```cpp
// motors.cpp: Source 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/>.

#include "../include/motors.hpp"

void motor_t::set(int _power) {
  power = motors::slew::list[port].power = _power * inverted * scale;
}

namespace motors {
  void set(motor_t motor, int power) { motor.set(power); }

  int get(motor_t motor) { return motor.power; }

  motor_t init(unsigned char port, int inverted, float slewRate, float scale) {
    motor_t motor;
    motor.port             = port;
    motor.inverted         = inverted;
    motor.slewRate         = slewRate;
    motor.scale            = scale;
    slew::list[motor.port] = motor;
    return motor;
  }

  namespace slew {
    motor_t list[11];
    TaskHandle handle;

    void _slew(void* none) {
      unsigned long int current;
      while (true) {
        current = millis();
        for (size_t i = 1; i <= 10; i++) {
```

```
        motorSet(i,
                 (int)(((list[i].power - motorGet(i)) * list[i].slewRate) +
                       ((list[i].power >= motorGet(i))
                            ? (current - list[i].tlast - slewWait)
                            : (-1 * (current - list[i].tlast - slewWait))) +
                       motorGet(i)));
        list[i].tlast = current;
      }
      delay(slewWait);
    }
    free(none);
  }

  void init(void) {
    motor_t default_motor;
    default_motor.inverted = 1;
    default_motor.slewRate = 1;
    default_motor.scale    = 0;
    for (size_t i = 1; i <= 11; i++) {
      list[i]            = default_motor;
      default_motor.port = i;
    }
    handle = taskCreate(&_slew, TASK_DEFAULT_STACK_SIZE, NULL,
                        TASK_PRIORITY_DEFAULT + 1);
  }
 } // namespace slew
} // namespace motors
```