Project 3

Project 3 Objectives

At the end of this project, you should be able to:

- control the speed and direction of DC motors through an H-bridge circuit,
- implement and tune a proportional-derivative control law that maintains the hovercraft's heading at some desired orientation, and
- implement a high-level control law that decides when to brake and when to use the PD control law.

Part 1: Circuit

- Mount motor amplifier board
- Connect ducted fans to the output side
- Connect microprocessor to the input side
- Keep away from the compass

Part 2: Fan Control Interface

Must implement:

- `void set_lift_direction(MotorDirection direction)`
  - Determines whether the lift fan is pushing air into or out of the chamber

- In project.h, define:

```
typedef enum {
   BRAKE,
   HOVER
} MotorDirection;
```

  - This new data type has two values: BRAKE and HOVER

Part 2: Fan Control Interface

Must implement:

- `void set_lift_magnitude(int16_t magnitude)`
  - Sets the duty cycle of the middle fan. Must ensure that `magnitude` is in the range [0 … 1023]

- `void set_lateral_magnitudes(int16_t magnitude_left, int16_t magnitude_right)`
  - Sets the duty cycle of the left and right fans. Must ensure that the magnitudes are in the valid range

- Initialization of the PWM channels (more on this today)

Part 2: testA()

Test function implements the following steps:

1. Set thrust direction to HOVER
2. Note the original heading (call it your "goal")
3. Slowly ramp the lift fan thrust upwards until the heading error is above 45 degrees
4. Slowly ramp the lift fan thrust downwards until at zero
5. Set thrust direction to BRAKE
6. Slowly ramp the lift fan thrust upwards until 50% duty cycle
7. Slowly ramp the lift fan thrust downwards until 0% duty cycle

Part 2: testB()

Test function implements the following step:

1. Slowly ramp the left fan up to a duty cycle of 25%
2. While ramping the left fan down to 0%, ramp the right fan up to a duty cycle of 25%
3. Slowly ramp the right fan down to 0% duty cycle

Part 3: Proportional-Derivative Control

Must implement:

-
  ```
  void pd_control(uint16_t forward_thrust, int16_t error,
                                  int16_t rotation_rate)
  ```
  - Implements the PD-control law: compute a left/right differential
  - Add this differential to forward_thrust to derive duty cycle signals for the left/right fans
  - Use the computed duty cycles to set the fan speed

Note: test slowly

Part 4: Main Program

- Start with the template in the project specification and fill in your own code as necessary

- The interrupt service routine sets the flag_timing variable to 1 every 49.152 ms
  - This allows us to ensure that we have ~20 control cycles per second.

Part 5: Hovercraft

- Mount the motor amplifier board
- Connect the board to the batteries (motor power pins only - not the logic pins!)
- No wires near the fans

Checkpoint

- 30 minute meeting by April 4th
- Have parts 1 & 2 completed and tested
- Have part 3 implemented and partially tested
- A successful checkpoint is worth 10% of the project grade

Full Project Due April 11th