```cpp
#include "rclcpp/rclcpp.hpp"
#include "lgsvl_msgs/msg/vehicle_control_data.hpp" // Publish to ->
#include "lgsvl_msgs/msg/can_bus_data.hpp" // Subscribe to <-

class PIDControllerNode : public rclcpp::Node
{
public:
    PIDControllerNode() : Node("pid_controller")
    {
        des_spd = 20.0; // Desired speed

        Kp = 2.0;
        Ki = 0.05;
        Kd = 0.5;

        err_pre = 0.0;
        err_int = 0.0;

        cmd_thr_publisher_ = this->create_publisher<lgsvl_msgs::msg::VehicleControlData>(
            "/lgsvl/vehicle_control_cmd", 100);
        mes_spd_subscriber_ = this->create_subscription<lgsvl_msgs::msg::CanBusData>(
            "/lgsvl/state_report", 100, std::bind(&PIDControllerNode::callbackMeasurement,
this, std::placeholders::_1));

        control_loop_timer_ = this->create_wall_timer(
            std::chrono::milliseconds(10), std::bind(&PIDControllerNode::controlLoop, this));

        dt = 0.01; // delta t = 10 ms
    }

private:
    void callbackMeasurement(const lgsvl_msgs::msg::CanBusData::SharedPtr mes)
    {
        mes_ = *mes.get(); // Measurment data
    }

    void publishCmdThr(double acceleration_pct, double braking_pct)
    {
        auto msg = lgsvl_msgs::msg::VehicleControlData();
        msg.acceleration_pct = acceleration_pct;
        msg.braking_pct = braking_pct;
        cmd_thr_publisher_->publish(msg);
    }

    void controlLoop()
    {

        double err = des_spd - mes_.speed_mps;
        err_int = err_int + err*dt;
        double err_der = (err - err_pre)/dt;

        u = Kp*err + Ki*err_int + Kd*err_der;

        err_pre = err;

        auto msg = lgsvl_msgs::msg::VehicleControlData();

        if (u > 1.0) // Braking Operation

        {
            msg.acceleration_pct = 1.0;
            msg.braking_pct = 0.0;
```

```
        }

        else if (u > 0.0 && u <= 1.0) // Braking Operation

        {
            msg.acceleration_pct = u;
            msg.braking_pct = 0.0;
        }

        else if (u > -1.0 && u <= 0.0) // Acceleration Operation

        {
            msg.acceleration_pct = 0.0;
            msg.braking_pct = -u;
        }

        else // Acceleration Operation

        {
            msg.acceleration_pct = 0.0;
            msg.braking_pct = 1.0;
        }

        cmd_thr_publisher_->publish(msg);
    }

    double des_spd;
    double Kp;
    double Ki;
    double Kd;
    double dt;
    double err_pre;
    double err_int;
    double u;
    lgsvl_msgs::msg::CanBusData mes_;

    rclcpp::Publisher<lgsvl_msgs::msg::VehicleControlData>::SharedPtr cmd_thr_publisher_;
    rclcpp::Subscription<lgsvl_msgs::msg::CanBusData>::SharedPtr mes_spd_subscriber_;
    rclcpp::TimerBase::SharedPtr control_loop_timer_;
};

int main(int argc, char **argv)
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<PIDControllerNode>();
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```