```
/**
 * @file auto.c
 * @brief The primary source for the autonomous operation period
 * 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/auto.h"

bool isAuto = true;

void autonNone() {}
void autoHerbLeft();
void autoHerbRight();
void autoMeatLeft();
void autoMeatRight();

int   selectedAuton         = 3;
Auton autons[MAX_AUTON + 1] =
{ {
                // index 0
                .name       = "none",
                .sensorName = "lDrv",
                .sensor     = &drive[0].sensor,
                .execute    = &autonNone,
        },{
                // index 1
                .name       = "herb l",
                .sensorName = "rDrv",
                .sensor     = &drive[1].sensor,
                .execute    = &autoHerbLeft,
        },{
                // index 2
                .name       = "herb r",
```

```
                .sensorName = "lIntake",
                .sensor     = &intake[0].sensor,
                .execute    = &autoHerbRight,
        },{
                // index 3
                .name       = "meat l",
                .sensorName = "rIntake",
                .sensor     = &intake[1].sensor,
                .execute    = &autoMeatLeft,
        },{
                // index 4
                .name       = "meat r",
                .sensorName = "gyroC",
                .sensor     = &gyro.child,
                .execute    = &autoMeatRight,
        },
};

void driveToPosition(int l, int r, unsigned long until) {
        driveSettings[0].target = l;
        driveSettings[1].target = r;
        until                  += millis();

        do {
                PID(&driveSettings[0]);
                PID(&driveSettings[1]);

                motorUpdate(&lift);
                sensorRefresh(lift.sensor);
                sensorRefresh(&gyro);

                for (int i = 0; i < 2; i++) {
                        motorUpdate(&drive[i]);
                        sensorRefresh(drive[i].sensor);
                }

                delay(10);
        } while ((!driveSettings[0].isTargetReached ||
                  !driveSettings[1].isTargetReached) &&
                   millis() < until);

        drive[0].power = 0;
        drive[1].power = 0;
        update();
} /* driveToPosition */
```

```
void driveToPositionAngle(int l, int r, int a, unsigned long until) {
        int gError;
        driveSettings[0].target = l;
        driveSettings[1].target = r;
        until                  += millis();

        do {
                gError = (a - gyro.averageVal) * 1.5;
                PID(&driveSettings[0]);
                PID(&driveSettings[1]);
                drive[0].power += gError;
                drive[1].power -= gError;

                motorUpdate(&lift);
                sensorRefresh(lift.sensor);
                sensorRefresh(&gyro);

                for (int i = 0; i < 2; i++) {
                        motorUpdate(&drive[i]);
                        sensorRefresh(drive[i].sensor);
                }

                delay(10);
        } while ((!driveSettings[0].isTargetReached ||
                  !driveSettings[1].isTargetReached) &&
                                                millis() < until);

        drive[0].power = 0;
        drive[1].power = 0;
        update();
} /* driveToPosition */


void turnTo(int angle, unsigned long until) {
        until += millis();

        gyroSettings[0].target = angle;
        gyroSettings[1].target = angle;

        do {
                PID(&gyroSettings[0]);
                PID(&gyroSettings[1]);

                update();
                delay(10);
        } while ((!gyroSettings[0].isTargetReached ||
```

```
                !gyroSettings[1].isTargetReached) &&
                                                millis() < until);

        driveSet(0, 0);
} /* turnTo */

void autonomous() {
        unsigned long startTime = millis();
        isAuto = true;
        reset();

        selectedAuton = clipNum(selectedAuton, MAX_AUTON, 0);
        if (autons[selectedAuton].execute != NULL)
                autons[selectedAuton].execute();

        #ifdef DEBUG_MODE
                printf("\n\n\rFinished autonomous in %ldms\n\n", millis() - startTime);
        #endif

        if (selectedAuton) {
                for (size_t i = 0; i < 2; i++) {
                        driveSettings[i].min = -(driveSettings[i].max = 127);
                        gyroSettings[i].min  = -(gyroSettings[i].max  = 127);
                }

                intakeSet(127);

                driveToPosition(1050, 1050, 2000);
        }

        while (isAutonomous()) {
                update();
                delay(10);
        }
} /* autonomous */

Task driveToPositionAngleT(void *triple) {
        Triple *t = (Triple *)triple;

        driveSettings[0].target = t->a;
        driveSettings[1].target = t->b;

        do {
                PID(&driveSettings[0]);
                PID(&driveSettings[1]);
                drive[0].power += (t->c - gyro.averageVal) * 2.3;
```

```
                drive[1].power -= (t->c - gyro.averageVal) * 2.3;

                motorUpdate(&lift);
                sensorRefresh(lift.sensor);
                sensorRefresh(&gyro);

                for (int i = 0; i < 2; i++) {
                        motorUpdate(&drive[i]);
                        sensorRefresh(drive[i].sensor);
                }

                delay(10);
        } while ((!driveSettings[0].isTargetReached ||
                  !driveSettings[1].isTargetReached));

        drive[0].power = 0;
        drive[1].power = 0;
        delete(triple);
        taskDelete(NULL);
} /* driveToPositionAngleT */
```