```c
/**
 * @file opcontrol.c
 * @brief Controls what happens in operator control
 * 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 <string.h>
#include "../include/robot.h"

#define MOGO_HOLD 300

extern bool isAuto;

int digital(unsigned char joyNum,
            unsigned char channel,
            unsigned char b1,
            unsigned char b2) {
        return joystickGetDigital(joyNum, channel, b2) * -1 +
               joystickGetDigital(joyNum, channel, b1) * 1;
} /* digital */

void moveDrive();
void moveMogo();
void moveIntake();
void moveLift();
void manipPID();
void moveManip();

void autonLeft22();
void autonLeft22T();

void operatorControl() {
        #ifdef DEBUG_MODE
```

```
        printf("Starting Driver Control...\n");
#endif
reset();
update();
isAuto = false;

manipSettings.target = manip.sensor->value;
liftSettings.target   = lift.sensor->value;

/*
 *   if (liftLimit[0].value) {
 *       liftSettings.target = ARM_QUARTER;
 *       PID(&liftSettings);
 *   }
 */

bool isSkills = strstr(autons[selectedAuton].name, "skills");

while (true) {
        if (joystickGetDigital(1, 7, JOY_LEFT) &&
            joystickGetDigital(2, 7, JOY_LEFT)) {
                exit(0);
        }

        if (isSkills) {
                // skillsMogo();
                if (joystickGetDigital(2, 7, JOY_DOWN)) {
                        reset();
                        sensorReset(drive[0].sensor);
                        sensorReset(drive[1].sensor);
                        sensorReset(lift.sensor);
                        sensorReset(mogo.sensor);
                        sensorReset(&gyro);
                        autonLeft22();
                }
        }

        moveDrive();
        moveMogo();
        moveIntake();
        manipPID();
        moveLift();
        update();

        delay(20);
}
```

```
} /* operatorControl */

void moveDrive() {
        drive[0].power = deadBand(joystickGetAnalog(1, 3), 10) +
                         127 * digital(1, 7, JOY_UP, JOY_DOWN) +
                         127 * digital(1, 7, JOY_RIGHT, JOY_LEFT);
        drive[1].power = deadBand(joystickGetAnalog(1, 2), 10) +
                         127 * digital(1, 8, JOY_UP, JOY_DOWN) +
                         127 * digital(1, 8, JOY_LEFT, JOY_RIGHT);
} /* moveDrive */

void moveMogo() {
        int power = 127 * digital(1, 5, JOY_UP, JOY_DOWN);

        if (((mogo.power == 127) || (mogo.power == 9)) && !power)
                power = 9; mogo.power = power;
} /* moveMogo */

void moveLift() {
        static unsigned long lastPress;

        if (digital(2, 6, JOY_DOWN, JOY_UP) + digital(1, 6, JOY_UP, JOY_DOWN) ||

                lift.power = 127 * (digital(1, 6, JOY_UP, JOY_DOWN) +
                                                                                di


                if (lift.power > 0 && manip.sensor->value <
                                                (MANIP_INTAKE + MANIP_HOVER) / 2 - 2
                        lift.power = 0;
                        manipSettings.target = MANIP_HOVER;
                } else if (lift.power) {
                        lastPress = millis();
                }

                if (liftLimit[0].value) {
                        sensorReset(lift.sensor);
                        lift.power = clipNum(lift.power, 0, -127);
                } else if (liftLimit[1].value) {
                        lift.sensor->zero = lift.sensor->value - 1000;
                        lift.power        = clipNum(lift.power, 127, 0);
                }
                liftSettings.target = lift.sensor->value;
        } else if (liftLimit[0].value) {
                  sensorReset(lift.sensor);
                liftSettings.target = 0;
                lift.power          = 0;
```

```
        } else if (liftLimit[1].value) {
                lift.sensor->zero   = lift.sensor->value - 1000;
                liftSettings.target = 1000;
                lift.power          = 0;
        } else {
                PID(&liftSettings);
        }
} /* moveLift */

void moveManip() {
        manip.power = 127 * digital(2, 8, JOY_UP, JOY_DOWN);
}

void moveIntake() {
        intake.power = 127 * digital(2, 5, JOY_UP, JOY_DOWN);
} /* moveIntake */

void manipPID() {
        static unsigned long lastPress;
        static int  power;

        if (joystickGetDigital(2, 7, JOY_UP))
                manipSettings.target = MANIP_PLACE;
        else if (joystickGetDigital(2, 7, JOY_RIGHT))
                manipSettings.target = MANIP_HOVER;
        else if (joystickGetDigital(2, 7, JOY_DOWN))
                manipSettings.target = MANIP_INTAKE;

        power = 127 * digital(2, 8, JOY_UP, JOY_DOWN);

        if (power) {
                if (manip.sensor->value < MANIP_PLACE - 50 && power > 0) {
                        manipSettings.target = MANIP_PLACE;
                        PID(&manipSettings);
                } else {
                        manip.power          = power;
                        manipSettings.target = manip.sensor->averageVal;
                        lastPress             = millis();
                }
        } else if (millis() - lastPress < 190) {
                manip.power = 0;
        } else {
                PID(&manipSettings);
        }
} /* manipPID */
```

```
void autonLeft22T(void *none) {
        autonLeft22();
        taskDelete(NULL);
} /* autonLeft22T */
```