```cpp
// drive.hpp: Header file for utilities relating to the drive
// 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/>.

#ifndef DRIVE_HPP
#define DRIVE_HPP

#include "joystick.hpp"
#include "motors.hpp"
#include "sensors.hpp"

/** Contains everything relating to the drive */
namespace drive {
  /** Class for a side of the drive */
  struct side_t {
    /** Top motor on the the side */
    motor_t topM;
    /** Middle motor on the side */
    motor_t midM;
    /** Bottom motor on the side */
    motor_t lowM;
    /** Sets all motors on the side to the given power */
    void set(int power);
    /** A pointer to the sensor on the side */
    sensors::quad_t* sensor;
  }; // struct side_t

  /** Multiplier for which 1 inch is used to convert into degreees rotation on
   * 4"
   * wheels */
  extern double inch;
  /** The left side of the drive */
  extern side_t left;
  /** The right side of the drive */
```

```
extern side_t right;

/** Set both sides of the drive at their requested powers */
void set(int lpower, int rpower);

/** Initialize the drive subsystem */
void init(void);

/** Drive a specific number of inches */
void inches(long inches);

/** Tank control that should be used in a while loop */
void tank(void);

/** Joystick accelerometer driving! */
namespace accel {
  /** Current x value of the joystick accel */
  extern int x;
  /** Current y value of the joystick accel */
  extern int y;
  /** Previous joystick accel x value */
  extern int prevX;
  /** Previous joystick accel y value */
  extern int prevY;

  /** Tilt control using the josytick accelerometer. Should be used in a while
   * loop */
  void drive(void);
} // namespace accel

namespace gyro {
  /** A driving that can allow an arc, or keep the robot straight */
  class drive {
  public:
    /** A reference to the gyro which will be used to get values from */
    sensors::gyro_t* gyro;
    /** The ideal heading of the robot (is absolute)*/
    int heading;
    /** The urgency/agressiveness of the arc */
    float urgency;
    /** Turn the arc off */
    void off(void);
    /** Use to initialize and run the task */
    drive(int heading, float urgency = 15.f, bool absolute = false,
          sensors::gyro_t* gyro = &sensors::gyro, unsigned int tolerance = 3);
```

```
    private:
      /** The task that runs, keeping the robot straight */
      void task(void* none);
      /** The initial  heading, as opposed to the ideal heading */
      int iHeading;
      /** TaskHandle for the gyro heading task */
      TaskHandle handle;
      /** The internal variable used for changing the pid values */
      float changer;
      /** Whether it is on or not */
      bool on;
      /** The tolerance for turning */
      int tolerance;
    }; // class drive

  } // namespace gyro

} // namespace drive

#endif /* end of include guard: DRIVE_HPP */
```