```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 skillsMogo();
void moveArm();
void moveClaw();
void realClawPID();
void limitArmMogo();

void autonLeft22();
void autonLeft22T();

void operatorControl() {
```

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

clawSettings.target = claw.sensor->value;
armSettings.target  = arm.sensor->value;

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

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

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

        moveDrive();
        moveMogo();

        if (isSkills) {
                // skillsMogo();
                if (joystickGetDigital(2, 7, JOY_DOWN)) {
                        reset();
                        sensorReset(drive[0].sensor);
                        sensorReset(drive[1].sensor);
                        sensorReset(arm.sensor);
                        sensorReset(mogo.sensor);
                        sensorReset(&gyro);
                        autonLeft22();
                }
        }
        moveArm();
        moveClaw();
        limitArmMogo();
        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, 6, JOY_UP, JOY_DOWN) +
                    127 * digital(2, 5, JOY_UP, JOY_DOWN);
        if ((mogo.power == 127 || mogo.power == 9) && !power)
                power = 9;
        mogo.power = power;
} /* moveMogo */

void skillsMogo() {
        if ((mogo.sensor->value <= MOGO_HOLD) &&
            !joystickGetDigital(1, 5, JOY_DOWN) &&
            !joystickGetDigital(2, 7, JOY_UP)) {
                mogo.power = clipNum(mogo.power,
                                     127,
                                     (MOGO_HOLD - mogo.sensor->value) * .9 + 13);
        }
} /* skillsMogo */

void moveArm() {
        static unsigned long lastPress, lastDPress;

        if (digital(2, 6, JOY_DOWN, JOY_UP) || (millis() - lastPress < 90)) {
                arm.power = 127 * digital(2, 6, JOY_UP, JOY_DOWN);

                if (arm.power) {
                        lastDPress = false;
                        lastPress = millis();
                }

                if (armLimit[0].value) {
                        sensorReset(arm.sensor);
                        arm.power = clipNum(arm.power, 0, -127);
                } else if (armLimit[1].value) {
                        arm.sensor->zero = arm.sensor->value - ARM_CONE;
                        arm.power        = clipNum(arm.power, 127, 0);
                }

                if (joystickGetDigital(2, 8, JOY_UP)) {
                        lastDPress = true;
                        armSettings.target = ARM_LOAD;
                        PID(&armSettings);
                } else if (!lastDPress)
                        armSettings.target = arm.sensor->value;
        } else if (armLimit[0].value) {
                sensorReset(arm.sensor);
                armSettings.target = 0;
                arm.power          = 0;
        } else if (armLimit[1].value) {
                arm.sensor->zero   = arm.sensor->value - ARM_CONE;
                armSettings.target = ARM_CONE;
                arm.power          = 0;
        } else {
                PID(&armSettings);
        }
} /* moveArm */

void limitArmMogo() {
        if (joystickGetDigital(2, 8, JOY_DOWN))
                return;

        if (mogo.sensor->value > MOGO_PART &&
                        arm.sensor->value > ARM_HALF - 50 &&
                        arm.sensor->value < ARM_3_5_QUARTER && arm.power < -25) {
                armSettings.target = ARM_HALF - 150;
                PID(&armSettings);
        } else if (mogo.sensor->value > MOGO_PART &&
                        arm.sensor->value >= ARM_3_5_QUARTER && arm.power > 28) {
                armSettings.target = ARM_CONE;
                PID(&armSettings);
        }


        if (arm.sensor->value > ARM_3_QUARTER &&
                        arm.sensor->value < ARM_CONE - 80 && mogo.power > 25)
                mogo.power = 0;
}

void moveClaw() {
        if (deadBand(joystickGetAnalog(2, 4), 10)) {
                claw.power = -joystickGetAnalog(2, 4);
                clawSettings.target = claw.sensor->value + (50 * sgn(claw.sensor->velocity))
        } else {
                realClawPID();
        }
} /* moveClaw */

void clawPID() {
        static unsigned long lastPress;
        static bool lastDir;
        static int power;

        power = -joystickGetAnalog(2, 4);

        if (power) {
                lastDir             = power > 0;
                claw.power          = -power;
                clawSettings.target = claw.sensor->value;
                lastPress           = millis();
        } else if (millis() - lastPress < 275) {
                clawSettings.target = claw.sensor->value + 50;
        } else if (lastDir) {
                claw.power = 0;
        } else {
                PID(&clawSettings);
        }
} /* clawPID */

void realClawPID() {
        switch (digital(2, 8, JOY_LEFT, JOY_RIGHT)) {
                case 1:
                        clawSettings.target = CLAW_CLOSED;
                        break;
                case -1:
                        clawSettings.target = CLAW_OPEN;
                        break;
                default:
                        break;
        }

        PID(&clawSettings);
}

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