```
// motors.hpp: Header file of hardware abstraction for motors and slewing
// Copyright (C) 2017 Ethan Wells
//
// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU Lesser General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU Lesser General Public License for more details.
//
// You should have received a copy of the GNU Lesser General Public License
// along with this program.  If not, see <http://www.gnu.org/licenses/>.

#pragma once
#include "API.h"

/** Class for motor objects */
struct motor_t {
  /** Port the motor is pluggin in to */
  unsigned char port;
  /** The invered status of the motor, should be 1 or -1 */
  char inverted;
  /** The requested power value of the motor */
  int power;
  /** A multiplier for setting the motor values */
  float scale;
  /* The rate at which motor power should increase for every
   * motors::slew::slewWait milliseconds */
  float slewRate;
  /** The last update time of the motor. Is managed by the slew task, so it
   * shouldn't need to be changed */
  unsigned long tlast;
  /** Set the motor to the specified power */
  void set(int power);
}; // struct motor_t

/** Namespace relating to the motors and setting them, initializing them,
 * slewing, etc */
namespace motors {
  /** Sets the motor to the power */
  void set(motor_t motor, int power);

  /** Gets the current power value requested of the motor, analogous of
```

```
   * motor.power */
  int get(motor_t motor);

  /** Returns an initialized motor_t object with the specified parameters, and
   * adds a duplicate of the motor to the motor list for slewing */
  motor_t init(unsigned char port, int inverted, float slewRate, float scale);

  /** Namespace relating to slewing the motors to save the gears and the PTCs */
  namespace slew {
    /** The wait time between each iteration of setting all of the motors */
    static const unsigned char slewWait = 10;
    /** The list of motors, as added to in motors::init() */
    extern motor_t list[11];
    /** The TaskHandle for handling the slewing task */
    extern TaskHandle handle;

    /** Initialization function for slewing. Call in initialize() */
    void init(void);
  } // namespace slew
} // namespace motors
```