Project 4: Motor Control
Eventually: 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

**Loop:**

```c
void loop()
{
    Static PeriodicAction fsm_task(50, fsm_step);

    // Check to see if it is time to execute the fsm_task
    fsm_task.step();
}
```
Component 3: Interface Functions

float bound(float value, float min_value, float max_value)

void set_motor(float val)
• The value is in the range -64 … 64
• The magnitude of the value determines the PWM duty cycle
• The sign of the value determines the state of INa/INb
Setting PWM Duty Cycle

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

- `pin =` Arduino pin (not Analog 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 motor up to 25% duty cycle,
  • Ramp motor down to -25% duty cycle,
  • Ramp motor up to 0% duty cycle
Coding

• fsms_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