```
#include "main.h"

void wheelMonitorTask(void *parameter){
    while(true){
        bool dLeftDirection = false;
        bool dRightDirection = false;
        bool leftDone = false;
        bool rightDone = false;
        bool gyroStepOne = false;
        bool gyroStepTwo = false;
        int targetOffset = 0;
        mutexTake(runWheelsMutex, 100);
        bool run = runWheels;
        mutexGive(runWheelsMutex);

        mutexTake(driveTicksMutex, 100);
        int target = wheelTargetTicks;
        mutexGive(driveTicksMutex);

        mutexTake(wheelDirMutex, 100);
        enum WheelDirection d = wheelDir;
        mutexGive(wheelDirMutex);

        mutexTake(useGyroMutex, 100);
        bool gyro = useGyro;
        mutexGive(useGyroMutex);

        while(run){
            mutexTake(runWheelsMutex, 100);
            run = runWheels;
            mutexGive(runWheelsMutex);

            int gyroAverage = (gyroGet(gyroOne) + gyroGet(gyroTwo)) / 2;
            if(gyro){
                if((!leftDone || !rightDone) && !gyroStepOne){
                    targetOffset =  abs(target) < 45 ? 30 : abs(target * .6);
                    if(gyroAverage < target - (targetOffset)){
                        dLeft(true, false);
                        dRight(false, false);
                    }else if(gyroAverage > target + (targetOffset)){
                        dLeft(false, false);
                        dRight(true, false);
                    }else{
                        gyroStepOne = true;
                        stopDrive();
                        delay(400);
                    }
                }else if((!leftDone || !rightDone) && !gyroStepTwo){
                    DRIVEBASE_POWER /= 2;
                    if(gyroAverage < target){
                        dLeft(true, false);
                        dRight(false, false);
                    }else if(gyroAverage > target){
                        dLeft(false, false);
                        dRight(true, false);
                    }else{
                        gyroStepTwo = true;
                    }
                    DRIVEBASE_POWER *= 2;
                }else{

                    leftDone = true;
                    rightDone = true;
                    stopDrive();
                }
```

```
                if(leftDone && rightDone){
                    run = false;

                    mutexTake(runWheelsMutex, 100);
                    runWheels = false;
                    mutexGive(runWheelsMutex);
                    stopDrive();
                }
            }else{
                if(abs(encoderGet(leftQuad)) < target){
                    switch(d){
                        case FORWARD: dLeftDirection = false; break;
                        case BACKWARD: dLeftDirection = true; break;
                        case LEFT: dLeftDirection = true; break;
                        case RIGHT: dLeftDirection = false; break;
                    }
                    dLeft(dLeftDirection, false);
                }else{
                    leftDone = true;
                    stopLeft();
                }

                if(abs(encoderGet(rightQuad)) < target){
                    switch(d){
                        case FORWARD: dRightDirection = false; break;
                        case BACKWARD: dRightDirection = true; break;
                        case LEFT: dRightDirection = false; break;
                        case RIGHT: dRightDirection = true; break;
                    }
                    dRight(dRightDirection, false);
                }else{
                    rightDone = true;
                    stopRight();
                }

                if(leftDone && rightDone){
                    run = false;

                    mutexTake(runWheelsMutex, 100);
                    runWheels = false;
                    mutexGive(runWheelsMutex);
                    stopDrive();
                }
            }
            if(leftDone && rightDone){
                run = false;

                mutexTake(runWheelsMutex, 100);
                runWheels = false;
                mutexGive(runWheelsMutex);
                stopDrive();
            }

            delay(20);
        }
        delay(20);
    }
}

void setSyncMove(enum WheelDirection d, int targetTicks, bool enableGyro){
    mutexTake(driveTicksMutex, 100);
    wheelTargetTicks = targetTicks;
    mutexGive(driveTicksMutex);

    mutexTake(wheelDirMutex, 100);
    wheelDir = d;
```

```
    mutexGive(wheelDirMutex);

    zeroDriveSensors();

    mutexTake(runWheelsMutex, 100);
    runWheels = true;
    mutexGive(runWheelsMutex);

    mutexTake(useGyroMutex, 100);
    useGyro = enableGyro;
    mutexGive(useGyroMutex);
}

void dLeft(bool backwards, bool bypassSlew){
    if(bypassSlew){
        mutexTake(motorMutexes[backLeft - 1], 100);
        motorSet(backLeft, backwards ?  -DRIVEBASE_POWER : DRIVEBASE_POWER);
        mutexGive(motorMutexes[backLeft - 1]);

        mutexTake(motorMutexes[backRight - 1], 100);
        motorSet(frontLeft, backwards ?  -DRIVEBASE_POWER : DRIVEBASE_POWER);
        mutexGive(motorMutexes[backRight - 1]);

        mutexTake(motorReqMutex, 100);
        motorReq[backLeft - 1] = backwards ?  -DRIVEBASE_POWER : DRIVEBASE_POWER;
        motorReq[frontLeft - 1] = backwards ? -DRIVEBASE_POWER : DRIVEBASE_POWER;
        mutexGive(motorReqMutex);
    }else{
        mutexTake(motorReqMutex,100);
        motorReq[backLeft - 1] = backwards ?  -DRIVEBASE_POWER : DRIVEBASE_POWER;
        motorReq[frontLeft - 1] = backwards ? -DRIVEBASE_POWER : DRIVEBASE_POWER;
        mutexGive(motorReqMutex);
    }
}

void dRight(bool backwards, bool bypassSlew){
    if(bypassSlew){
        mutexTake(motorMutexes[frontRight - 1], 100);
        motorSet(frontRight, backwards ?  DRIVEBASE_POWER : -DRIVEBASE_POWER);
        mutexGive(motorMutexes[frontRight - 1]);

        mutexTake(motorMutexes[backRight - 1], 100);
        motorSet(backRight, backwards ? DRIVEBASE_POWER : -DRIVEBASE_POWER);
        mutexGive(motorMutexes[backRight - 1]);

        mutexTake(motorReqMutex, 100);
        motorReq[backRight - 1] = backwards ?  DRIVEBASE_POWER : -DRIVEBASE_POWER;
        motorReq[frontRight - 1] = backwards ? DRIVEBASE_POWER : -DRIVEBASE_POWER;
        mutexGive(motorReqMutex);
    }else{
        mutexTake(motorReqMutex, 100);
        motorReq[backRight - 1] = backwards ?  DRIVEBASE_POWER : -DRIVEBASE_POWER;
        motorReq[frontRight - 1] = backwards ? DRIVEBASE_POWER : -DRIVEBASE_POWER;
        mutexGive(motorReqMutex);
    }
}

void analogDrive(){
    mutexTake(motorReqMutex, 100);
    if((joystickGetAnalog(1, 2) > 0 && joystickGetAnalog(1, 3) < 0) || (joystickGetAnalog(1, 2) < 0 && joystickGetAnalog(1, 3) > 0)){
        if(abs(joystickGetAnalog(1, 2)) > ANALOG_DEADZONE){
            motorReq[backRight - 1] = -joystickGetAnalog(1, 2) * TURN_MULTIPLIER;
            motorReq[frontRight - 1] = -joystickGetAnalog(1, 2) * TURN_MULTIPLIER;
        }else{
            motorReq[backRight - 1] = 0;
            motorReq[frontRight - 1] = 0;
```

```
        }

        if(abs(joystickGetAnalog(1, 3)) > ANALOG_DEADZONE){
            motorReq[backLeft - 1] = joystickGetAnalog(1, 3) * TURN_MULTIPLIER;
            motorReq[frontLeft - 1] = joystickGetAnalog(1, 3) * TURN_MULTIPLIER;
        }else{
            motorReq[backLeft - 1] = 0;
            motorReq[frontLeft - 1] = 0;
        }
    }else{
        if(abs(joystickGetAnalog(1, 2)) > ANALOG_DEADZONE){
            motorReq[backRight - 1] = -joystickGetAnalog(1, 2);
            motorReq[frontRight - 1] = -joystickGetAnalog(1, 2);
        }else{
            motorReq[backRight - 1] = 0;
            motorReq[frontRight - 1] = 0;
        }

        if(abs(joystickGetAnalog(1, 3)) > ANALOG_DEADZONE){
            motorReq[backLeft - 1] = joystickGetAnalog(1, 3);
            motorReq[frontLeft - 1] = joystickGetAnalog(1, 3);
        }else{
            motorReq[backLeft - 1] = 0;
            motorReq[frontLeft - 1] = 0;
        }
    }
    mutexGive(motorReqMutex);
}

void strafeLeft(int millis){
    if(millis != 0){
        mutexTake(motorReqMutex, 100);
        motorReq[backRight - 1] = -127;
        motorReq[frontRight - 1] = 127;
        motorReq[backLeft - 1] = 127;
        motorReq[frontLeft - 1] = -127;
        mutexGive(motorReqMutex);

        delay(millis);

        stopDrive();
    }else{
        mutexTake(motorReqMutex, 100);
        motorReq[backRight - 1] = -127;
        motorReq[frontRight - 1] = 127;
        motorReq[backLeft - 1] = 127;
        motorReq[frontLeft - 1] = -127;
        mutexGive(motorReqMutex);
    }
}

void strafeRight(int millis){
    if(millis != 0){
        mutexTake(motorReqMutex, 100);
        motorReq[backRight - 1] = 127;
        motorReq[frontRight - 1] = -127;
        motorReq[backLeft - 1] = -127;
        motorReq[frontLeft - 1] = 127;
        mutexGive(motorReqMutex);

        delay(millis);

        stopDrive();
    }else{
        mutexTake(motorReqMutex, 100);
        motorReq[backRight - 1] = 127;
```

```
        motorReq[frontRight - 1] = -127;
        motorReq[backLeft - 1] = -127;
        motorReq[frontLeft - 1] = 127;
        mutexGive(motorReqMutex);
    }
}

void stopLeft(){
    if(isAutonomous()){
        mutexTake(motorMutexes[frontLeft - 1], 100);
        motorStop(frontLeft);
        mutexGive(motorMutexes[frontLeft - 1]);

        mutexTake(motorMutexes[backLeft - 1], 100);
        motorStop(backLeft);
        mutexGive(motorMutexes[backLeft - 1]);

        mutexTake(motorReqMutex, 100);
        motorReq[backLeft - 1] = 0;
        motorReq[frontLeft - 1] = 0;
        mutexGive(motorReqMutex);
    }else{
        mutexTake(motorReqMutex, 100);
        motorReq[backLeft - 1] = 0;
        motorReq[frontLeft - 1] = 0;
        mutexGive(motorReqMutex);
    }
}

void stopRight(){
    if(isAutonomous()){
        mutexTake(motorMutexes[frontRight - 1], 100);
        motorStop(frontRight);
        mutexGive(motorMutexes[frontRight - 1]);

        mutexTake(motorMutexes[backRight - 1], 100);
        motorStop(backRight);
        mutexGive(motorMutexes[backRight - 1]);

        mutexTake(motorReqMutex, 100);
        motorReq[backRight - 1] = 0;
        motorReq[frontRight - 1] = 0;
        mutexGive(motorReqMutex);
    }else{
        mutexTake(motorReqMutex, 100);
        motorReq[backRight - 1] = 0;
        motorReq[frontRight - 1] = 0;
        mutexGive(motorReqMutex);
    }
}

void stopDrive(){
    stopLeft();
    stopRight();
}
```