Project Startup: Getting Started with Your Algorithm Branch 🚀

Hey team, here are the Git steps to get you set up and working on your assigned localization algorithm. The `main` branch has all the common files we need (Webots world, map, supervisor, configs), and your empty branch (`ekf`, `markov`, or `amcl`) is already created on GitHub.

Step 1: Clone or Update Your Local Repo 💻

- **If you haven't cloned the repo yet:**

  git clone https://github.com/herish23/G23-Intelligent-Robotics.git

  cd G23-Intelligent-Robotics

- **If you *already* cloned it:** Make sure you're up-to-date.

  # Make sure you don't have uncommitted changes first!

  git checkout main

  git pull origin main

  git fetch origin # Important: This gets info about the new branches

---

Step 2: Switch to YOUR Assigned Branch 🌿

- Replace `<your-branch-name>` with **your specific branch** (`ekf`, `markov`, or `amcl`). This command checks out the branch from the remote repository and sets it up for you locally.

  git checkout <your-branch-name>

  (Example: `git checkout ekf`)

- **Verify:** Run `git branch`. You should see a `*` next to your branch name.

Step 3: Start Coding! 👩‍💻

- You're now working safely in your **own branch**.
- Find the **common files** (map, `.wbt`, trajectories, configs) that were pulled from `main`.
- Create your **controller file** (e.g., `ekf_controller.py`) inside the appropriate folder.
- Make sure your code reads the shared **map** and **trajectory files**.
- Implement the **logging** for your **estimated pose** and **computation times** as per the agreed format.

Step 4: Commit & Push Your Work (Often!) 💾➡️☁️

- **Commit changes locally:** Add the files you worked on and commit with a clear message.

  git add . # Or specify files like 'ekf_controller.py'

  git commit -m "Describe your progress here (e.g., Added DT calculation)"

- **Push your branch to GitHub:** Keep your work backed up and visible.

  git push origin <your-branch-name>

  (Example: `git push origin ekf`)

Step 5: Getting Updates from `main` Later (If Needed)

- If new common files are added to `main`, you'll need them. **First, commit your current work** on your branch. Then:

  git fetch origin # Get the latest info

  git merge origin/main # Merge the updated main branch into your current branch

  # If Git reports merge conflicts, open the conflicted files, fix them, then:

  # git add .

```
# git commit -m "Resolved merge conflicts from main"

git push origin <your-branch-name> # Push the updated branch
```
