```
// sensors.cpp: Source file for hardware abstraction of sensors
// 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/main.h"

namespace sensors {

  quad_t left(1, 2, false);
  quad_t right(3, 4, false);
  pot_t lift(1, false);
  gyro_t gyro(2, 197);

  quad_t::quad_t(unsigned char port1, unsigned char port2, bool _inverted) {
    ports[0] = port1;
    ports[1] = port2;
    inverted = _inverted;
    zero     = 0;
    request  = 0;
  }
  void quad_t::init(void) {
    enc = encoderInit(quad_t::ports[0], quad_t::ports[1], quad_t::inverted);
  }
  long quad_t::value(void) { return (encoderGet(enc) - zero); }
  void quad_t::reset(void) {
    zero    = encoderGet(enc);
    request = 0;
  }

  gyro_t::gyro_t(unsigned char _port, unsigned int _calibration) {
    port        = _port;
    calibration = _calibration;
    zero        = 0;
    request     = 0;
```

```
}
void gyro_t::init(void) { gyro_t::gyro = gyroInit(port, calibration); }
long gyro_t::value(void) { return (gyroGet(gyro_t::gyro) - zero); }
void gyro_t::reset(void) {
  zero    = gyroGet(gyro_t::gyro);
  request = 0;
}

pot_t::pot_t(unsigned char _port, bool _inverted) {
  port     = _port;
  inverted = _inverted;
  zero     = 0;
  request  = 0;
}
void pot_t::init(void) { analogCalibrate(port); }
long pot_t::value(void) {
  return ((analogReadCalibrated(port) - zero) * ((inverted) ? -1 : 1));
}
void pot_t::reset(void) {
  zero    = analogReadCalibrated(port);
  request = 0;
}

sonic_t::sonic_t(unsigned char port1, unsigned char port2) {
  ports[0] = port1;
  ports[1] = port2;
}
void sonic_t::init(void) {
  sonic = ultrasonicInit(sonic_t::ports[0], sonic_t::ports[1]);
}
long sonic_t::value(void) { return ultrasonicGet(sonic); }

button_t::button_t(unsigned char _port, bool _inverted) {
  port     = _port;
  inverted = _inverted;
}
void button_t::init(void) { pinMode(port, INPUT); }
bool button_t::value(void) {
  return (digitalRead(port)) ? ((inverted) ? false : true)
                             : ((inverted) ? true : false);
}

void init(void) {
  left.init();
  right.init();
  lift.init();
```

```
    gyro.init();
  }

  void reset(void) {
    left.reset();
    right.reset();
  }
} // namespace sensors
```