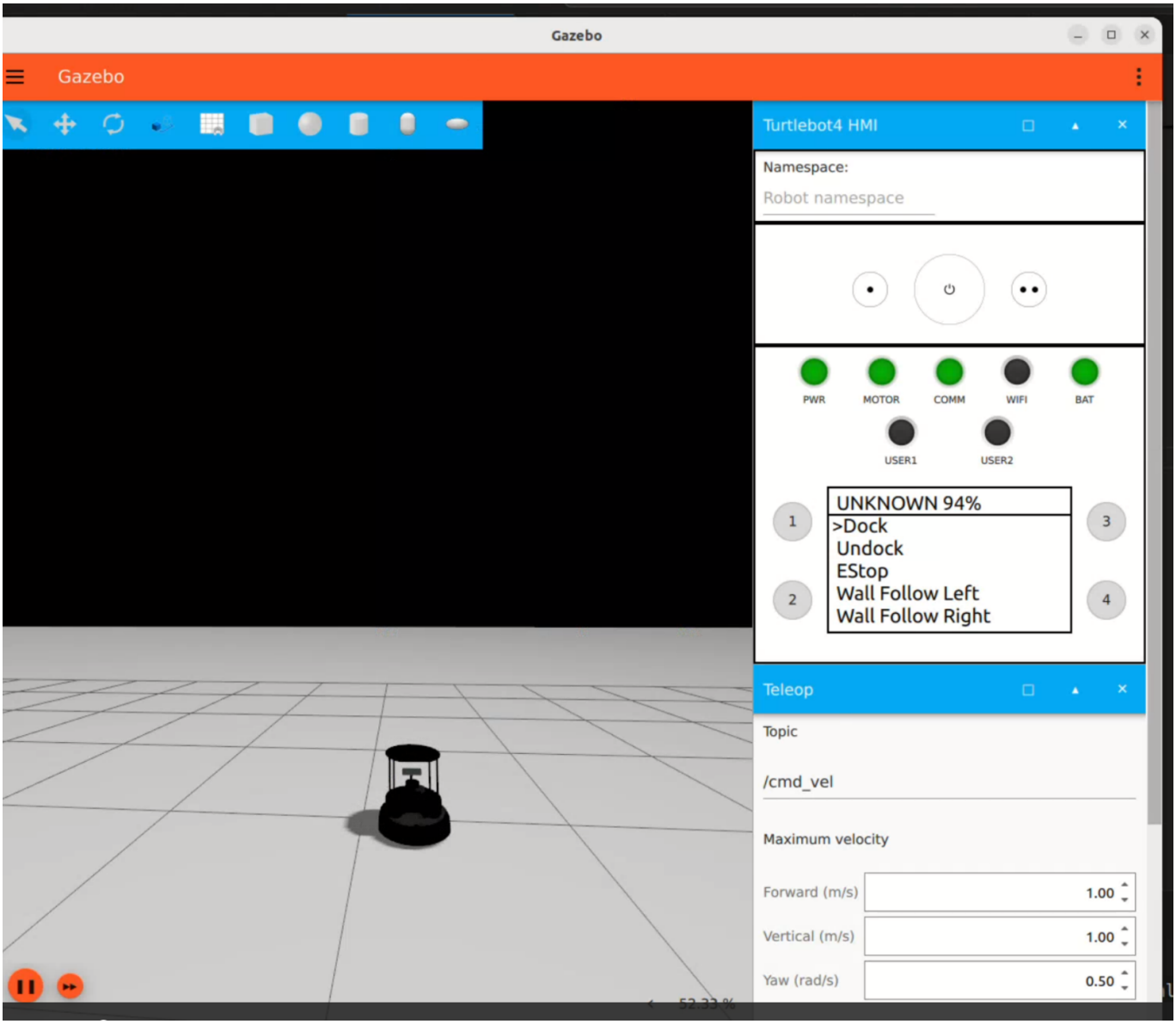
Homework2

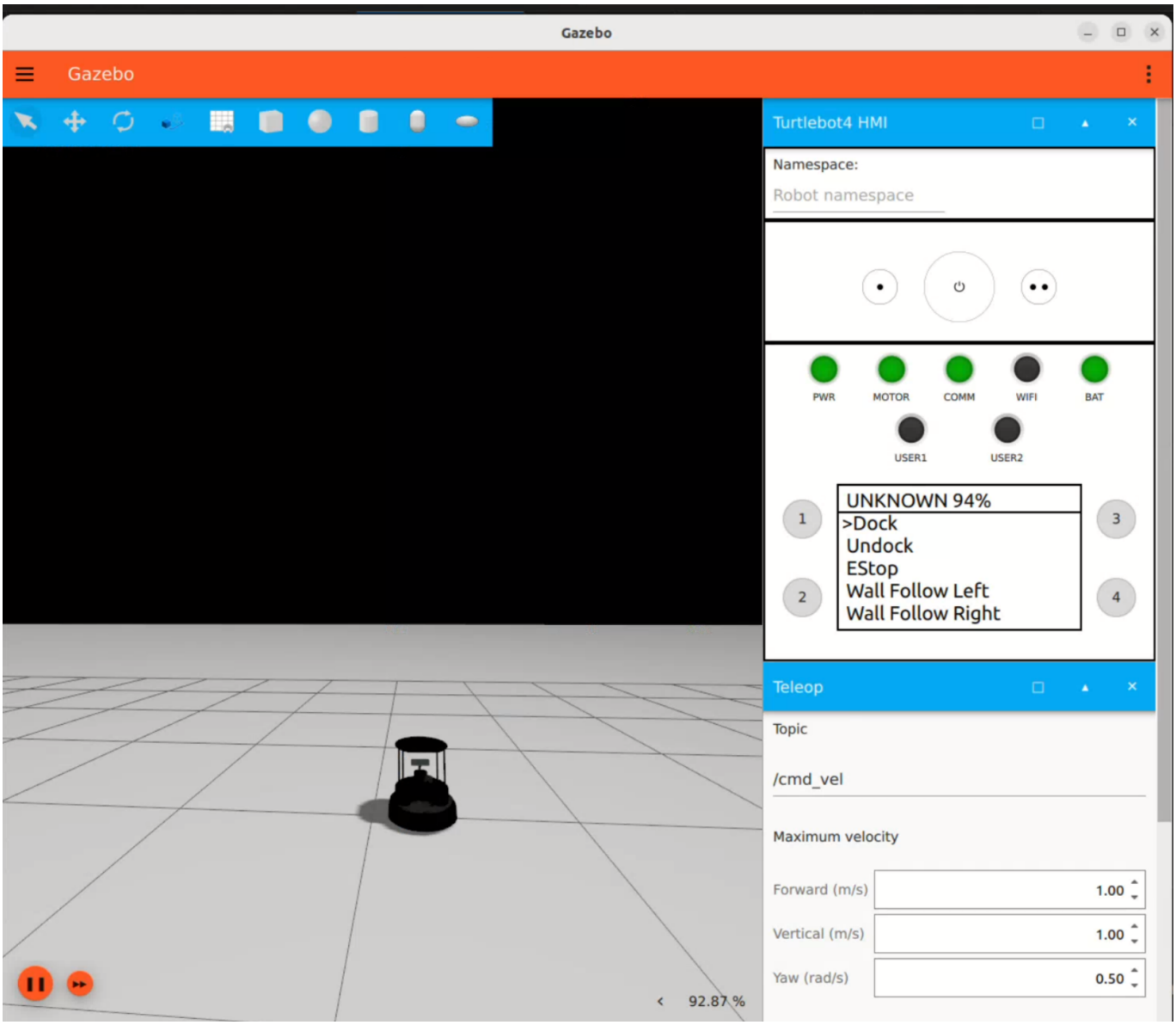
by y.cao

about

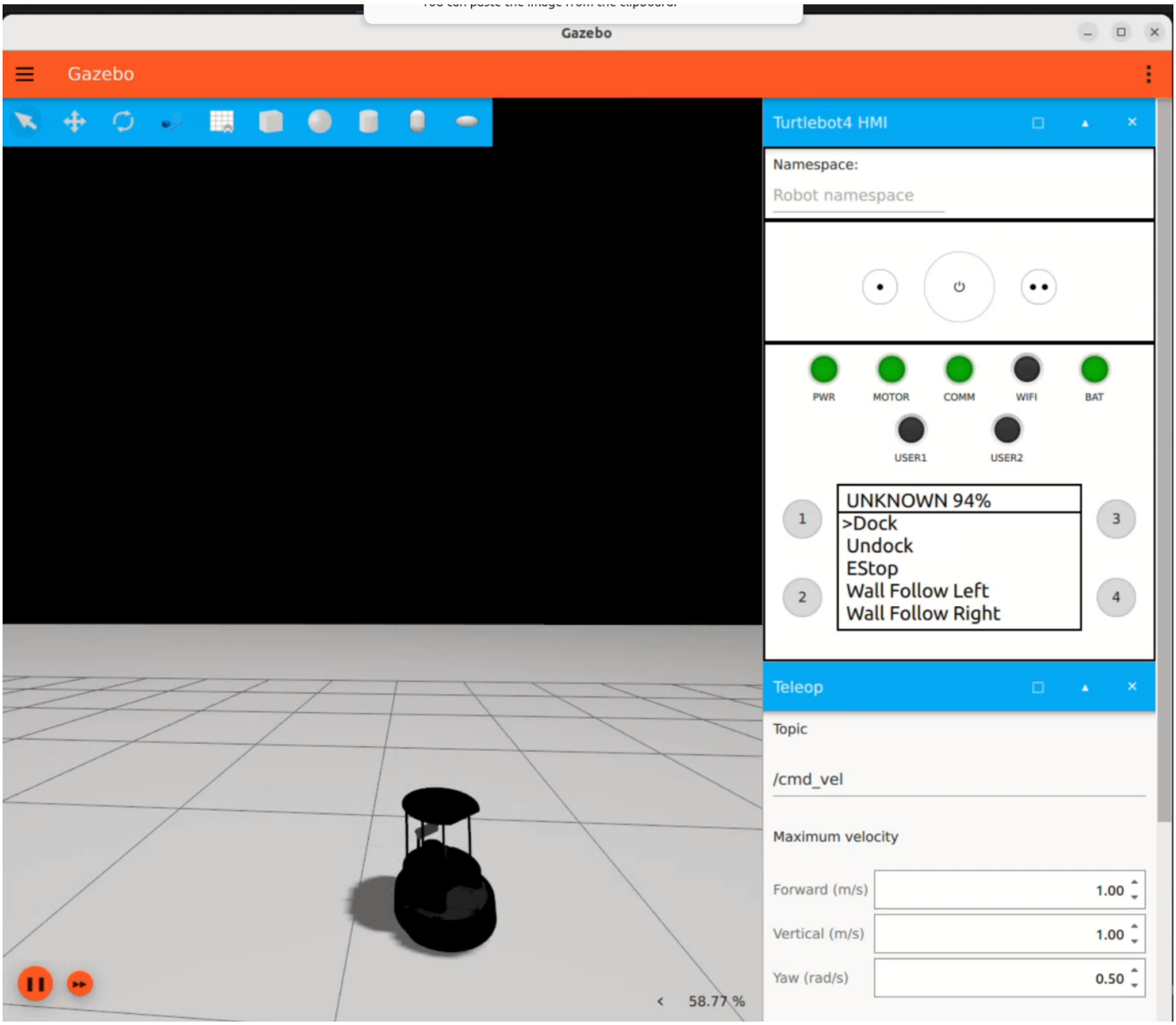
在这次作业中，我使用了C++作为主要语言。首先，先将代码从Python改成C++。 下面是初始的截图。



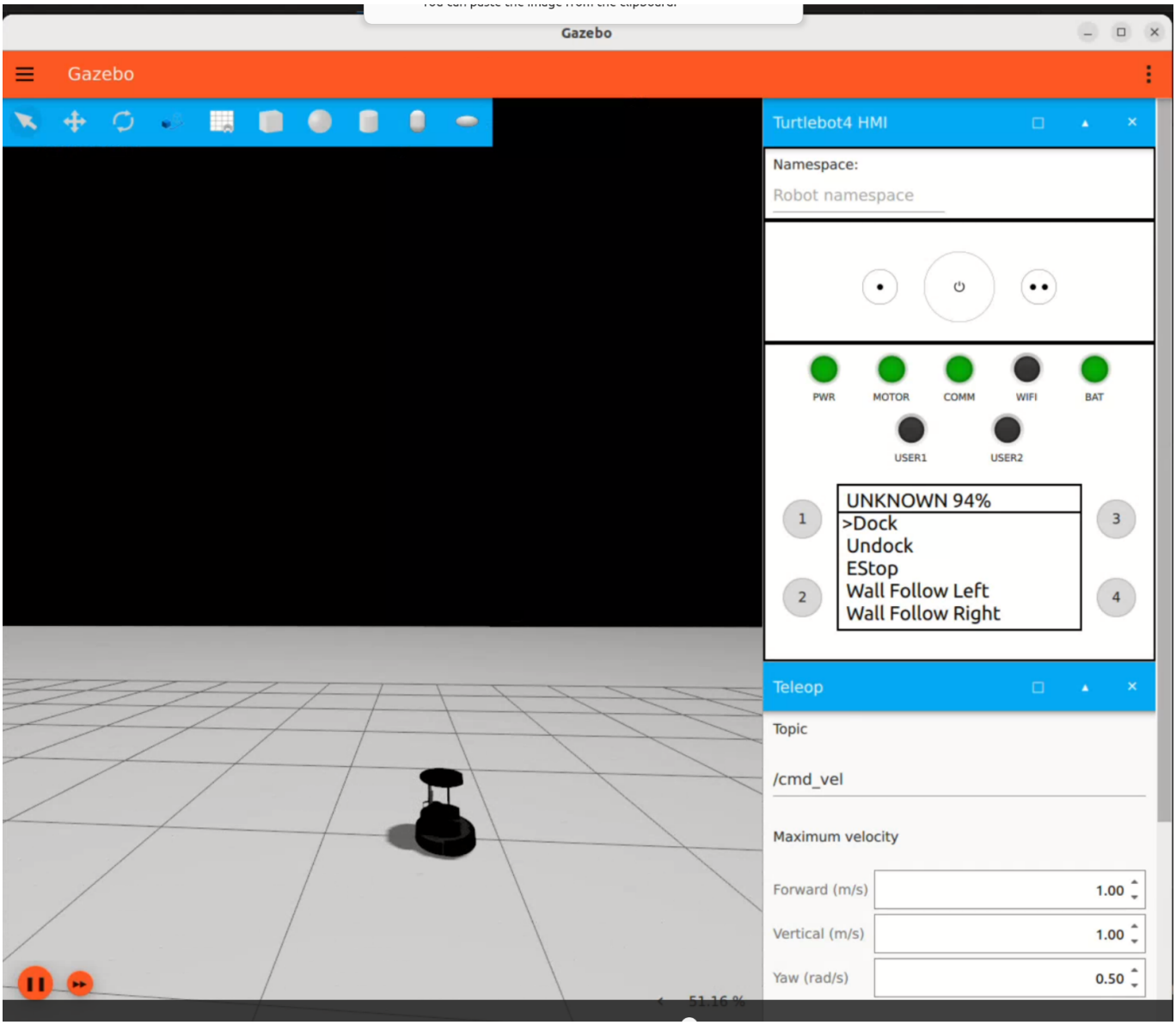
然后，小车向前移动，以1m/s的速度运行1秒。



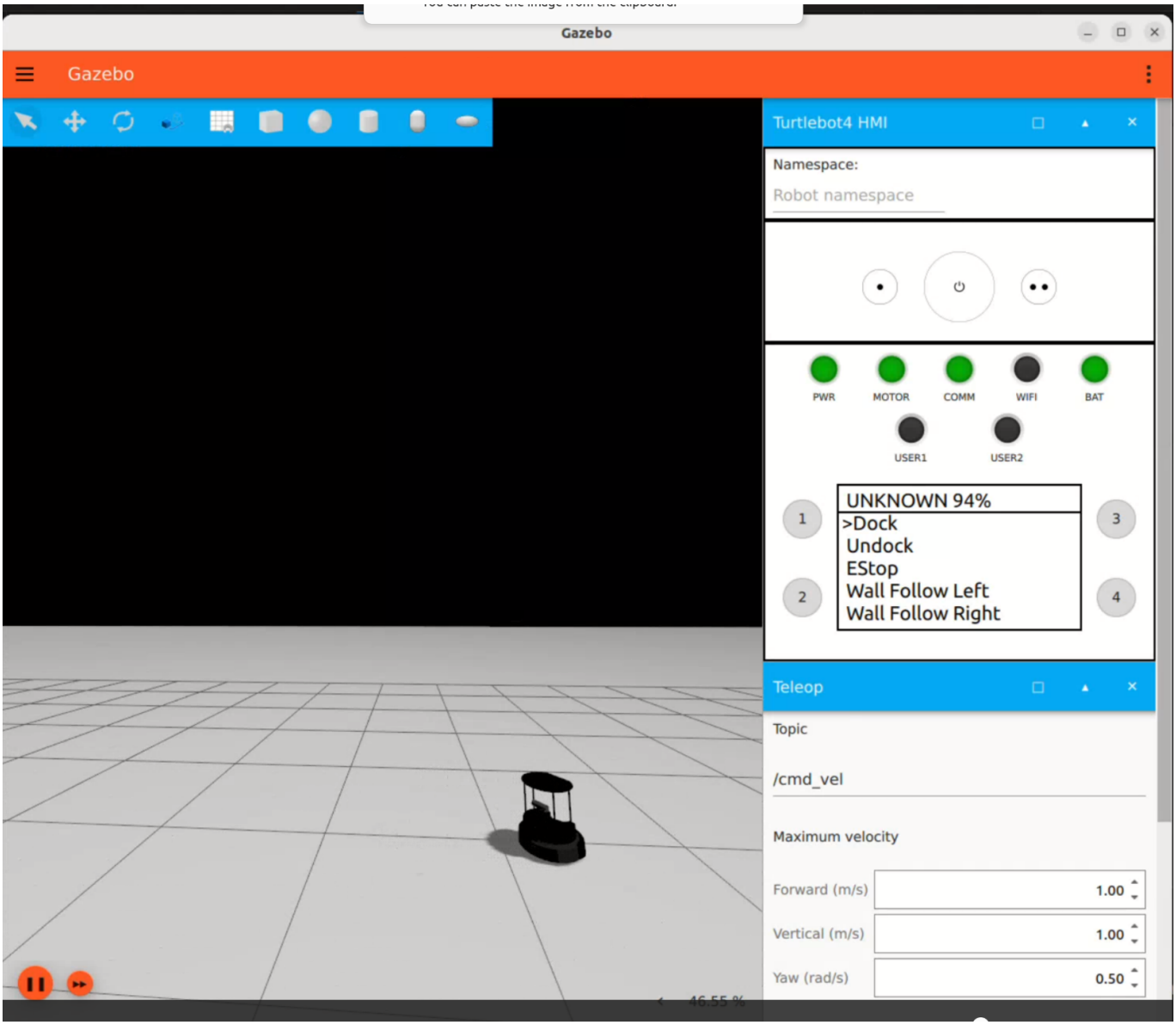
小车进行第一次旋转，并且向前继续移动。



小车进行第二次旋转，并继续向前运动。



小车进行第三次旋转，并继续向前运动。



可以看出，在开环控制下，小车的运动十分不精确，噪声很大。 我将代码附在下面。

```
#include<iostream>
#include "rclcpp/rclcpp.hpp"
#include "geometry_msgs/msg/twist.hpp"
#include <cmath>

class Turtlebot : public rclcpp::Node
{
private:
    /* data */
    rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr vel_pub_;
    rclcpp::TimerBase::SharedPtr timer_;
    int counter_;
    void run();

public:
    Turtlebot(const std::string& node_name);
    ~Turtlebot();
};
```

```
Turtlebot::Turtlebot(const std::string& node_name) : Node(node_name),
counter_(0)
{
    RCLCPP_INFO(this->get_logger(), "Press ctrl+c to terminate");

    // publisher for cmd_vel topic

    vel_pub_= this->create_publisher<geometry_msgs::msg::Twist>("cmd_vel",
10);

    timer_ = this->create_wall_timer(std::chrono::milliseconds(100),
std::bind(&Turtlebot::run, this));
}

Turtlebot::~Turtlebot()
{
}

void Turtlebot::run()
{
    geometry_msgs::msg::Twist vel;
    vel.linear.y = 0;
    if(counter_ >=0 && counter_<=10)
    {
        vel.linear.x = 1.0;
        vel.linear.y = 0;
        vel.linear.z = 0;
        vel.angular.x = 0;
        vel.angular.y = 0.0;
        vel.angular.z = 0.0;
    }
    else if (counter_>=10&& counter_<50)
    {
        vel.linear.x = 1.0;
        vel.linear.y = 0;
        vel.linear.z = 0;
        vel.angular.x = 0;
        vel.angular.y = 0.0;
        vel.angular.z = 0.0;
    }
    else if (counter_>=50 && counter_<60)
    {
        vel.linear.x = 0.0;
        vel.linear.y = 0.0;
        vel.linear.z = 0.0;
        vel.angular.x = 0.0;
        vel.angular.y = 0.0;
        vel.angular.z = M_PI_2;
    }
    else if (counter_>=60 && counter_<100)
    {
        vel.linear.x = 1.0;
        vel.linear.y = 0.0;
        vel.linear.z = 0.0;
```

```
            vel.angular.x = 0.0;
            vel.angular.y = 0.0;
            vel.angular.z = 0.0;
        }
        else if (counter_>=100 && counter_<110)
        {
            vel.linear.x = 0.0;
            vel.linear.y = 0.0;
            vel.linear.z = 0.0;
            vel.angular.x = 0.0;
            vel.angular.y = 0.0;
            vel.angular.z = M_PI_2;
        }
        else if (counter_>=110 && counter_<150)
        {
            vel.linear.x = 1.0;
            vel.linear.y = 0.0;
            vel.linear.z = 0.0;
            vel.angular.x = 0.0;
            vel.angular.y = 0.0;
            vel.angular.z = 0.0;
        }
        else if (counter_>=150 && counter_<160)
        {
            vel.linear.x = 0.0;
            vel.linear.y = 0.0;
            vel.linear.z = 0.0;
            vel.angular.x = 0.0;
            vel.angular.y = 0.0;
            vel.angular.z = M_PI_2;
        }
        else if (counter_>=160 && counter_<200)
        {
            vel.linear.x = 1.0;
            vel.linear.y = 0.0;
            vel.linear.z = 0.0;
            vel.angular.x = 0.0;
            vel.angular.y = 0.0;
            vel.angular.z = 0.0;
        }
        else
        {
            vel.linear.x = 0.0;
            vel.linear.y = 0.0;
            vel.linear.z = 0.0;
            vel.angular.x = 0.0;
            vel.angular.y = 0.0;
            vel.angular.z = 0.0;
        }
        vel_pub_->publish(vel);
        counter_++;

    }
```

```
int main(int argc, char* argv[])
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<Turtlebot>("turtlebot_move");
    rclcpp::spin(node);

    rclcpp::shutdown();
    return 0;
}
```