```c
/**
 * @file pid.h
 * @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/>
 */

#pragma once

#include "motors.h"
#include "sensors.h"

/**
 * The settings for step-based PID
 */
typedef struct PIDSettings {
        /**
         * p multiplier for the proportion of the error
         */
        float kP;

        /**
         * i multiplier for the compound of the error
         */
        float kI;

        /**
         * d multiplier for the change in error
         */
        float kD;

        /**
         * The ideal position, or goal value
         */
```

```
float target;

/**
 * Maximum value to be assigned to the controlled system
 */
int max;

/*
 * Minimum value to be assigned to the controlled system
 */
int min;

/**
 * The maximum value the integral will be limited to (-1 for none)
 */
int integralLimit;

/**
 * The amount of distance from target to still be considered *at* the target
 */
int tolerance;

/**
 * How long the sensor must be near it's target, as defined by tolerance, to
 * be considered reached it's target
 */
unsigned long precision;

/**
 * The system the pid controls
 */
Motor *root;

/**
 * Whether or not the instance has remained at it's target, within the range
 ****of tolerance, longer than precision
 */
bool isTargetReached;

/**
 * A sensor to use instead of root->sensor
 */
Sensor *sensor;

/**
 * The output of millis() at the point in time which target within tolerance
```

```
         * was reached. 0 if not currently at target within tolerance
         */
        unsigned long _reached;

        /**
         * The last recorded time
         */
        unsigned long _time;

        /**
         * The integral
         */
        int _integral;

        /**
         * The error
         */
        int _error;

        /**
         * The derivative
         */
        float _derivative;
} PIDSettings;

/**
 * The default PID settings
 */
#define DEFAULT_PID_SETTINGS \
  .kP            = 1,        \
  .kI            = 0,        \
  .kD            = 0,        \
  .target        = 0,        \
  .max           = 127,      \
  .min           = -127,     \
  .integralLimit = 10,       \
  .tolerance     = 5,        \
  .precision     = 220,      \
  .sensor        = NULL

/**
 * Use the Settings to achieve the target, one step at a time
 *
 * @param settings a pointer to the settings to be used
 */
void PID(PIDSettings *settings);
```