Relazione di laboratorio
INFORMATICA

Classe:	4Di	**Alunno:**	BELLAMIA Antonio	**Data:**	10/mag/2024
Titolo dell'esercitazione:					
IMPLEMENTAZIONE IN LINGUAGGIO JAVA DI UN PANNELLO DI CONTROLLO PER UN IMPIANTO DOMOTICO REALIZZATO CON MICROCONTROLLORE ARDUINO					

1. Analizza la Traccia e/o il Problema.

Per quanto riguarda la disciplina di Informatica, nell'ambito dell'area di progetto, il focus principale è sull'implementazione di codice in Java per interagire, attraverso un'interfaccia grafica (GUI), con il microcontrollore Arduino. Il gruppo avrà il compito di svolgere una serie di attività mirate, tra cui l'implementazione delle seguenti funzionalità:

1. Connessione al Microcontrollore Arduino: Utilizzando la libreria "jSerialComm", sarà necessario sviluppare la logica per stabilire una connessione affidabile e stabile con Arduino attraverso la porta seriale. Questo passaggio è fondamentale per garantire la comunicazione bidirezionale tra il software Java e il microcontrollore.

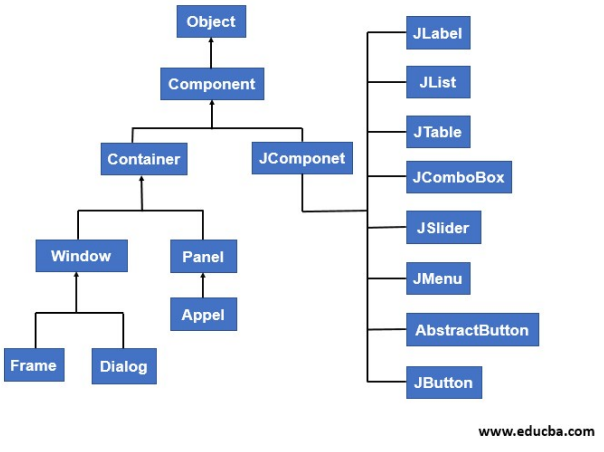
2. Lettura e Scrittura di Dati: Ogni gruppo dovrà creare le procedure necessarie per leggere e scrivere dati da e verso Arduino. Questo può includere il design di protocolli di comunicazione efficienti e sicuri, nonché la gestione degli errori e delle eccezioni durante le operazioni di lettura/scrittura.

3. Visualizzazione dei Dati tramite GUI: Utilizzando Java Swing per la creazione di un'interfaccia grafica intuitiva, sarà importante mostrare i dati provenienti da Arduino in modo chiaro e comprensibile per l'utente finale. Inoltre, la GUI dovrà consentire all'utente di interagire con il microcontrollore, ad esempio modificando le impostazioni o il comportamento del dispositivo in tempo reale.

L'obiettivo primario è quindi quello di creare un'applicazione Java robusta e funzionale che faciliti l'interazione e il controllo del microcontrollore Arduino attraverso un'interfaccia utente intuitiva.

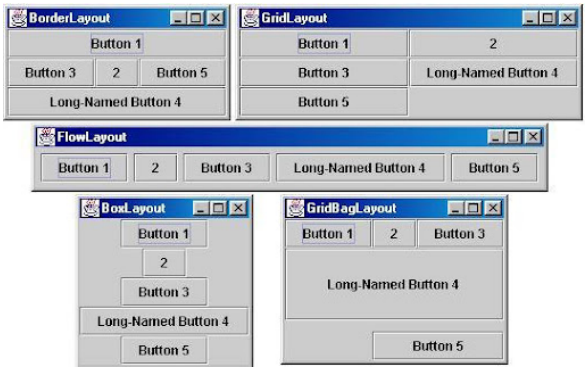
2. Inserisci i Cenni Teorici per rappresentare e/o risolvere il problema.

Per realizzare un'interfaccia grafica con il linguaggio Java ci dobbiamo servire di alcune classi specifiche importate attraverso le librerie Swing (javax.swing) e AWT (java.awt).AWT è la libreria di interfaccia grafica originale fornita con il linguaggio di programmazione Java. Essa fornisce un insieme di classi per la creazione di finestre, bottoni, etichette, campi di testo e altri elementi di interfaccia utente.
AWT utilizza i componenti di interfaccia utente nativi del sistema operativo sottostante, il che significa che le app AWT avranno un aspetto e un comportamento simile all'ambiente operativo in cui vengono eseguite.
Swing, invece, è una libreria di componenti di interfaccia grafica più recente e più flessibile rispetto ad AWT. Essa è stata progettata per essere indipendente dalla piattaforma, il che significa che le app Swing avranno lo stesso aspetto e comportamento su qualsiasi piattaforma Java. Swing fornisce una vasta gamma di componenti di interfaccia utente, tra cui finestre, bottoni, pannelli, barre di scorrimento, caselle di controllo, menu e molti altri. Inoltre, Swing offre un'elevata personalizzazione e flessibilità nell'aspetto degli elementi di interfaccia utente. Ogni componente grafico è di fatto un oggetto, creato a partire da una specifica classe.



Tutte le classi che vanno a definire i componenti dell'interfaccia grafica ereditano dalla classe padre "Component", i metodi comuni forniti da questa classe includono la gestione degli eventi di input, la gestione della grafica e la gestione del layout.
Container è una sottoclasse di Component che rappresenta un contenitore per altri componenti. Questa classe fornisce metodi per aggiungere, rimuovere e disporre i componenti al suo interno. I contenitori possono essere organizzati attraverso i layout manager che determinano come i componenti sono disposti all'interno del contenitore. La classe container viene estesa da Window e da Panel. Le window rappresentano una finestra di livello superiore dell'applicazione. Le finestre sono contenitori specializzati che possono contenere altri componenti di interfaccia utente. Dalla classe window ereditano JFrame e le finestre di dialogo. JFrame è di fatto la finestra che contiene l'intera GUI. JComponent, invece, è una sottoclasse di Container e rappresenta la classe base per tutti i componenti Swing.

Esistono varie classi derivate dalla più generica JComponent, alcune di esse sono:
-JButton: Rappresenta un pulsante cliccabile.
-JLabel: Rappresenta un'etichetta di testo o un'immagine non interattiva.
-JTextField: Fornisce un campo di testo modificabile per l'input dell'utente.
-JTextArea: Fornisce un'area di testo multi-linea modificabile.
-JComboBox: Fornisce una casella a discesa di selezione con opzioni predefinite.
-JCheckBox e JRadioButton: Rappresentano rispettivamente caselle di controllo e bottoni di opzione.
La classe JFrame mette a disposizione diversi metodi come ad esempio il metodo setTitle(String title)con il quale è possibile attribuire un titolo alla finestra. Il metodo setSize(int width, int height) serve a definire la dimensione di una finestra Swing. Con il metodo setDefaultCloseOperation(int operation), è possibile configurare il comportamento predefinito della finestra quando l'utente tenta di chiuderla. Questo metodo offre varie opzioni, consentendo di definire il comportamento desiderato in risposta alla chiusura della finestra. Ad esempio, utilizzando setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE), l'applicazione terminerà automaticamente quando l'utente chiude la finestra principale.Il metodo setVisible(boolean visible) serve a controllare se una finestra deve essere resa visibile o nascosta all'utente. Ad esempio, utilizzando setVisible(true), è possibile rendere visibile una finestra appena creata, consentendo agli utenti di interagire con il suo contenuto. Il metodo add(Component comp) offre la possibilità di arricchire dinamicamente una finestra con nuovi componenti di interfaccia utente. Utilizzando questo metodo, si possono aggiungere bottoni, etichette, campi di testo e altri elementi alla finestra per fornire funzionalità aggiuntive o presentare informazioni all'utente. Infine, il metodo setLayout(LayoutManager manager) offre un controllo preciso sulla disposizione dei componenti all'interno di una finestra Swing. Attraverso l'uso di layout manager, gli sviluppatori possono organizzare i componenti in modo flessibile. Ad esempio, utilizzando un BorderLayout, è possibile assegnare a ciascun componente una posizione specifica all'interno della finestra. Il BoxLayout è un layout progettato per gestire il posizionamento dei componenti in una singola riga o colonna, mentre il GridLayout serve ad organizzare i componenti in una griglia regolare di righe e colonne. Ogni cella della griglia può contenere un singolo componente, e tutte le celle hanno le stesse dimensioni. In CardLayout, invece, è un layout manager che gestisce più componenti sovrapponendoli uno sopra l'altro, consentendo di visualizzarne solo uno alla volta. Funziona come una pila di carte, dove è possibile navigare tra i componenti come pagine di un libro. È utile quando si desidera mostrare diversi pannelli o viste nello stesso spazio e passare da uno all'altro in modo dinamico. Si può navigare attraverso i



componenti usando metodi come next() e previous(), o specificando direttamente il nome del componente da visualizzare. Inoltre abbiamo il
GridBagLayout che permette di organizzare i componenti in una griglia flessibile. Ogni componente può occupare più celle della griglia e avere vincoli di posizione e dimensione personalizzati. Questo layout è estremamente potente e versatile, consentendo di creare interfacce utente complesse e ben strutturate. Puoi specificare come posizionare e ridimensionare i componenti utilizzando oggetti GridBagConstraints, che forniscono dettagliate informazioni sui vincoli di layout. Il GridBagLayout è particolarmente utile quando si desidera un controllo preciso sulla disposizione dei componenti all'interno di un contenitore, adattandosi dinamicamente alle dimensioni della finestra e ai cambiamenti nel contenuto.
Le interazioni degli utenti con l'interfaccia grafica prendono il nome di eventi. Quando un evento viene generato, viene inviato a un oggetto chiamato "listener" che ha registrato interesse per quell'evento specifico. Il listener quindi esegue un'azione o una serie di azioni in risposta all'evento. Ad esempio, un listener di clic su un pulsante potrebbe eseguire un'azione quando l'utente preme il pulsante. Gli eventi vengono gestiti da un listener, implementato con l'interfaccia ActionListener. ActionListener è un'interfaccia in Java utilizzata per gestire gli eventi generati dagli oggetti Swing. Questa interfaccia contiene un solo metodo, actionPerformed(ActionEvent e), che deve essere implementato da qualsiasi classe che desidera gestire gli eventi di azione. Quando un oggetto Swing genera un evento di azione (ad esempio, quando un pulsante viene premuto), viene chiamato il metodo actionPerformed() associato a quell'oggetto. All'interno di questo metodo, il codice per gestire l'azione desiderata può essere inserito.

3. Descrivi la Strategia Risolutiva che hai adottato.

Riflettendo sul progetto, la prima sfida è stata capire come far comunicare in modo efficiente il microcontrollore con l'interfaccia grafica. Demandando tutte le considerazioni riguardanti il protocollo e la scheda Arduino alla relazione realizzata per la disciplina Sistemi e Reti, di seguito verranno indicati i passaggi che ci hanno portato alla realizzazione della GUI e delle classi per l'input/output sul canale seriale.

Per la realizzazione dell'interfaccia grafica del progetto, si è adottato un approccio strutturato che mira a garantire usabilità e coerenza nell'interazione con l'utente. La progettazione dell'interfaccia segue i principi di chiarezza e immediatezza, utilizzando componenti Swing nell'ambiente Java per creare un'esperienza utente intuitiva e reattiva.

- Progettazione dell'Interfaccia: L'interfaccia grafica è stata concepita per fornire un feedback visivo immediato dello stato dei componenti controllati dal microcontrollore Arduino. Si è scelto di utilizzare un layout organizzato in pannelli, ognuno dei quali è dedicato al controllo di specifiche funzionalità del sistema, come la gestione dei LED, del servo-motore e del sensore di luminosità. L'uso di colori e

icone facilita la comprensione dello stato attuale dei componenti, migliorando l'interazione con l'applicazione.

- Gestione della Comunicazione Seriale: La comunicazione seriale tra l'applicazione Java e il microcontrollore Arduino è fondamentale per il trasferimento dei dati. Questa viene gestita attraverso un protocollo seriale che definisce il formato dei messaggi scambiati. Ogni messaggio è composto da un byte suddiviso in 8 bit, dove ciascun bit rappresenta una specifica informazione o comando. Il bit più significativo (MSB) identifica la natura dell'istruzione (lettura o scrittura), mentre i bit successivi codificano il componente di destinazione e lo stato o il valore da trasmettere.

L'implementazione software prevede l'uso di un buffer condiviso e di listener per la ricezione e l'invio dei dati seriali. Il buffer assicura che i messaggi siano correttamente indirizzati ai componenti dell'interfaccia, mentre i listener permettono di gestire gli eventi di comunicazione in modo efficiente. Inoltre, la presenza di un sistema di eccezioni garantisce la robustezza dell'applicazione in caso di errori o disconnessioni.

Il BufferDataListener è un componente chiave del software che permette di collegare l'interfaccia grafica alla logica di ricezione dei dati seriali.
Esso serve alle classi dell'interfaccia grafica come mezzo per ricevere 'notifiche' quando nuovi dati sono disponibili nel buffer.
Le classi che necessitano di ricevere dati dal buffer implementano l'interfaccia BufferDataListener. Quando nuovi dati arrivano nel buffer, il metodo fireBufferDataChange viene chiamato per notificare tutti gli ascoltatori registrati.
Gli ascoltatori possono filtrare gli eventi generati dal buffer utilizzando parole chiave come instanceOf per determinare se il messaggio ricevuto è rilevante per loro.
La classe Buffer è il cuore del sistema di comunicazione. Contiene un array di booleani che rappresenta il messaggio inviato da Arduino e utilizza metodi sincronizzati per gestire l'accesso concorrente alle risorse.
La classe BufferDataComponent è la superclasse di Buffer e permette al buffer di essere "ascoltabile" da vari osservatori, che si possono registrare tramite il metodo addBufferDataListener.

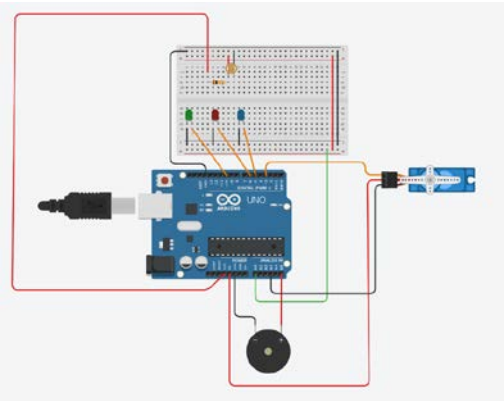
Infine, il SerialCommInputManager è il thread che legge i dati dalla porta seriale e li inoltra al buffer, mentre la classe SendData si occupa di inviare messaggi dalla GUI ad Arduino attraverso la porta seriale. Queste classi lavorano insieme per garantire una comunicazione fluida tra la GUI e il microcontrollore.

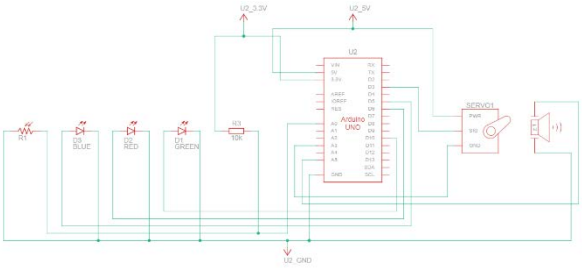
Ogni pannello nell'interfaccia grafica ha un codice univoco basato sui componenti che regola secondo il protocollo. Ad esempio, StatusPanels ha il codice 2, AutosPanel ha il codice 3, LedsPanel ha il codice 0, e ServoPanel ha il codice 1, in più troviamo il FooterPanel che ha il compito di mostrare il tempo di esecuzione e lo stato della comunicazione.

I pannelli sono così organizzati visivamente:

Per tutti i dettagli riguardanti l'interfaccia grafica si consiglia di consultare la documentazione ufficiale del progetto, allegata con il nome di "GRUPPO_A_DOCUMENTAZIONE_PROTOCOLLO_E_SOFTWARE_AREA_DI_PROGETTO .pdf"

4. Elenca gli Strumenti e/o il Materiale che hai utilizzato.





Ci siamo serviti, per la realizzazione del progetto, per quanto riguarda il lato hardware di:

- Scheda compatibile con Arduino Uno R3
- Breadboard
- Fotoresistore (LDR)
- Resistore (10KΩ)
- 3 LED
- Servo-Motore SG-90
- Buzzer attivo (opzionale)
- Cavi

Per quanto riguarda il lato software abbiamo fatto uso di:

- IDE IntelliJ IDEA (per la programmazione in java)
- Arduino IDE (per la programmazione del microcontrollore)

5. Descrivi le eventuali difficoltà che hai riscontrato e, se ci sei riuscito, come le hai superate.

Le maggiori difficoltà che abbiamo riscontrato sono emerse quando la realizzazione del software era quasi conclusa, quindi in fase di test e debug. Abbiamo riscontrato alcuni problemi nell'elaborazione dei dati letti dal seriale da parte dei vari pannelli, causati da sviste in fase di codifica o da mancati aggiornamenti a seguito di cambiamenti nel protocollo. Tutte le difficoltà sono state superate grazie ad una massiva fase di debug che ha portato alla revisione dell'intero codice per verificarne la correttezza.

6. Descrivi in che modo la Soluzione che hai implementato si potrebbe migliorare.

Il sistema presenta un unico punto debole, ovvero l'impossibilità di riconnettersi al seriale a run-time, dopo che la connessione è andata perduta. Ci sono stati vari tentativi di risolvere questo problema, invano. Per ovviare a questo è stata implementata una notevole gestione delle eccezioni definendo stati come "Safe Mode" mostrati nel FooterPanel.

7. Descrivi in cosa senti di essere migliorato grazie a quest'esercitazione.

Come per ogni progetto, posso dire di aver allenato le mie capacità logiche e di problem-solving. In più ho appreso il funzionamento dei Listener personalizzati e della loro creazione ed inoltre ho acquisito conoscenze in merito al funzionamento della comunicazione tramite una porta seriale da un'ambiente Java da e verso l'esterno.