```c
/**
 * @file pid.c
 * @brief A PID implementation
 * Copyright (C) 2017 Ethan Wells
 *
 * This program is free software: you can redistribute it and/or modify it
 * under the terms of the GNU 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 General Public License for more
 * details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program. If not, see <https://www.gnu.org/licenses/>
 */

#include "../include/pid.h"
#include <math.h>

void PID(PIDSettings *settings) {
        float error;
        float power;

        error = settings->target - (settings->sensor ?
                                    settings->sensor->average :
                                    settings->root->sensor->average);

        if (sgn(error) != sgn(settings->_error))
                settings->_integral = 0;
        settings->_integral  += error / (millis() - settings->_time);
        settings->_time       = millis();
        settings->_derivative = error - settings->_error;
        settings->_error      = error;

        power = clipNum(
          (settings->kP * error) +
          (settings->kI * ((settings->integralLimit == -1) ? settings->_integral :
                           clipNum(settings->_integral, settings->integralLimit,
                                   -settings->integralLimit))) +
          (settings->kD * settings->_derivative),
          settings->max,
          settings->min);
```

```
        if (!mutexTake(settings->root->_mutex, 5)) {
                return;
        }

        settings->root->power = round(power);
        mutexGive(settings->root->_mutex);

        if (abs((int)(error + .5)) <= settings->tolerance) {
                if (settings->_reached) {
                        if (millis() - settings->_reached >= settings->precision) {
                                settings->isTargetReached = true;
                        } else {
                                settings->isTargetReached = false;
                        }
                } else {
                        settings->_reached        = millis();
                        settings->isTargetReached = false;
                }
        } else if (!settings->_reached || settings->isTargetReached) {
                settings->_reached        = 0;
                settings->isTargetReached = false;
        }
} /* PID */
```