THRUSTER CONTROL SYSTEM

Autonomous Catamaran – IIT Kharagpur

Overview

This project implements a robust, real-time keyboard interface for controlling the left and right thrusters of an autonomous surface vehicle (catamaran) using a Raspberry Pi 3B. The system is designed for safety, reliability, and extensibility, with all commands logged for analysis and future development. It is ready for remote, headless operation and future sensor integration.

System Architecture

Hardware

- **Raspberry Pi 3B**
  - Raspberry Pi OS 32-bit (headless, SSH-enabled)
  - 32GB MicroSD Card (Class 10)
  - Powered via USB from a computer
- **Propulsion**
  - 2 × T-100 Thrusters (Blue Robotics)
  - 2 × Electronic Speed Controllers (ESCs, compatible with T-100)
    - Left ESC: GPIO 17
    - Right ESC: GPIO 18
- **Power**
  - 4 × 3.7V $LiFePO_4$ (Lithium Iron Phosphate) batteries (for ESCs/thrusters)
  - Jumper wires and secure connectors
- **Networking**
  - WiFi via mobile hotspot for SSH access
- **Ground Station Computer**
  - Ubuntu OS, AMD Ryzen 7, NVIDIA RTX 3050 GPU
  - Provides SSH access, Pi power, and is used for code development, data analysis, and future compute-intensive tasks

---

Software Dependencies

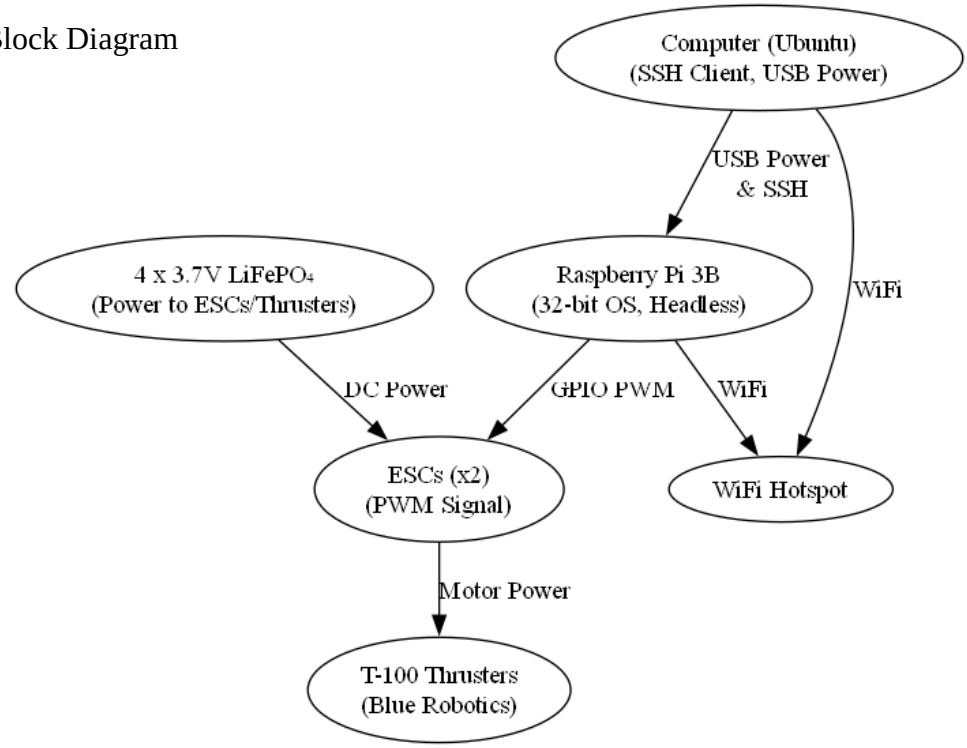
- Python 3.x
- pigpio (for GPIO PWM control)
- curses (for keyboard-driven terminal UI)
- csv (for logging)
- time, sys (for timing, system control)
- pigpio daemon (sudo pigpiod must be running)

Features

- **Real-time keyboard control** of left and right thrusters (individually or together)
- **Safety mechanisms:**
    - Emergency stop (toggle)
    - Pause/resume
    - ESC arming sequence at startup
- **Comprehensive logging** of all command events (timestamped, saved to thruster_log.csv)
- **Live status display** in the terminal
- **Graceful shutdown:** Neutralizes thrusters and cleans up GPIO on exit

---

Hardware Block Diagram



Bill of Materials (BOM)

Item	Quantity	Model/Spec	Notes
Raspberry Pi 3B	1	3B	Main controller
MicroSD Card	1	32GB, Class 10	OS, code, data storage
T-100 Thruster	2	Blue Robotics	Propulsion
ESC (for T-100)	2	Blue Robotics	PWM controlled
$LiFePO_4$ Battery	4	3.7V, 18650 or similar	For ESC/thruster power
Jumper Wires	Several	Male-Female	GPIO and ESC connections
USB Cable	1	Micro USB	Power for Pi
Computer	1	Ubuntu, Ryzen 7, RTX 3050	Ground station, SSH, compute
Mobile Hotspot	1	Any	WiFi for SSH

GPIO Pin Assignments

Function	GPIO Pin
Left Thruster (ESC)	17
Right Thruster (ESC)	18

Usage Instructions

Prerequisites

- Ensure the pigpio daemon is running:

```
bash
sudo pigpiod
```

- Connect ESCs and thrusters to GPIO 17 and 18.

Run the Script

```
bash
python thruster_control.py
```

Follow On-Screen Instructions

- Arm ESCs by pressing Enter.
- Use arrow keys or WASD for control.
- Refer to the on-screen guide for all commands.

Shutdown

- Press q to quit and save the log.
- On exit, all GPIOs are cleaned up and thrusters are set to neutral.

Code Walkthrough

1. **Configuration & Initialization**

- Assign GPIO pins for ESCs.
- Set PWM parameters (neutral, min, max, step).
- Connect to pigpio daemon; exit if not found.

2. **PWM Sending Function**

```
python
def send_pwm(l, r):
pi.set_servo_pulsewidth(ESC_LEFT_PIN, l)
pi.set_servo_pulsewidth(ESC_RIGHT_PIN, r)
now = time.strftime('%Y-%m-%d %H:%M:%S')
writer.writerow([now, l, r])
csvfile.flush()
```

- Sets PWM signals and logs the command with a timestamp.

3. **User Interface (curses-based)**

- **Startup:** Sends neutral PWM to arm ESCs; waits for Enter key.
- **Main Loop:**
    - Arrow keys/WASD: Incremental thruster control.
    - Space: Pause/resume.
    - r: Reset both to neutral.
    - x: Emergency stop toggle.

    - q: Quit and save log.
- **Live Status Display:** Shows current state, PWM values, and safety status.

4. **Safety & Error Handling**

- Emergency stop: Immediate neutral signal, lockout until released.
- Graceful shutdown: Neutralizes thrusters, stops pigpio, closes log file.

5. **Logging**

- All PWM commands are logged to thruster_log.csv with timestamps for traceability and analysis.

Key Design Decisions

- **Terminal UI (curses):** Enables real-time, responsive control without GUI complexity.
- **Comprehensive Logging:** Essential for debugging, safety audits, and performance analysis.
- **Safety First:** Emergency stop and pause features prioritized for safe field operation.

Extensibility

- **Modular Design:**
    - send_pwm() and control logic can be extended for more thrusters or different vehicle configurations.
- **Integration Ready:**
    - Prepared for higher-level autonomy modules, sensor integration, or remote control interfaces.
- **Ground Station:**
    - High-performance computer (Ubuntu, Ryzen 7, RTX 3050) is ready for future compute-intensive tasks (e.g., simulation, ROS, AI).

Known Limitations

- Requires pigpio daemon to be running.
- Terminal-based UI only (no GUI).
- No built-in input validation for extreme/faulty hardware states.

Safety Considerations

- Always power down before hardware changes.
- Double-check all wiring, especially power and ground.
- Use fuses or BMS with $LiFePO_4$ batteries.
- Monitor battery voltage and temperature during operation.

Conclusion

This thruster control script is a **mission-critical tool** for safe, precise, and logged manual operation of the Autonomous Catamaran's propulsion system. It is designed for reliability, operator safety, and ease of use in both lab and field environments, and is ready for the next phase of sensor integration and autonomy.

For any issues or feature requests, please open an issue on GitHub or contact the maintainers.

Prepared by:
Team, Autonomous Catamaran, IIT Kharagpur
Date: 2025-06-07
