Controllo di un Motore DC con MOSFET e Potenziometro (Progetto in 3 file: `Project.ino`, `MotorController.h`, `MotorController.cpp`)

Corso di Robotica – Istituto Tecnico

Riepilogo del progetto

Obiettivo: realizzare un sistema per il controllo di un motore DC usando un MOSFET di potenza (N-channel), PWM da Arduino e un potenziometro per il controllo manuale della velocità. L'attività è strutturata in molte fasi: prima si costruisce e verifica il circuito con un **LED** al posto del motore; successivamente si sostituisce il LED con un **motore alimentato a** 12 V. Il software sarà organizzato in tre file:

- `Project.ino` — codice principale (setup loop) e istanziazione della classe;
- `MotorController.h` — dichiarazione della classe `MotorController`;
- `MotorController.cpp` — implementazione della classe `MotorController`.

Panoramica delle fasi

Ho suddiviso il lavoro in molte fasi, ciascuna con obiettivi chiari: materiali, test LED, software, test potenziometro, inserimento motore 12 V, misure, miglioramenti, documentazione finale.

Fase 0 — Gruppi e Materiali

Formate gruppi da 4 persone. Scegliete una postazione comoda per lavorare (si possono utilizzare anche i computer portatili). Recuperate tutto il materiale necessario:

- Alimentatore (chiedere al docente), cavo di alimentazione, cavi banana-coccodrillo rosso e nero;
- Arduino, cavo USB, jumper vari e breadboard;
- LED, 2 resistori da 1 kΩ, MOSFET IRFZ44N, diodo 1N4007, motore DC (chiedere al docente), potenziometro;
- Quaderno per lo schema elettrico.

Fase 1 — Disegno del circuito

Sul quaderno, disegnate il circuito per il LED. Il controllo deve avvenire tramite un **MOSFET** collegato all'Arduino. Il LED servirà come simulazione del motore; il principio di funzionamento rimane identico: il MOSFET permette di gestire il flusso di corrente verso il carico senza far passare direttamente corrente attraverso l'Arduino.
Nel circuito, aggiungete un **potenziometro** collegato a un pin analogico dell'Arduino. Questo servirà per regolare la luminosità del LED o la velocità del motore, modulando il segnale PWM inviato al Gate del MOSFET.
Disegnate un simbolo semplice del MOSFET, indicando chiaramente i tre terminali: **Gate**, **Drain** e **Source**.

- **Gate**: riceve il segnale dall'Arduino per controllare il MOSFET;
- **Drain**: collegato al carico (LED);
- **Source**: collegato alla massa o al negativo dell'alimentazione.

Annotate anche la corrispondenza tra i terminali simbolici e i pin reali del componente, così da poterlo collegare correttamente nel circuito fisico. In questo modo, il disegno sarà completo e pronto come riferimento pratico per il montaggio.

Fase 2 — Montaggio del circuito di test con LED

- Inserire il MOSFET sulla breadboard.
- Collegare il **Gate** del MOSFET a un resistore da 1 kΩ, quindi a un pin digitale PWM dell'Arduino (es. D9).
- Collegare il **Source** del MOSFET al **GND** della breadboard.
- Collegare il **GND** dell'alimentatore a 12 V allo stesso **GND** della breadboard.
- Collegare il **Drain** del MOSFET al terminale negativo del LED.
- Collegare il terminale positivo del LED a un resistore da 1 kΩ, quindi al **+12 V** dell'alimentatore.
- Inserire il potenziometro sulla breadboard.
- Collegare un terminale laterale del potenziometro al **GND** della breadboard.
- Collegare l'altro terminale laterale del potenziometro al **+5 V** dell'Arduino.
- Collegare il terminale centrale (wiper) del potenziometro a un pin analogico dell'Arduino (es. A0).

Fase 3 — Software preliminare (test PWM su LED)

Scrivere un programma semplice (senza programmazione ad oggetti) per far lampeggiare il LED tramite comandi digitali alti e bassi, in modo da testare il corretto funzionamento del MOSFET e del collegamento al pin PWM.

Fase 4 — OOP | Programmazione ad oggetti

Creare i 3 file spiegati nella sezione **Riepilogo del progetto** del seguente documento e scrivere il codice per gestire innanzitutto il funzionamento del MOSFET creando una classe così strutturata:

```
class CARICO_IN_POTENZA{
    private:
        // Vostro codice
        // Implementate lo stato del carico
        // così da sapere se è acceso o spento
    public:
        CARICO_IN_POTENZA(...){
            // Logica del costruttore
        }

        // Metodi da implementare

        void accendi(){}      // Accende con un segnale HIGH
        void spegni(){}       // Spegne con un segnale LOW
        void on_dimmer(...){} // Accende con un segnale analogico 0-255
};
```

E la creazione dell'oggetto verrá fatta in questa maniera:

```
CARICO_IN_POTENZA led(9); // creazione esempio dell'oggetto
```

Nella `void loop(){}` inserire un esempio di codice che esegua le seguenti operazioni:

1. Accende il LED.
2. Attende 2 secondi.
3. Spegne il LED.
4. Attende 2 secondi.
5. Accende il LED con una luminosità del 40%.
6. Attende 2 secondi.
7. Spegne nuovamente il LED.
8. Attende infine 5 secondi.

Fase 5 — Miglioramento software: `soft_start`

All'interno della classe, creare il metodo:

```
void soft_start(int val)
```

Se il carico è spento, il metodo deve accendere il carico in modo graduale, variando la luminosità da 0 fino al valore `val`. Se il parametro `val` non viene specificato, il metodo deve comunque accendere completamente il carico, assumendo val = 255.

Fase 6 — Implementazione del potenziometro nel codice

Modificare la classe creata in precedenza permettendo la lettura del valore del potenziometro. Il valore che deve restituire deve essere un valore tra 0 e 100 (intendasi percentuale). Le modifiche da effettuare al codice saranno:

```
class CARICO_IN_POTENZA{
    private:
        ...
    public:
        CARICO_IN_POTENZA(..., ...){}

        // Metodi già creati da non cancellare

        // Metodi DA AGGIUNGERE
        int lettura_potenziometro(int val...){}

};

CARICO_IN_POTENZA led_pot(9, A0);
```

Scrivere quindi all'interno della `void loop(){}` un codice che stampi nella seriale il valore letto dal potenziometro.

Fase 7 — Controllo automatico con potenziometro

A questo punto si modifica la classe `CARICO_IN_POTENZA` in modo da permettere il controllo diretto del carico tramite il valore letto dal potenziometro. L'obiettivo è che la luminosità vari in tempo reale al variare della posizione del potenziometro.

- Aggiungere all'interno della classe un metodo che legga il valore del potenziometro e regoli automaticamente il carico:

  ```
  void aggiorna_da_potenziometro(){}
  ```

- All'interno della `loop()` scrivere:

  ```
  led_pot.aggiorna_da_potenziometro();
  ```

In questo modo il carico seguirà in tempo reale la variazione del potenziometro, permettendo di controllare manualmente la luminosità del LED o la velocità del motore.

Fase 8 — Sostituzione del LED con il motore DC

Dopo aver verificato il corretto funzionamento con il LED, è ora possibile sostituirlo con il **motore DC da** 12 V. Prestare la massima attenzione ai collegamenti elettrici:

- Scollegare l'alimentazione prima di modificare il circuito;
- Sostituire il LED e la sua resistenza con il motore DC;
- Inserire un **diodo 1N4007** in antiparallelo al motore (catodo verso il +12 V, anodo verso il Drain del MOSFET) per proteggere il circuito dai picchi di tensione generati dal carico induttivo;