Guidance for Students: Virtual Robot Maze Simulator

Introduction to Rust and the Project

Welcome to the **Virtual Robot Maze Simulator** project! In this project, you'll not only learn the basics of the Rust programming language but also understand how programming works in real-world embedded systems, like robots or even medical devices.

Why Learn Rust?

Rust is a modern programming language that combines high performance with memory safety. It is used in many areas, including:

- **Robotics**: Rust is increasingly popular in robotics for its efficiency, safety, and speed.
- **Medical Devices**: Rust's safety features make it an excellent choice for embedded systems in critical applications, such as medical devices.
- **Versatility**: Even if you need to code in languages like C later, the principles you learn with Rust—like control logic, error handling, and concurrent programming—will still be applicable.

By learning Rust, you are gaining skills that are valuable for both software development and embedded systems programming.

Project Overview

The main objective of this project is to program a virtual robot to navigate through a 2D maze. The robot's movement is simulated using a virtual motor, which acts like a "black box." You will write Rust code to interact with this motor's API, controlling the robot's speed, direction, and other parameters.

This project simulates real-world embedded systems challenges and teaches you:

- **Control Logic**: How to manage the robot's movement using logical conditions and decision-making.
- **Algorithmic Thinking**: How to develop algorithms to navigate the maze effectively.
- **Efficient Coding**: How to write code that interacts efficiently with hardware components.

Project Structure

The project is divided into three main components:

1. **Black Box Simulator**: This is the core component, simulating a virtual motor and sensors of the robot. The "black box" mimics the behavior of real hardware, providing a practical interface for your Rust code.
2. **Robot Control Code**: You will write Rust code to send commands to the virtual motor, adjusting the robot's speed and direction.
3. **Virtual Maze Environment**: The environment consists of a simple 2D grid maze. Your goal is to program the robot to traverse this maze successfully.

Hands-On Coding Experience with Rust

This project provides a practical way to learn Rust while simulating embedded systems programming:

- **Control Flow**: You will use loops and conditionals to manage the robot's movements, reinforcing Rust's strong type system.
- **Error Handling**: Rust's `Result` and `Option` types will be used to handle errors and manage optional values, helping you write safer code.
- **Concurrency** (optional, for advanced students): If you progress well, you will be introduced to concurrent programming, enabling you to simulate more realistic motor control.

Getting Started

Follow these steps to begin:

1. **Set Up a GitHub Account**
   - Go to GitHub's signup page and create an account.
   - Use the **Sign up with Google** option if you have a Google account.
   - Set up your GitHub profile and explore the interface to understand where to find repositories, issues, and your profile.
2. **Fork the Project Repository**
   - Go to the main project repository on GitHub: `https://github.com/mkstreet/virtual-robot-maze.git`
   - Click the **Fork** button in the top right corner of the page.
   - Forking creates an independent copy of the repository under your GitHub account.
   - Once the forking process is complete, you will be redirected to your forked version of the repository.
3. **Launch GitHub Codespaces**
   - Go to your forked repository on GitHub.
   - Click the **Code** button, then select the **Codespaces** tab.
   - Click on **Create codespace on main** to launch a new Codespace environment.
   - Codespaces will set up a development environment using the `devcontainer.json` file, installing Rust and necessary extensions.
   - Once ready, you will have an online development environment similar to VS Code, with the project files already loaded.

Next Steps

Once you have set up GitHub Codespaces, start exploring the project by navigating to the `src/main.rs` file. Begin writing Rust code to interact with the virtual robot's black box API and program the robot to navigate the maze.