

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



(CO3109)

---

## Web Application for Code Generation for Robotics Control

---

Nhóm 3: L01 \_ CNPM \_ 17

<b>GVHD:</b>	PGS.TS. Lê Hồng Trang	
<b>Sinh viên:</b>	Ngô Quang Bách	- 2110770
	Nguyễn Minh Diễm	- 2111056
	Nguyễn Quốc Tuấn	- 2112585
	Đinh Hoàng Dũng	- 2110096
	Mai Trọng Dũng	- 2113064

TP. Hồ Chí Minh, 14/3/2023

# Mục lục

<b>PHÂN CÔNG NHIỆM VỤ</b>	<b>2</b>
<b>1 GIỚI THIỆU</b>	<b>3</b>
1.1 Tổng quan . . . . .	3
1.2 Mục tiêu của đề tài . . . . .	3
1.3 Các nền tảng . . . . .	4
1.3.1 Ubuntu . . . . .	4
1.3.2 Gazebo . . . . .	4
1.3.3 ROS2 . . . . .	5
1.3.4 React Blockly . . . . .	5
<b>2 THIẾT KẾ</b>	<b>6</b>
2.1 Tổng quan kiến trúc . . . . .	6
2.1.1 Mô phỏng & điều khiển Robot . . . . .	6
2.1.2 Front-end . . . . .	6
2.1.3 Back-end . . . . .	6
2.2 Quy trình hoạt động . . . . .	7
<b>3 HIỆN THỰC</b>	<b>8</b>
3.1 Thiết lập hệ thống . . . . .	8
3.1.1 Thiết lập môi trường mô phỏng . . . . .	8
3.1.2 Thiết lập Front-end . . . . .	8
3.1.3 Thiết lập Back-end . . . . .	9
3.1.4 Kết quả kiểm thử . . . . .	12
<b>4 KẾT LUẬN</b>	<b>13</b>
4.1 Về tổng quan hệ thống . . . . .	13
4.1.1 Ưu điểm . . . . .	13
4.1.2 Nhược điểm . . . . .	13
4.2 Về quá trình hiện thực . . . . .	13
4.2.1 Những khó khăn . . . . .	13
4.2.2 Hướng mở rộng . . . . .	14
<b>TÀI LIỆU THAM KHẢO</b>	<b>15</b>



## PHÂN CÔNG NHIỆM VỤ

STT	Họ & tên	MSSV	Nhiệm vụ	Hoàn thành
1	Ngô Quang Bách	2110770	<ul style="list-style-type: none"><li>Viết React Blockly</li><li>Ráp BE - FE</li></ul>	100
2	Nguyễn Minh Điềm	2111056	<ul style="list-style-type: none"><li>Viết React, API</li><li>Viết hướng dẫn thiết lập</li></ul>	100
3	Nguyễn Quốc Tuấn	2112585	<ul style="list-style-type: none"><li>Viết BE xử lý FE</li><li>Viết báo cáo, demo</li></ul>	100
4	Đinh Hoàng Dũng	2110096	<ul style="list-style-type: none"><li>Viết ROS Node</li><li>Viết báo cáo</li></ul>	100
5	Mai Trọng Dũng	2113064	<ul style="list-style-type: none"><li>Viết ROS Node</li><li>Viết báo cáo</li></ul>	100

## 1 GIỚI THIỆU

### 1.1 Tổng quan

Ngành robot đang trải qua một giai đoạn phát triển sôi động, với tốc độ tăng trưởng cao, mở ra nhiều cơ hội cho các ứng dụng mới trong một loạt các lĩnh vực như giáo dục, nghiên cứu, công nghiệp và dịch vụ. Khả năng sáng tạo và ứng dụng linh hoạt của robot hứa hẹn mang lại sự thay đổi lớn trong cách chúng ta làm việc và sống.

Trong thế giới của robot, lập trình thường là một thách thức khó khăn đối với người mới bắt đầu và những người không chuyên về robot. Các phương pháp lập trình truyền thống thường liên quan đến cú pháp phức tạp và đòi hỏi kiến thức lập trình sâu rộng, đặc biệt là cho các nhiệm vụ như thu thập dữ liệu cảm biến và thuật toán tìm đường, điều này tạo ra một bức tường khó vượt qua đối với những người mới vào lĩnh vực này.

Để thu hẹp khoảng cách này, nhu cầu về các giao diện lập trình trực quan để đơn giản hóa điều khiển robot ngày càng tăng. Bằng cách đơn giản hóa quá trình lập trình, chúng ta có thể mở khóa tiềm năng cho sự tham gia rộng rãi hơn vào lĩnh vực robot. Các giao diện thân thiện với người dùng này nên cho phép người dùng tạo và sửa đổi hành vi của robot mà không cần chuyên môn lập trình cao.

Ứng dụng web tạo mã điều khiển robot bằng ROS2 là một công cụ hữu ích không chỉ để đơn giản hóa quá trình lập trình, mà còn để mở rộng cơ hội và khuyến khích sự tham gia rộng rãi trong lĩnh vực robot. Với tiềm năng thúc đẩy sự phát triển của cộng đồng và đem lại nhiều lợi ích cho xã hội, ứng dụng này hứa hẹn là một công cụ quan trọng trong tương lai của ngành công nghiệp robot.

### 1.2 Mục tiêu của đề tài

**Đơn giản hóa Quá trình Lập trình Robot:** Mục tiêu chính của dự án là tạo ra một ứng dụng web có khả năng tạo mã điều khiển robot một cách dễ dàng và linh hoạt. Thay vì phải sử dụng các công cụ phức tạp và yêu cầu kiến thức lập trình chuyên sâu, người dùng có thể tạo ra các chương trình điều khiển robot thông qua một giao diện trực quan và thân thiện.

**Mở rộng Khả năng tiếp cận:** Dự án nhằm mục tiêu mở rộng khả năng tiếp cận và tham gia trong lĩnh vực robot. Bằng cách cung cấp một ứng dụng web dễ sử dụng, chúng tôi hy vọng thu hút sự quan tâm của người mới bắt đầu và những người không chuyên về lập trình, từ sinh viên đến người yêu thích và các chuyên gia.

**Khuyến khích sự Sáng tạo và Hợp tác:** Dự án cũng nhằm mục tiêu khuyến khích sự sáng tạo và hợp tác trong cộng đồng robot. Bằng cách chia sẻ kiến thức và kinh nghiệm thông qua ứng dụng, chúng tôi mong muốn tạo ra một môi trường hỗ trợ cho việc phát triển và chia sẻ các giải pháp robot mới.

**Tăng cường Hiệu quả và Năng suất trong Lập trình Robot:** Cuối cùng, mục tiêu của dự án là tăng cường hiệu quả và năng suất trong quá trình lập trình robot. Bằng cách giảm bớt thời gian và công sức cần thiết để tạo ra các chương trình điều khiển robot, chúng tôi hy vọng giúp người dùng tập trung hơn vào việc thử nghiệm và phát triển các ứng dụng robot sáng tạo và hiệu quả.

## 1.3 Các nền tảng

### 1.3.1 Ubuntu

**Ubuntu** là một hệ điều hành mã nguồn mở dựa trên Linux và cộng đồng toàn cầu. Ubuntu được biết đến với tính dễ sử dụng, ổn định và bảo mật cao.

Lợi ích của việc sử dụng Ubuntu:

- **Hỗ trợ đa nền tảng:** Ubuntu hỗ trợ nhiều nền tảng phần cứng khác nhau, bao gồm máy tính và cả các thiết bị nhúng. Điều này giúp các lập trình viên phát triển ứng dụng có thể chạy trên nhiều thiết bị khác nhau.
- **Cộng đồng lớn:** Ubuntu có cộng đồng người dùng và nhà phát triển lớn trên toàn thế giới. Cộng đồng này cung cấp và hỗ trợ tài nguyên cho các lập trình viên gặp sự cố trong quá trình phát triển ứng dụng.
- **Môi trường phát triển ổn định:** Ubuntu cung cấp môi trường phát triển ứng dụng ổn định và an toàn, được đánh giá cao về tính bảo mật. Hệ điều hành này trang bị nhiều tính năng bảo mật giúp bảo vệ dữ liệu của các lập trình viên.

### 1.3.2 Gazebo

**Gazebo** là một môi trường mô phỏng robot mạnh mẽ và linh hoạt được sử dụng rộng rãi trong cộng đồng robot học và nghiên cứu. Được phát triển bởi Open Source Robotics Foundation, Gazebo cung cấp một nền tảng mô phỏng đa mục đích cho robot, từ robot di động đến robot có cánh, robot công nghiệp và nhiều hơn nữa.

Lợi ích của việc sử dụng Gazebo:

- **Mô phỏng vật lý thực tế:** Gazebo sử dụng công cụ mô phỏng vật lý PhysX để tạo ra môi trường mô phỏng thực tế, giúp người dùng có thể kiểm tra hiệu suất của robot trong môi trường tương tự như thực tế.
- **Hỗ trợ đa robot:** Gazebo cho phép mô phỏng nhiều robot trong một môi trường duy nhất, giúp người dùng thử nghiệm và phát triển các hệ thống đa robot phức tạp.
- **Tích hợp ROS:** Gazebo được tích hợp chặt chẽ với ROS, cung cấp một giao diện tiện lợi cho việc phát triển và kiểm tra các ứng dụng robot.
- **Mô hình hóa chính xác:** Gazebo cung cấp một thư viện các mô hình robot và môi trường phong phú, giúp người dùng mô phỏng các robot với độ chính xác cao và chi tiết.
- **Hỗ trợ mô phỏng cảm biến và hành vi:** Gazebo cho phép mô phỏng cảm biến và hành vi của robot trong một môi trường 3D, từ việc thu thập dữ liệu từ các cảm biến đến việc điều khiển robot di chuyển.

### 1.3.3 ROS2

**Robot Operating System 2 (ROS2)** là hệ thống mã nguồn mở cung cấp các công cụ và thư viện cho phép phát triển các ứng dụng robot một cách hiệu quả và dễ dàng. ROS2 được xây dựng dựa trên kiến trúc phân tán, giúp tăng cường khả năng mở rộng và hiệu suất cho các ứng dụng robot phức tạp.

Lợi ích của việc sử dụng ROS2:

- **Mã nguồn mở:** ROS2 được phát triển dưới dạng mã nguồn mở, cho phép cộng đồng robot phát triển và đóng góp các tính năng mới.
- **Kiến trúc phân tán:** ROS2 được xây dựng dựa trên cấu trúc phân tán, giúp tăng cường khả năng mở rộng và hiệu suất cho các ứng dụng robot.
- **Hỗ trợ đa ngôn ngữ:** ROS2 hỗ trợ nhiều ngôn ngữ lập trình phổ biến như C++, Python, Java... giúp mở rộng cơ hội phát triển và sử dụng cho lập trình viên.
- **Cộng đồng lớn:** ROS2 sở hữu cộng đồng robot lớn và hoạt động tích cực, cung cấp và hỗ trợ tài nguyên cho người dùng.

### 1.3.4 React Blockly

**React Blockly** là một thư viện mã nguồn mở cho phép bạn tạo ra các giao diện người dùng tương tác bằng cách xây dựng các khối (block) trực quan. Đây là sự kết hợp sức mạnh của React với sự đơn giản của Blockly để tạo ra các ứng dụng web mạnh mẽ và dễ sử dụng.

Lợi ích của việc sử dụng React Blockly:

- **Nhiều thư viện hỗ trợ:** Các thư viện Blockly cung cấp các API và công cụ đầy đủ, giúp các nền tảng lập trình tạo ra một môi trường có thể tùy chỉnh theo nhu cầu riêng biệt.
- **Giao diện trực quan:** Editor của Blockly sử dụng các khối hình học để đại diện cho các khái niệm lập trình như biến, biểu thức logic hay vòng lặp. Điều này cho phép người dùng lập trình mà không cần lo lắng về cú pháp.
- **Hỗ trợ đa nền tảng:** Blockly là một thư viện không phụ thuộc nền tảng, được các lập trình viên ưa chuộng vì tính nhất quán, linh hoạt và tiện lợi. Blockly tương thích với hầu hết các trình duyệt web và có thể hoạt động cả trên thiết bị di động.

## 2 THIẾT KẾ

### 2.1 Tổng quan kiến trúc

Đồ án tập trung vào việc phát triển một ứng dụng Web giúp tạo mã điều khiển cho Robot. Hệ thống được chia thành 3 phần chính: **Front-end**, **Back-end** và **Mô phỏng & điều khiển Robot**. Mỗi phần sẽ đảm nhận các chức năng cụ thể.

#### 2.1.1 Mô phỏng & điều khiển Robot

##### 1. ROS2

- **Quản lý Robot:** Sử dụng ROS2 để điều khiển Robot.
- **Xử lý thông tin cảm biến và hành động:** Nhận và xử lý dữ liệu từ cảm biến, gửi lệnh điều khiển tới Robot.
- **Tích hợp với Gazebo:** Kết nối với Gazebo để mô phỏng môi trường và kiểm tra mã điều khiển trong môi trường ảo trước khi áp dụng vào thực tế.

##### 2. Gazebo

- **Mô phỏng môi trường:** Cung cấp môi trường ảo mô phỏng các tình huống thực tế.
- **Kiểm tra và đánh giá:** Cho phép người dùng kiểm tra hoạt động của mã điều khiển, quan sát hành vi của Robot trong môi trường mô phỏng và đánh giá độ hiệu quả của thuật toán điều khiển trước khi áp dụng vào môi trường thực tế.

#### 2.1.2 Front-end

##### 1. React

- **Giao diện người dùng (User Interface):** Sử dụng React để xây dựng giao diện người dùng, cung cấp các thành phần giao diện động và thân thiện.
- **Quản lý trạng thái (State Management):** Sử dụng Redux/Context API để quản lý trạng thái của ứng dụng, đảm bảo tính nhất quán và dễ dàng theo dõi.
- **Tương tác với Blockly:** Tích hợp với Blockly để cung cấp giao diện lập trình kéo thả, giúp người dùng tạo ra mã điều khiển Robot một cách trực quan.

##### 2. Blockly

- **Khối lệnh lập trình:** Cung cấp các khối lệnh cho phép người dùng kéo thả để xây dựng logic điều khiển Robot.
- **Chuyển đổi khối lệnh thành code:** Chuyển đổi các khối lệnh thành mã JSON để gửi tới Back-end xử lý.

#### 2.1.3 Back-end

##### 1. Python

- **Xử lý yêu cầu từ Front-end:** Nhận và xử lý các yêu cầu từ giao diện người dùng như yêu cầu tạo mã điều khiển hoặc cập nhật cấu hình Robot.
- **Giao tiếp với ROS2:** Chuyển đổi các khối lệnh từ Blockly thành mã điều khiển bằng Python và ROS2.

##### 2. Các package, config, validation...

## 2.2 Quy trình hoạt động

1. **Người dùng truy cập ứng dụng Web:** Truy cập giao diện web được xây dựng bằng React.
2. **Tạo mã điều khiển bằng Blockly:** Người dùng sử dụng giao diện kéo thả để tạo ra các khối lệnh điều khiển.
3. **Gửi yêu cầu tới Back-end:** Các object thuộc loại JSON được gửi từ Front-end xuống Back-end để xử lý.
4. **Chuyển đổi code:** Back-end nhận object và gọi API tương ứng bằng Flask.
5. **Gửi code tới ROS2:** API trên sẽ gọi tới các phương thức để tạo các ros node nhằm gửi các lệnh điều khiển tới ROS2 ở dạng file JSON để mô phỏng.
6. **Kiểm tra trong Gazebo:** Gazebo sẽ hiển thị kết quả mô phỏng, cho phép người dùng quan sát và điều chỉnh.



### 3 HIỆN THỰC

#### 3.1 Thiết lập hệ thống

##### THIẾT LẬP CHUNG

- Cài đặt hệ điều hành **Ubuntu 22.04 LTS** (dual-boot hoặc WSL, máy ảo...).
- Mặc định mọi thiết lập sau ở home ("`cd ~`").
- **CLONE REPOSITORY** "`git clone https://github.com/diem23/controlRobotBlockly`".
- Trong lúc cài đặt và chạy, nếu vướng lỗi "`bash: <abcXYZ>: No such file or directory`" thì kiểm tra, cập nhật lại `.bashrc`, các file `setup.bash` hoặc chạy "`cd <address>`" vào đúng địa chỉ.

##### 3.1.1 Thiết lập môi trường mô phỏng

1. Cài đặt software để phát triển robot: **ROS2 Humble**.
  - Cài đặt software theo hướng dẫn ở link sau.
  - Sau khi cài đặt, chạy lệnh "`cd /opt/ros/humble`" để kiểm tra cài đặt đã thành công chưa.
  - Thêm lệnh "`source /opt/ros/humble/setup.bash`" vào vùng cuối file `.bashrc` bằng cách chạy lệnh "`gedit .bashrc`".
  - Sau khi chỉnh sửa, chạy lệnh "`source .bashrc`" để có thể cập nhật.
2. Cài đặt software để mô phỏng môi trường và mô hình robot: **Gazebo**.
  - Đầu tiên, kiểm tra phiên bản Ubuntu bằng lệnh "`lsb_release -a`".
  - Chọn gói cài đặt tương ứng với phiên bản Ubuntu ở link sau.
3. Cài đặt robot cho quá trình mô phỏng: **Linorobot**.
  - Cài đặt robot theo hướng dẫn ở link sau.
  - Lưu ý, dòng lệnh nào có `<robot_type>` sẽ thay bằng `2wd`, `<laser_sensor>` sẽ thay bằng `a1`, `<robot_computer_ws>` sẽ thay bằng đường dẫn tới folder `linorobot2_ws`.
  - Tạo một folder có tên `hostmachine_ws`, dòng lệnh nào có `<host_machine_ws>` thay bằng đường dẫn tới folder `hostmachine_ws`.
  - CHẠY THỬ MÔ PHỎNG "`ros2 launch linorobot2_gazebo gazebo.launch.py`".

**KẾT QUẢ THIẾT LẬP:** Gazebo được mở có map và Robot.

##### 3.1.2 Thiết lập Front-end

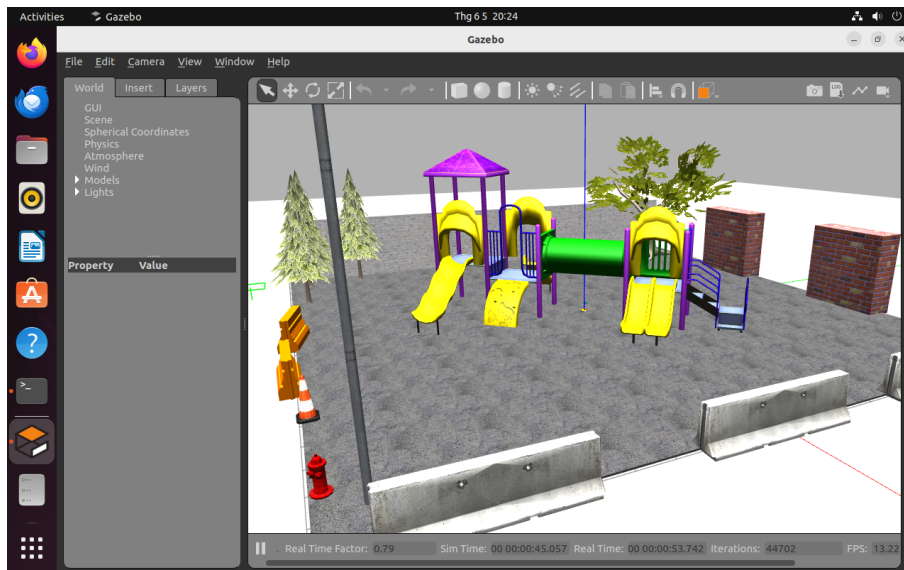
Để sử dụng React-Blockly trên Ubuntu, đầu tiên cần cài đặt Node.js và npm nếu chưa có. Sau đó, có thể tạo một dự án React mới và cài đặt React-Blockly như một phụ thuộc của dự án đó. Dưới đây là các bước cụ thể.

1. Cài đặt **Node.js** và **npm** bằng cách chạy các lệnh dưới đây
  - "sudo apt update"
  - "sudo apt install nodejs"
  - "sudo apt install npm"
2. Tạo một dự án **React** mới bằng lệnh
  - "npx create-react-app robot\_control"
3. Cài đặt **React-Blockly** bằng lệnh
  - "npm install react-blockly blockly"
4. Sau khi đã clone git ở trên, để **KHỞI CHẠY WEB APP**, chạy các lệnh
  - "cd controlRobotBlockly/frontend"
  - "npm start"

**KẾT QUẢ THIẾT LẬP:** Web react với các khối lệnh di chuyển.

### 3.1.3 Thiết lập Back-end

1. Cài đặt môi trường Python Flask handle request trong Back-end
  - "cd controlRobotBlockly/Backend".
  - **Cài đặt pipenv** "pip install -user pipenv". Sau đó, chạy "pipenv shell".
  - **Cài đặt các package cần thiết** "pip install -r requirements.txt". Trong lúc thực thi, nếu terminal báo không tìm thấy package, chạy thêm "pipenv install <package>" hoặc "pip install <package>".
  - **Xây dựng các package ros2** (đã cài đặt ở phần 4.1.1) "colcon build". blue



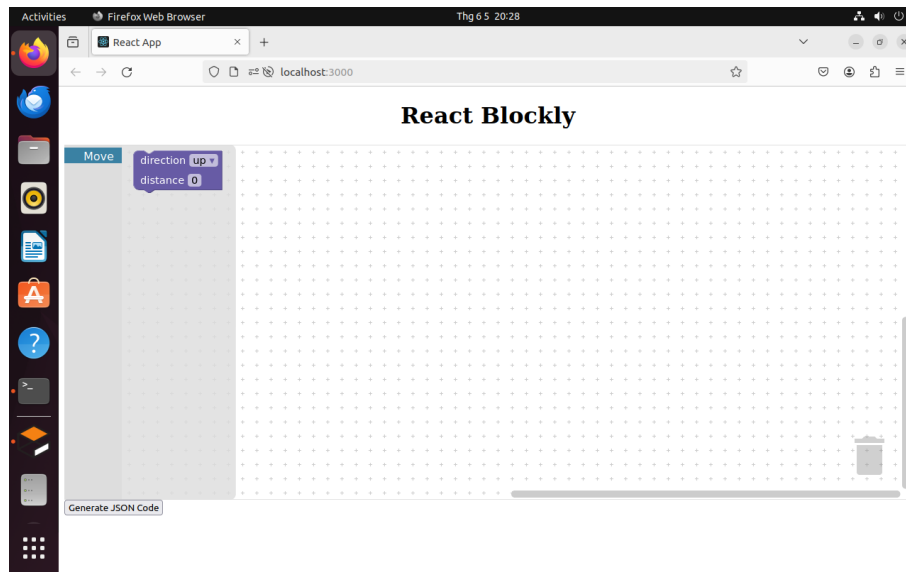
Hình 1: Map & robot mô phỏng

- Tiếp theo, điều hướng đến folder **Backend** và chạy lệnh sau để có thể chạy **Back-end**

```
dungmai@dungmai-virtual-machine: ~/controlRobotBlockly/...
dungmai@dungmai-virtual-machine: ~ x dungmai@dungmai-virtual-machine: ~/c... x
dungmai@dungmai-virtual-machine:~$ cd controlRobotBlockly/Backend/
dungmai@dungmai-virtual-machine:~/controlRobotBlockly/Backend$ python3 main.py
* Serving Flask app 'main'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 399-622-872
```

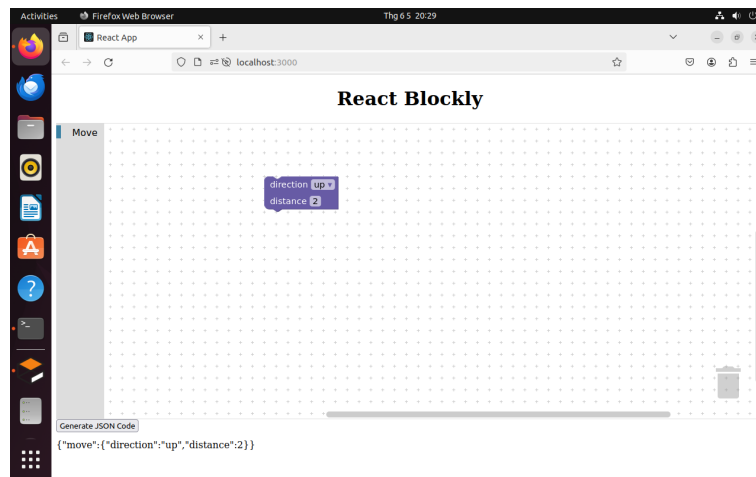
Hình 2: Khởi chạy main.py

- Sau khi **Back-end** đã được chạy, điều hướng đến folder **frontend** và chạy lệnh "npm start" để có thể chạy **Front-end**



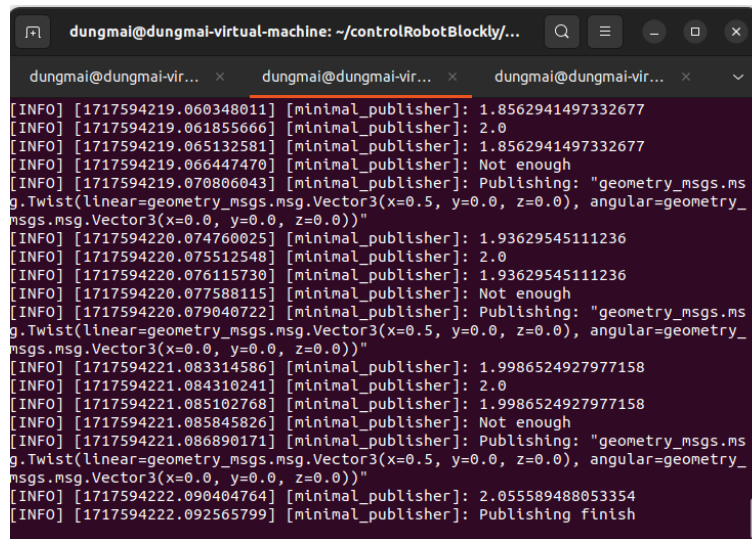
Hình 3: Giao diện Web App

- Sau khi hoàn tất thiết lập, ở Web App, ta sẽ kéo khối Block và điều chỉnh các thông số điều khiển ở trên Block và ấn "Generate JSON Code" để gửi file json chứa các thông số điều khiển robot đến cho Back-end xử lý và điều khiển robot chạy cho phù hợp.



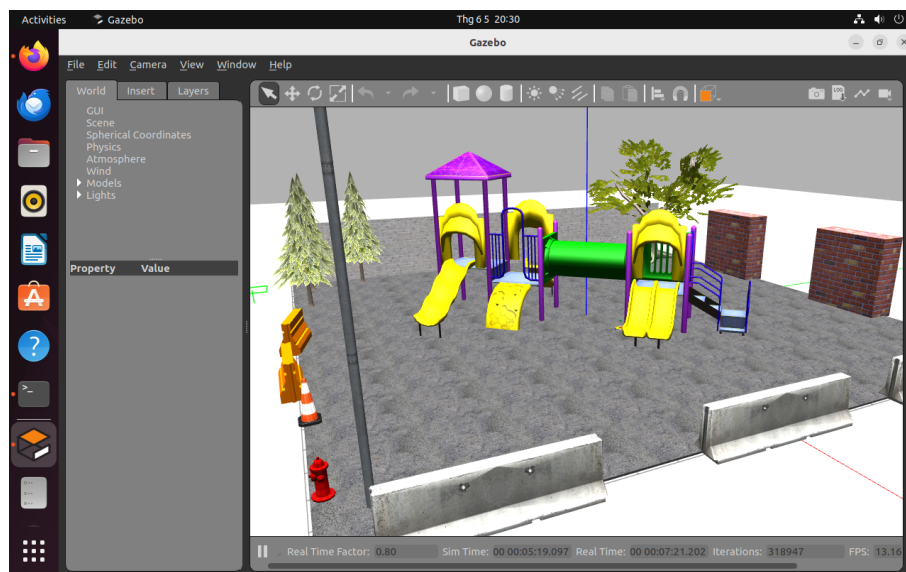
Hình 4: Chỉnh sửa thông số điều khiển

- Linorobot sẽ sử dụng 2 topic, 1 topic là "/cmd\_vel" để nhận các thông số điều khiển từ Back-end và 1 topic là "/odom" để có thể gửi các thông số vị trí cho Back-end, thông số vị trí này sẽ được dùng để kiểm tra quãng đường robot đã đi.



```
dungmai@dungmai-virtual-machine: ~/controlRobotBlockly/...  
[INFO] [1717594219.060348011] [minimal_publisher]: 1.8562941497332677  
[INFO] [1717594219.061855666] [minimal_publisher]: 2.0  
[INFO] [1717594219.065132581] [minimal_publisher]: 1.8562941497332677  
[INFO] [1717594219.066447470] [minimal_publisher]: Not enough  
[INFO] [1717594219.070806043] [minimal_publisher]: Publishing: "geometry_msgs.  
g.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_  
msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))"  
[INFO] [1717594220.074760025] [minimal_publisher]: 1.93629545111236  
[INFO] [1717594220.075512548] [minimal_publisher]: 2.0  
[INFO] [1717594220.076115730] [minimal_publisher]: 1.93629545111236  
[INFO] [1717594220.077588115] [minimal_publisher]: Not enough  
[INFO] [1717594220.079040722] [minimal_publisher]: Publishing: "geometry_msgs.  
g.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_  
msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))"  
[INFO] [1717594221.083314586] [minimal_publisher]: 1.9986524927977158  
[INFO] [1717594221.084310241] [minimal_publisher]: 2.0  
[INFO] [1717594221.085102768] [minimal_publisher]: 1.9986524927977158  
[INFO] [1717594221.085845826] [minimal_publisher]: Not enough  
[INFO] [1717594221.086890171] [minimal_publisher]: Publishing: "geometry_msgs.  
g.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y=0.0, z=0.0), angular=geometry_  
msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))"  
[INFO] [1717594222.090404764] [minimal_publisher]: 2.055589488053354  
[INFO] [1717594222.092565799] [minimal_publisher]: Publishing finish
```

Hình 5: Thông tin terminal của Backend



Hình 6: Quãng đường robot đã chạy được

### 3.1.4 Kết quả kiểm thử

Case	Tham số đầu vào	Mong đợi	Thực tế
1	"up" 1	Tiến lên 1 đơn vị	1.05; 1.06; 1.02
2	"down" 1	Lùi xuống 1 đơn vị	1.00; 1.10; 1.01
3	"left" 1	Xoay trái 1 khoảng nhỏ (90 độ qua 5~8 request)	Như mong đợi
4	"right" 1	Xoay phải 1 khoảng nhỏ (90 độ qua 5~8 request)	Như mong đợi
5	"left" 10	Xoay trái khoảng 90 độ	Nhiều lần như mong đợi, vài lần fail
6	"right" 10	Xoay phải khoảng 90 độ	Nhiều lần như mong đợi, vài lần fail

Bảng 1: Kết quả kiểm thử điều khiển robot

## 4 KẾT LUẬN

### 4.1 Về tổng quan hệ thống

#### 4.1.1 Ưu điểm

- Hệ thống chạy được.
- Giao diện Web thân thiện, dễ tương tác.
- Kết quả khá sát với mong đợi.

#### 4.1.2 Khuyết điểm

- Quy trình chạy rườm rà.
- Chưa hiện thực được cụm khối.
- Còn vài cảnh báo và lỗi.

### 4.2 Về quá trình hiện thực

#### 4.2.1 Những khó khăn

- Một số thành phần xung đột, không chạy được với nhau.
- Một số thành phần bị lỗi thời.
- Một số thành phần có rất ít tài liệu tham khảo.



#### 4.2.2 Hướng mở rộng

- Cho người dùng tự tùy chỉnh nhiều thông số hơn (quãng đường cần đi, thời gian di chuyển...)
- Cải thiện để sự di chuyển thực tế của robot sát với mong đợi nhất có thể
- Tích hợp thêm các chuyển động phức tạp hơn: nâng hạ ben (đối với một số loại robot), tự động dừng khi gặp vật cản...



## TÀI LIỆU THAM KHẢO

- [1] Installation Guide (*Ubuntu*)  
<https://ubunchuu-truong-us.github.io/docs/category/installation-guide>
- [2] Installing Gazebo with ROS  
[https://gazebo-sim.org/docs/latest/ros\\_installation](https://gazebo-sim.org/docs/latest/ros_installation)
- [3] linorobot2 (*Guide*)  
<https://github.com/linorobot/linorobot2>
- [4] react-blockly (*Installation Guide*)  
<https://www.npmjs.com/package/react-blockly>
- [5] ROS 2 Documentation  
<https://docs.ros.org/en/humble/index.html>