Relazione di laboratorio
SISTEMI E RETI

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 Sistemi e Reti, nell'ambito dell'area di progetto, il focus principale è sull'implementazione di un protocollo di comunicazione seriale per permettere lo scambio di byte tra il microcontrollore Arduino e l'applicazione realizzata in Java. Il gruppo avrà il compito di svolgere una serie di attività mirate, tra cui l'implementazione delle seguenti funzionalità:

1. Connessione tramite Porta Seriale: Una delle prime attività sarà quella di stabilire una connessione affidabile tra il microcontrollore Arduino e l'applicazione Java tramite porta seriale. Questo richiede la scrittura di codice in grado di gestire la comunicazione bidirezionale tra i due dispositivi, garantendo stabilità e sicurezza durante lo scambio di dati.

2. Lettura e Scrittura dei Dati: Ogni gruppo dovrà creare le procedure necessarie per leggere e scrivere dati da e verso l'applicazione Java. Questo implica la progettazione di algoritmi efficienti per l'acquisizione e la trasmissione dei dati, nonché la gestione di eventuali errori durante le operazioni di lettura/scrittura.

3. Implementazione di un Protocollo di Comunicazione: Un'altra sfida significativa sarà l'implementazione di un protocollo di comunicazione tra il microcontrollore Arduino e l'applicazione Java. La complessità di questo protocollo è a discrezione di ciascun gruppo, ma sarà fondamentale progettare una soluzione che garantisca una comunicazione fluida e affidabile tra i dispositivi.

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

Arduino è un'azienda hardware, open source, che produce microcontrollori programmabili.

Arduino serve principalmente per creare dei prototipi, che una volta testati, potrebbero essere implementati, dalle aziende, in circuiti più piccoli e specifici per il caso. Arduino inoltre può risultare utile per l'insegnamento delle materie STEM nelle scuole e rappresenta un ottimo hobby per gli appassionati del settore.

Il progetto si basa su una scheda programmabile, attraverso il proprio ambiente di sviluppo, su un computer. Il linguaggio di programmazione è molto simile al C++.

Arduino serve per creare delle automazioni che, basate sui dati raccolti dall'esterno, agiranno in un senso o in un altro. I dispositivi che permettono di interfacciarci con il mondo reale, sono i sensori. Un sensore per definizione è un "Dispositivo meccanico, elettronico o chimico, che in apparecchiature o meccanismi rileva i valori di una grandezza fisica e ne trasmette le variazioni a un sistema di misurazione o di controllo".

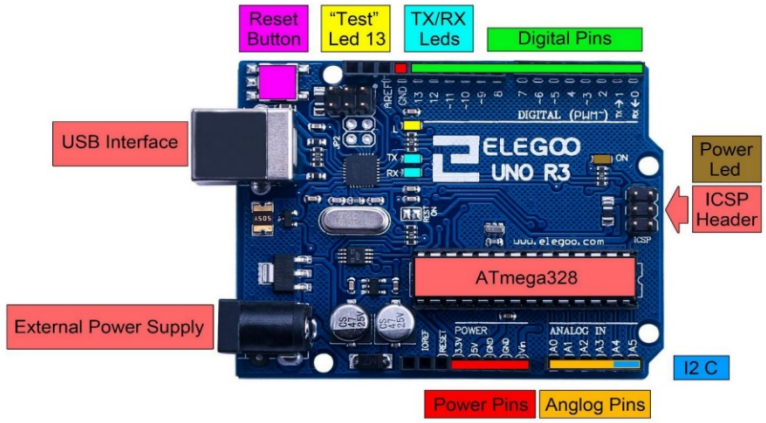
La realtà in cui viviamo, è una realtà analogica, dove tra due valori ce ne sarà sempre un altro, matematicamente in un intervallo compreso tra $-\infty$ e $+\infty$. Esistono per questo i sensori analogici, che 'delegano', alla parte software, la gestione digitale dei dati. Oltre ai sensori analogici però, esistono anche i sensori digitali, che riescono a rendere discreta l'informazione, fornendo al microcontrollore un segnale input digitalizzato.

L'utente si aspetta, come è giusto che sia, una risposta del nostro 'automa' agli input. Esistono quindi dei dispositivi di output, che spaziano dai LED ai Display, dai buzzer ai componenti motorizzati. Si aggiungono ai componenti di input e output, anche gli elementi circuitali, come i resistori, i condensatori, i transistor o i diodi.

Da questa panoramica generale, possiamo capire come, utilizzando un microcontrollore del tipo Arduino, possiamo creare innumerevoli progetti, adatti a tutte le nostre idee.

Arduino, come detto in precedenza, è Open Source, il che permette alle aziende di creare il proprio microcontrollore e perfezionarlo secondo i propri criteri. Anche le creazioni basate su Arduino saranno quindi libere e fruibili a tutti.

L'azienda cinese ELEGOO, dal 2015 produce microcontrollori e kit di sensori basati sul modello Arduino. Le schede ELEGOO sono totalmente compatibili con l'Arduino 'originale' e sono meno costose. La nostra scuola è dotata di kit ELEGOO con microcontrollore UNO R3, che abbiamo utilizzato per l'area di progetto.



Innanzitutto, il programma viene caricato sulla scheda tramite una porta USB di tipo B, che fornisce anche l'alimentazione necessaria alla scheda. Il programma viene caricato nella piccola memoria del microcontrollore, dove permane fin quando non viene sovrascritto. Una volta memorizzato il programma, potremmo decidere di fornire l'alimentazione elettrica necessaria tramite un "Barrel Jack", connesso alla porta apposita. Il programma viene elaborato dal microprocessore ATmega328 (Che lavora alla frequenza di clock di 16Mhz). Come abbiamo detto in precedenza, per questo tipo di progetti avremo bisogno dei sensori. Sulla scheda troviamo 14 pin input/output digitali e 6 pin di input analogico. Questi ultimi pin, possono essere usati come digitali, nel caso vengano dichiarati, nel programma da eseguire, come pin 14, 15, 16, 17, 18 o 19. I pin digitali 0 e 1, contrassegnati anche come RX e TX, servono per la trasmissione seriale dei dati, quando è presente un monitor seriale. I pin 3, 5, 6, 9, 10 e 11 sono definiti PWM, cioè in grado di manipolare il segnale e generarne uno ad onda quadra con "duty cycle" variabile. Esistono poi dei pin prettamente dedicati alla funzione elettrica. Accanto ai pin digitali ne troviamo uno GND, o di messa a terra. Nella parte inferiore ci sono due pin GND, uno che eroga una tensione elettrica di 5 Volt, uno che eroga una tensione elettrica di 3,3 Volt e il pin Vin. Quest'ultimo pin serve per fornire al microcontrollore un ulteriore afflusso elettrico attraverso una batteria esterna.

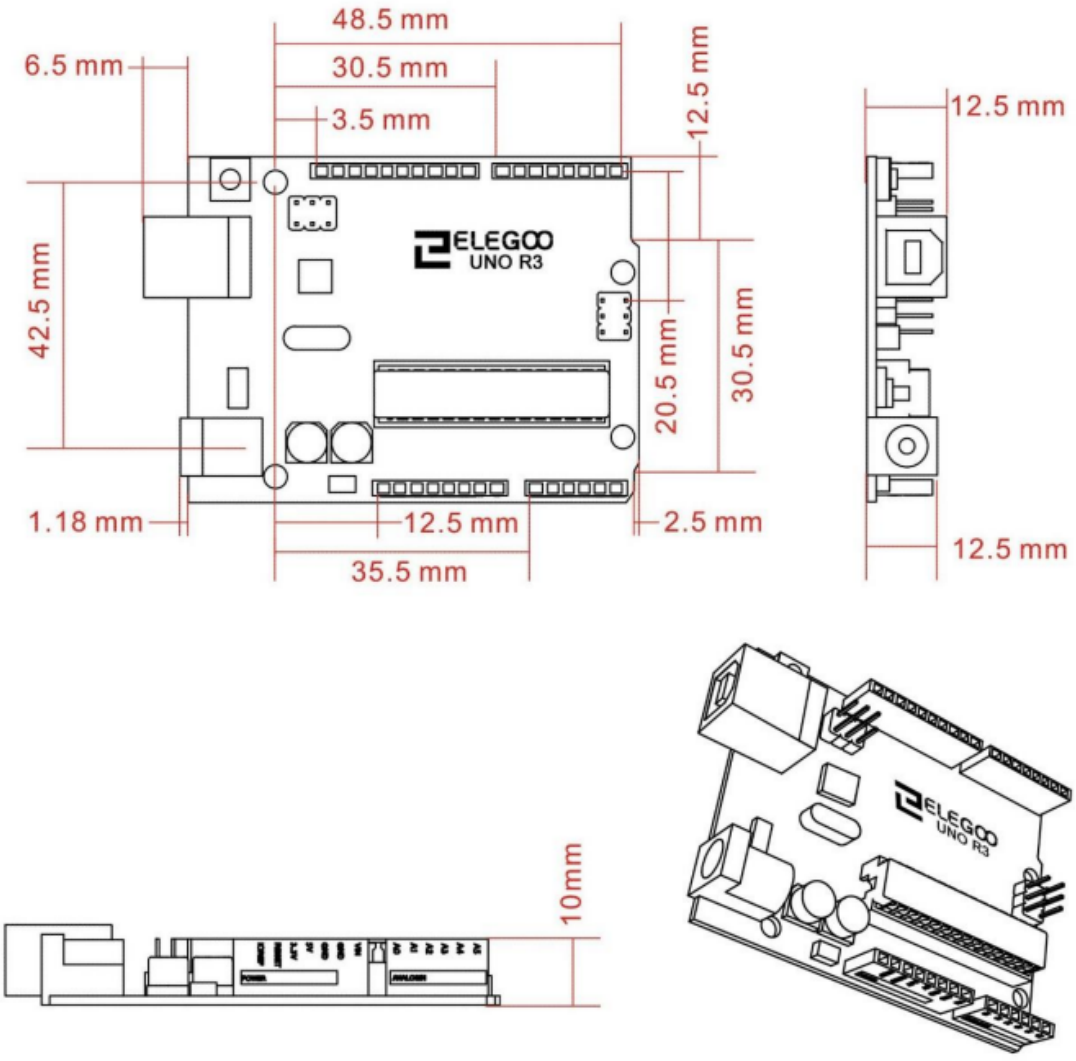
Troviamo poi 4 led sulla scheda. Il LED di test, indicato con ‘L’, lampeggia quando il pin 13 è impostato su HIGH: ci serve per capire se la scheda funziona correttamente.

I led TX e RX lampeggiano quando sono in corso attività di trasmissione o ricezione di dati, in seriale, tramite la porta USB-B collegata al computer. Lavorano ovviamente con i pin 0 e 1.

Il LED ‘ON’ serve per capire se la scheda è alimentata (led attivo) oppure è spenta.

Possiamo notare anche un pulsantino rosso, vicino alla porta USB, quello di “Reset”. Questo pulsante, se premuto, disattiva tutte le periferiche e riavvia l'esecuzione del programma caricato. Il fenomeno è lo stesso che si ottiene togliendo e ricollegando l'alimentazione della scheda.

Nella prossima figura, sono mostrate le dimensioni della scheda.



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. I due componenti dovevano scambiare messaggi ognuno di un byte secondo un protocollo di comunicazione da noi definito. Dopo una serie di iterazioni siamo giunti alla realizzazione definitiva del protocollo che può essere riassunta dalla seguente tabella:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
L/S	*LED/MOT/SENS/AUTO*						
S	LED		ID LED*		OFF/ON/AUTO**		0
L	LED		ID LED*		AUTO 0/1	ON/OFF	0
S	MOTORINO		OFF/ON*	VALUE**			
L	MOTORINO		AUTO 0/1	ROT VALUE			
L	SENS		REALTIME VALUE%				
S	SENS		SET MINIMUM VALUE% (vedi tabella)				0
L	AUTO/SENS		GET MINIMUM VALUE% (vedi tabella)				0
S	AUTO		1	1	1	1	1

Il byte è stato suddiviso in 8 bit ed ogni bit rappresenta un'informazione comprensibile sia dal microcontrollore che dal software Java.
Di seguito sono riportate alcune caratteristiche fondamentali per comprendere il funzionamento del protocollo:

- Lettura e Scrittura: Il bit più significativo indica se il messaggio è di lettura (messaggio inviato da Arduino verso Java) o di scrittura (messaggio inviato da Java verso Arduino). Arduino ignora i byte con MSB a 0, mentre Java ignora quelli con MSB a 1.
- Codice Univoco per i Componenti: I bit 6 e 5 sono usati per codificare i componenti del sistema (LED, Servo-Motore, sensore di luminosità, automazione), assegnando a ciascuno un codice binario univoco.
- Messaggio Associato ai LED: I bit 4 e 3 identificano il singolo LED, mentre i bit 2 e 1 indicano lo stato del LED (spento, acceso, automatico)[3].
- Messaggio Associato al Servo Motore: Il bit 4 indica se l'automazione è attiva, e i restanti bit rappresentano il grado di rotazione del servo-motore.
- Messaggio Associato al Sensore di Luminosità: Differisce tra scrittura (impostazione del valore minimo di luminosità) e lettura (valore attuale di luminosità ambientale).
- Modifica Valore Minimo: La GUI invia ad Arduino un valore tra 0 e 10 per impostare la soglia minima di luminosità. Arduino utilizza una tabella di corrispondenza per stabilire la luminosità effettiva.
- Invio Valore Luminosità: Arduino invia alla GUI il valore della luminosità ambientale, espresso in una scala da 0 a 31, che la GUI converte in percentuale.
- Conferma Valore Minimo: Arduino conferma alla GUI la ricezione del valore minimo di luminosità, utilizzando la stessa tabella di corrispondenza del punto precedente.
- Attivazione Automazione: Un messaggio speciale con tutti i bit impostati su 1 attiva l'automazione per tutti i componenti del sistema.

- Verifica Connessione: Arduino invia periodicamente il valore della luminosità alla GUI come test di connessione. Se la GUI non riceve dati, si presume che la connessione seriale sia interrotta.

Questo protocollo permette una comunicazione efficiente e strutturata tra i componenti hardware e il software, garantendo un controllo preciso e personalizzabile del sistema.

Per tutti i dettagli riguardanti il protocollo seriale, si invita a consultare la documentazione ufficiale del progetto, allegata con il nome di GRUPPO_A_DOCUMENTAZIONE_PROTOCOLLO_E_SOFTWARE_AREA_DI_PROGETTO.pdf

Il software in Arduino è progettato per gestire un sistema di illuminazione e un motore servo basato su input da un sensore di luminosità e comandi dalla porta seriale. Si procede ad una breve analisi:

Inizialmente, vengono dichiarate le costanti e le variabili globali necessarie, insieme alle impostazioni dei pin e alla configurazione dei LED, del buzzer e del servo motore.

Nella funzione setup(), vengono impostati i pin come input o output e inizializzati i componenti come il sensore, il buzzer e il servo. Viene anche inizializzata la connessione seriale e disattivati eventuali suoni del buzzer.

Le funzioni setBrightness() e setServoRotation() vengono utilizzate per controllare la luminosità dei LED e la rotazione del servo motore, rispettivamente, in base agli input ricevuti. La funzione eseguiScrittura() instrada i comandi ricevuti dalla porta seriale alle funzioni corrispondenti.

Le funzioni leggiSeriale() e sendData() gestiscono la comunicazione seriale tra il software e altri dispositivi, inviando e ricevendo dati.

La funzione loop() è il cuore del programma, dove viene continuamente letto il valore del sensore di luminosità, inviato alla GUI, gestiti i comandi dalla seriale e controllati i componenti di illuminazione e servo motore in base alle condizioni.

Inoltre, ci sono funzioni per inviare i dati di luminosità, luminosità minima e rotazione del servo motore alla GUI e per gestire il controllo automatico della luminosità dei LED e della rotazione del servo motore.

Quando viene ricevuto un byte dalla porta seriale, il software elabora questo byte bit per bit. Il byte ricevuto viene diviso in 8 bit, e ognuno di questi bit viene elaborato separatamente.

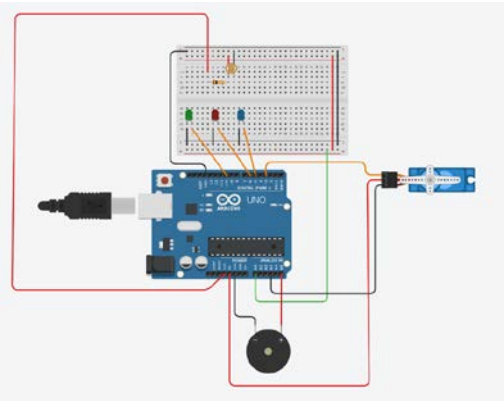
Il byte ricevuto è memorizzato come un array di booleani chiamato bits, dove ogni elemento dell'array rappresenta uno dei bit del byte. Questo array viene popolato utilizzando la funzione bitRead() per leggere ogni bit del byte ricevuto.

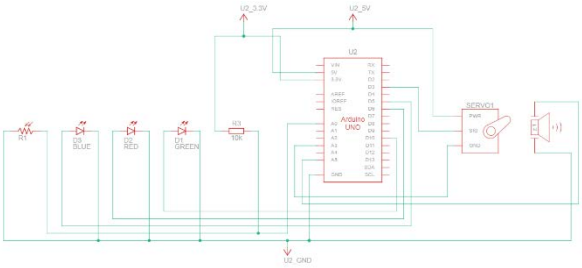
Una volta che l'array bits è stato popolato con i bit del byte ricevuto, viene eseguita una serie di operazioni per interpretare il significato del byte ricevuto. Questo viene fatto principalmente nella funzione eseguiScrittura(), che esegue un'operazione specifica in base ai primi due bit del messaggio ricevuto.

La funzione eseguiScrittura() è responsabile di interpretare il byte ricevuto dalla porta seriale e di instradare il programma verso le operazioni appropriate in base ai primi due bit del messaggio ricevuto. Di seguito è indicato nel dettaglio il suo funzionamento.

- Lettura del byte: La funzione inizia leggendo il byte ricevuto dalla porta seriale.
- Decodifica dei primi due bit: I primi due bit del byte ricevuto determinano il tipo di operazione da eseguire. Questi due bit vengono letti e interpretati come un valore numerico che determina il tipo di azione da intraprendere.
- Instradamento verso l'operazione corretta: Utilizzando il valore numerico ottenuto dalla decodifica dei primi due bit, la funzione dirige il programma verso l'operazione corretta. Questo avviene tramite un'istruzione switch-case che confronta il valore numerico con possibili casi predefiniti. Ogni caso corrisponde a un tipo di operazione da eseguire, come ad esempio impostare le caratteristiche dei LED, del servo, il valore di luminosità minima o l'automazione del sistema. Ad esempio, se i primi due bit indicano "00", viene eseguita un'operazione per impostare le caratteristiche dei LED; se indicano "01", viene eseguita un'operazione per impostare le caratteristiche del servo; se indicano "10", viene eseguita un'operazione per impostare il valore di luminosità minima; se indicano "11", viene eseguita un'operazione per impostare l'automazione del sistema.
- Esecuzione dell'operazione corrispondente: Una volta identificata l'operazione da eseguire, la funzione passa il controllo al blocco di codice corrispondente. Ad esempio, se il valore numerico corrisponde all'operazione per impostare le caratteristiche dei LED, viene chiamata la funzione setLeds(). Se corrisponde all'operazione per impostare il servo, viene chiamata la funzione setServo(), e così via.

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 da inviare all'applicazione java, causati da sviste in fase di codifica o da mancati aggiornamenti a seguito di cambiamenti nel protocollo. Tutte le difficoltà sono state superate grazie alla 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.

Considerata la lentezza delle operazioni di lettura ed invio dati da parte di Arduino, data da una probabile congestione del canale seriale, si potrebbe implementare una soluzione che permetta di inviare valori che differiscono da quelli precedentemente inviati, evitando di inviare informazioni già conosciute dalla GUI.

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 della comunicazione seriale verso l'esterno.