Project 4: Motor Control
Project 4: Motor Driver Control

Four ducted fans for our hovercrafts:

• Three lateral fans:
  • Brushed motors
  • Bidirectional control
  • H-Bridges

• One lift fan:
  • Brushless motor
  • Unidirectional control
  • Electronic Speed Control (ESC) unit
Component 1: Circuit

• Right side:
  • H-bridge to battery power
  • H-bridge to fans

• Left side: H-bridge to Teensy
  • Teensy power (+5V) and ground
  • For each fan: PWM magnitude and 2 direction control signals
    • Lift fan: hard-wire direction to push air into the lower chamber

Be careful with direct battery power!
Component 2: Supporting Types/Implementation

Top of program:
// Promise that we will implement this function later
void fsm_step();
// Create a task that will be executed once per 50 ms
PeriodicAction fsm_task(50, fsm_step);

// Gains to be used for reverse thrust
const float FAN_GAIN[] = {1.0, 1.0, 1.0};

Loop:
void loop()
{
    // Check to see if it is time to execute the fsm_task
    fsm_task.step();
}

Andrew H. Fagg: Embedded Real-Time Systems: Project 4
Component 3: Interface Functions

```c
float clip(float value, float min_value, float max_value)
```

```c
void set_lateral_fan_magnitudes(float magnitude[3])
```

- Each magnitude in the range \(-63 \ldots 63\)
- For reverse thrust: magnitude is multiplied by corresponding FAN_GAIN before setting the PWM duty cycle
Setting PWM Duty Cycle

```c
analogWrite(pin, duty);
```

- `pin` = Arduino pin
- `duty` in `[0 ... 255]` (0% to 100%)
  - This is an int! Make sure that you convert your float to an int before calling this function

- Note: negative duty cycles do not make sense & will likely lead to strange behavior
Component 4: Finite State Machine

fsm_step() will implement the following behavior:

When switch is pressed:

• Lateral fans:
  • Ramp left up to 25% duty cycle, then down to 0%,
  • Right: same sequence
  • Back: same sequence
  • Ramp left down to -25% duty cycle, then back to 0%,
  • Right: same sequence
  • Back: same sequence
Coding

• `fsm_step()`:
  • Called once every 50ms
  • Do not include for, while or sleep. Instead, rely on the fact that the function will be called regularly

• Make sure that each function that you implement does exactly what the specification says & no more

• Stick to the documentation specification