Section A

I use python, since I am not sure how to implement genAI stuff in C (like how to handle json, send API request...). The general idea is to use prompt enigneer, to a structured API from any LLM provider (like: https://platform.openai.com/docs/guides/structured-outputs) , which enforce the output to be in structured format of json, also there is a safety net mechnism json_validator which verify if the json function call is valid. The prompt template is in txt. addiontionally, if the prompt engineering method don't work well, I also write a concept example template for fine tuning.

Section B

This part use Cpp (with a little bit of help from Gemini). The motion control consist of 3 main part, 1: motion planner that calculate the velocity during the process using non linear function to create a smooth movement transition; 2: a pid controller to control and adjust the movement to minimize error between target posistion and current position; 2: a kalmen filter to estimate the state and filter potential noise from sensor (position /velocity sensor). but this work on the premise, (especially for motion planner) that the robot how many step to take or given a time limit to finish the movement. but then thinking of the previous conversation, the system might also not be good enough to successfully execute the function in given time, rather it need time to keep on adjusting until to read the optimal (target position), in this case, we write the addiontional md to demonstarte how to do it in cencpet. the motion planner logic need to be update, and a safety mechnism like dead zone need to be implement.

Opt: How you might integrate these two levels of control into a unified system.

Ultimately, the section A should be done in C for the system to work seamlessly, so the one class object can call the class object in the next step if previous step is been validate as doable. A intemidiate method can be to write the json output (from the section A which is already validated) in a file (in some folder like 'User/APPDATA') where the lower level module can read; Then we need a Controller Logic to parse the json command into function call to motor control module. It would be sth like this:

```
 // 1. Get current position from the encoder
float current_position = encoder.get_position(); // e.g., 10.0 degrees

// 2. Parse the target from the LLM JSON
float target_position = json_command.parameters.target_position; // 90.0 degrees

// 3. Calculate the required displacement
float displacement = target_position - current_position; // 80.0 degrees

MotionController.set_target(displacement);
```

we initialize the Motion controller in section B, then the control loop now runs the motion using the logic defied in section B