei suoi prodotti ponendo sempre più attenzione alle mutevoli esigenze dei suoi clienti e a rispettarli attraverso la vendita esclusiva di prodotti conformi alle normative italiane ed internazionali.

A tal proposito l'azienda a deciso di procedere con la sanificazione esclusivamente attraverso le lampade UVC, le quali si sono dimostrate le uniche efficaci per l'eliminazione di virus e batteri.

Altri cambiamenti sono da individuare nella parte elettrica e nella componentistica utilizzata per la realizzazione del prototipo. Infatti:

- Arduino mega verrà sostituito con Controllino Mini. La scelta è scaturita dall'esigenza di disporre certificazione sia UE che UL (utilizzata negli USA). Inoltre, Controllino mini dispone di sufficienti ingressi e uscite per governare la macchina
- Inserimento di LED dinamici per rendere l'esperienza d'utilizzo ancor più piacevole
- Migrazione del canale di comunicazione. A seguito di successive analisi, il reparto ha optato nell'utilizzare un modulo wifi ESP8266 per connettere applicazione smartphone a controllino
- Il mercato americano ha richiesto discrete volte la creazione dell'app anche per i sistemi iOS. L'azienda intende far fronte alle richieste nei tempi e modi più consoni.

Indice delle Immagini

Figura 1 Andamento Temperatura Caldaia 5
Figura 2 UML classi - ingressi digitali 6
Figura 3 UML classi - ingressi analogici 6
Figura 4 UML classi - uscite digitali 6
Figura 5 Modulo lettura termocoppia Tipo K 7
Figura 6 Isteresi temperatura caldaia 8
Figura 7 UML Gestione Task 8
Figura 8 UML gestione Interrupt 9
Figura 9 UML Task principali 13
Figura 10 Rappresentazione resistenza di Pull-down 14
Figura 11 Screenshot Home Fragment 15
Figura 12 Screenshot Device Fragment 15
Figura 13 Screenshot Setting Fragment 15
Figura 14 Screenshoot Notifica 16

Indice delle tabelle

Tabella 1 Significato colori ciclo fotocatalisi 11
Tabella 2 Significato colori ciclo Nebulizzazione e ciclo notturno 12